PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

ECOLE NATIONALE POLYTECHNIQUE

ELECTRONICS ENGINEERING DEPARTMENT

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR

ENGINEER'S DEGREE
(ELECTRONICS ENGINEERING)

Classification of REST API methods to solve the interoperability problem

Sofiane CHETOUI

Presented and defended publicly on 06/18/2018

Members of the Jury

President	Pr. Adel Belouchrani	ENP
Supervisor	Dr. Mourad Adnane	ENP
Examiner	Dr. Rabah Sadoun	ENP

ENP 2018

10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

ECOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

ELECTRONICS ENGINEERING DEPARTMENT

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR

ENGINEER'S DEGREE
(ELECTRONICS ENGINEERING)

Classification of REST API methods to solve the interoperability problem

Sofiane CHETOUI

Presented and defended publicly on 06/18/2018

Members of the Jury

President	Pr. Adel Belouchrani	ENP
Supervisor	Dr. Mourad Adnane	ENP
Examiner	Dr. Rabah Sadoun	ENP

ENP 2018

10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie

Acknowledgments

I would like first to express my eternal gratitude towards my parents, my family, my professors and my friends who have made me the person I am today. I would like also to thank my supervisor and teacher Dr. Mourad ADNANE for his support and help. I would like also to remind Dr. Rabah SADOUN and Pr. Adel BELOUCHRANI for the enormous gratitude that we express towards them, I feel proud for having you as a teacher.

I would particularly like to express my sincere gratitude to my supervisor at Nokia Dr. Michel LE PALLEC for the continuous support during my internship, for his guidance, motivation and interesting ideas. I would like to thank all the members of the Self-Adaptive Virtual Chaining team and the members of the Future Space team for their help and kindness.

ملخص

ويركز هذا العمل على قضايا التشغيل البيني ، حيث نقترح نهجاً جديداً للتعامل مع التوصيل البيني لأجهزة إنترنت الأشياء ، التي من المفترض أن تخفف من مشكلة قابلية التشغيل البيني. استنادًا إلى تقنيات معالجة اللغات الطبيعية وتقنيات التعلم الآلي ، يهدف الحل المقترح إلى تصنيف أساليب واجهة برمجة التطبيقات من أجل تحديد الإجراء المحدد المتعلق بكل طريقة تلقائيًا دون أي معرفة مسبقة. قد تكون هذه المساهمة جزءاً هاماً من حل أكبر يضمن قابلية التشغيل البيني لأجهزة إنترنت الأشياء دون فرض أي شيء على مختلف أصحاب المصلحة في إنترنت الأشياء. يمكن استخدام التقنية المقترحة أيضًا للكشف عن الإمكانات والواجهات المادية المتاحة لأي جهاز ، والتي يمكن أن تساعد أيضًا في تحسين تقنيات التوصية بالخدمات.

الكلمات الدالة : انترنت الاشياء ، التشغيل البيني ، تصنيف ، معالجة اللغات الطبيعية ، تقنيات التعلم الآلي

Résumé

Ce travail se concentre sur les problèmes d'interopérabilité, où nous proposons une nouvelle approche pour traiter l'interfaçage des objets connectés, ce qui est censé atténuer le problème de l'interopérabilité. Basée sur les techniques de traitement du langage naturel et d'apprentissage automatique, la solution proposée vise à classifier les méthodes API afin d'identifier automatiquement l'action spécifique liée à chaque méthode sans aucune connaissance préalable. Cette contribution peut être un élément important d'une solution plus large qui assure l'interopérabilité des des objets connectés sans rien imposer aux différentes parties prenantes de l’internet des objets. La technique proposée peut également être utilisée pour détecter les capacités et les interfaces physiques disponibles de n'importe quel appareil, ce qui peut également aider à améliorer de manière les techniques de recommandation de service.

Mots clés : Internet des objets, interopérabilité, classification, traitement du langage naturel, apprentissage automatique

Abstract

This work focuses on interoperability issues, where we propose a novel approach to deal with the interfacing of internet of things devices, which is supposed to alleviate the interoperability issue. Based on Natural Language Processing and Machine Learning techniques, the proposed solution aims to classify API methods in order to automatically identify the specific action related to each method without any prior knowledge. This contribution may be an important piece of a larger solution that ensures the interoperability of IoT devices without imposing anything to the different IoT stakeholders. The proposed technique can be also used to detect the capabilities and the available physical interfaces of any device, which can also can help in improving the service recommendation techniques.

Keywords : Internet of things, interoperability, classification, natural language processing, machine learning

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
APIM	Application Programming Interface Methods
CoAP	Constrained Application Protocol
CRG	Control Research Group
DA	Discriminant Analysis
ESA	Explicit Semantic Analysis
GP	General Profile
HAL	Hyperspace Analogue to Language
IoT	Internet of Things
ISP	Internet Service Providers
LSA	Latent Semantic Analysis
ML	Machine Learning
MLAs	Machine Learning Algorithms
NLP	Natural Language Processing
NLTK	Natural Language Tool Kit
PCA	Principal Component Analysis
REST	Representation State Transfer
SDN	Software Defined Network
SOAP	Simple Object Access Protocol
SP	Specific Profile
SVM	Support Vector Machine
VO	Virtual Object

General Introduction

The Internet of Things is seen by all the experts as the next most influencing technological revolution since the internet. Internet of Things will impact in every corner of our planet with more than 50 billion connected objects by 2025. However, its early stages and the way ahead is still long. To realize the vision of truly connected things, there are still major issues to overcome.

The real value of Internet of Things is certainly not about connecting things to the Internet, rather is about making things work together in order to create new services and valuable data, and being able to make connected things of diverse nature communicate and work together. With regards to the exponential growth of Internet of Things devices in gender and number made, Interoperability is one of the major issues in Internet of Things.

Things will need to communicate with each other, things will need to discover the capabilities of other things, things will need to use the capabilities of other things, things will need to exchange data with cloud services, and applications will need to understand the data from things. This work focuses on interoperability issues, we began with a thorough state of the art about the related works and efforts made in this area, and it has been realized that a great effort mainly in standardization is made so far in order to reach the vision of full interoperability wherein devices work together regardless of their nature or implementation. However, all the standardization efforts fail to propose a universal and scalable solution to ensure the interoperability of Internet of Things devices. In this work we propose, a novel approach to deal with the Internet of Things devices interfacing which is supposed to alleviate the interoperability issue, based on Natural Language Processing (NLP) and Machine Learning (ML) techniques, the proposed solution aims to classify API methods in order to automatically identify the specific action related to each API method without any prior knowledge. This contribution may be an important piece of a larger solution that ensures the interoperability of Internet of Things devices without imposing anything to the different IoT stakeholders (users, suppliers, App developers).

CHAPTER 1

Nokia Bell Labs

Chapter 1 Nokia Bell Labs

This chapter gives a brief description about Nokia Bell Labs and the recent projects of the IoT department where my internship took part.

1.1 Bell Labs

Bell labs was originally founded in the late 19th century by the scientist Alexander Graham Bell in the USA. Since its creation it has made seminal scientific discoveries and revolutionized the ICT world in different sectors (computer science, wire-line and wireless networks, optics and so on). Nokia Bell Labs is focused on all the key elements of today's and tomorrow's communication networks. From the physical components within the networks to the mathematical underpinnings describing how networks can operate, the expertise of Bell Labs researchers' expertise in optics, wireless, software, statistics and quantum computing sets our work apart. Projects at Bell Labs generally consist of groupings of experts from diverse disciplines converging to find solutions for the difficult and complex challenges of digital communications. To find out real solutions for many of the complex communication challenges of the industry, Bell Labs researchers often collaborate with both other research organizations all around the world and the academic world.

1.2 Bell-Labs organization

Nokia Bell-Labs gathers 6 Research Labs spread over multiple locations in the world.

- '<u>Access</u>' developing new breakthrough technologies for the communications systems that directly connect with end user,
- '<u>Application Platforms & Software Systems</u>' solving problems of service complexity in future networks by automating and simplifying delivery of cloud and sensor systems with intelligent, self-deploying, and self-healing control,
- '<u>Emerging Materials, Components, and Devices</u>' solving the great device challenges of the future. Research areas include III-V devices, silicon devices, hybrid modules, and emerging materials.
- '<u>IP and Optical Networks</u>' delivering disruptive innovations that significantly improve the scale, flexibility, and agility of IP and optical networks.
- '<u>End to End Mobile Network Solutions</u>' transforming how mobile networks are used, through creation of future-proof, adaptive and massively scalable E2E solutions. It develops new architectures exploiting 5G and beyond,

And finally our Lab: NAACS

- Network Algorithms, Analytics, Control and Security, developing disruptive, breakthrough solutions to critically hard industry problems. It accomplishes this in network protocols, algorithms addressing networks and other complex systems, data and user-security, network control systems, and useful and real-time analytics.
- Research Labs are then organized in several Research Groups. My internship took place in the Control Research Group (CRG) within the NAACS Lab.

1.3 Nokia Bell-Labs France

Nokia Bell labs France, as one Bell-Labs location, counts about 200 researchers located at Villarceaux (Nozay, Essonne), the biggest center of Nokia in Europe with more than 4000 employees, most of them being engineers. Nokia Bell-Labs France partnerships with many of the French and world's most known research institutes and academic universities. My internship took place in the department called IoT-Control belonging to the Control Research group belonging to the NAACS Lab itself. The IoT-Control research department, which is composed of ten's researchers, is focused on problems related to networking, cloud computing and digital security (from a networking perspective). It is composed of different projects addressing several key issues related to the IoT control by considering Software Based Networks (SDN).

1.4 Research projects

Bell Labs research is organized in Projects classified according their degree of maturity (also known as Research Life Cycle). Following the Bell Labs research project organization, The IoT control department especially contributes to a cross-location project called *Future Space* which is part of *FX projects aiming at* consolidating advanced research trough prototyping (Proof of Concept).

The Self Adaptive Virtual Object Chaining-SAVOC project I worked on is part of *10X projects* representing the first step of the Bell Labs Research Life Cycle with the study of disruptive research concepts. Basically, while considering a given issue, a 10X improvement or gain is expected with respect to the known State-of-the-Art. My internship is a contribution to the 10X SAVOC project.

1.4.1 Majord’Home

SAVOC and Future Space are projects originating from the same ‘root’ idea introduced in 2014: the *Majord’Home* concept.

The *Majord’Home* is a major innovation in the control of networks to support IoT services. Indeed, as stated in [1] and [2] the project aims at removing the burden of home network management from the end-users by delegating it (mainly) to Internet Service Providers (ISPs) having view and control of both connected objects and network elements. Interestingly, the *Majord’Home* solution allows users to keep control of their smart-home through a simplified interface, on which they particularly express their expectations. The latter are then translated into low level network configuration instructions, by following the Software-Defined Network Paradigm.

1.4.2 S.A.V.O.C 10X project

Today, *Future Spaces* has a development and implementation focus, centered on the dynamic establishment of Software-Defined LANs involving different smart spaces. The 10X SAVOC project held by our team is more focused on advanced research matters targeting the automated recommendation and deployment of IoT services. This particularly includes the mathematical modeling of IoT services and the development of theoretical tools in order to render the system more intelligent and self-organized.

The main objective of the 10X SAVOC project is the development of autonomic mechanisms for the recommendation and composition of personalized IoT services involving several connected objects. The first outcome of this project is the modeling of Virtual Objects (VO) abstracting connected objects and then the modelling of their composition, by adapting a Graph Theory Framework (Typed Attributed Graph) with structured sets of attributes and types. This work includes a key aspect: the modeling of the physical environment, interacting with connected devices and thus with IoT services. Among the autonomic mechanisms, an automated characterization of IoT services has been demonstrated (Automated building of an IoT service catalog) with respect to *physical interfaces, representing the way connected devices interact with their physical environment.* Such a characterization is then used to perform automated recommendations of IoT services to end-users. Capitalizing on this work, we have investigated the automated characterization and deployment of missing digital functions to make (recommended) IoT service work.

After a brief presentation of the ‘Nokia Bell Labs’ entity and its research projects, the following chapter positions our contribution by first describing the known State-of-the Art in the IoT.

CHAPTER 2

State of the art

Chapter 2 State of the art

This chapter gives a big picture of the issue that we are dealing with in this project. In 2.1 we give a general overview about IoT, where we present the importance and the expected impact of IoT in different domains. Then, under the same section we present briefly the main issues in IoT. Afterwards, in 2.2 we extend on the research topic that our team in Nokia Bell labs is focusing on. Finally, in 2.3 and 2.4 we present the issue that we are trying to deal with in this project as well as the related works to this topic. In 2.5, we sum-up what has been discussed in this chapter.

2.1 General view

The Internet of Things refers to the general idea of connecting things (e.g. watch, light, vehicle) of their ability to interact with their physical environment by sensing, communicating, producing new events and information, thereby, creating new services.

The internet of things (IoT) is seen by all the experts as the next most influencing technological revolution since the creation of the Internet. IoT will completely change the way we live by impacting in every corner of our planet. As an illustration, Cisco reports [41] that IoT should generate about $14.4 trillion in value across all industries in the next decade, with more than 1 trillion connected devices by 2025.

Healthcare will be greatly enhanced by IoT, through wearable devices. For instance, it will allow to monitor the personal health of the patients remotely by sending the patient data to a health monitoring center. In some hospitals, it is already used for security purposes. Newborn babies are given connected wristbands, allowing a wireless network to locate them at any time. If a newborn is taken too close to an exit door without being signed out, elevators would stop working and the exit doors would lock. In the neonatal intensive care unit, nurses receive critical alerts on cell phones about their patients' medical conditions, including heart rate and oxygen changes that sensors have detected.

IoT is also revolutionizing manufacturing by improving the efficiency and productivity of manufacturing operations, according to TATA Consultancy Survey [42], manufacturers utilizing IoT solutions in 2014 saw an average 28.5% increase in revenues between 2013 and 2014. Typically, manufacturers are currently using IoT solutions to track products in their factories, improve their control functionalities, and increase their analytics functionality through predictive maintenance. For instance, it makes logistics much easier, it can be used to monitor every detail such as commodity details, purchasing of raw materials, production and sales of product after sale service, having information about stock, customer's satisfaction etc.

Smart cities is also new concept issued from the IoT, which will solve major problems faced by people living in the cities like pollution, traffic congestion and energy saving. Smart Belly trash [43] will send alerts to municipal services when a bin needs to be emptied. Using web applications, citizens can find free available parking slots across the city. It will help in monitoring the water supply and ensuring that there is adequate water supply for the resident. It will also help to discover if there is any water loss. Tokyo, for example, has calculated a saving of $170 million each year by detecting water leakage problems early. As an evidence, a lot of new services have not been discovered or imagined yet.

Despite the countless devices and applications that will be generated by IoT, and on which we can write several books just to list them, the real stake isn't the devices neither the applications, actually the big corporations don't expect to make big profits on devices themselves. One main value of IoT relates with sensors and especially all the data generated by these devices, which contains a rich information on people's behavior, including: the shows people watch, the products they consume, the ads that influence their buying behavior and the apps they use. All this information allows for recommending personalized services to end-users.

All these examples, are an evidence that IoT will impact greatly our daily lives. Nevertheless, before the IoT being widely accepted, there are some fundamental issues and challenges that need to be addressed in order to benefit from its offered versatile functionalities. In the following we state the major issues in IoT:

1- Heterogeneity: the IoT is expected to make interwork a huge number of devices of diverse nature, coffee machines, smartphones, air conditioner, sensors, actuators, etc. which actually have different capabilities, properties, and different communication technologies (as different 'languages'). The current solutions fail in correctly solving this problem by presenting a universal solution. For instance, many efforts have been spent to develop protocols for ubiquitous and pervasive networking (e.g. ZigBee, Bluetooth), but each solution has its own specific characteristics and application domains.

2- Scalability: regarding the exponential growth of IoT devices in gender and number, scalability remains an important requirement. When designing an IoT system, current and future needs have to be taken into account and if not, the is likely to be left with unusable services and devices that must be replaced, which is an expensive prospect. Scalability issues are present at different levels, including: naming, data communication and networking, information and knowledge management, service provisioning and management.

3- Security and privacy: it is considered as the most concerning issue for IoT systems, especially in some critical applications like healthcare where experts demonstrated that we can easily control the connected devices used for care, which can be a real risk to the life of a patient. Security and privacy requirements can be classified as follows: resilience to attacks, data authentication, access control and client privacy.

4- Constrained resources: owing to requirements on energy consumption, most of devices which will be used in the IoT are expected to be extremely constrained in terms of memory and computing resources. Currently, one of the most used solutions is the 'fog/edge/cloud' computing which provides a big relief for end-devices but at the same time it accentuates other issues, like security and privacy. As an illustration, the blockchain approach is being investigated to provide a level of 'trustability' in the use of unknown devices [51]

5- Search and discovery: with the expected number and diversity of the connected devices in any environments, it is obvious that end-users won't be aware of all available resources, their capabilities and available services, adding to that, the configuration and deployment of these kind of services is becoming more complex and difficult for a typical end-user. This is what makes of the automatic discovery and recommendation mechanisms and deployment of IoT services a key element for the widespread of the IoT. These techniques aim at taking the end-user out of the complex tasks (at the expense of a fined-grained control), while providing efficient and interoperable solutions. By now, current approaches hardly comply with all the specific requirements, capabilities and characteristics of constrained devices.

Each issue consists an integral research field, and what makes of the IoT one of the most challenging technological revolutions is its different related requirements on all aspects, including security, interoperability, scalability, search and discovery, etc. In fact, what makes the IoT issues really challenging is that every solution solving a given issue, highly affects the efficiency of the solutions addressing other ones. For instance, some semantic techniques have been proposed to describe IoT devices as services, which offer a quiet good solution to the heterogeneity and interoperability issues. Unfortunately, all these solutions are far way from being lightweight solutions and they usually require high processing and memory capabilities, which is challenging for constrained devices. To overcome this limitation, a solution is to use the ‘fog’ computing in order to lessen the burden of the processing and memory storage of these devices, but this solution raises new privacy and security issues. Hence, the main idea to be concluded is that to develop an efficient and suitable solution for one IoT issue, having a holistic view is highly recommended.

2.2 IoT services

According to many studies [3][4], a home network is expected to have an average of 30 to 60 connected devices. These devices are of different nature and issued from different manufacturers, which greatly affects the ability of the end-users to manage all these connected devices and to benefit from all the richness of the IoT services they could support. Indeed, a pool of connected devices working individually can only offer a limited set of services compared to what they could offer if they were combined. Regarding their growing number and diversity, interfacing connected devices remains a cumbersome task for end-users. Therefore, assisting techniques in the selection and composition of services are seeing as a key element for the full adoption of the IoT.

Through different scenarios, several techniques have been developed to address the service assistance issue, by providing different levels of assistance to the end-user. One of the most critical elements of these different levels of assistance is the knowledge owned by the end-user, and more precisely the knowledge related to both available services (service awareness) and the way to compose (and deploy) these services. In the following we present these scenarios.

The first scenario suggests that the end-user has the full awareness and composition knowledge of the service he would like to benefit from. In this scenario, the user request is performed through a dedicated *user interfa*ce (e.g. WEB Graphical User Interface) by selecting the right set of connected devices from the available ones. Nevertheless, this scenario requires that all end-users have the suitable expertise, which is obviously impossible due to the growing number and the different nature of connected devices in a single smart-environments, and in any smart-environments.

A second scenario suggests the presentation of personalized IoT services based on the knowledge of the available connected devices. In this approach, a limited set of suitable IoT services is presented to the end-users, compared to the presentation of all known service classes with all their instances. This scenario is more convenient than the first one, but its benefit is also influenced by the number of the available personalized services. Basically, it loses all its value in a user context involving multiple locations containing an important number of available services.

A remarkable and common element between all the different scenarios is that they are based on the use of a service catalog, which describes the different classes of IoT services and their classified instantiations involving connected devices and digital functions to make services work. Thus, the challenge about the service assistance techniques is related to the autonomic building of a service catalog, this catalog describing how to compose IoT services. The rapid growth in the number of devices and thus in the number and diversity of services requires that this catalog should be updated and build automatically. The building of a such catalog and its efficiency are mainly related to the modeling and characterization of the different services. The characterization task aims at ‘identifying’ IoT services and ‘understanding’ how known IoT services are composed.

Current solutions represent (and then characterize) an IoT service as a set of connected devices, which means that IoT services are identified by the types of the connected devices composing them. These solutions fail to efficiently address the characterization of IoT services. These solutions suffer from the service polymorphism problem [reference], since the same service may be supported by different types of connected devices. For instance, a video-conference may be supported by: a connected TV and a webcam, a smartphone, a PC etc. The service polymorphism does not allow representing the IoT services in a unambiguous way, which means that same types services may be classified in different service classes, since they are composed of different connected devices. Conversely, the same type of devices may lead to different IoT services, which adds more ambiguity in the representation and characterization of IoT services.

As a solution to these issues, a new physical based service characterization is proposed by Nokia Bell-Labs in [1], which focuses on the way entities of the real word (People, Animals, Plants, etc.) physically interact with connected devices. For instance, a smartphone may have the following *physical interfaces*: a screen, a camera, an accelerometer sensor, a microphone, etc, allowing an interaction with the physical environment. This new service characterization efficiently helps in resolving the problem of service polymorphism and service ambiguity. For instance, a video-broadcasting service is characterized as follows: 1 camera in 1 location and n screens in other different locations, while video-surveillance is characterized as follows: n cameras in different locations and 1 screen in another location. Based on this approach, a service classification algorithm has been proposed in [2], which computes a signature that characterizes and defines the classes of services (e.g. video conference, video surveillance, remote health control). Using this classification algorithm, a computed signature defines each service class. Accordingly, a service catalog is obtained with the classified service instances according to their service class (a name may be assigned by an expert to each service class), e.g. ‘video surveillance’ class). This signature also encodes the types of physical functions of connected devices required to build an instance of a given IoT service class.

Using the obtained signatures (one per service class), a pattern matching algorithm has been proposed in [1], in order to recommend IoT services to users by matching signatures of service classes listed in the catalog with the available connected devices. If a user wants to know what he/she can do with one or several connected objects, this algorithm can recommend him/her the services which signatures are supported by the available connected devices.

This approach, solving the polymorphism and ambiguity problems, represents a pillar of our research work. Nevertheless, there are still crucial elements and issues that must be taken into consideration to make it fully functional. For instance, the discovery of connected devices, their resources, metadata, properties and capabilities is an essential requirement in any IoT ecosystem [5] (By the way, physical functions required to compose IoT services are identified but the way to use the capabilities of connected devices is not addressed by this method). the related discovery mechanisms depend on many building blocks, such as configuration management, registration/un-registration, service exposition, semantic integration [6]. Thus, the issue should be resolved by considering all the involved elements and blocks. It is for this reason that a multitude of frameworks appeared to fix these issues by providing a set of building principles, protocols and standards [7].

Globally, the current IoT based systems don’t provide universal discovery, recommendation and deployment mechanisms for IoT services [8]. Many experts consider this as an important challenge in IoT, since it aims at interfacing all the objects around us, given that most of these objects are not dedicated to directly interact with end-users (as people).

Therefore, in this section related works to compose connected devices to build IoT services were presented, with a focus on assisting techniques. The main goal of this section was to explain how important and challenging the composition of connected devices by using abstract models is, which is going to lead us in the next section to talk about a main constraint which is the work context of this project: how to discover and use services offered by connected devices

2.3 Work context and open issues

Service recommendation in the IoT requires to discover and understand ‘things’ in terms of capabilities (and the way to use these capabilities), properties and metadata. Actually, this requirement is a fundamental one in any Internet of Things ecosystem. This is not restricted to service recommendation; it is generally classified under the ‘interoperability’ umbrella. More generally, Interoperable devices must be able to interwork regardless of their models, manufacturers or industries. The main goal is that the communication between people, processes and things works no matter what screen type, browser or hardware in use [9]. A report made by Digital McKinsey [10] states that Interoperability between IoT systems is critical and this report estimates that it represents 40% of potential value across IoT applications, with that number as high as nearly 60% in some settings, which is making interoperability a hot topic at conferences around the world. However, this fundamental requirement is far away from being met, which means that the IoT still in its infancy in this area and the way ahead is still long. The reason why things are unable to discover and work with each other is mainly related to their heterogeneity, in terms of their supported actions, descriptions and communication protocols (ie‘data structure and languages’).

Hence, to realize the vision of truly connected things the real problem to overcome in service recommendation and composition, and in the IoT in general is the universality of the solution in other words, there must be universal mechanisms for automatic discovery of resources, their properties and capabilities, as well as means to access them. Therefore, this work will focus on the interoperability issues, and more precisely on the interfacing of connected devices. The next chapter will present the related works on this topic.

Actually most of the IoT devices nowadays contain an Application Programming Interface (API) that allows to access the different features of the device in order to integrate it in third party products. These APIs offer a simplified way to connect the different devices, and therefore create an IoT ecosystem. Fig.1 shows a set of different devices supported by Amazon Alexa thanks to their APIs, which actually plays the role of an interface that connects the ends of different devices as depicted in Fig.2.

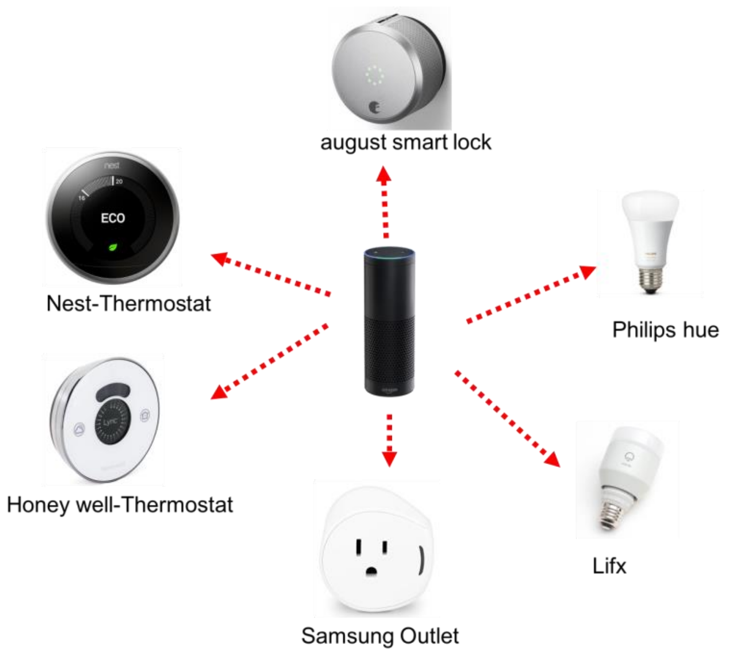


Figure 1. Devices that work with Amazon Alexa

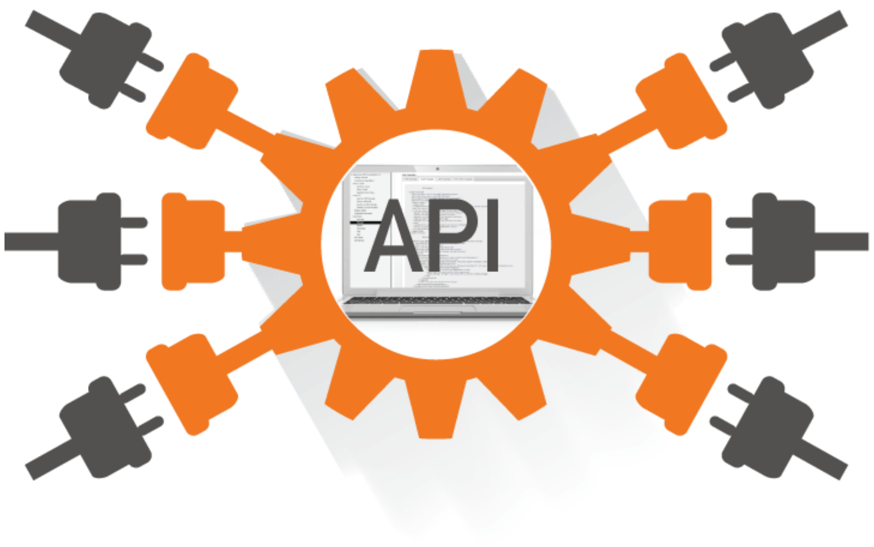


Figure 2. Equivalent diagram that shows the role that plays an API

However, these APIs are not governed by rules or protocols, and as result these APIs don't really resolve the interoperability issues, because basically to be able to get two devices work with each other it's necessary to go through the documentation of the API of both devices and to understand the architecture of each API and finally write the necessary codes, this procedure has to be done each time we need to connect two devices, which makes it highly costly and time consuming, and doesn't scale with the true vision of IoT that aims to connect billions of heterogonous devices.

2.4 Interoperability and related works

A great effort, mainly in standardization, has been made so far in order to reach the vision of all connected things to allow the connected devices to work together no matter what to form a smart environment. The reality, however, is that the IoT is fragmented and lacks interoperability. All these standardization efforts can be divided into two categories:

- 1- Syntactic approach: specific formats or protocols are used in order to describe the devices, their resources and the mechanisms for accessing these resources. Many examples have been given in the last chapter about these formats, such as the CoRE Link Format [12], the Constrained Application Protocol (CoAP) [13] and XMPP IoT Discovery [14]. However, the disadvantage of this approach is the lack of flexibility. Forinstance by defining a specific format, descriptions of device capabilities and proprieties are automatically restricted to the ones already defined in this format, which does not meet at all IoT requirements from the point of view of the scalability and adaptivity of a solution.

- 2- Semantic approach, presenting some flavors of Graph Theory, aim at representing knowledge within a domain as a set of concepts related to each other using ontologies which are defined as “a formal, explicit specification of a shared conceptualization” [15]. An ontology is composed of: Classes, relations, attributes and ‘individuals'. Classes are the main concepts to describe. Each class can have one or several children, known as subclasses, used to define more specific concepts. Classes and subclasses have attributes that represent their properties and characteristics. Individuals are instances of classes or their properties. Finally, relations are the ’Edges’ that connect all the presented components [16]. In other words, rather than defining a format, the semantic approach defines concepts, which gives a real flexibility compared to the syntactic approach. Nevertheless, semantic approach is not sufficient to ensure the interoperability between devices (and related data). Using the semantic approach will just ensure the full interoperability for those using and sharing the same ontology. Furthermore, this approach is unlikely to be used widely in the IoT since it requires high computational performance (lot of Vertices and Edges to describe connected devices and then IoT services), whereas the IoT is often based on constrained devices.

Despite all that, the disadvantages previously mentioned of both approaches are far away from being the most critical; indeed, the nonexistence of a universal syntactic or semantic approach is the main reason of the inconvenience of both approaches to create an environment of interoperable devices. For instance, in [17] we can enumerate more than 400 different ontologies. Indeed from project to project, instead of re-using concepts, many concepts are redefined. Therefore, the two approaches, rather than alleviating the interoperability issues, are making it more difficult to overcome.

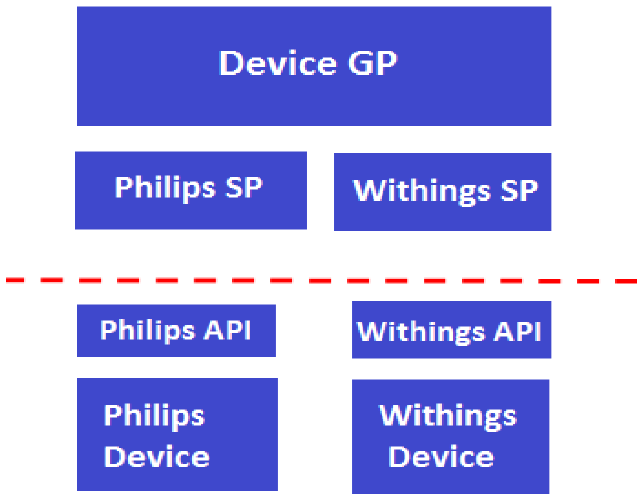


Figure 5. Translation of standard API requests to specific ones

Satoshi Asano et al. in [17] [18] dealt differently with the issue to ensure device interoperability, by proposing a framework to translate standard API requests to device specific API ones, as depicted in fig. 5, the authors propose a general profile (GP) which defines a standard set of API for a specific device class, for example a unique GP is defined for air-conditioners, and a specific profile (SP) that defines the conversion rules needed for the mapping between a vendor API and the GP. Based on this idea, they proposed a collaboration framework supposed to interconnect anything connected regardless of the device specific API, as depicted in fig.6.

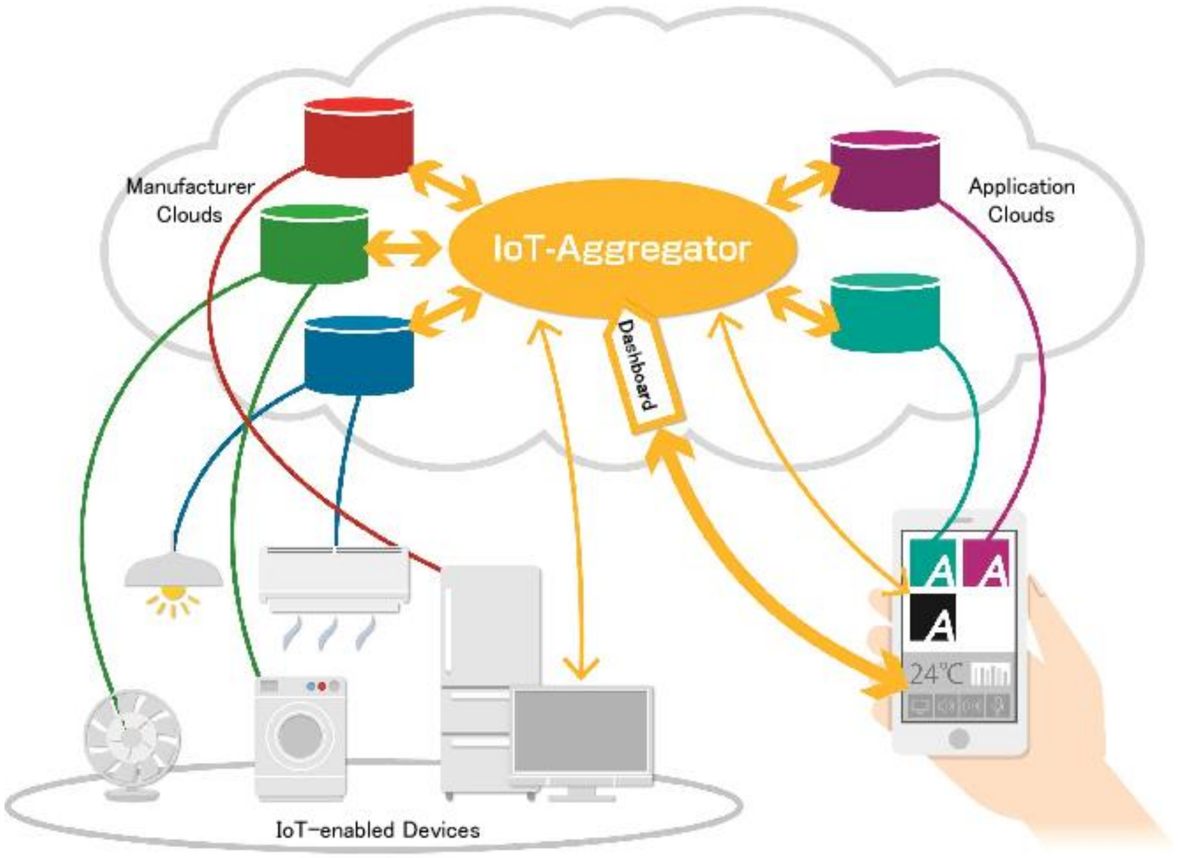


Figure 6. Interoperability collaboration framework [17].

The architecture of the presented solution is interesting but Satoshi Asano et al have simply moved the problem on: without resolving it:

- Their approach imposes a generic profile to any manufacturer, which must be adopted unanimously and which is unlikely to happen since many protocols have failed using this standardization-like approach
- Each manufacturer should develop its own new SP according to the GP for each class of devices.
- Finally, the GP may not support all the functionalities provided by a device manufacturer, and in this case, they suggest that the related manufacturer adds an extension to the provided GP class.

Hence, the API translation effort is ensured by experts performing thus an combined syntaxic/ semantic translation. As such, it does not meet the automated translation requirement targeted in this presented research work. However, from an architectural perspective, the concept of a Generic Profile, abstracting the control interface of connected devices makes sense. We will follow this rationale in our study by targeting an automated classification of API methods under a generic description of control commands.

2.5 Conclusion

The Internet of things is in its early stages and the way ahead is still long, several problems to overcome to realize the vision of all connected things. The real value of the IoT is certainly not about connecting things to the internet, rather is about making things work together in order to create new services. Things will need to communicate with each other, things will need to discover the capabilities of other things, things will need to use the capabilities of other things, things will need to exchange data with cloud services (as other things), and applications will need to understand the data from things. All this allows us to understand not only the importance of the interoperability but also the difficulty that it has to deal with, since the heterogeneity exists at several levels. The related works are unable to solve this issue, either because they fail to take into consideration the scalability of their solution or because they try to impose a new standard to all IoT stakeholders. Therefore, a first step is to propose a scalable and adaptive solution, without imposing anything to the different stakeholders.

CHAPTER 3

Proposed Solution

Chapter 3 Proposed solution

In this chapter, a brief summary about the definition of Application Programming Interface (API) and its importance is given initially, and on which the proposed solution relies, then the proposed solution and its building blocks are explained in the remainder of the chapter.

3.1 Application programming interface

An API is an interface giving a set of methods for discovering and using the different software resources. APIs aren’t new. They have been used by programs in order to facilitate their integration and use in other programs. Currently all the big firms such as Google, Microsoft, Facebook and Twitter have their own APIs for their developers to integrate projects into one another with ease. Today they are considered as a vital tool because they enable companies to grow their businesses more quickly than ever. A good example of a company that has used APIs to grow its business quickly and to move to a market that it has never considered before is Uber. Instead of reinventing the wheel by building its own mapping, payment or communication services, Uber used the best of those programs and connected them all via APIs.

The IoT needs to connect a huge set of connected objects such as cars, medical devices, smart grids and thermostats to the Internet, to make them accessible to other objects and applications. APIs are the key element to achieve this with a minimum of effort. APIs used in the IoT space, are mostly web service APIs which can come in different forms such as Simple Object Access Protocol (SOAP), Representation State Transfer (REST) or XML/JSON, in this work we will focus our study on the REST (Representational State Transfer) technology. since it represents the most used architectural style for APIs and it is more and more adopted as depicted in figure 7,8.

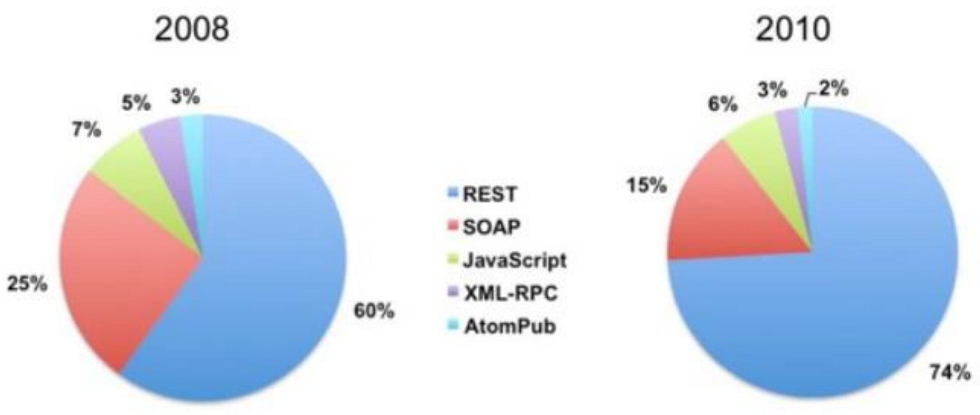


Figure 7. Repartition of architectural styles for API. [52]

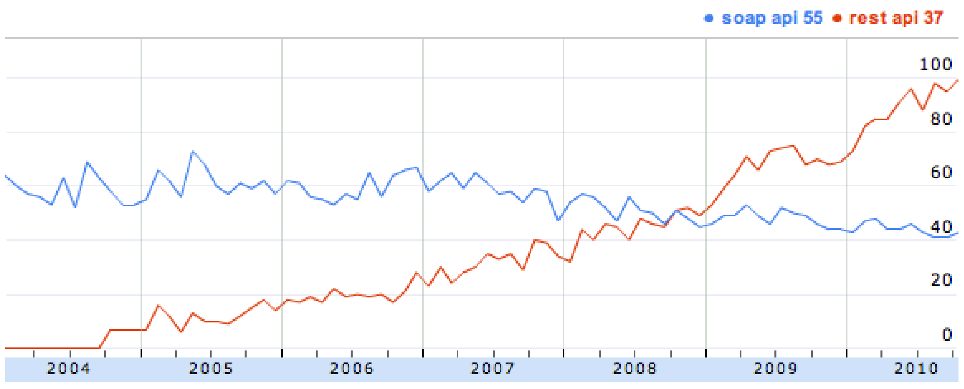


Figure 8. The use of REST and SOAP APIs between 2004-2010. [52]

3.2 REST APIs

Unlike SOAP, REST provides a set of architectural principles rather than an explicit protocol. Some of the features required for a REST API include :

- Aclient-server principle which defines the two entities that interact in a REST API as a client and a server. A client sends a query, and the server returns a response. The latter must have as much information as possible about the client, because it is important that they are able to work independently of each other
- The fact of being "stateless" means that the server has no idea of the client's status between two queries. From the point of view of the server, each request is an entity distinct from the others
- The resources need to be identified using URLs, they can have several representation formats (e.g., HTML, JSON) negotiated at run time using HTTP.
- The client sends an HTTP call to a specific URL with a given verb (GET, POST, PUT or DELETE.)

Just like any HTTP request, REST requests contain a URL, a method, a header and a body, as shown in fig.9 :

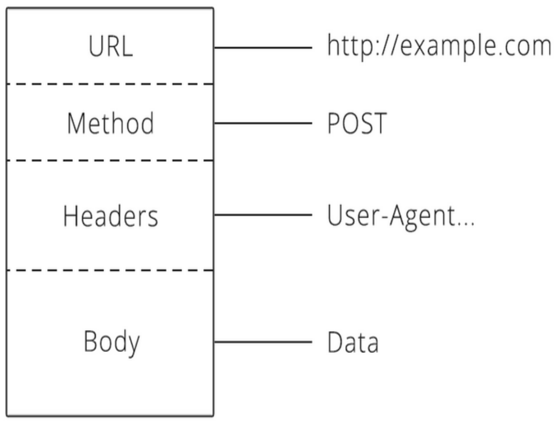


Figure 9. REST request format.

The four methods commonly found in REST APIs are:

- GET - Requests server to find resource,
- POST - Requests the server to create a new resource,
- PUT - Requests the server to modify or update an existing resource,
- DELETE - Requests the server to delete a resource.

whilst the related headers provide meta-information about a query. For example, the type of device used, the time at which the client sent the request and the size of the body, which contains the data that the client wishes to send to the server. Most of the APIs are REST-based APIs, but it doesn't mean that we can control and access any device in an easy way. Indeed, REST is far from being a protocol, rather, it's more a set of guidelines for an architectural style. This means that to be able to manipulate any device or to integrate a set of devices in any environment, the user or the developer must learn how to use each API via the APIs documentation, which is a difficult and a costly solution. Although it concerns the same class of devices, for instance air-conditioner, each manufacturer defines its own URLs and resources in a different way. This yields to new efforts trying to standardize the API design, in order to reduce the development cost, like in [19], but without presenting an universal solution solving the interoperability issues.

3.3 The proposed solution

The proposed solution deals with an important aspect of the interoperability issue which concerns accessing and controlling devices regardless of their type or manufacturer by using their APIs. This solution was proposed after studying many IoT APIs of different manufacturer, like Philips Hue API [20], LIFX API [21], Nuki API [22] Lockitron API [23] and Withings API [24], and after making these observations :

- The APIs use human-readable words in URLs and body parameters
- Different APIs of a specific type of devices use a limited set of keywords
- Different APIs of a specific devices type use similar key words

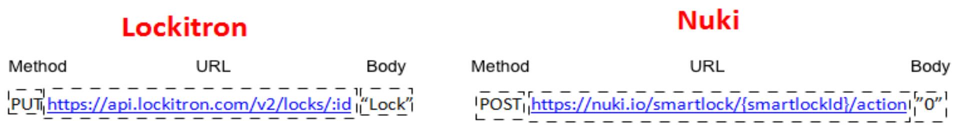

Lockitron
Method
URL
Body
PUT
https://api.lockitron.com/v2/locks/:id
"Lock"
Nuki
Method
URL
Body
POST
https://nuki.io/smartlock/{smartlockId}/action
"0"

Figure 10. REST methods to turn on the light for LIFX and PHILPS.

An illustration of this observation is depicted in Figure 10 and 11, where we can clearly see the existing similarity between API methods when it concerns the same action (e.g. fig. 10 set light on, fig. 36 lock). The existence of such a similarity is quite logical, since manufacturers always try to make their APIs as easy as possible for developers to manipulate and to learn them. One of the most important features of an easy-to- learn and esay-to-integrate API is an API that uses human readable words in the URLs or in the body parameters that relates to the specific use of this method.

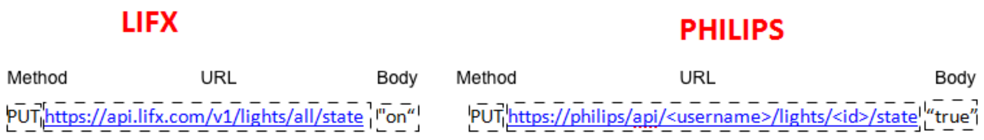
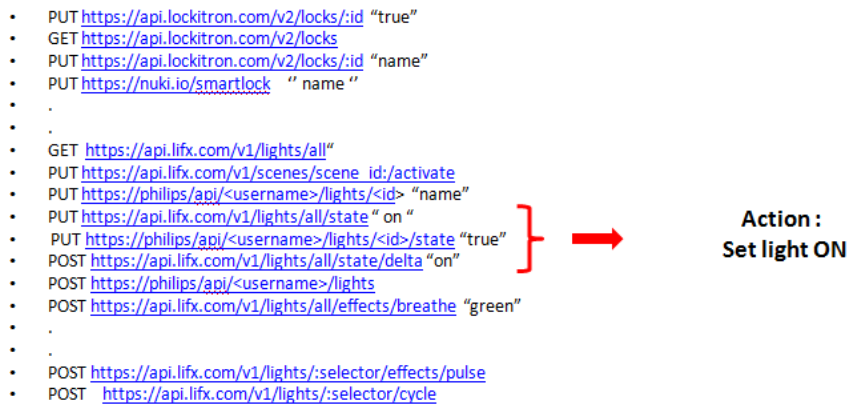

LIFX
Method
URL
Body
PUT
https://api.lifx.com/v1/lights/all/state
"on"
PHILIPS
Method
URL
Body
PUT
https://philips/api/<username>/lights/<id>/state
"true"

Figure 11. REST methods to lock the door for Lockitron and Nuki.

Based on these observations, the proposed solution consists of being able to characterize per action each API method by processing API methods using natural language techniques and machine learning. Fig. 12 :



- PUT https://api.lockitron.com/v2/locks/:id "true"
- GET https://api.lockitron.com/v2/locks
- PUT https://api.lockitron.com/v2/locks/:id "name"
- PUT https://nuki.io/smartlock " name "
- .
- .
- GET https://api.lifx.com/v1/lights/all"
- PUT https://api.lifx.com/v1/scenes/scene_id:/activate
- PUT https://philips/api/<username>/lights/<id> "name"
- PUT https://api.lifx.com/v1/lights/all/state " on "
- PUT https://philips/api/<username>/lights/<id>/state "true"
- POST https://api.lifx.com/v1/lights/all/state/delta "on"
- POST https://philips/api/<username>/lights
- POST https://api.lifx.com/v1/lights/all/effects/breathe "green"
- .
- .
- POST https://api.lifx.com/v1/lights/:selector/effects/pulse
- POST https://api.lifx.com/v1/lights/:selector/cycle

Figure 12. Recognition of the associated API method action.

3.4 Building blocks

In order to define the building blocks required to measure the similarity between API methods, it is necessary to identify the different scenarios of similarity measure that the solution has to deal with, and their corresponding difficulty level. In the following, different scenarios of similarity measure that can occur are enumerated:

- Semantic similarity measure: it relates to a measure of similarity between two words in terms of semantic meaning. It concerns mainly all the words that we can find in a dictionary, an example of a such similarity measure would be between the words "bulb" and "lamp" or between "door" and "light"

- Syntactic similarity measure: it concerns a measure of similarity between a word that has a semantic meaning and another one that hasn't. This kind of similarity is useful when trying, for instance, to measure the similarity between a word and it's misspelled or abbreviated version. An example of a such similarity measure would be between the words "bri" and "bright" or between "smart-light" and "light"

- Other similarity measures: it concerns all similarity measures that do not belong neither to the semantic nor to the syntactic measures. It concerns mainly words which have no semantic or syntactic similarity such us “smart-light” and “lamp”, the word “smart-light” has no semantic meaning, since it doesn’t belong to any dictionary, but also the two words has no syntactic similarity as well.

Therefore, to be able to measure the similarity between different API methods while considering the different scenarios that may occur, the solution must be able to measure the similarity between words that have or have not a semantic meaning, hence the following building blocks were proposed :

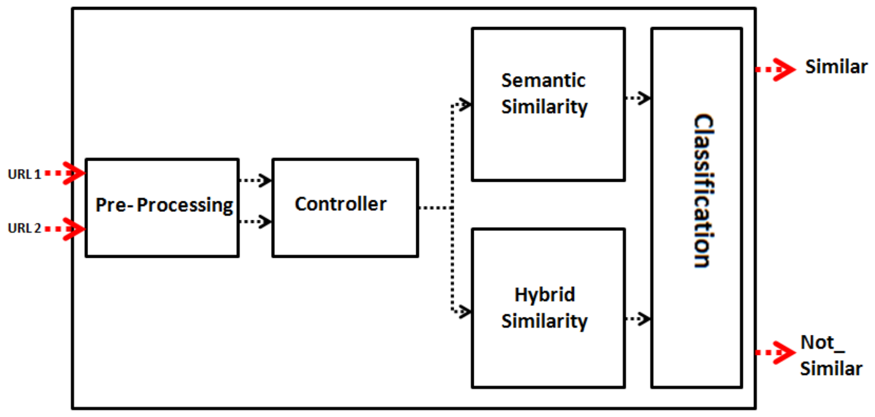


Figure 13. Building blocks

A better explanation of the essential building blocks (Semantic Similarity, Hybrid Similarity and Classification) and their respective functions is given in the next chapters in order to understand the details of the proposed solution and the reasons why such building blocks have been chosen. While, the pre-processing and the controller ensure the simple function of extracting the words from the URLs and sending them to the right block of similarity measure based on whether they have a semantic meaning or not.

CHAPTER 4

Semantic Similarity

Chapter 4 Semantic similarity

Measuring the semantic similarity between words is a very important research area given the necessity of such measures in different tasks, such as document clustering, plagiarism detection, automatic essay scoring and text summarization. In this work we will use a semantic similarity measure to determine whether words like “light” , “lamp” and “bulb” are similar or not. In the following, a short state-of-the-art about the existing semantic similarity measures is presented, then we will evaluate some semantic similarity measure techniques in our use-case, in order to select the most suitable techniques.

4.1 Semantic measures

In semantic measures, there are mainly two categories: **the corpus-based similarity** that uses large corpora of written texts to extract useful information in order to measure the similarity between two words, and **knowledge-based similarity** that uses information derived from *semantic networks*.

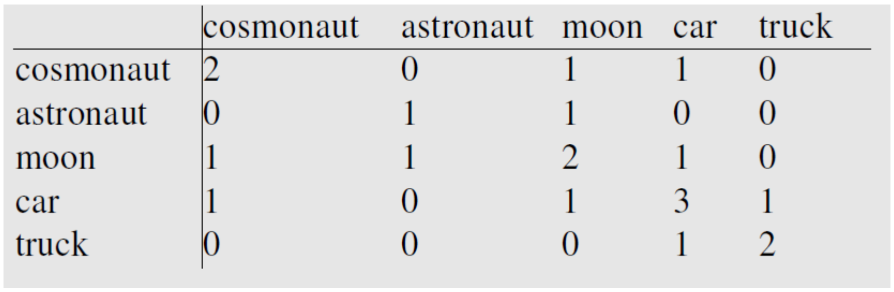
4.1.1 Corpus based similarity techniques

The most known and used corpus-based similarity techniques are :

- **Hyperspace Analogue to Language (HAL):** [25, 26] based on the fact that words with similar meaning repeatedly occur closely in different texts. As an example, in a large corpus of text one could expect to see the words car, automobile and vehicle appear often close to each other. The same might be true for mouse, cat and dog. A word-by-word matrix (fig.14 [49]) is formed with each matrix element is the strength of association between the word represented by the row and the word represented by the column. The user of the algorithm then has the option to drop out low entropy columns from the matrix. As the text is analyzed, a focus word is placed at the beginning of a ten-word window that records which neighboring words are counted as co-occurring. Matrix values are accumulated by weighting the co-occurrence, inversely proportional to the number of words between the current word and the focus word; closer neighboring words are thought to reflect more of the focus word's semantics and so are weighted higher. HAL also records word-ordering information by treating the co-occurrence differently based on whether the neighboring word appeared before or after the focus word.

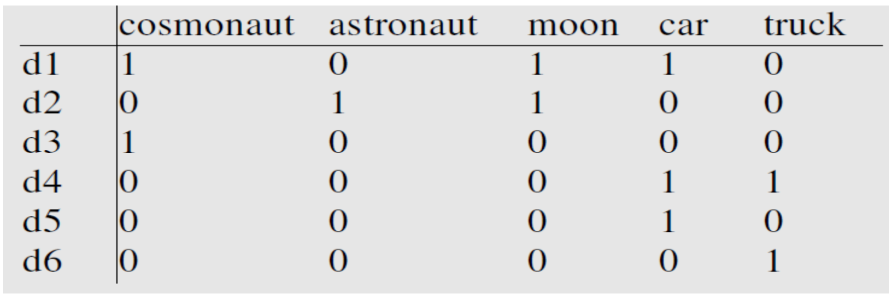
- **Latent Semantic Analysis (LSA):** [27, 28] this technique assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis). A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) or word count per document is constructed (fig.15) from a large piece of text and singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns, words are then compared by taking the cosine [54] of the angle between the two vectors formed by any two rows.

- **Explicit Semantic Analysis (ESA)**: [25, 29] used to calculate the semantic relatedness between two terms or texts by representing a word as a column vector in the term frequency–inverse document frequency matrix(tf-idf) [48] of the text and representing the text as the centroid of the vectors representing its words, just like fig.15 [49] but using the tf-idf. In general, the English version of Wikipedia is used as text corpus to construct the matrix. The semantic relatedness between the two terms (or texts) is expressed by the cosine measure [30] between the corresponding vectors.



	cosmonaut	astronaut	moon	car	truck
cosmonaut	2	0	1	1	0
astronaut	0	1	1	0	0
moon	1	1	2	1	0
car	1	0	1	3	1
truck	0	0	0	1	2

Figure 14. Word by word matrix.

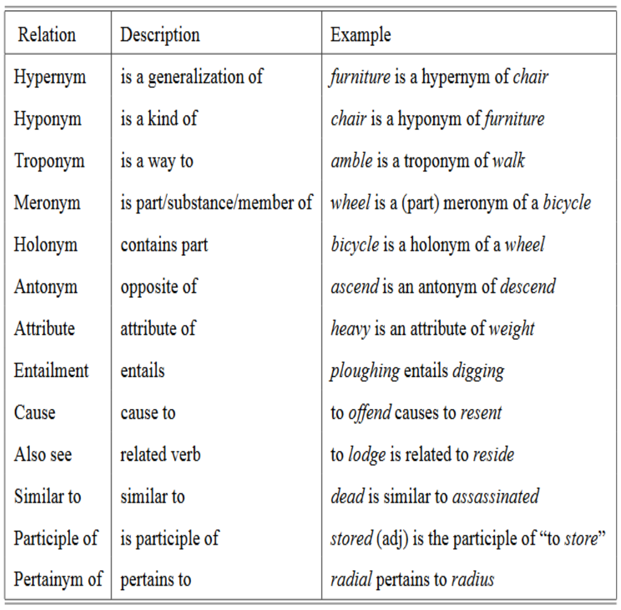


	cosmonaut	astronaut	moon	car	truck
d1	1	0	1	1	0
d2	0	1	1	0	0
d3	1	0	0	0	0
d4	0	0	0	1	1
d5	0	0	0	1	0
d6	0	0	0	0	1

Figure 15. Word by document matrix.

4.1.2 Knowledge based similarity techniques

The second category of semantic measures, the knowledge-based one, uses *semantic networks* in order to extract useful information to compute the similarity between words. **WordNet [31]** is the most used *semantic networks* (graphs) in the area of natural language processing, **which is a large lexical database of English nouns**, verbs, adjectives and adverbs, **which are grouped into sets of cognitive synonyms (synsets),** each expressing a distinct concept and interlinked between each other based on many semantic relations, as shown in Table.1 [50], such as hypernyms, hyponyms, meronym and holonym for nouns, and such as hypernym, troponym and entailment for verbs. That's to say, WordNet can be visualized as a large graph or semantic network, where each node of the network represents a real-world concept and each node is essentially a set of synonyms that represent the same concept, and interlinked with other concepts based on different relations, as depicted in Fig.16 [50].



Relation	Description	Example
Hypernym	is a generalization of	*furniture* is a hypernym of *chair*
Hyponym	is a kind of	*chair* is a hyponym of *furniture*
Troponym	is a way to	*amble* is a troponym of *walk*
Meronym	is part/substance/member of	*wheel* is a (part) meronym of a *bicycle*
Holonym	contains part	*bicycle* is a holonym of a *wheel*
Antonym	opposite of	*ascend* is an antonym of *descend*
Attribute	attribute of	*heavy* is an attribute of *weight*
Entailment	entails	*ploughing* entails *digging*
Cause	cause to	to *offend* causes to *resent*
Also see	related verb	to *lodge* is related to *reside*
Similar to	similar to	*dead* is similar to *assassinated*
Participle of	is participle of	*stored* (adj) is the participle of "to *store*"
Pertainym of	pertains to	*radial* pertains to *radius*

Table 1. WordNet semantic relations.

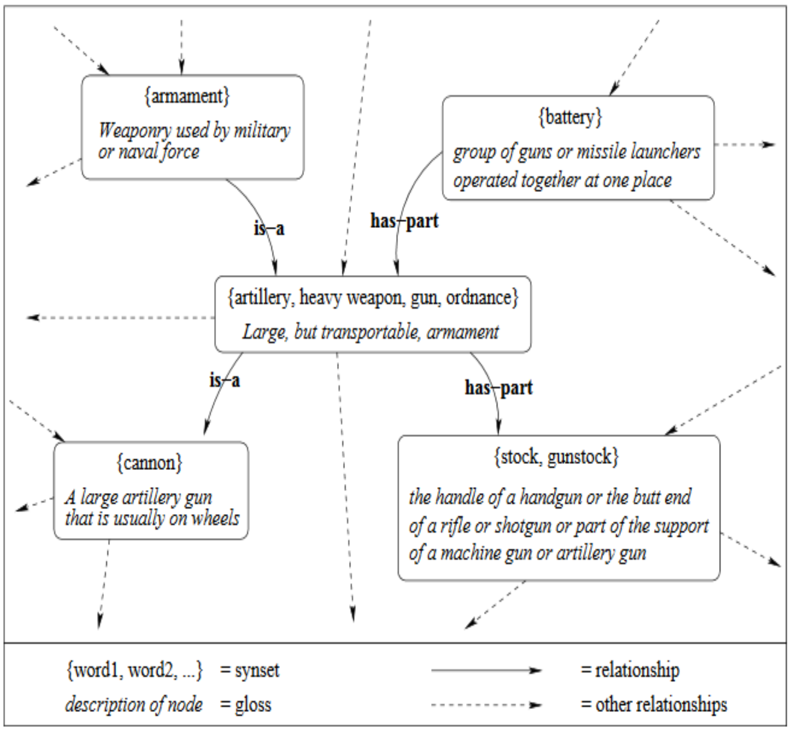

{armament}
Weaponry used by military or naval force
{battery}
group of guns or missile launchers operated together at one place
is–a
has–part
{artillery, heavy weapon, gun, ordnance}
Large, but transportable, armament
is–a
has–part
{cannon}
A large artillery gun that is usually on wheels
{stock, gunstock}
the handle of a handgun or the butt end of a rifle or shotgun or part of the support of a machine gun or artillery gun
{word1, word2, ...} = synset
description of node = gloss
= relationship
= other relationships

Figure 16. WordNet graph semantic network.

WordNet with its 117,000 synsets interlinked based on rich semantic relations offers a real treasure for many knowledge based techniques, which can be divided into two groups: measures of semantic similarity and measures of semantic relatedness. Fig. 17 [25].

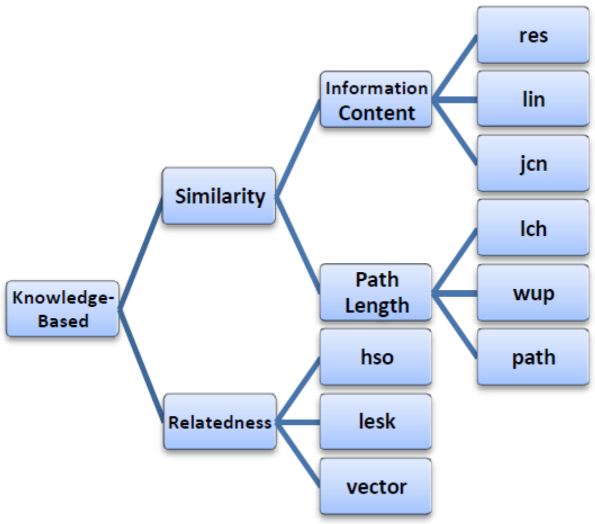


Figure 17. Knowledge based semantic measures.

Semantically similar concepts are deemed to be related on the basis of their likeness. Semantic relatedness, on the other hand, is a more general notion of relatedness, not specifically tied to the shape or form of the concept. In other words, semantic similarity is a kind of relatedness between two words, it covers a broader range of relationships between concepts that includes extra similarity relations such as is-a-kind-of, is-a-specific example-of, is-a-part-of, is-the-opposite-of [25]. The measures of semantic relatedness are :

- **St.Onge (hso):** This measure, reports the relatedness between words and not between word senses or concepts, for example, the words 'red' and 'green' are highly related since both of them are colors, but they aren't similar**.** it is used when a partial understanding of a text is only needed, its main advantage consists in that it requires only a light representation of context, and it measures the similarity by finding lexical chains linking the two word senses based on all the relations defined in WordNet [32]

- **Lesk (lesk):** It is based on the use of the definition of each sense of a word in text(a gloss) , which are compared to all the glosses of each word in the sentence, and then the word is assigned the sense whose gloss shares the largest number of words with the glosses of the other words, while the adapted and generalized version of this measure technique works by finding overlaps in the glosses of the two synsets and the relatedness score is the sum of the squares of the overlap lengths.[33]

- **Vector pairs (vector):** This technique considered ways of augmenting the words in the glosses with data from external sources by using an alternate matching scheme of WordNet concepts that is not as short and not as exact as a WordNet gloss, but describes the concept in a broader sense. It starts by generating gloss vectors by creating a word space, a list of words that would form the dimensions of the vectors. This list of words should contain words that are highly topical, having great potential to discriminate topics. Then, each concept in WordNet is represented by a gloss vector, which is essentially a context vector formed by considering a WordNet gloss as the context. Finally the semantic relatedness of two concepts is simply the cosine of the angle between the corresponding normalized gloss vectors [34].

The second category of the knowledge based techniques, and which is used for the similarity measures gathers six techniques. The first three techniques (Resnik, Lin and Jiang & Conrath) are mainly based on the concept of information content(IC), which is build based on statistical information estimation from large corpora. Information content of a concept measures the specificity or the generality of that concept, i.e. how specific to a topic the concept is. These three techniques are [34, 35]:

- **Resnik measure (res):** It defines the semantic similarity of two concepts as the amount of information they share in common, while the quantity of information common to two concepts is equal to the information content (IC) of the Least Common Subsumer (most informative subsumer), which is a concept in a lexical taxonomy (e.g. WordNet), which has the shortest distance from the two concepts compared. For example, animal and mammal both are the subsumers of cat and dog, but mammal is lower subsumer than animal for them. The upper bound on the value is generally quite large and varies depending upon the size of the corpus used to determine information content values. To be precise, the upper bound should be ln (N) where N is the number of words in the corpus.
- **Lin Measure (lin):** It is based on Resnik's similarity and considers the information content of lowest common subsumer (lcs)of the two compared concepts. The similarity value returned by the lin measure is a number equal to 2 * IC(lcs) / (IC(concept1) + IC(concept2)). Where IC(x) is the information content of x. Which means that the similarity value will be greater than or equal to zero and less than or equal to one. If the information content of any of either concept1 or concept2 is zero, then zero is returned as the similarity score, due to lack of data. Ideally, the information content of a concept would be zero only if that concept were the root node, but when the frequency of a concept is zero, the value of zero is used as the information content because of a lack of better alternatives.

- **The Jiang-Conrath Measure (jcn):** It takes into account the information content of both concepts, along with that of their lowest common subsumer. The measure is a distance measure that specifies the extent of unrelatedness of two concepts. It combines features of simple edge counting with those of information content introduced in the Resnik measure. The similarity value returned by the jcn measure is equal to 1 / jcn_distance, where jcn_distance is equal to IC(concept1) + IC(concept2) - 2 * IC(lcs). There are two special cases that need to be handled carefully when computing similarity; both of these involve the case when jcn_distance is zero. In the first case, we have ic(concept1) = ic(concept2) = ic(lcs) = 0. This would only happen when the concepts are the root node. In the second case, we have ic(concept1) + ic(concept2) = 2 * ic(ics). This is almost always found when concept1 = concept2 = lcs (i.e., the two input concepts are the same)

The remaining other techniques of semantic similarity measure are based on path length measure [34, 35]:

- **Leacock & Chodorow (lch):** It uses an intuitive method to measure the semantic similarity of word senses taking advantage of the tree-like structure of WordNet, by counting up the number of links between the two synsets. In fact, the measure is based on the shortest path that connects the synsets and the maximum depth of the taxonomy in which the synset connection occurs, and it’s equal to -log (length / (2 * D)), where length is the length of the shortest path between the two concepts (using node-counting) and D is the maximum depth of the taxonomy.

- **Wu & Palmer (wup):** it is based on the depth of the two senses in the taxonomy and that of their Least Common Subsumer. The formula is score = 2*depth(lcs) / (depth(s1) + depth(s2)). This means that 0 < score <= 1. The score can never be zero because the depth of the LCS is never zero (the depth of the root of a taxonomy is one). The score is one if the two input concepts are the same.

- **Path (path):** This technique measures the semantic similarity between words by a simple node-counting scheme (path). The similarity score is inversely proportional to the number of nodes along the shortest path between the concepts. It chooses the shortest path that connects the senses in the (hypernym/hypnoym) taxonomy. The shortest possible path occurs when the two concepts are the same, in which case the length is 1. Thus, the maximum similarity value is 1.

4.2 Semantic measures evaluation

In order to choose the best semantic measure that fits the best our application, an evaluation of the accuracy of the different measures is needed. However, the brief study which was made about the different semantic measures allows us to evaluate only a limited set of semantic measures, which is a considerable time saving, given the large number of available techniques. For our application, the semantic measures which must be carried out in order to characterize API **methods deal with semantic similarity rather than semantic relatedness**. The semantic relatedness is defined as: “the strength of the semantic interactions between two elements with no restrictions on the types of semantic links considered “ , while the semantic similarity is defined as : “subset of the notion of semantic relatedness only considering given taxonomic relationships in the evaluation of the semantic interaction between two elements” [61]. For instance, the two concepts Tea and Cup are therefore highly related despite the fact that they are not similar: the concept Tea refers to a Drink and the concept Cup refers to a Vessel. Therefore, the evaluation will only <u>be restricted on the semantic similarity measures of the knowledge based category (Fig 17).</u> <u>Moreover, the corpus based measures are excluded from the evaluation tests, since they are mainly based on the co-occurrence calculation between (frequent) words</u>, for example the word Sugar is more likely to appear with the word Coffee in text, which makes the two words highly related even if they are different, which isn’t at all appropriate to our application.

The best tool to perform <u>natural language processing</u> is Natural Language Tool Kit (NLTK) [37]. It provides easy-to-use open source interfaces to over 50 corpora and lexical resources such as WordNet, **which in our case will be used as a semantic network.** NLTK contains a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP (Natural Language Processing) libraries all written in Python.

The accuracy of the proposed solution is the unique criterion in this work, other criterion like time processing won't be considered. Therefore, the semantic similarity measures will be evaluated only in relation to the level of difficulty to differentiate between similar and dissimilar words. To do so, a function for each of the six semantic similarity measures has been implemented (index 1) using NLTK (Natural Language Toolkit**) and WordNet as semantic network**. Then a set of measures were carried out using a set of similar and dissimilar words, for each semantic measure. It's worth mentioning that the set of similar and dissimilar words were chosen from the different words that we could come across while dealing with APIs, and the reason for that, is simply the fact that these words are more relevant to our application. The similarity or the dissimilarity of these words is defined based on what we consider as a similar or dissimilar API methods, for instance, two API methods which are both used to turn the light on, and where one of these API methods employs the word 'light' while the other employs the word 'bulb', so in this case we consider the word 'light' similar to the word 'bulb'. Afterwards, to evaluate how easy it is to differentiate between these words, the measures were modeled by a normal distribution for similar words, and a normal distribution for dissimilar words, after calculating the mean and the standard deviation for each measure set. Finally, using the Matlab code (index 2), the overlap area between the two normal distributions normalized to the overall area is computed. A large normalized overlap area between the normal distribution of the similar words and the normal distribution of the not similar words corresponds to a semantic measure that differentiates hardly between similar and not similar words, while a small normalized overlap distribution corresponds to a semantic measure that differentiates easily between similar and not similar words. The obtained results are shown in the following figures:

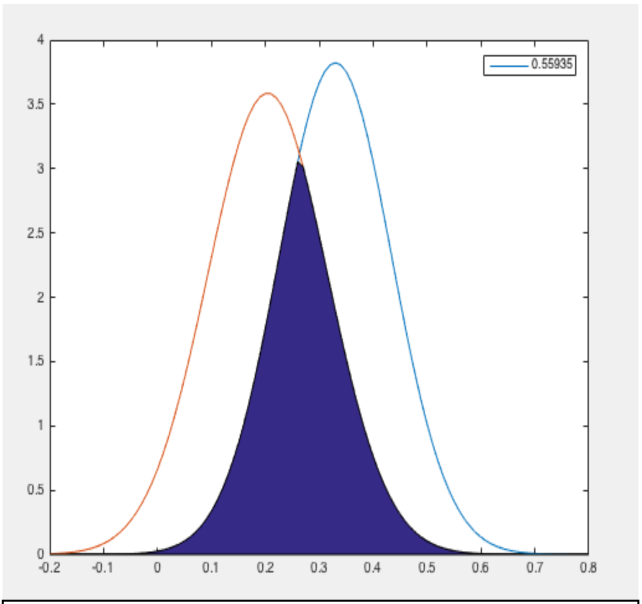


Figure 18. Overlap area : Path length

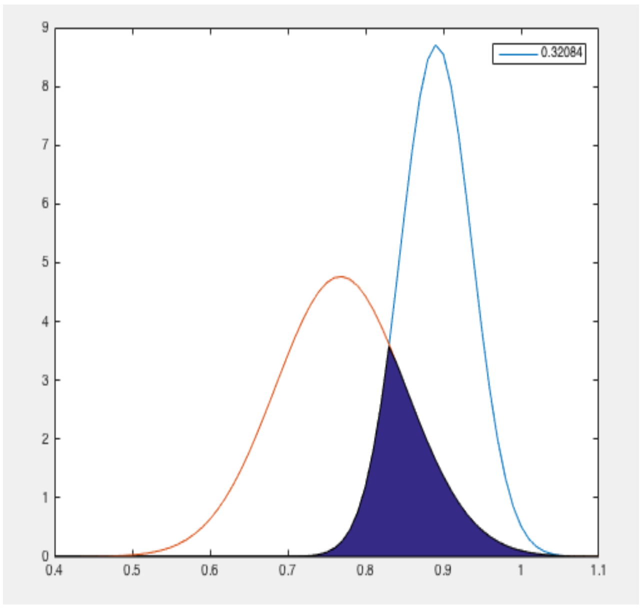


Figure 19. Overlap area: Wu and Palmer

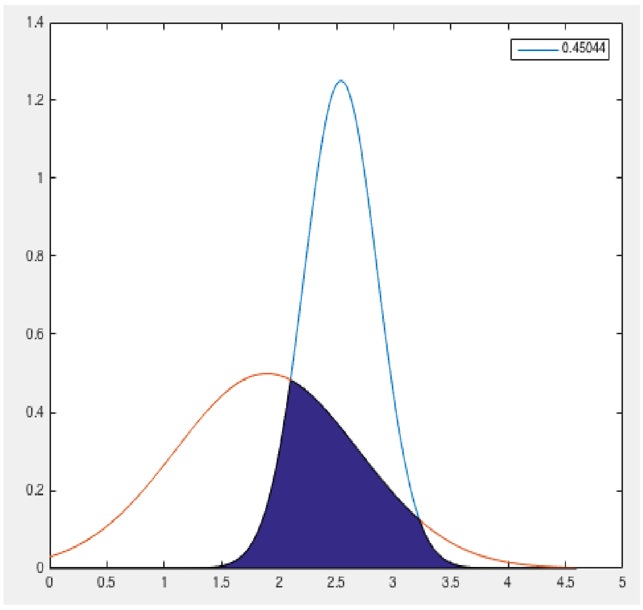


Figure 20. Overlap area: Leacock & Chodorow

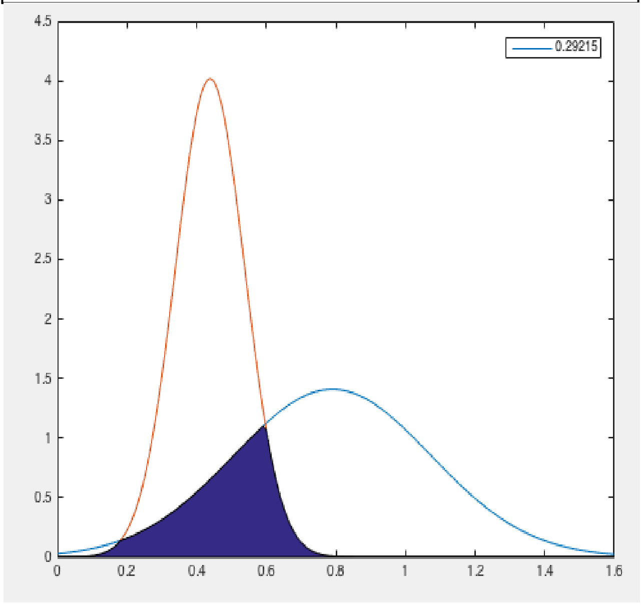


Figure 21. Overlap area : Lin

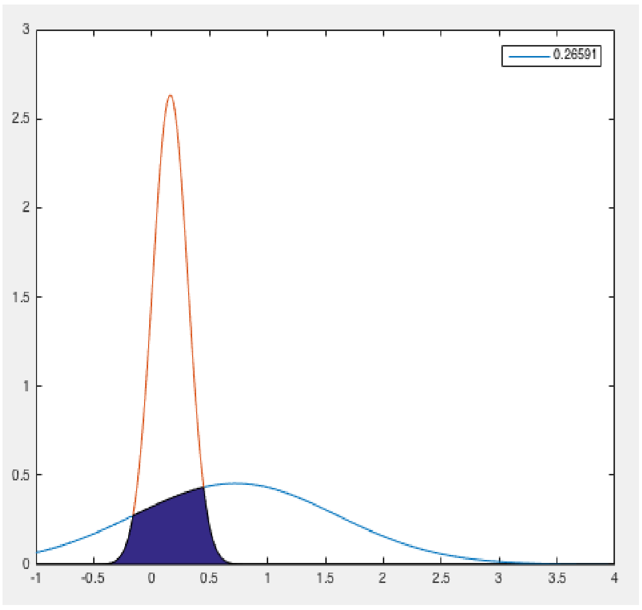


Figure 22. Overlap area : Jiang-Conrath

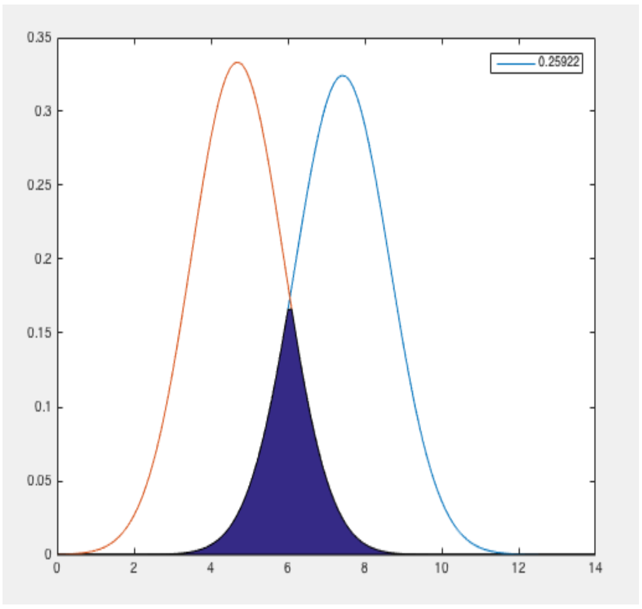


Figure 23. Overlap area : Resnik

Path Length	Wu and Palmer	Leacock & Chodorow	Lin	Jiang-Conrath	Resnik
0.55935	0.32084	0.45044	0.29215	0.26591	0.25922

Table 2. Normalized overlap of areas of the evaluated similarity measures

The table shows the high overlap area of the path based similarity measures, namely Path Length, Wu Palmer and Leacock & Chodorow which is synonymous to a poor differentiation between similar and dissimilar words, in the other hand information content based similarity measures, namely Lin, Jiang-Conrath and Resnik outperform clearly the other based methods.

4.3 Conclusion

The results shown on the previous figures and summarized by table 2, allow us to confirm on the one hand the better performance of Information Content measures (Lin, Jiang & Conrath and Resnik) compared to path semantic measures (Path, Wu & Palmer and Leacock & Chodorow) and on the other hand, it allows us also to confirm that the Resnik measure is one of the most accurate semantic measures when figuring out semantic similarity. Furthermore, using such a method based on the representations of the computed values as two normal distributions of similar and not similar sets and then evaluating the semantic measures based on the normalized overlap area between the two normal distributions is believed to be a new technique to evaluate the semantic measures (at least in all the referenced research papers in this document). Usually, corpora of words or phrases are used for evaluation. For instance, in [38] the Microsoft paraphrase corpus consisting of 4076 training and 1725 test pairs is used to evaluate these semantic measures based on their accuracy on the testing set., they also conclude that the Resnik measure is one of the most accurate semantic measures in detecting paraphrases, which gives more credibility to our results and evaluation technique. Hence, the Resnik method will be used asthe chosen method for semantics similarity measures in our building block.

CHAPTER 5

Hybrid Similarity

Chapter 5 Hybrid similarity

In the third chapter, all the similarity measures between different words that may be dealt with, in this work have been listed: namely the semantic measures and the syntactic measures, adding a new third similarity measure type. This chapter deals with this third type of similarity measures, which is the most difficult similarity measure among the three types. Indeed, this third type relates to all similarity measures that do not belong neither to the semantic nor to the syntactic measures. It concerns mainly words which have no semantic or syntactic similarity such us "smart-light" and "lamp", the word "smart-light" has no semantic meaning, since it doesn't belong to any dictionary, but also the two words has no syntactic similarity such as the one that exists for instance between "smart-light" and "light". To our knowledge, no work have been reported to deal with such an issue since the similarity measures in natural language processing generally are used for datasets that doesn't contain such cases, which let us think about building our own solution. It should be noted that what's called as a hybrid similarity measure in this chapter is completely different from what is found in literature about hybrid techniques. Existing hybrid similarity measures usually combine two different semantic measures. For instance in [39], a knowledge-based is combined with a corpus-based semantic measure in order to create a hybrid similarity measure with a better accuracy.

5.1 Proposed Idea

The proposed idea to detect the similarity between two words which have no semantic neither syntactic similarity, (e.g. "smart-light" and "lamp"), is to use a hybrid similarity measure by combining a semantic with a syntactic technique, illustrated here.

An example of a hybrid measure of similarity between the word "light" and "smart-lamp" is illustrated in the following to easily explain how the hybrid similarity measure works. In the first step, the semantic similarity measure is used along with WordNet to extract all the similar words of the word "light" (among which there is certainly the word "lamp") then a syntactic measure is used between the word "smart-lamp" and all the similar words of the word "light", finally only the best similarity measure is taking into consideration to decide on the similarity between "smart-lamp" and "light" (Fig.24) .

The semantic measure will be based on the Resnik technique, since the results of the last chapter support this decision. A similarity measure will be performed on the WordNet word corpus by choosing a simple threshold value for the Resnik similarity measure, based on which we **extract all the similar words of the word "light",** this value was figures out after several tests, during which we observed the words at the output while tweaking the threshold value.

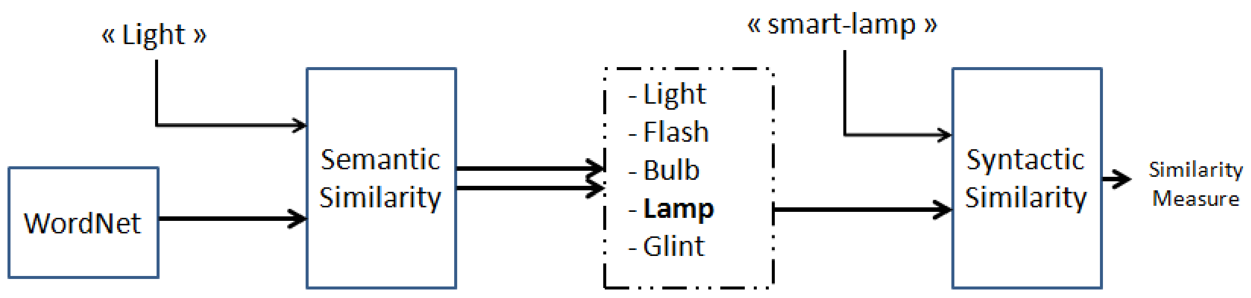


Figure 24. Hybrid similarity block

5.2 Syntactic similarity measures

There are several techniques of syntactic similarity measures which are, for instance, widely used in text typing prediction or in automatic orthography correction. They are based on the calculation of a string metric to measure the distance between two words based on their character composition, in the following the most used character based techniques[35]:

- **Longest Common SubString**[45] **:** this algorithm considers the similarity between two strings is based on the length of the longest subsequence common to the two strings.
- **Jaro-Winler**[46]**:** is based on the number and order of the common characters between two strings, while using also a prefix scale to give better ratings to the words that match from the beginning. It takes into account typical spelling deviations.
- **N-gram**[35] : is a sub-sequence of n items from a given sequence of text. It compares the n-grams from each character or word in two strings and then the distance is computed by dividing the number of similar n-grams by maximal number of n-grams.
- **Dice's coefficient**[35]: is defined as twice the number of common terms in the compared strings divided by the total number of terms in both strings.

- **Overlap coefficient**[35]: is defined as the size of the intersection divided by the smaller of the size of the two sets. It is similar to the Dice's coefficient, but considers two strings a full match if one is a subset of the other, for instance, the words ‘sleep’ and ‘sleepless’ are considered as a full match because the first one is a subset of the second one.

- **Levenshtein distance**[47]**:** it is the most used string based measure, it considers the distance between two words as the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

For the choice of the syntactic approach to be used, no performance test will be done to compare the previsouly listed syntactic approches, since it is not a very complicated problem, an intuitive approach like the Levenshtein distance will be sufficient for this application. However, a modification has been made to the Levenshtein distance by normalising the distance based on the length of the longest word. <u>It’s to be noted that for the Levenshtein similarity measure, the less important the symilarity measure is, the more similar the words are.</u>

Mathematically, the Levenshtein distance between two strings A,B of length |A| and |B| respectively, is given by lev(|A|,|B|) where :

Lev(i,j)= max(i,j)   if(min(i,j)=0);

Otherwise :

Lev(i,j)= min (Lev(i-1,j)+1, Lev(i,j-1)+1, Lev(i-1,j-1)+ 1(Ai≠Bi))

Where 1(Ai≠Bi) is equal to 0 when i=j and 1 otherwise.

5.3 Evaluation of the hybrid approach : maximum syntactic similarity

The algorithm (index 3) as explained and depicted in fig.24 has given the following results while measuring the **maximum** similarity between complex (strings) and dissimilar words:

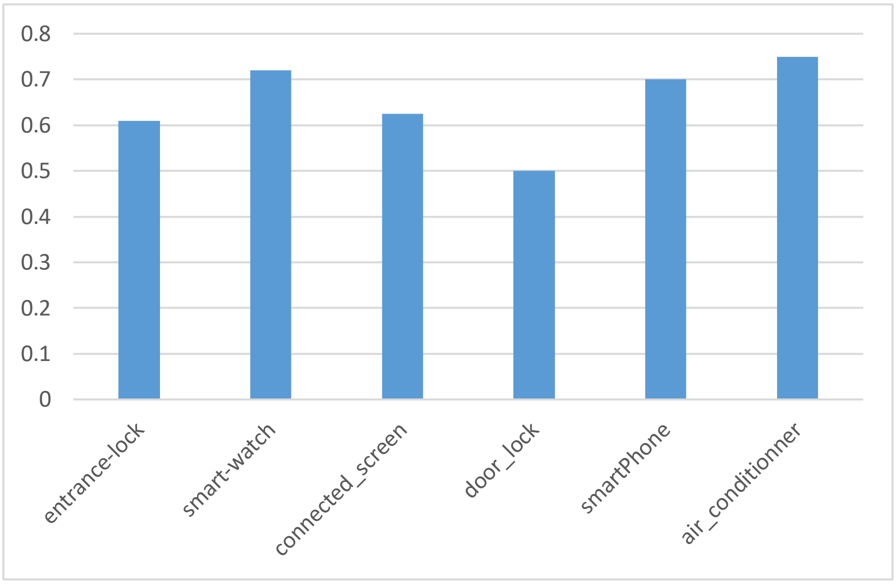


Figure 25.a Similarity measures of dissimilar words while using the word "light" as reference word

The similar and dissimilar words were chosen based on the similarity or the dissimilarity of publicly available API methods, we proceeded in this way because these words are more relevant to our case. It's worth reminding also that for the syntactic similarity measure (a "distance") that we use, the more similar the words are, the less the distance value of the similarity measure is.

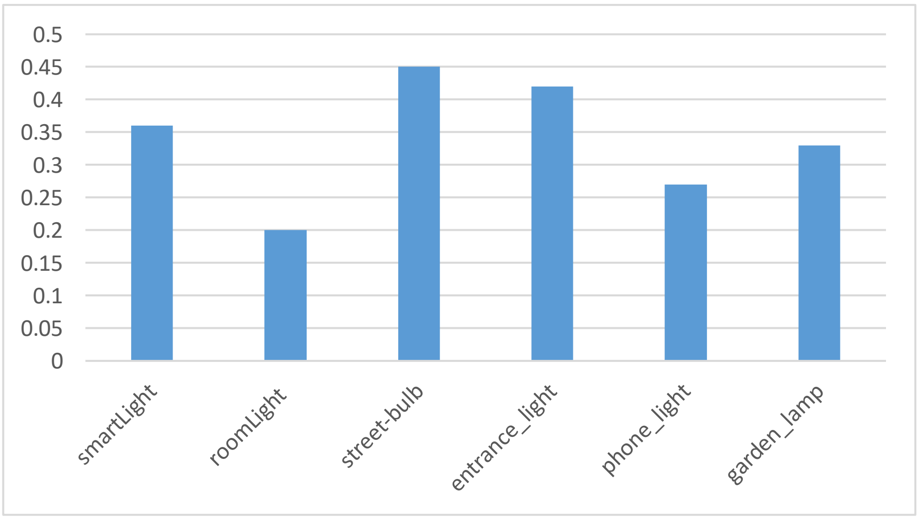


Figure 25.b. Similarity measures of similar words while using the word "light" as reference word

An apparent difference in magnitude between the similarity values of similar words and the similarity values of dissimilar words has been obtained. For instance, in figure 25.b, we obtain a set of similarity values between similar words and in which the value varies between 0.2 and 0.45, while in figure 25.a the similarity values of dissimilar words varies between 0.5 and 0.75. However, unsatisfactory results are obtained for further similarity measures.

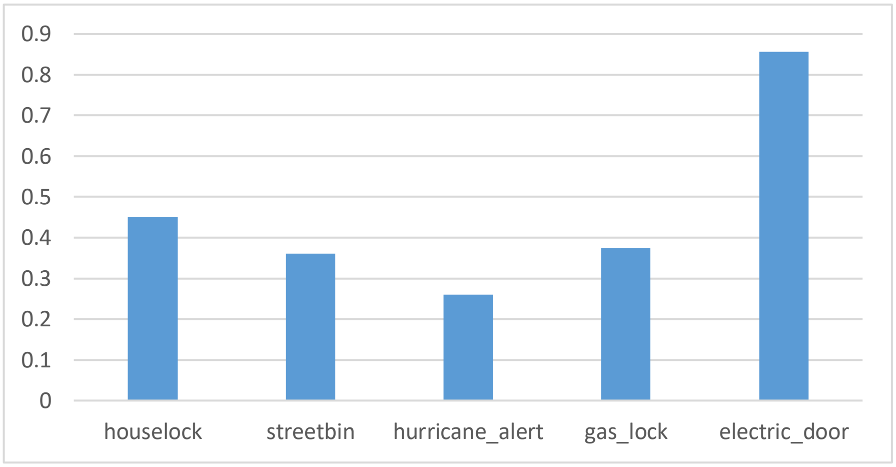


Figure 26. Similarity measures of disimilar words using the word “light” as reference word

As illustrated by the figure 26, the similarity values of dissimilar words are close to the similarity values of similar words shown in fig 25.b.

Discussion:

After analysis, these results are due to the fact that WordNet also contains compound words, such as “ house_lamp”,”street_light” or “gas_light”. These words belong to the list of similar words of the word “lamp”, extracted from the WordNet corpus by the Resnik techque (Fig.27). For instance for the synonym word “street_light”, the first part “street” distorts the similarity measures, since the focus is on the word “light” and not the word “street”, therefore when performing a syntactic similarity measure, for instance between the word “lamp” and “streetbin” a value indicating a strong similarity is obtained, since there is a strong syntactic similarity between “streetlight” and “streetbin”.

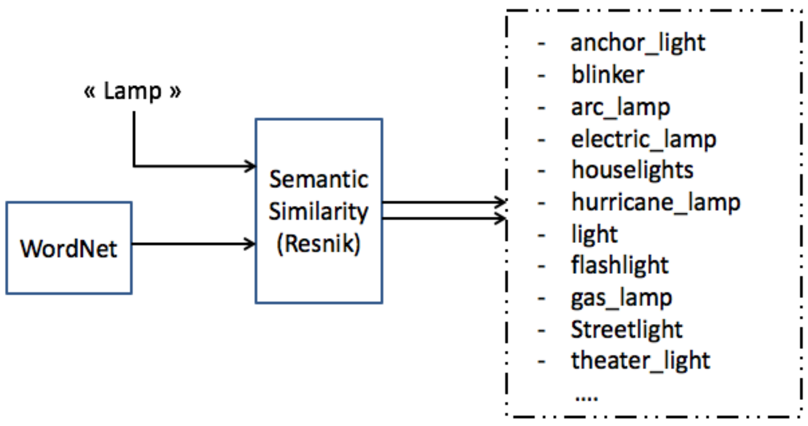


Figure 27. Compound words contained in WordNet

5.4 Algorithm optimization: mean syntactic similarity

To overcome this issue, the idea is to take into account the syntactic similarity values of all the extracted words in the first step by calculating the mean of all the syntactic values. For instance, to figure out that the word "lamp" is similar to "street_light" and dissimilar to the word "streetbin", the idea is to extract all the similar **words to the word "lamp**" from WordNet using the Resnik similarity measure, these words will include among others the words mentioned in the fig. 27, then adding to the best syntactic similarity value we calculate the mean of all the syntactic measures between the word "street_light" and the extracted similar words of the word "lamp" (the word 'light' is likely to appear several times in the synonyms of the word 'lamp') , then we do the same with the word "streetbin". The mean of the syntactic measures is really helpful to distinguish similar and dissimilar words (other filtering strategies may be possible as well). In other words it helps to distinguish between informative and not informative words, and it's based on the fact that the word "light" is more likely to occur among the similar words of the word "lamp" than the word "street", for instance in the example of the fig.27 the informative word "light" occurs 6 times more than the no informative word " street", which helps therefore to figured out easily that the word "lamp" is similar to the word "street_light" and dissimilar to the word "streetbin". Figure 28 shows the syntactic similarity measure values between similar words and the corresponding mean of each measure, while figure 29 shows the same values for dissimilar words.

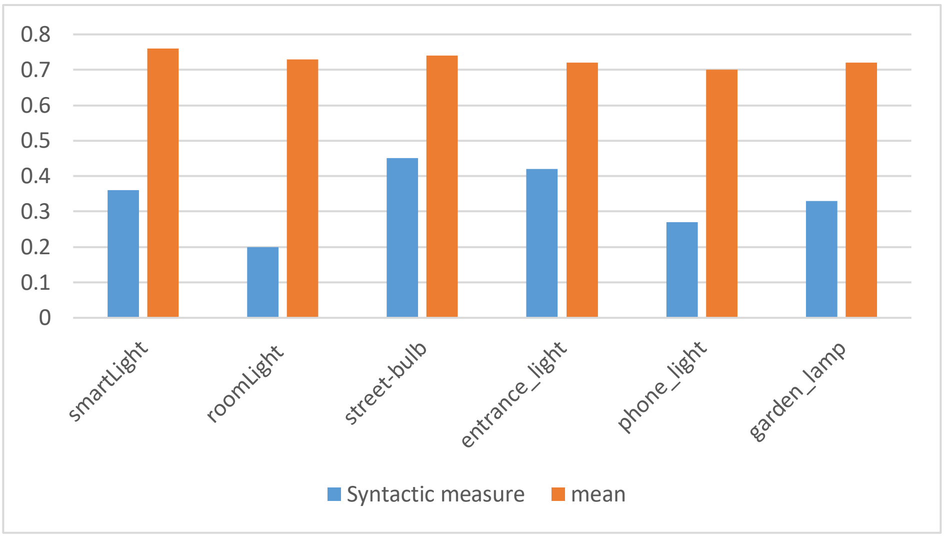


Figure 28. Syntactic measures and their corresponding mean (similar words)

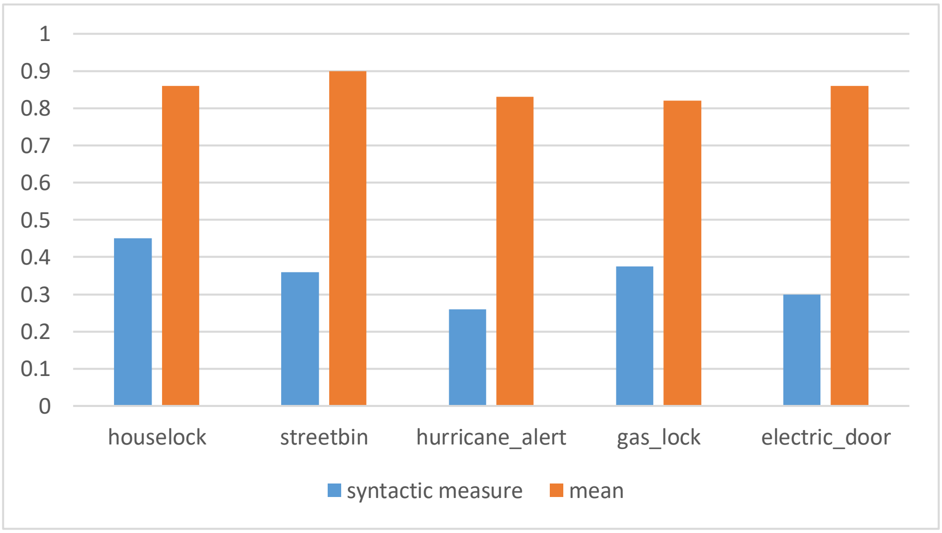


Figure 29. Syntactic measures and their corresponding mean (dissimilar words)

From figure 28 and figure 29, we confirm that we can’t distinguish between similar and dissimilar words by only using the best syntactic value, while by adding the mean value as a second indicator the differentiation between similar and dissimilar words is more unambiguous, in figure 28 the mean value of the syntactic measures of similar words varies between 0.7 and 0.75, while the mean value of the syntactic measures of dissimilar words in fig.55 varies between 0.82 and 0.9. Therefore, in this chapter, an Hybrid similarity measure was designed to detect the similarity between words that don’t share any syntactic or semantic similarity, in the last two chapter we only focused on generating similarity values that are quite separated for similar and dissimilar words, in the next chapter we are going to use the best evaluated techniques to classify API methods.

CHAPTER 6

Classification

Chapter 6 Classification

The main goal of the proposed solution consists of being able to recognize the associated action to each API method by processing them by using natural language techniques and machine learning, as depicted in figure 30.

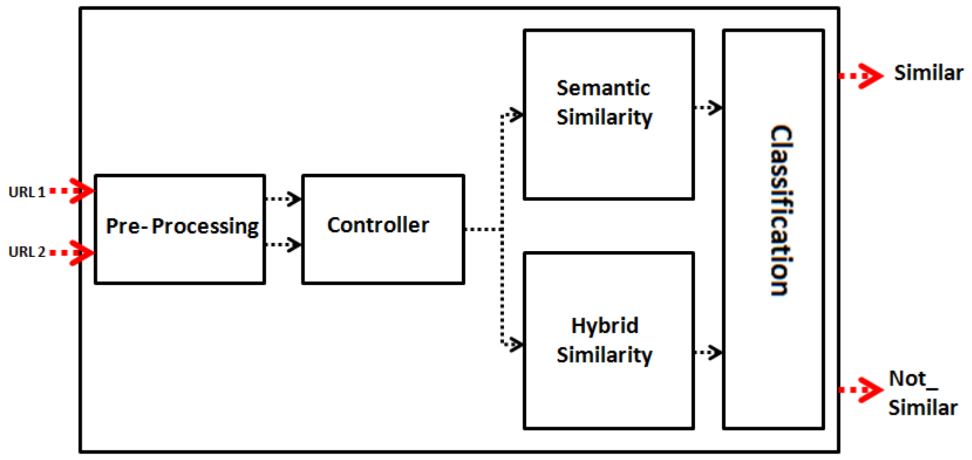


Figure 30. Building blocks

In the last chapters**, a semantic similarity block** has been developed to perform a similarity measure between the words that have a semantic meaning, the choice of the semantic similarity technique to be used was done after having tested several semantic similarity measures. Another block was developed to measure the similarity between words which may not have a semantic meaning based on a Hybrid similarity technique which was developed based on a semantic and a syntactic similarity measure.

So far, the building blocks depicted on Fig.31 were developed in a such a way to provide similarity values which can make the differentiation between similar and dissimilar cases as unambiguous as possible. The next step is to work on the classification block which based on the similarity values provided by the **SEMANTIC** and the hybrid similarity blocks, should be able to distinguish between similar and dissimilar API methods in a classification objective.

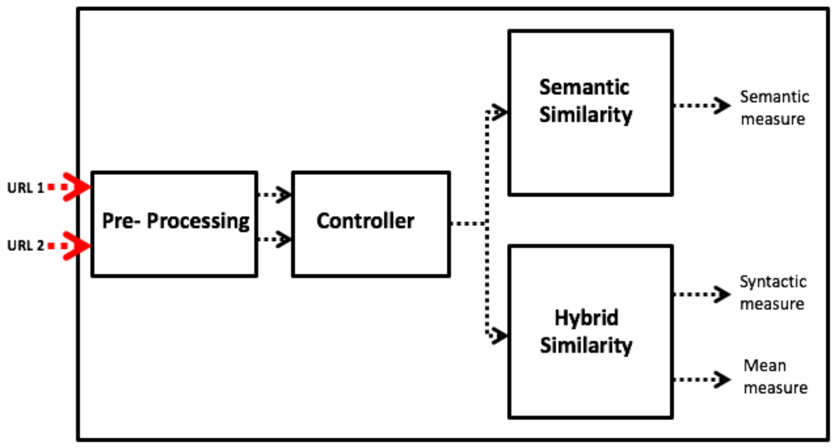


Figure 31. Already designed building blocks

6.1 Machine learning

Machine learning algorithms (MLAs) can be grouped in many ways, such as by following the algorithm learning style. This grouping type will be used in our case to give a general view of the MLAs. There are three different learning styles in machine learning algorithms:

6.1.1 Supervised learning

In this kind of learning style, a model is computed thanks to a training process that uses an input data and its corresponding output. During this process the algorithm is expected to make predictions about the output according to a given input data and corrects its predictions according to the correct output until it achieves an acceptable level of performance. Usually the supervised learning algorithms are used for two 'types' of problems:

A) Classification: the main goal in classification problems is to assign a label or a categorical variable to each dataset, as examples of classification problems: recognize email spams ("spam" or "not a spam"), predicting diseases ("disease" or " not a disease"), predicting the correctness of the responses ("true" or "false"). The most used classification algorithms are [40] :

- Naïve Bayes : it's based on the Bayes Theorem, which means that this classification algorithm assumes that the features or the variables are independent, that's from where the Naïve word comes from. Basically, this technique aims to maximize the likelihood using the naïve Bayes rule by calculating the number of times a certain feature appears within a certain class or label, as well as the number of times this label or class appears in the data, these numbers are then converted to probabilities and plugged-in to the Naïve Bayes formula, to finally calculate the probability to get a certain label or class given a certain input. We do the same for each class or label, then at the end we choose the class with the highest probability. It can be used in binary classification and multi-class classification problems. The logic is the same for continuous inputs, the only difference is that we have to use a distribution to represent the data. That being said, for each feature we compute it’s distribution for being in both classes, then we use these distribution along with the Naïve Bayes formula to predict to which class the input belongs.

- Decision Tree: in this algorithm, a tree model is created in order to classify different datasets, in which the leaves contain the class labels and branches contains the values of the features that lead to each class label. Basically, it predicts the label of an instance by travelling from the root node to a leaf. The general framework for growing a decision tree is to start with a single leaf and assign a label to it according to the majority of the labels over the training set, then we examine the effect of splitting based on each feature based on a gain measure that we define (e.g. train error, information gain, gini index, etc.), and the feature that provides the highest gain is chosen to split the data of this node. The previous iteration is then repeatedly done on all the nodes, until we use all the available the features to split the data, or we reach a max depth defined as hyper-parameter. The decision trees have the advantage of being simple to understand and interpret, adding to the fact of being able to handle both numerical and categorical data. One of the hyper-parameters that we can tweak for decision trees is the min-leaf size, which represents the minimum number of samples perf leaf.

- K-Nearest Neighbors: It works by simply using the k-nearest neighbors, which are defined as the k most similar training datasets, the class assigned to the dataset is computed by a majority vote of its neighbors. This algorithm only assumes that the distance between data instances is useful in making predictions. The common values for k are 3, 7 and 11, this value may get larger for large size datasets. A simple example to explain the approach of this algorithm, let´s suppose that we try to classify a certain point in space is belonging whether to the red or the green color using the K-nearest neighbors approach, for k=3 we have to find the 3 nearest neighbors to this point, and then we simply assign this point to the class that has the biggest number of nearest neighbors, for instance if we found out that 2 of the 3 nearest neighbors are red, then the point is classified as belonging to the red class. The hyper-parameters that can be tuned to improve the performance for this technique are the k, which represents the number of neighbors to take into consideration for each classification, and the distance metric used to define the nearest k neighbors.

- Support Vector Machine (SVM): It is mainly used for binary classification problems which calculates a line that best separates the data into groups, which is done using an optimization process on the training dataset. More precisely the optimization process aims to find weights such that, if multiplied by a certain input it gives either a positive or a negative number, getting either ones should correspond to the class of the input. A margin is added to make the constrain easier by allowing some data to be misclassified, since in real problems a line cannot be drawn to neatly separate the data into two groups. A non-linear classification is widely used generally by using different kernels to map the data into high dimensional spaces. Thus, among the hyper-parameters that we can tweak to optimize the SVM is the box-constraint which specifies how hard or soft our margin is. Adding to that, we can use also the kernel scale, which is a constant that changes the behavior of the kernel function.

- Discriminant Analysis: This method is based on a relatively simple approach based on the statistical features of the data. The math in DA is quite complex, so we won't dive into it but we are going to give the intuition behind it: It simply assumes that the data is Gaussian, then it calculates the variance and the mean of each class, finally it tries to maximize a cost function defined by the difference of the mean of each class, divided by their variance, which means in other words that DA tries to find the line that ensures the maximum data separability. Regularization is the process of finding a small set of predictors that yield an effective predictive model and it can be used for DA, where two variables delta and gamma can be used to control regularization, these two parameters are the the hyper-parameters for DA.

B) Regression : The main difference between the classification and regression problems is that the output variables in regression problems are numbers rather than categories. There are some of the classification techniques that could be used also in regression problems, like the decision trees and the SVM, however the most used regression techniques won't be mentioned in this work since we are dealing with a classification problem rather than a regression one.

6.1.2 Unsupervised learning

In this kind of learning, a model is computed to deduce structures in the input data without going through a training phase, and the main difference with the supervised learning is that the input data in the unsupervised kind is unlabeled. It's mostly used in clustering problems where the data is divided in different unlabeled groups based on the computed model.

6.1.3 Semi-supervised learning

As indicated by its name, this method is a mix of supervised and unsupervised learning in the sense that the data is a mixture of labeled and unlabeled sets. In this kind of learning, the model learns the structure of the data and in the same time makes predictions.

The problem to deal with in this project is much more about a classification task for which algorithms work aim at figuring out l the API Methods (APIM) associated with a specific action, by grouping similar ones. For instance, starting from a big set of APIM, the algorithm picks up a random one [an action should be assigned to this method] which will be compared to all the other APIM by classifying them into two groups 'similar' and 'dissimilar' ones, at the end the algorithm will have created different groups of APIM in each of them there are only similar APIM.

6.2 Classification algorithms evaluation

In this last part, an evaluation of the mostly used classification algorithms will be performed in order to choose the algorithm that gives the most accurate results, so the evaluation will be done only on the basis of accuracy, without taking into account the other aspects like the prediction speed. In machine learning , there is basically no ideal algorithm outperforming the other ones for all applications. It really depends on the data and the context, and only an evaluation of these algorithm based on the specific application data t would let us know which one performs better.

It has to be mentioned that only the URLs will be taking into consideration for the evaluation, in other words, the API method will be reduced to the URL part only. The reason for such a decision is for the sake of simplicity, because the URL is the most difficult part a natural language processing has to deal with, the other parts like the method (post, get, put, delete) or the body parameters are more straightforward when applying classical natural language processing techniques.

The URLs were generated using a python function (index 4), in which the URLs were created using all the possible combinations of words from three different sets of words (we chose the words based on the different API methods that we came through): the first set is composed of similar nouns like ‘street_lamp’, ‘house_light’ and ‘bulb’, the second set is composed of words that would give a more particular sense to the URL like ‘state’, ‘set-state’ and ‘all’, and the third set of words is composed of words with no valuable information about the associated action to the URL, and which represents in this case a noise, words like ‘id’, ‘v1’ and ‘link’ were used in this set of words. The same idea was used to generate dissimilar URLS yielding to in total, 2380 similar and dissimilar URLs. An simple example to illustrate how the URLs were generated is explained in the following, while taking the used tables (table1, table2 and table3) from index 4 to generate dissimilar URLs (or API methods in general) :

table1=['lock','desktop-computer','house-lock','smart_lock','watch','phone','connected-lock','connected_watch','room_lock','room-conditioner','security-lock','emergency-alarm','television1','screen2','speaker3']

table2=['states','all-states','screen2-state','watch-state','status','<id>','user','all']

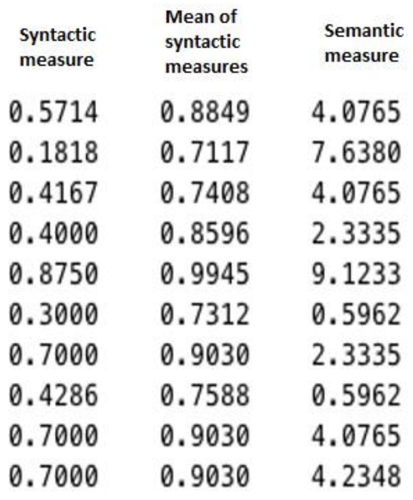
table3=['api','v1','v3','link','REST','1']

Based on these tables, a first URL may be composed of (remember a URL is composed of only one word from each table) : the words “lock” from table 1, “states” from table 2 and “api”from table 3. Another URL would be composed of: “lock”,” all-states” and “api” or “lock”, “all” and “v1”, and so on, until we go through all the possible cases, whose number equals the multiplication of the tables dimensions.

As a reminder the generated URLs are only used for the training and testing steps, in order to validate our method. In a real use case the algorithm is going just to take as input a set of API methods and outputs different groups of API methods, each of which corresponds to a certain action (turn on the light for instance), and this is done based on similar and dissimilar classification.

The generated URLs were used in the main program (index 4) to calculate the similarity values, namely the semantic similarity, the syntactic similarity and the mean of the syntactic similarity measures. Remember that the python program in index 4 only generates the similarity measures, then the classification is done using machine learning within Matlab. In the following a detailed explanation of how the similarity values are measured using the semantic and hybrid methods explained in the previous chapters (for a better understanding make sure to read the program in index 4):

The program takes as input a set of URLs and should output a similarity matrix like the one in the figure 31, which actually abstracts the similarities existing between the URL of reference and all the other URLs fed in as input. The matrix shown in Fig. 31 is just an example, in the following we are going to present two different approaches, where the similarity matrix in both cases has a different form.



Syntactic measure	Mean of syntactic measures	Semantic measure
0.5714	0.8849	4.0765
0.1818	0.7117	7.6380
0.4167	0.7408	4.0765
0.4000	0.8596	2.3335
0.8750	0.9945	9.1233
0.3000	0.7312	0.5962
0.7000	0.9030	2.3335
0.4286	0.7588	0.5962
0.7000	0.9030	4.0765
0.7000	0.9030	4.2348

Figure 31. Matrix of similarity measures

It has to be mentioned that for the program in index 4 no URLs are fed in as input, rather they are generated inside the program itself as already explained above. As a first step, the program chooses the URL of reference by choosing a URL which contains as much as possible of semantic words(words that can be found in a dictionary , the reason for that, is that it has been observed that the semantic measure seem to be more accurate than the syntactic measure, which makes sense, in other words detecting the similarity between the words “light” and bulb is easier than detecting it for “smartlight” and “bulb”, adding to that, it’s hard to measure the similarity between two words neither of which belongs to a dictionary, while not having any syntactic similarity, as example of such example is the words “smartbulb” and “roomlight” (they don’t belong to any dictionary and the don’t have any syntactic similarity, while being similar though). That’s why the URL of reference should contain as much as possible words that have a semantic meaning, if for example the three words of the reference URL have a semantic meaning, it means that the program will never face the case where two words with no semantic neither syntactic similarity are compared. As a reminder, it has been mentioned previously that the hybrid similarity technique is used when there is no semantic neither a syntactic similarity between two words (for example “light” and “smartbulb”), but keep in mind that it works only if one of the two words has a semantic meaning (in the last example, “ light” has a semantic meaning). Therefore, the only assumption that we are making here is that: **to be able to extract all the similar API methods for a certain action, we assume that there is at least one API method of them, that is composed only of words that have a**

semantic meaning, this assumption is not a deal breaker though, based on the structure of the API methods observed in the literature.

After choosing the URL of reference, the program goes through all the other URLs and generates the similarity measures. To illustrate this more easily let's take the example below:

Reference URL : /light/state/api

URL 2 : /bulb/all-states/v1

The program measures the similarity between the different URLs by taking and considering all the possible tuples of words between the two URLs, for example the word "light" from the reference URL should be compared to all the words from URL2, and not only to the word "bulb", this may look senseless at first sight, but if you think of it, it totally makes sense, actually all the possible tuples are considered because simply we don't have any prior knowledge about the structure of the URLs, if you take as example the word "state" in the reference URL and ask the following question : should it be compared to the word "bulb" in URl2 (through a semantic measure) ? or should it be compared to the word " all-states" (through a hybrid measure)? The answer is : we don't know and we can't know unless we start making assumptions (we could have assumed for instance that the words should be compared in respect of their position in the URL, for instance the second word from the reference URL "state" with the second word from URL2 "all-state") but we did not want to make assumptions that may turn out unrealistic, therefore we ended up choosing two approaches:

a- Maximum similarity approach:

In this approach we measure the similarity between all the possible tuples and at the end, we only consider the values that provide the maximum similarity. In other words, after comparing all the tuples if we end up, having many values for the semantic measure, we only take the best one, it can be considered as " a similarity measure by maximization". To illustrate this, let's take the example of the URLs above: it starts by taking the word "light" from the URL of reference and the word "bulb" from URL2. The next step is to figure out which similarity measure should be applied, this is done by checking whether the words from URL 2 belong to the WORDNET corpus (have a semantic meaning), if both of the words belong to the corpus (which is the case in this example) then a semantic similarity measure is performed as explained in the last chapters using Resnik technique. The obtained semantic measure between the last two words is stored as we move to the next word in URL2 by measuring the similarity between the word "light" and "all-states", which in this case is going to be performed through a hybrid similarity measure since the word " all-states" doesn't have a semantic meaning, afterward the measure values are stored as we move to the next comparison between the word "light" and "v1" (which represents the noise), after performing a hybrid similarity between the two words, the program compares between the obtained measure values and the previously obtained value in the already performed hybrid measures and it keeps only the best measure. The next words to be compared are “state” with “ bulb” then “state” with “all-states” and so on. After comparing all the possible tuples, the program in this example should keep for the semantic measure the values obtained by comparing “light” and “bulb” (because it’s supposed to be the two tuples with the highest semantic similarity), and it’s supposed to keep the values obtained by comparing “state” with “all-states” for the hybrid similarity (because they are supposed to yield the best hybrid similarity values). At end, the result of a such comparison is three values, one represent the sematic similarity measure and the two others represent the values of the hybrid similarity measure, these values are what we find in each row of the similarity matrix.

b- All similarity measures approach:

This approach is pretty much similar to the last approach except the fact that, rather than taking only the maximum similarity values, we are going to keep all the similarity measures. As explained previously, each URL in our example is composed of 3 words (not necessarily words with a semantic meaning), thus each word in the first URL should be compared to all the words in the second URL. As results, we end-up having 9 similarity measures, these 9 measures can be all semantic measures, or all hybrid measures or something in between, since the kind of similarity measure to use depends on the word nature. As result, to be able to consider all the cases, we have to consider the boundary cases: if all the measures are semantic measures, than in this case we have 9 values that sums up the similarity between URL1 and URL2, but if all the 9 measures are hybrid, then in this case 18 values (recall each hybrid measure outputs two values: 2*9=18) are used to abstract the similarity measure between both URLs. Thus overall, we have 27 features (18 for hybrid and 9 for semantic), these values are initially set to default values. Note that for certain similarity measures, we may end up having neither a semantic similarity measure or hybrid similarity measure, in other words some values among the 27 features have always the default values, so as result these values are not informative and prevent makes the training of some of the machine learning techniques harder because the variance of certain features in regard of the labels is zero, thus we got rid of the features whose variance is zero all over the data.

6.2.1 Results

The generated data from the python program that measures the similarity between different URLs is then divided into training dataset and testing dataset in a Matlab code (index 5) to evaluate the different algorithms of machine learning according to their accuracy : **2/3 for training and 1/3 for testing.**

6.2.1.1 Maximum similarity approach:

Fig. 32 depicts the classification accuracy percentage of each machine learning **algorithm for a testing dataset size of 650**, that contains similar and dissimilar URLs:

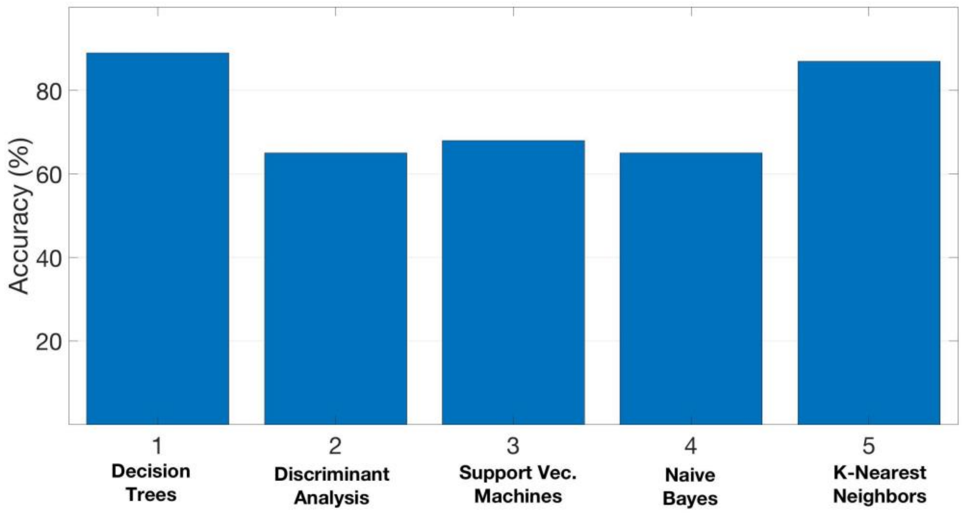


Figure 32. Classification accuracy percentage

The graph bars show that the Discriminant Analysis and the Naïve Bayes algorithm are the ones that perform the worst with an accuracy percentage of 65%, while the Support Vector Machine performs slightly better with an accuracy percentage of 68%.

In the other hand, the Decision Trees and Nearest Neighbor gave a decent result and significantly better than the other algorithms with an accuracy percentage of 89% and 87% respectively.

Recent published works in machine learning have shown that a good tuning of the hyper-parameters may lead to a significant increase in performance, in some cases a good tuning of the parameters can even make the regular ML techniques outperform the state of the art techniques. Thus, in order to optimize the hyper-parameters of aforementioned techniques to get as much accuracy as possible, we had to explore the different used techniques for such procedure. In the following a brief explanation of the most used techniques for hyper-parameters optimization:

- Grid search: it's usually the most used approach, which consists in exhaustively searching through all the hyper-parameters space. For instance, if for a certain ML technique there is two hyper-parameters, then the Grid search in this case consists in going through all the possible pairs of hyper-parameters. At the end, we pick up the pair that provides the minimum loss over the searched hyper-parameters space, such space is usually discretized to break it down to a finite space. This approach is highly greedy in computation.
- Random search: This approach is pretty similar to the previous approach, except the fact that the parameters chosen at each run are randomly selected. This method usually outperforms the Grid search optimization.
- Bayesian optimization: this approach is considered according to the literature as the state of the art when it comes to hyper-parameters tuning. Its concept is based on Gaussian Processes (GPs), which is kind of similar to the regression problems, where given a certain function, we estimate the best function parameters that let this function best fit the data, however the difference here, is that rather than building a model for the parameters, we build a model of a function, where this function is then used to give an estimation of the performance (objective function) given a certain hyper-parameters. In other words, it is a non-parametric approach, in that it finds a distribution over the possible functions $f(x)$ that are consistent with the observed data. This approach, outperforms both the Grid and Random search approaches.

After several readings about these three approaches we ended-up choosing the Bayesian optimization approach, since it's seemed to be the most efficient and less time consuming approach. Actually, Matlab includes the necessary function to perform a Bayesian Optimization, which is actually called through the following parameter 'OptimizeHyperparameters' inside the fitting functions used to train the models previously. It has also to be mentioned that it uses k-cross validation to optimize the parameters, where the original data is randomly partitioned into k subsamples, where one of these subsamples is used for the validation and the remaining k-1 subsamples are used for training. Afterwards this process is repeated k times, and the returned loss (or objective function) is computed by taking the mean of the loss over the k folds. Each of the next figures show the objective function variation in function of the hyper-parameters that we optimized over. Actually, each ML technique contains usually lot of hyper-parameters, and it's not convenient to explore all the hyper-parameter space, so in the following results we chose to use 'auto' parameter while calling the fitting function, which is a feature in

Matlab that allows to pick-up for each ML technique the typically used parameters to optimize over, it has also to be mentioned that the used parameters for the optimization have already been explained in 6.1.1. In the following we show the results of the optimization on each one of the previously studied techniques:

- **Decision Trees:**

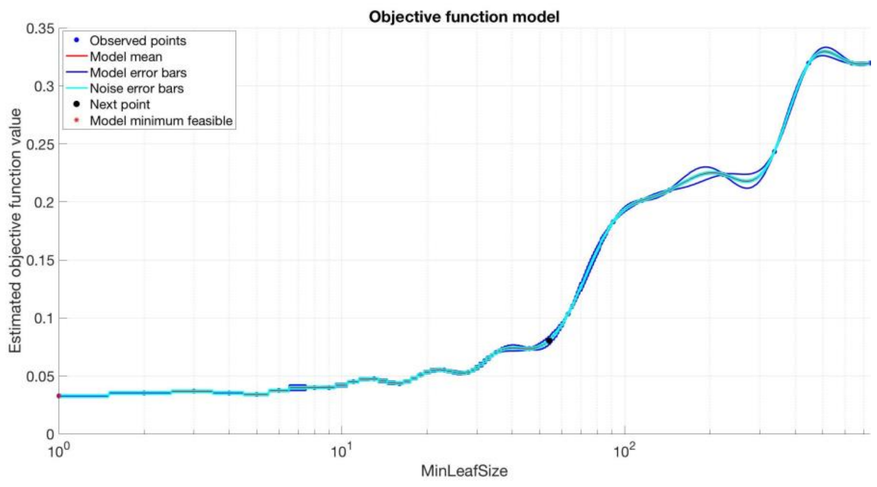


Figure 33. Decision trees objective function for different values of the MinLeafSize

- **Discriminant analysis:**

I

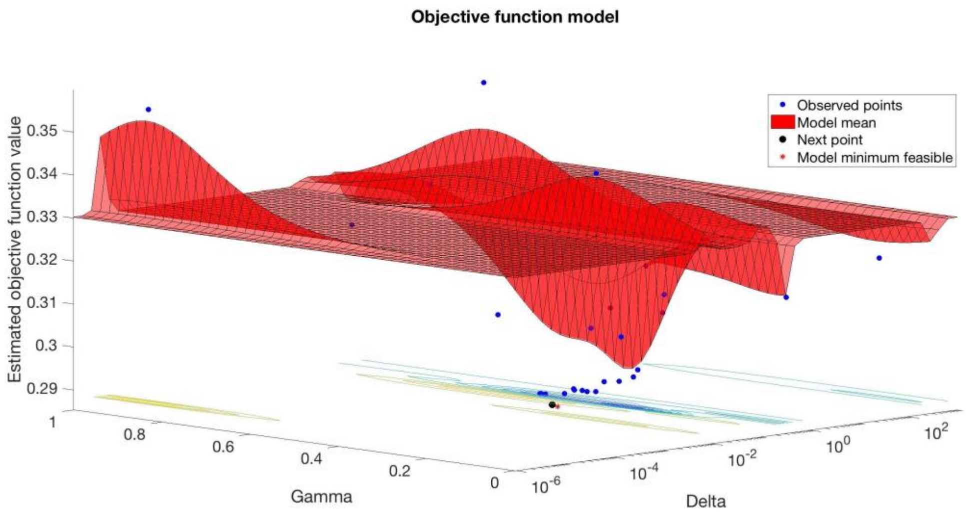


Figure 34. Discriminant Analysis objective function for different values of Gamma and Delta

- **Support vector machines:**

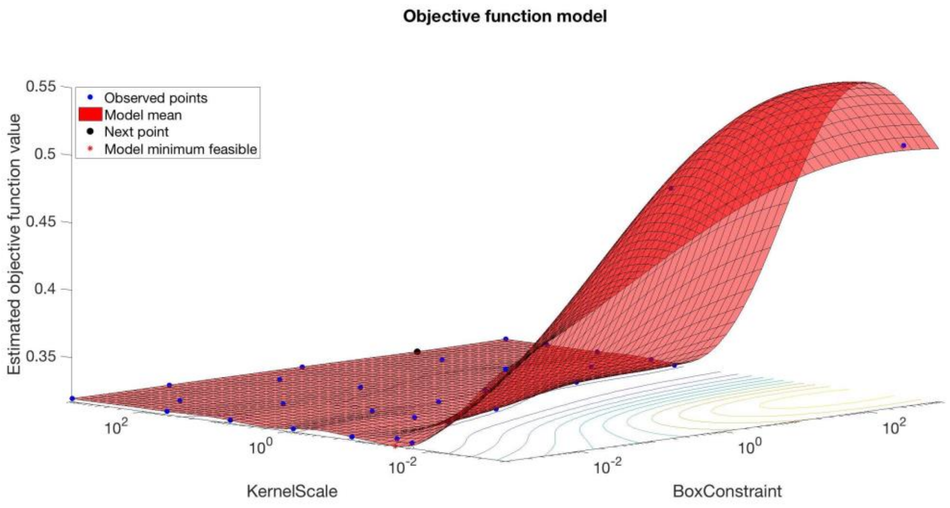


Figure 35. Support Vector Machines objective function for different values of the KernelScale and BoxConstraint

- **Naïve Bayes:**

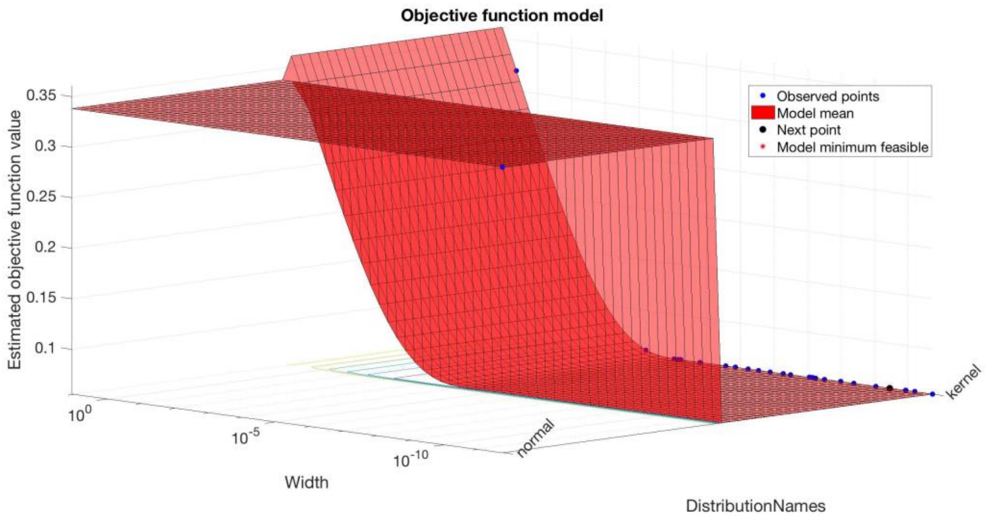


Figure 36. Naïve Bayes objective function for different distributions and different Width values

- **K-Nearest Neighbors:**

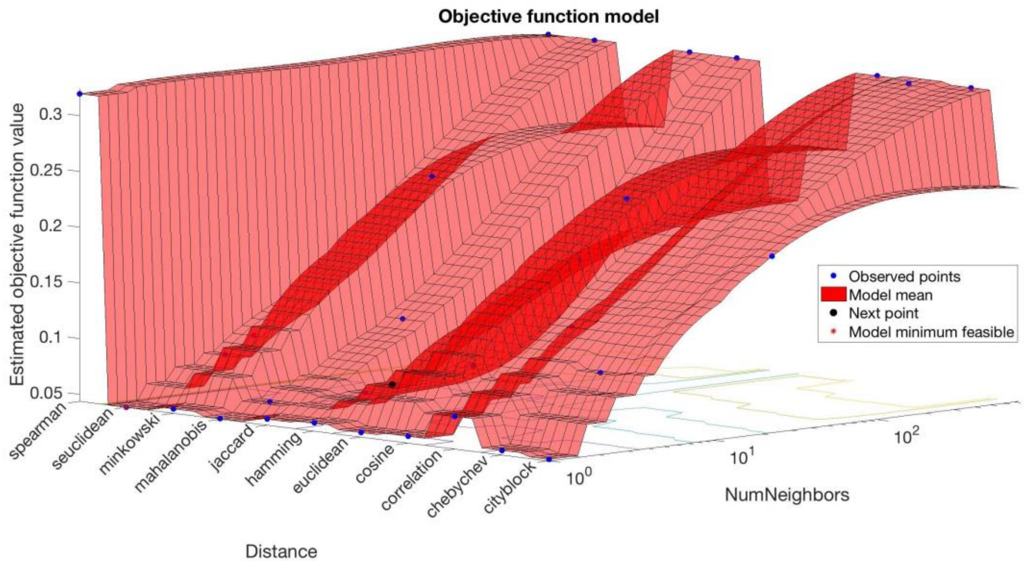


Figure 37. k-Nearest Neighbors objective function for different distance metrics and k values

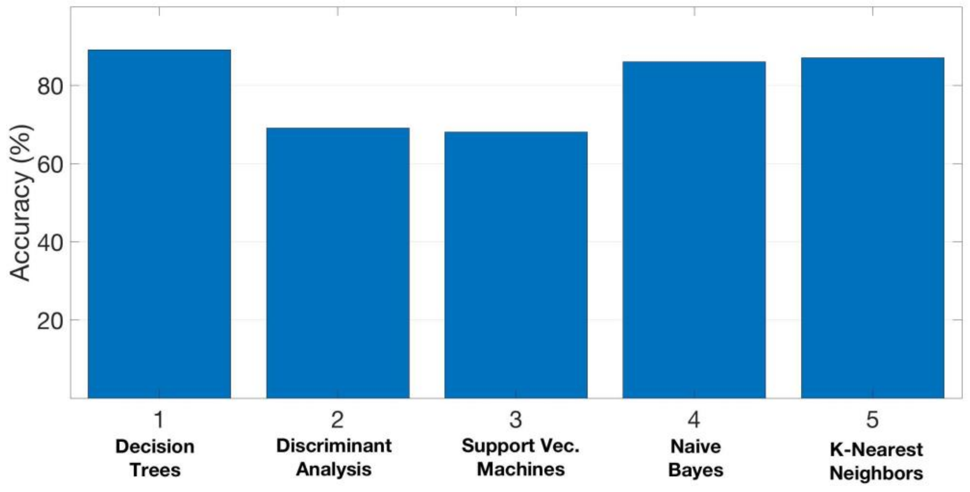


Figure 38. The accuracy of the different techniques after optimization

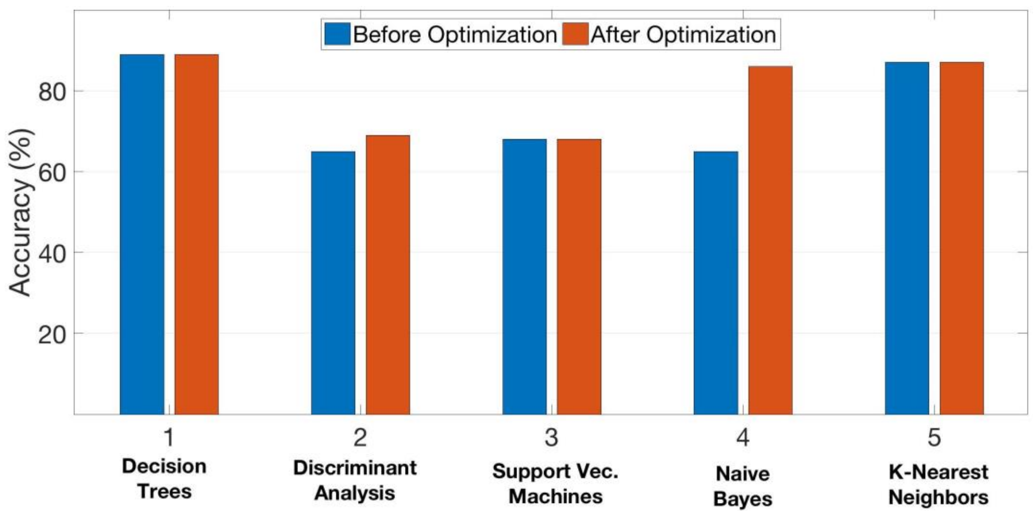


Figure 39. The accuracy of the different techniques before vs after optimization

The obtained accuracy after optimization is depicted in Fig. 38, while the comparison between both accuracies, before and after optimization is depicted in Fig. 39:

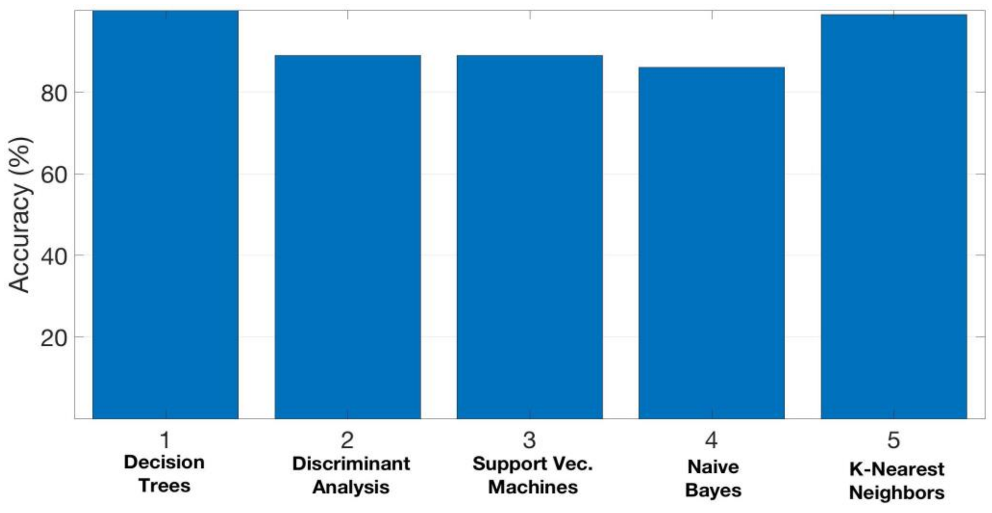


Figure 40. The accuracy of the different techniques based on the all similarity measures approach

The resulted-in improvement that we got thanks to the optimization process is as low as 0.4% for the decision trees or as high as 21% for the Naïve Bayes, but which still not performing better than the decision trees. Overall, the accuracy of the best performing ML technique didn't increase significantly.

6.2.1.2 All similarity measures approach:

The accuracy of the all similarity measures approach is depicted in Fig. 40, these accuracies have already been optimized using the Bayesian Optimization, while the comparison between the accuracy of this approach and the previous results is depicted in Fig.41 :

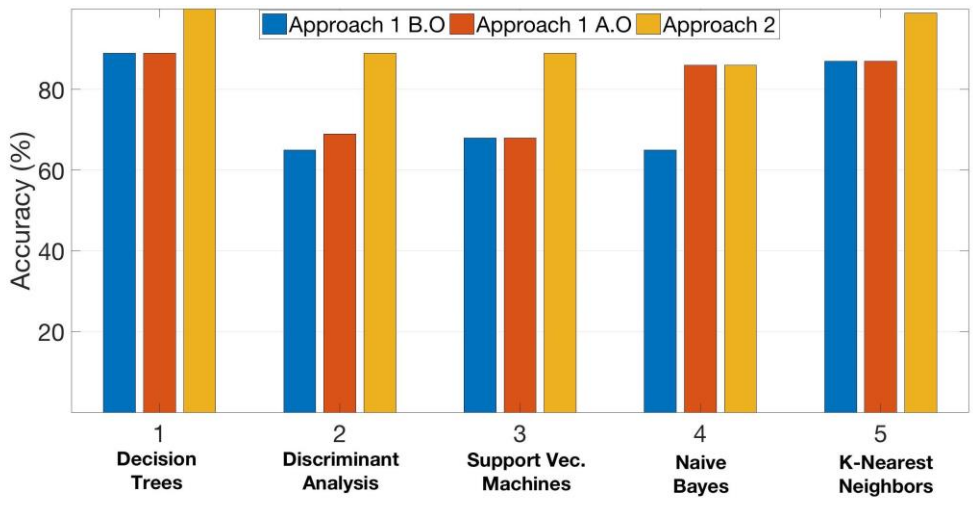


Figure 41. The accuracy of the different techniques based on both approaches

The all similarity approach outperforms all the other investigated techniques, especially when it comes to the decision trees, which classifies all the test data perfectly. The reason for which the all similarity approach outperforms the other investigated technique is because that it doesn’t have any loss any the information brought by the features, since we are taking all the similarity measures, while for the maximization approach we were taking only the best syntactic measure and the best hybrid measure.

6.2.2 Dimensionality reduction using PCA:

Principal Component Analysis (PCA) is the main technique used for dimensionality reduction. The data in the lower-dimensional space is chosen in such way that the variance of the data is maximized. If we assume that we want to reduce the dimensionality of the data from m dimensions to d dimensions (where $d < m$), the algorithm starts by computing the covariance matrix of the data, then it computes the eigenvalues of the covariance matrix, finally it picks up the d eigenvectors related to the highest d eigenvalues of the covariance matrix. One of the metrics to explain how good the chosen dimensions describe the data is the proportion of variance (PoV), which equals the sum of the eigenvalues of the chosen dimensions divided by the sum of all the eigenvalues.

In our use-case, dimensionality reduction is useful to reduce the number of features that need to be used for the classification, which can break down a greedy algorithm to an efficient one. In the case of all similarity approach we used 16 features knowing that the used URLs contain only three elements, which means that in real use case the number of features may increase incredibly because the URLs contain sometimes higher number of elements, as result the problem may become intractable.

Thus we chose to reduce the dimensionality and retrain the different MLAs. Figure 42 shows a set of similar and dissimilar URLs reduced to their two principal components. We can see from the figure that only 2 dimensions (knowing that the original data had 16 dimensions) separate most of the two classes quite well. Figure 43 shows the same things but for three dimensions, and it shows a better separability than the two dimensional case. We tried this for 2,3,4 and 5 dimensions, which gave a PoV equal to 67%, 80% ,92% and 98%. After retraining the network based on the new dimensions (five dimensional case), we got the accuracies depicted in the Fig. 44 :

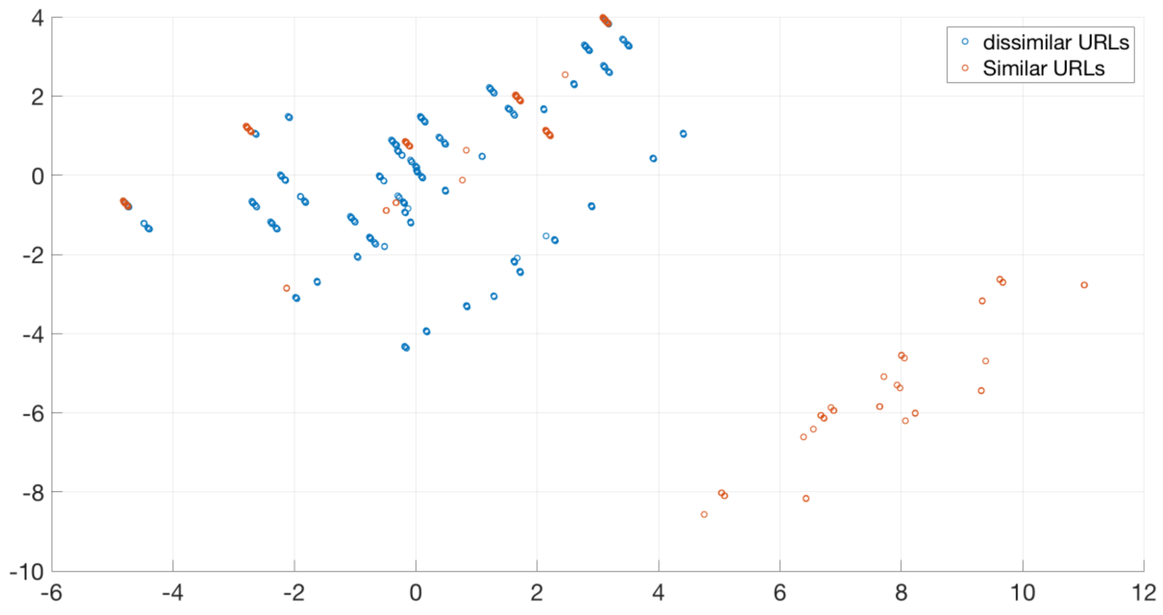


Figure 42. similar and dissimilair URLs plotted using their 2 principal components

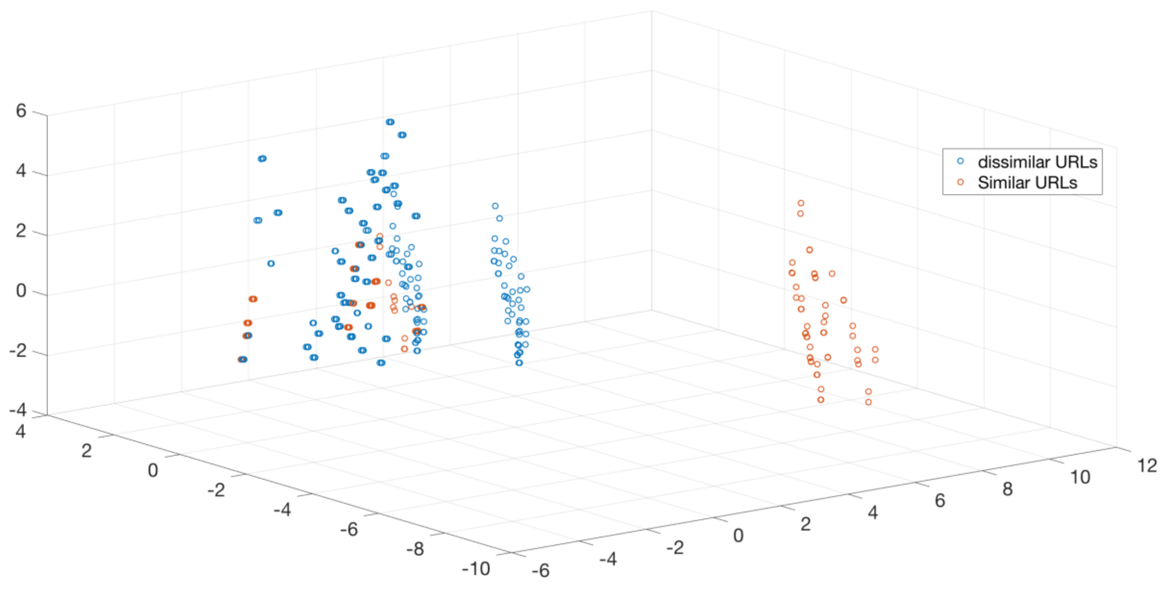


Figure 43. similar and dissimilair URLs plotted using their 3 principal components

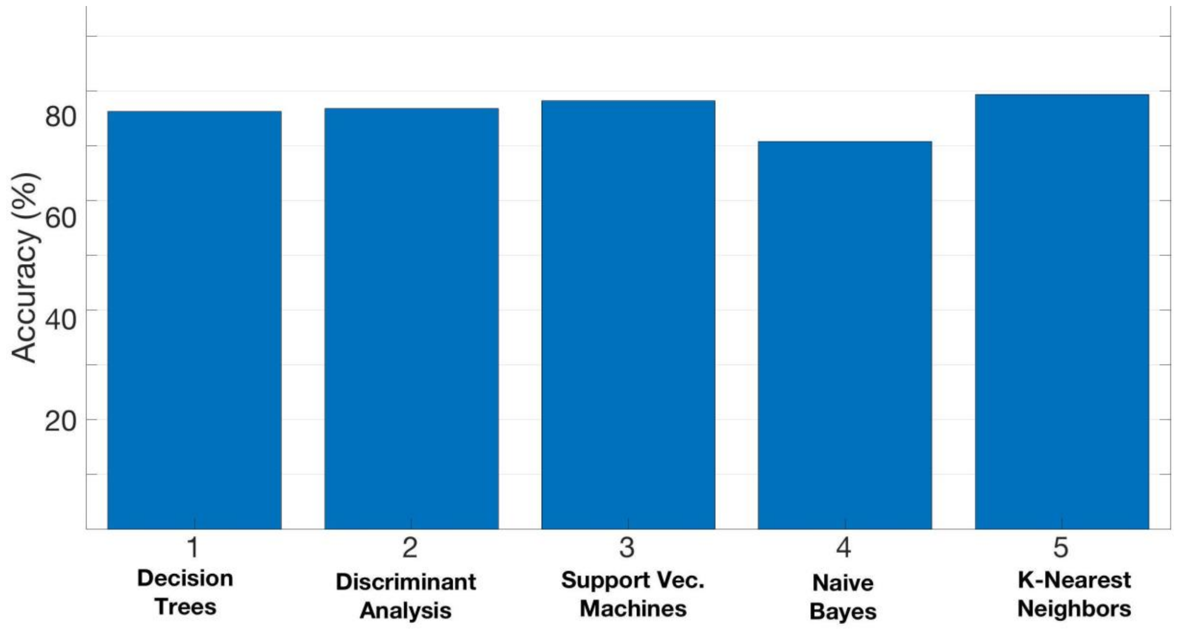


Figure 44. Accuracy of the different MLAs using 5 principal components

The results show a significant decrease in performance compared to the case where we used the 16 dimensions (or features). The decrease in performance is due to the PCA which is seems to be not convenient for the used data. We can explain the decrease in performance technically by the fact that the PCA considers that the most informative dimensions are those with the highest variance, so it tends to look for the useful information for the classification within the variance values, while the information may be hidden in the mean rather than the variance. This part may need more investigation using another technic of dimensionality reduction, which is based on the mean or other metrics beside the variance as in the PCA.

6.3 Conclusion

In this work, an original solution was proposed to deal with the interfacing of IoT devices based on APIs, which is believed to alleviate the interoperability issues.

After a thorough study of related works, it was concluded that internet of things still in its early stages and still have important challenges to overcome before reaching the true vision of IoT, and interfacing IoT devices regardless of their nature or manufacturer is one of the serious issues related to interoperability. So far, this problem has been dealt with by proposing different standards, but all of them failed in proposing a universal standard capable of interfacing the different IoT devices while considering their tremendous diversity and constrained capabilities.

The proposed idea in this work consists in using natural language processing and machine learning to classify the different API methods based on the specific action that they are supposed to perform, because being able to infer the related action to an API method regardless of the architecture of the API or the nature of the device using it, is going to be an important step in universally interfacing IoT devices.

After studying the different cases that we may come through while processing the different elements of the API methods, the different building blocks of the solution were identified, including namely the semantic and the hybrid similarity blocks, among others. Afterwards, the different natural language processing techniques were studied, evaluated and adapted to our application such that the differentiation between similar and dissimilar API methods is as easier as possible. The evaluation of both the syntactic similarity measures and the semantic similarity measures show the maturity of these methods and their usefulness in our use case. Then, rather than setting simple thresholds on the similarity measures to classify the API methods, a more sophisticated solution based on machine learning was chosen in order to get the highest possible accuracy. The different machine learning algorithms were studied and evaluated using real data from the different similarity measures. These algorithms were evaluated using two different approaches in retaining the similarity values of the URLs, the first similarity technique called "similarity maximization where we only keep the best semantic and hybrid similarity values, the second approach keeps all the similarity values between the two URLs. The results show the total feasibility of a such method, the all-similarity values approach along with the decision trees was able to classify all the test dataset correctly, knowing that the used URLs contained 30% of non-informative words. A possible significant amelioration of the results may be obtained not by focusing on the different similarity measures nor on the machine learning techniques since they are already performing well, rather, to improve the results (on real data) more informative information should be fed in as input, an example of this would be to reinforce the API methods by additional information from the API documentation. Finally, we tried Principal Component Analysis for dimensionality reduction in order to use less features for the classification of the API methods, but the obtained results show a poor performance compared to the other approaches.

Besides the proposed solution for interfacing IoT devices based on APIs, this approach can be used also to detect the capabilities and physical interfaces of the connected devices by sampling processing their API documentation, which can be highly useful in alleviating the interoperability issues and improving the service recommendation techniques. This work opens also the way to addressing interoperability issues, and in some cases, standardization in general, by using natural language processing and machine learning techniques without the need to impose anything to the different stakeholders except using meaningful key-words and words. However, important work remains to be done in this direction, especially for the GET API methods, because basically the work focused only on the PUT, POST and DELETE methods which can be associated to an action, while the GET methods are used to retrieve information which usually contains more than one single sub-information presented in different formats, therefore making the device interfacing hard to achieve.

7 References

[1] Noirie, Ludovic, et al. “Towards automated IoT service recommendation.” *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2017

[2] M. Le Pallec, M. O. Mazouz, and L. Noirie, “Physical-Interface-Based IoT Service Characterization,” in *Internet of Things (IoT’16)*, Stuttgart, Germany, Nov. 2016

[3] www.ETCIO.com. “Households have 10 connected devices now, will rise to 50 by 2020 - ET CIO.” *ETCIO.com*, 18 Aug. 2016, cio.economictimes.indiatimes.com/news/internet-of-things/households-have-10-connected-devices-now-will-rise-to-50-by-2020/53765773

[4] The Author Joseph BrookesJoseph Brookes is a writer for Which-50.com and the Which-50 Digital Intelligence Unit. “Typical households will have 29 connected devices each by 2020.” *Which-50*, 1 Oct. 2016, which-50.com/typical-households-will-29-connected-devices-2020/.

[5] Bröring, A., Datta, S. K., & Bonnet, C. (2016). A Categorization of Discovery Technologies for the Internet of Things. Proceedings of the 6th International Conference on the Internet of Things - IoT'16. doi:10.1145/2991561.2991570

[6] Maurel, Y., Lalanda, P., & Diaconescu, A. (2011). Towards a Service-Oriented Component Model for Autonomic Management. *2011 IEEE International Conference on Services Computing*. doi:10.1109/scc.2011.36

[7] Bröring, Arne, et al. “A Categorization of Discovery Technologies for the Internet of Things.” *Proceedings of the 6th International Conference on the Internet of Things - IoT16*, 2016

[8] Mihailovic, A. (2016). Liberalising Deployment of Internet of Things Devices and Services in Large Scale Environments. Wireless Personal Communications, 92(1), 33-49. doi:10.1007/s11277-016-3837-0

[9] Tracy, Phillip. “IoT interoperability: Where it stands and what comes next.” *RCR Wireless News*, 31 Oct. 2016, www.rcrwireless.com/20161031/internet-of-things/iot-interoperability-tag31-tag99.

[10] James Manyika, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin, and Dan Aharon. “Unlocking the potential of the Internet of Things.” *McKinsey & Company*, www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world.

[11] *The connected consumer : top trends in IoT*. www.parksassociates.com/bento/shop/whitepapers/files/ParksAssoc-ConnectedConsumer-TopTrends-in-IoT-2015.pdf.

[12] “Constrained RESTful Environments (CoRE) Link Format.” *IETF Tools*, tools.ietf.org/html/rfc6690

[13] “The Constrained Application Protocol (CoAP).” *IETF Tools*, tools.ietf.org/html/rfc7252.

[14] “Internet of Things - Discovery.” *XMPP*, xmpp.org/extensions/xep-0347.html.

[15] N. Guarino, M. Carrara, and P. Giaretta, An ontology of meta-level categories,4th International Conference Principles of Knowledge Representation and Reasoning.Citeseer, 1994, pp. 270-280

[16] Hachem, Sara, et al. “Ontologies for the internet of things.” *Proceedings of the 8th Middleware Doctoral Symposium on - MDS 11*, 2011

[17] *SWoT: Semantic Web of Things*, www.sensormeasurement.appspot.com/?p=ontologies.

[18] Asano, Satoshi, et al. “Device collaboration framework in IoT-Aggregator for realizing smart environment.” *2016 TRON Symposium (TRONSHOW)*, 2016.

[19] *Towards a standard API design for open services in smart buildings - IEEE Conference Publication*, ieeexplore.ieee.org/document/7842883/

[20] “Philips hue API.” *Philips hue API | Philips Hue API*, www.developers.meethue.com/philips-hue-api.

[21] “Introduction · LIFX HTTP Remote Control API.” *LIFX HTTP Remote Control API*, api.developer.lifx.com/

[22] “API.” *Nuki*, nuki.io/fr/api/.

[23] “HomeDocumentation.” *Lockitron*, api.lockitron.com/

[24] *Nokia apiDoc*, developer.health.nokia.com/api/doc

[25] H.gomaa, Wael, and Aly A. Fahmy. “A Survey of Text Similarity Approaches.” *International Journal of Computer Applications*, vol. 68, no. 13, 2013, pp. 13–18

[26] “Hyperspace Analogue to Language (HAL) Introduction - Semantikoz.” *Big Data Science and Cloud Computing*, 16 Nov. 2014, www.semantikoz.com/blog/hyperspace-analogue-to-language-hal-introduction/

[27] “Latent semantic analysis.” *Wikipedia*, Wikimedia Foundation, 6 Oct. 2017, en.wikipedia.org/wiki/Latent_semantic_analysis.

[28] Landauer, Thomas K., and Susan T. Dumais. “A solution to Platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.” *Psychological Review*, vol. 104, no. 2, 1997, pp. 211–240

[29] Explicit semantic analysis.” *Wikipedia*, Wikimedia Foundation, 3 Jan. 2017, en.wikipedia.org/wiki/Explicit_semantic_analysis.

[30] “Cosine similarity.” *Wikipedia*, Wikimedia Foundation, 29 Sept. 2017, en.wikipedia.org/wiki/Cosine_similarity.

[31] University, Princeton. “What is WordNet?” *Princeton University*, Trustees of Princeton University © 2017, 17 Mar. 2015, wordnet.princeton.edu/.

[32] Fellbaum, Christiane. *WordNet: an electronic lexical database*. MIT Press, 2000.

[33] Banerjee, Satanjeev, and Ted Pedersen. “An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet.” *Computational Linguistics and Intelligent Text Processing Lecture Notes in Computer Science*, 2002, pp. 136–145

[34] Patwardhan, Siddharth. “Incorporating dictionary and corpus information into a context vector measure of semantic relatedness.” *University of Minnesota, Duluth*, 2003.

[35] “Similarity Measures.” *Measures*, atlas.ahc.umn.edu/umls_similarity/similarity_measures.html.

[36] Sebastien Harispe. Knowledge-based Semantic Measures: From Theory to Applications. Computer Science [cs]. Universit ́e de Montpellier, 2014. English.

[37] “Natural Language Toolkit¶.” *Natural Language Toolkit — NLTK 3.2.5 documentation,* www.nltk.org/.

[38] Islam, Aminul, and Diana Inkpen. “Semantic text similarity using corpus-Based word similarity and string similarity.” *ACM Transactions on Knowledge Discovery from Data,* vol. 2, no. 2, Jan. 2008, pp. 1–25.,

[39] Nitish, A., Kartik, A. & Paul, B. (2012). DERI&UPM: Pushing Corpus Based Relatedness to Similarity: Shared,Task System Description. First Joint Conference on Lexical and Computational Semantics (*SEM), pages 643-647, Montreal, Canada, June 7-8, 2012 Association for Computational Linguistics

[40] “Modern Machine Learning Algorithms: Strengths and Weaknesses.” *EliteDataScience*, 16 Sept. 2017, elitedatascience.com/machine-learning-algorithms.

[41] Bradley, J., Barbier, J., & Handler, D. (2013). Embracing the Internet of Everything To Capture Your Share of $14.4 Trillion. Cisco and/or its affiliates.

[42] Internet of Things: The Complete Reimaginative Force, conducted by Research Now, Tata Consultancy Services, 22 July 2015, as cited by eMarketer,

[43] Smart City Solutions. (n.d.). Retrieved October 28, 2017, from http://bigbelly.com/

[44] “Eddystone (Google).” Wikipedia, Wikimedia Foundation, 30 Aug. 2017, en.wikipedia.org/wiki/Eddystone_(Google).

[45] “Longest common subsequence problem.” Wikipedia, Wikimedia Foundation, 16 Oct. 2017, en.wikipedia.org/wiki/Longest_common_subsequence_problem.

[46] "Jaro–Winkler distance." Wikipedia. November 03, 2017. Accessed November 04, 2017.

[47] Levenshtein distance. (2017, October 26). Retrieved November 04, 2017, from https://en.wikipedia.org/wiki/Levenshtein_distance

[48] Tf–idf. (2017, December 29). Retrieved January 03, 2018, from https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[49] Constant, M. (n.d.). Similarité entre les mots (Rep.). Traitement Automatique des Langues Master Informatique Université Paris-Est Marne-la-Vallée

[50] Patwardhan, S. (2003). Incorporating dictionary and corpus information into a context vector measure of semantic relatedness (Unpublished master's thesis). University of Minnesota, Duluth.

[51] “Blockchain.” Wikipedia, Wikimedia Foundation, 28 Jan. 2018, en.wikipedia.org/wiki/Blockchain.

[52] Is REST Successful in the Enterprise? (n.d.). Retrieved from https://www.infoq.com/news/2011/06/Is-REST-Successful

Index 1

```
from nltk.corpus import wordnet as wn
from nltk.corpus.reader import NOUN
from nltk.corpus import wordnet_ic
import random
brown_ic = wordnet_ic.ic('ic-brown.dat')

def Path(wordA,wordB):
    Syns1=wn.synsets(wordA,NOUN)
    Syns2=wn.synsets(wordB,NOUN)
    if ( len(Syns1)==0 or len(Syns2)==0):
        return 0.0
    max_sim=Syns1[0].path_similarity(Syns2[0])
        for j in range(0,len(Syns1)):
        for i in range(0,len(Syns2)):
            if (max_sim< Syns1[j].path_similarity(Syns2[i])):
                max_sim=Syns1[j].path_similarity(Syns2[i])
    return max_sim

def Lch(wordA,wordB):
    Syns1=wn.synsets(wordA,NOUN)
    Syns2=wn.synsets(wordB,NOUN)
    if ( len(Syns1)==0 or len(Syns2)==0):
        return 0.0
    max_sim=Syns1[0].lch_similarity(Syns2[0])
    for j in range(0,len(Syns1)):
        for i in range(0,len(Syns2)):
            if (max_sim< Syns1[j].lch_similarity(Syns2[i])):
                max_sim=Syns1[j].lch_similarity(Syns2[i])
    return max_sim

def Wup(wordA,wordB):
    Syns1=wn.synsets(wordA,NOUN)
    Syns2=wn.synsets(wordB,NOUN)
    if ( len(Syns1)==0 or len(Syns2)==0):
        return 0.0
    max_sim=Syns1[0].wup_similarity(Syns2[0])
    for j in range(0,len(Syns1)):
        for i in range(0,len(Syns2)):
            if (max_sim< Syns1[j].wup_similarity(Syns2[i])):
                max_sim=Syns1[j].wup_similarity(Syns2[i])
    return max_sim

def Resnik(wordA,wordB):
    Syns1=wn.synsets(wordA,NOUN)
    Syns2=wn.synsets(wordB,NOUN)
    if ( len(Syns1)==0 or len(Syns2)==0):
        return 0.0
    max_sim=Syns1[0].res_similarity(Syns2[0],brown_ic)
    for j in range(0,len(Syns1)):
        for i in range(0,len(Syns2)):
            if (max_sim< Syns1[j].res_similarity(Syns2[i],brown_ic)):
                max_sim=Syns1[j].res_similarity(Syns2[i],brown_ic)
    return max_sim

def Jcn(wordA,wordB):
    Syns1=wn.synsets(wordA,NOUN)
    Syns2=wn.synsets(wordB,NOUN)
    if ( len(Syns1)==0 or len(Syns2)==0):
        return 0.0
    max_sim=Syns1[0].jcn_similarity(Syns2[0],brown_ic)
    for j in range(0,len(Syns1)):
        for i in range(0,len(Syns2)):
            if (max_sim< Syns1[j].jcn_similarity(Syns2[i],brown_ic)):
                max_sim=Syns1[j].jcn_similarity(Syns2[i],brown_ic)
    return max_sim
```

```
def Lin(wordA,wordB):
    Syns1=wn.synsets(wordA,NOUN)
    Syns2=wn.synsets(wordB,NOUN)
    if ( len(Syns1)==0 or len(Syns2)==0):
        return 0.0
    max_sim=Syns1[0].lin_similarity(Syns2[0],brown_ic)

    for j in range(0,len(Syns1)):
        for i in range(0,len(Syns2)):
            if (max_sim< Syns1[j].lin_similarity(Syns2[i],brown_ic)):
                max_sim=Syns1[j].lin_similarity(Syns2[i],brown_ic)
    return max_sim
```

Index 2

```
% numerical integral of the overlapping area of two normal distributions:
% s1,s2...sigma of the normal distributions 1 and 2
% mu1,mu2...center of the normal distributions 1 and 2
% xstart,xend,xinterval...defines start, end and interval width
% example: [overlap] = calc_overlap_twonormal(2,2,0,1,-10,10,0.01)
function [overlap2] = calc_overlap_twonormal(s1,s2,mu1,mu2,xstart,xend,xinterval)
clf
x_range=xstart:xinterval:xend;
plot(x_range,[normpdf(x_range,mu1,s1)' normpdf(x_range,mu2,s2)']);
hold on
area(x_range,min([normpdf(x_range,mu1,s1)' normpdf(x_range,mu2,s2)']'));
overlap=cumtrapz(x_range,min([normpdf(x_range,mu1,s1)' normpdf(x_range,mu2,s2)']'));
overlap2 = overlap(end);
legend([num2str(overlap2)]);
```

Index 3

```
 from nltk.corpus import wordnet as wn
from nltk.corpus.reader import NOUN
from nltk.corpus import wordnet_ic
def levenshtein(s1, s2):
    if len(s1) < len(s2):
        return levenshtein(s2, s1)
    # len(s1) >= len(s2)
    if len(s2) == 0:
        return len(s1)
    previous_row = range(len(s2) + 1)
    for i, c1 in enumerate(s1):
        current_row = [i + 1]
        for j, c2 in enumerate(s2):
            insertions = previous_row[j + 1] + 1 # j+1 instead of j since previous_row and
current_row are one character longer
            deletions = current_row[j] + 1       # than s2
            substitutions = previous_row[j] + (c1 != c2)
            current_row.append(min(insertions, deletions, substitutions))
        previous_row = current_row
    return previous_row[-1]
def leven_norm(w1, w2):
    return float(levenshtein(w1,w2))/(max(len(w1),len(w2)))
brown_ic=wordnet_ic.ic('ic-brown.dat')
word1='lamp'
word2='lamp_lock'
sys_word1=wn.synsets(word1,NOUN)
allnouns = [x for x in wn.all_synsets('n')]
min_sim=1
for i in range(0,len(allnouns)):
    for j in range(0,len(sys_word1)):
        res_sim=sys_word1[j].res_similarity(allnouns[i],brown_ic)
```

```
if(res_sim > 8):
    lev_sim=leven_norm(word2,allnouns[i].name().split(".")[0])
    if(min_sim> lev_sim):
        min_sim=lev_sim
        #print(sys_word1[j])
        #print(allnouns[i])
        print(min_sim)
```

Index 4

```
'''this program generates  similarity measures ( semantic + hybrid ) for similar or dissimilar words, then the generated values are used by another program within Matlab to train and test the different machine learning algorithms'''

from nltk.tokenize import word_tokenize

from nltk.corpus import wordnet as wn

from nltk.corpus.reader import NOUN

from nltk.corpus import wordnet_ic

import random

brown_ic = wordnet_ic.ic('ic-brown.dat')'''importing the information content

file, which is needed for the Resnik similarity measure. More information may

be found on the report or in this link :
https://stackoverflow.com/questions/18705778/what-is-the-use-of-brown-corpus-in-measuring-semantic-similarity-based-on-wordne

'''

'''Filter function needed to perform the preprocessing step, by getting rid of the different special characters

that we may encouter while dealing with URLs'''

def Filter (Link) :

   URL=Link

   F_URL=''

   for i in range(0,len(URL)):

      if ( URL[i]!='/' and URL[i]!='<' and URL[i]!='>'):

         F_URL+=URL[i]

      else : F_URL+=' '

   return(word_tokenize(F_URL))

' a function that checks whether the string w1 is a word or not, it does this by only looking in the WordNet corpus'

def IsWord (w1):

   return wn.synsets(w1)

'Semantic_sim is a function that outputs the semantic similarity measure between wordA and wordB'

def Semantic_sim(wordA,wordB):

   Syns1=wn.synsets(wordA,NOUN)'get all the Noun synsets of the WordA'

   Syns2=wn.synsets(wordB,NOUN)
```

```
    if ( len(Syns1)==0 or len(Syns2)==0):
        return 0.0
'''since each synset associated to a word contains many words, we define the semantic
similarity as the Resnik similarity between
the two words in each synset that would give us the maximum Resnik similarity, thus in
the remaining part of this function
we will be going through the different words of each synset to look for the maximum
Resnik value'''
    max_sim=Syns1[0].res_similarity(Syns2[0],brown_ic)


    for j in range(0,len(Syns1)):
        for i in range(0,len(Syns2)):
            if (max_sim< Syns1[j].res_similarity(Syns2[i],brown_ic)):
                max_sim=Syns1[j].res_similarity(Syns2[i],brown_ic)
    return max_sim
'''Levenshtein function is very famous function, it is used in this case as a syntactic
measure, more information can be found on the internet about this algorithm, it is also
known as The Edit Distance Algorithm'''
def levenshtein(s1, s2):
    if len(s1) < len(s2):
        return levenshtein(s2, s1)
    # len(s1) >= len(s2)
    if len(s2) == 0:
        return len(s1)
    previous_row = range(len(s2) + 1)
    for i, c1 in enumerate(s1):
        current_row = [i + 1]
        for j, c2 in enumerate(s2):
            insertions = previous_row[j + 1] + 1 # j+1 instead of j since previous_row and
current_row are one character longer
            deletions = current_row[j] + 1       # than s2
            substitutions = previous_row[j] + (c1 != c2)
            current_row.append(min(insertions, deletions, substitutions))
        previous_row = current_row
    return previous_row[-1]
```

```
'''as explained in the report, we noticed that normalizing the levenshtein measure by the length of the longest word may in some cases give better results than the standard Levenshtein measure'''
def leven_norm(w1, w2):
    return float(levenshtein(w1,w2))/(max(len(w1),len(w2)))
'a function used to generate URLs, in this example these words are used to generate dissimilar URLs'
def generate_sim():
    table1=['lock','desktop-computer','house-lock','smart_lock','watch','phone','connected-lock','connected_watch','room_lock','room-conditioner','security-lock','emergency-alarm','television1','screen2','speaker3']
    table2=['states','all-states','screen2-state','watch-state','status','<id>','user','all']
    table3=['api','v1','v3','link','REST','1']
    links=[]
    for i in range(0,len(table1)):
        for j in range(0,len(table2)):
            for k in range(0,len(table3)):
                links=links+['/'+table1[i]+'/'+table2[j]+'/'+table3[k]]
    return links
'''a function that measures the hybrid similarity as explained in the report, for instance it can be used to measure the similarity

between the word 'light' and 'smart-bulb', it firstly gets as input all the similar words to the word 'light' i the array allnouns and 'smart-bulb' is passed to the function through the string w2'''
def hybrid_sim(allnouns,w2):
    min_sim=1.0
    sum_sim=0.0
    count=0.0
    res=[]
    for i in range(0,len(allnouns)):
        lev_sim=leven_norm(w2,allnouns[i])
        sum_sim+=lev_sim
        count+=1
        if(min_sim> lev_sim):
            min_sim=lev_sim
    if(count!=0.0):
        res=[min_sim,sum_sim/count] ' we return the best syntactic value and the mean of all
the syntactic measures '
    else :
        res=[1.0 , 1.0 ]
    return res
'''This is the main function, which relies actually on all the previously defined functions, it
returns a three dimensional array containing the three similarity measures that we already
talked about, the similarity measure is performed between the URL and URL1 given as
input ... it gets also as input s1,s2 and s3 which are vectors containing all the similar words
to the three words of URL1 and which are needed to computed the Hybrid
similarity...s1,s2 and s3 can be computed inside the compute_sim function rather than
passing them as input parameters, for the this example we prefered to pass them as input
parameters to reduce the running time, because if these parameters are computed inside the
compute sim function, it means that they will be computed each time we call the function
compute_sim..this solution reduced drastically the running time...an alternative solution is
to compute the values of s1,s2 and s3 inside the function, but while storring their values in
global variables'''
def compute_sim(URL,URL_1,s1,s2,s3):
    URL_1=Filter(URL_1)
    URL_2=Filter(URL)
    semantic_simi=0.0
    hybrid_simi=[1.0 , 1.0]
    for i in range(0,len(URL_1)):
        for j in range(0,len(URL_2)):
            if(IsWord(URL_1[i]) and IsWord(URL_2[j])): ' if the two element of each URL
are words, then we will perform a semantic similarity'
                res_int=Semantic_sim(URL_1[i],URL_2[j])
                if(semantic_simi<res_int):
                    semantic_simi=res_int
            else:                          'otherwise we will perform a hybrid similarity'
                if (i==0):
                   res_int2=hybrid_sim(s1,URL_2[j])
                if(i==1):
                    res_int2=hybrid_sim(s2,URL_2[j])
                if(i==2):
                    res_int2=hybrid_sim(s3,URL_2[j])
            if(res_int2[0]<hybrid_simi[0] or res_int2[1]<hybrid_simi[1]):
               hybrid_simi=res_int2
   res_final= hybrid_simi + [semantic_simi]   ' we return the hybrid and semantic 
similiarity values'
   return res_final
'''this function is used to extract from the WordNet corpus the similar words to the word 
stored in the variable word, the output of this function
is needed for the hybrid similarity function'''
def similar_words(word):
   w1=word
   simi_words=[]
   sys_word1=wn.synsets(w1,NOUN)
   allnouns = [x for x in wn.all_synsets('n')]
   for i in range(0,len(allnouns)):
       for j in range(0,len(sys_word1)):
          res_sim=sys_word1[j].res_similarity(allnouns[i],brown_ic)
          if(res_sim > 8):
             simi_words=simi_words+[allnouns[i].name().split(".")[0]]
   return simi_words
' we will be looking for all the similar URLs to URL_1'
URL_1='/light/lamp/state'
'as explained earlier, in order to reduce the running time, we extract from WordNet all the 
similar words to the words of URL_1'
s1=similar_words(Filter(URL_1)[0])
s2=similar_words(Filter(URL_1)[1])
s3=similar_words(Filter(URL_1)[2])
links=generate_sim() ' generate URLs'
train=[] 'it is the vector which gonna hold the similarity values between URL_1 and all the 
generated URLs'
for i in range(0,len(links)):
   train=train+[compute_sim(links[i],URL1,s1,s2,s3)]
   print(i)
random.shuffle(train) '  just to add randomness ..'
```

Index 5

Creating the models

```
Mdl_tree= fitctree(X_train,Y_train); %decision trees
Mdl_discr= fitcdiscr(X_train,Y_train); % discriminant analysis
Mdl_svm= fitcsvm(X_train,Y_train); % support vector machine
Mdl_nb= fitcnb(X_train,Y_train) ; % naive bayes
Mdl_nn= fitcknn(X_train,Y_train) ; % nearest neighbor

% computing the accuracy of each model based on a dataset of 202 URLs

ac_tree= (1-loss(Mdl_tree,X_test2,Y_test2))*100;
ac_discr=(1-loss(Mdl_discr,X_test2,Y_test2))*100;
ac_svm=(1-loss(Mdl_svm,X_test2,Y_test2))*100;
ac_nb=(1-loss(Mdl_nb,X_test2,Y_test2))*100;
ac_nn=(1-loss(Mdl_nn,X_test2,Y_test2))*100;

%plotting the accuracy graph bars
accuracy_vect=[ac_tree ac_discr ac_svm ac_nb ac_nn];
bar(accuracy_vect)
set(gca,'xticklabel',{'tree','discr','svm','nbayes','neighboor'});
ylabel('accuracy percentage')
xlabel('different algorithms')
figure;
%plotting the accuracy variation as function of the size of the testing
%dataset

X_axe= [ 50 , 100 , 150 , 200 , 250 , 300, 350 , 400 , 450 , 500 , 550 , 600 ];

Vac_tree=[];
Vac_discr=[];
Vac_svm=[];
Vac_nb=[];
Vac_nn=[];
% computing the accuraccy of each algorithm while variying the testing
% dataset size, the values are stored into vectors
for i=1:12
  Vac_tree= [Vac_tree, (1-loss(Mdl_tree,X_test(1:i*50,:),Y_test(1:i*50,:)))*100];
  Vac_discr= [Vac_discr, (1-loss(Mdl_discr,X_test(1:i*50,:),Y_test(1:i*50,:)))*100];
  Vac_svm= [Vac_svm, (1-loss(Mdl_svm,X_test(1:i*50,:),Y_test(1:i*50,:)))*100];
  Vac_nb= [Vac_nb, (1-loss(Mdl_nb,X_test(1:i*50,:),Y_test(1:i*50,:)))*100];
  Vac_nn= [Vac_nn, (1-loss(Mdl_nn,X_test(1:i*50,:),Y_test(1:i*50,:)))*100];

end

plot(X_axe,Vac_tree,'b','LineWidth',2)
hold on
plot(X_axe,Vac_discr,'y','LineWidth',2)
hold on
plot(X_axe,Vac_svm,'r','LineWidth',2)
hold on
plot(X_axe,Vac_nb,'g','LineWidth',2)
```

```
hold on
plot(X_axe,Vac_nn,'k','LineWidth',2)
grid on
ylabel('accuracy percentage')
xlabel('testing dataset size')
```