Réf: 2018/II2/ N° of subject:

Ministry of Higher Education and Scientific Research

University of Manouba
National School of Computer Sciences

Design and Development Project Report

Subject

DESIGN AND DEVELOPMENT OF CLOUD SERVICES SEARCH ENGINE

Directed By

Omar ABDELLI

Mohamed Ahmed MHENNI
Houssem Eddine GHARBI

Supervised By **: Rim DRIRA**

Co-supervised By **: Hamdi GABSI**

Academic Year : 2017/2018

Abstract

THis work comes within the scope of the second year's design and development project at the T National School of Computer Science. The project covers the design and the development of a Cloud services search engine. We aim to assist Cloud services users in searching and selecting appropriate services meeting their requirements. Keywords: Cloud services search engine, search, user requirements.

Signature

Acknowledgement

In conducting this report, we have received meaningful assistance from many quarters which we like to put on record here with deep gratitude and great pleasure.

First and foremost, we express our sincere gratitude to our supervisor, Miss Rim Drira, who extended her complete support and helped to make us deliver our best, as well as our co-superviser Mr Hamdi Gabsi for their guidance and valuable pieces of advice during different phases of the elaboration of the project. Simply put, we could not have done this work without the amount of help and trust we received from them. We appreciate their patience in checking and reviewing our work.

We would also like to thank all of our teachers at the National School of Computer Science for their continuous help and treasurable training during our study years.

Finally, special thanks to the jury members who honored us by examining and evaluating this modest contribution.

Contents

General Introduction **2**

1 Preliminary Study **3**

1.1 Presentation of the project 3

1.2 Main concepts 3

1.3 Problem setting 4

1.4 Existing Solutions 5

1.4.1 Cloud Services Market 5

1.4.2 Serchen 6

1.5 Proposed solution 7

1.6 Working Methodology 8

1.6.1 Agile Methodology 9

1.6.2 Scrum 9

1.6.3 Kanban 10

1.6.4 Chosen methodology 10

2 Theoretical Study **12**

2.1 Text Classification 12

2.1.1 Supervised text classification: 12

2.1.2 Unsupervised text classification: 13

2.2 Clustering 18
2.2.1 K-means 18
2.2.2 Elbow method 19

3 Requirements Analysis and Specification 21

3.1 Requirements Analysis 21
3.1.1 Actors 21
3.1.2 Functional Requirements 21
3.1.3 Non functional requirements 22
3.2 Requirement Specification 23
3.2.1 Use Case Diagrams 23
3.2.2 System Sequence Diagram 24

4 Design 26

4.1 Global design 26
4.1.1 Physical architecture 26
4.1.2 Logical architecture 27
4.2 Detailed design 28
4.2.1 Class Diagram 29
4.2.2 Sequences Diagram 29
4.3 External Scripts 31

5 Achievement 33

5.1 Sprint1: Data collection 33
5.2 Sprint2: Data processing 36
5.3 Sprint3: Data modeling 38
5.4 Sprint4: Decision making and Evaluate Algorithms 39

5.5 Sprint5: Cloud Services Search Engine 42
5.6 Sprint6: Search engine platform 43
5.6.1 Development 43
5.6.2 Interfaces 45

General Conclusion **50**

Bibliography **52**

List of Figures

1.1 Cloud service market overview [N12] 6

1.2 Serchen home page 7

1.3 Classes Unifying 8

1.4 Work-flow of scrum methodology [N2] 9

1.5 kanban to-do list [N2] 10

1.6 An example of to do list of our project 11

2.1 Text classification pipeline [B1] 13

2.2 Unsupervised Text classification main steps 13

2.3 English stop words 14

2.4 Stemming result 15

2.5 TF-IDF equation N[7] 15

2.6 one-hot encoding N[8] 16

2.7 Word2Vec distributed representation N[8] 16

2.8 Word2Vec distributed representation N[9] 17

2.9 Doc2Vec distributed representationN[10] 17

2.10 k-means process N[11] 19

2.11 Sample of data set 19

2.12 Elbow method 19

3.1 Use case diagram 23

3.2 Sequence diagram of the use case"Create a customized clustering " 24

4.1 3-tiers architecture 27

4.2 MVT architecture 28

4.3 Class diagram 29

4.4 Sequences diagram 30

4.5 Example of running a script 31

5.1 AWS products 34

5.2 IBM products 34

5.3 Google products 34

5.4 Oracle product 35

5.5 Amazon data 35

5.6 Google data 36

5.7 Stop words implementation 37

5.8 Tokenize and stem the descriptions 37

5.9 Synonyms of management by babelnet 38

5.10 Automatic Detection for the optimal K 40

5.11 Cos similarity presentation 40

5.12 Clustering result after TF-IDF 41

5.13 Clustering result after doc2vec 42

5.14 whoosh ranking function 42

5.15 Database Models 43

5.16 Relation URLs/Views 44

5.17 Our Base.html 44

5.18 Login interface 45

5.19 Start interface 46

5.20 My services interface 46

5.21 Classes interface 47

5.22 Class details interface 47

5.23 My classes interface 48

5.24 Updates interface 48

5.25 API interface 49

5.26 API Response 49

List of Tables

General Introduction

In the era of constant evolution, the Cloud Computing is one of the most developed technologies that offers fast and efficient services at low costs. Currently, the number of cloud providers and services is growing fastly. Each cloud provider offers a rich catalog of services using a his specific vocabulary and many cloud services could be candidates for fulfilling a same given business requirement. Therefore, searching and selecting suitable cloud services is becoming a hard and time-consuming task. To address these challenges, we propose a new solution inspired from data science strategy in order to develop a search engine of Cloud services . First of all, we collect cloud services of different providers catalogs in a common repository which is regularly updated according to catalogs updates. Then, we propose a semantic Cloud service clustering. The proposed clusters will be used to enhance and improve Cloud services search engine's results and performances. Third, we develop a user friendly platform offering assisted search of cloud services and enabling the rating of used services. To the best of our knowledge, there is no tool which supports such functionalities for assisting the search and the selection process of cloud services. Our work is presented according to the following outline. In the first chapter we shed the light on some of the most important key concepts related to our work, then we present our problem setting followed by a study of the existing solution which focused on the same issues and finally we present our proposed solution as well as the working methodology. In the second chapter, we went through a theoretical study of the different algorithms and techniques that we used during our work. In the third chapter, we make a list of the functional and non-functional requirements and we present the use case and the system sequence diagram related to our work. In the fourth chapter, we describe the project design and architecture.

Besides, during the fifth chapter we describe deeply each sprint done in our solution. And we present the web-platform interfaces. Finally we conclude our work by giving a sum up and some potentials upgrades for our proposed platform.

Chapter 1

Preliminary Study

This chapter introduces the general context of our work. We will present our project by defining its context, its main concepts as well as the expected results.

1.1 Presentation of the project

This project entitled "Cloud Services Search Engine" was conducted in the context of a second year study project. It is specified as one of the main subjects in the official program of the National School of computer Science. It is supervised by our teachers within four months during the second semester of the second year.

1.2 Main concepts

In this section, we define the most important concepts related to our project.

Cloud Computing

According to the National Institute of Standards and Technology (NIST) "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be

rapidly provisioned and released with minimal management effort or service provider interaction. "[NIST1] Cloud computing offers three main services modes which are ;

- **SAAS** "(Software as a Service): The capability provided to the consumer is to use the provider's applications running on a Cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying Cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings."[NIST1]
- **PAAS** "(Platform as a Service): The capability provided to the consumer is to deploy onto the Cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying Cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations."[NIST1]
- **IAAS** "(Infrastructure as a Service): The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying Cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls)".[NIST1]
- **A Cloud provider** is a company that can provide and maintain cloud services typically the three mentioned Cloud service models (IAAS, PAAS and SAAS). In our work we focused on four Cloud providers among the leaders of the cloud market which are Amazon (AWS), Google , IBM and Oracle.

1.3 Problem setting

Nowadays, the number of Cloud providers and their offered services is rapidly growing. Consequently, a huge number of **heterogeneous** services are available. This diversity makes manual searching and selection of suitable services meeting users functional requirements time-consuming and challenging. In fact, each Cloud provider offers a wide range of resources and

diverse type of services which can respond to the same functional requirements. This leaves users in the agony of choice and lead to a steep documentation to compare one Cloud services to another to find the most suitable service responding a functional requirement. Added to that, it requires a high level of user expertise to make educated and suitable decisions. Cloud services are regularly evolving, and to handle this evolution during the selection stages, it is known to be an error-prone process and a complex problem. Thus, it is crucial to assist Cloud users during their search and selection for suitable Cloud services meeting their functional requirements.

1.4 Existing Solutions

Some interesting literature has dealt with Cloud services search and selection.

1.4.1 Cloud Services Market

We present "Cloud Services Market" N[12] that provide a comprehensive Cloud services directory and provide a search engine of the Cloud services. the figure 1.1 present an overview of the website.

⋆ **Advantages :**

✓ The website provide a large number of Cloud services provider.

⋆ **Disadvantages :**

✓ The collection of the cloud services is done manually and is not up to date. For example, only six services from amazon web service were cited when actually AWS offers 144 services

✓ The Cloud services are not arranged into classes which can make the user's search easier.

✓ The search provided by the website is not efficient. For example if we put storage as a key word we get as a result only one Cloud service but in reality there is much more storage services.

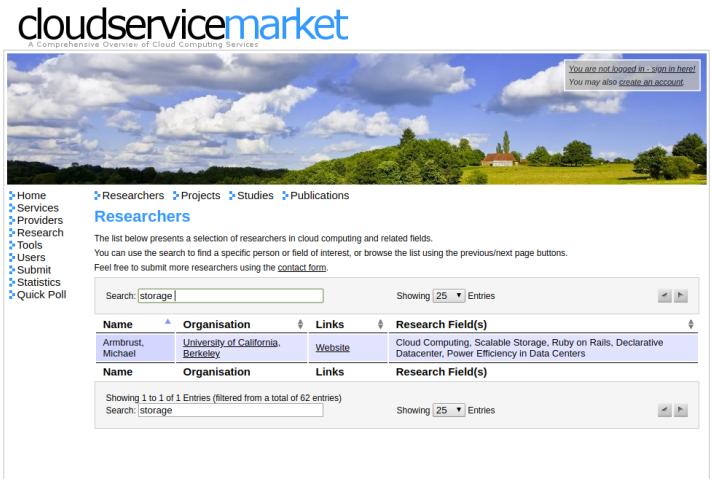


Figure 1.1: Cloud service market overview [N12]

1.4.2 Serchen

Also we found another website entitled "Serchen" N[15] that provides also important Categories of Cloud services in addition to a Cloud search engine the figure 1.2 presents the home interface of the website.

⋆ **Advantages :**

- ✓ The website provides a large number of Cloud services.
- ✓ All the Cloud services are defined and classed into many categories according to their functionality.
- ✓ The Cloud search engine is meaningful and efficient.

⋆ **Disadvantages :**

- ✓ The website doesn't provide an automatic collection of the Cloud services.
- ✓ We can't found the services related to the Cloud leading the market provider such as AWS, Google, IBM, etc.

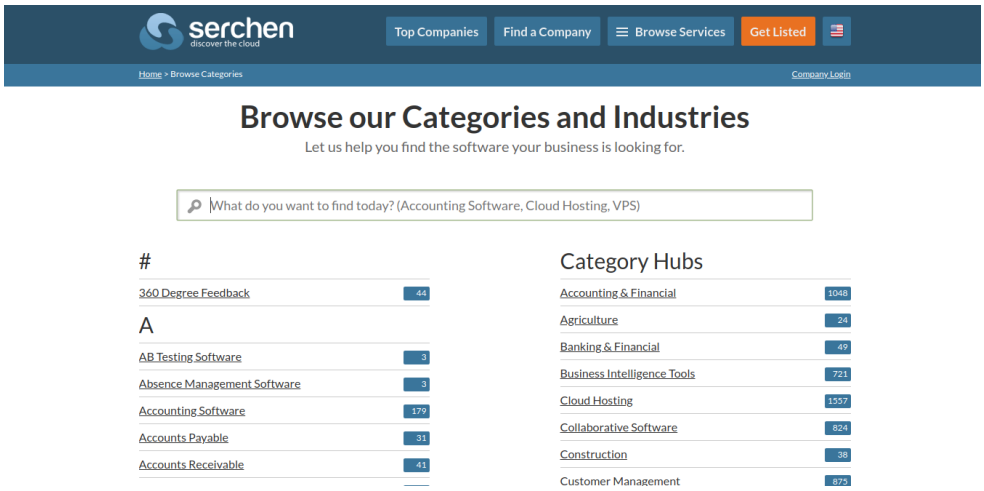


Figure 1.2: Serchen home page

1.5 Proposed solution

In this work, we propose, a Cloud services search engine "CSOOGLE" aiming to assist Cloud users in searching and selecting suitable Cloud services. We focused on user's functional requirements as the search criteria. In fact, the user can introduce key words related to his functional requirements such as; storage, data, etc. Our Cloud services search engine is based on the ranking function "Okapi BM25" which is basically a TF-IDF enhanced function. It will propose for the user the Cloud services that perfectly match with his key words. In order to have satisfactory results and enhance our Cloud services search engine's performance, we decided to not only collect and unify Cloud services meta-data from various cloud providers essentially, Amazon, Google, IBM and Oracle, but also, we propose a clustering approach inspired from machine learning strategy in order to bundle cloud services that can satisfy the same functional requirement together. To do so ,we proceeded as follows:

- **Data collection:** We put forward a new strategy for Cloud services collection and storage which can flexibly manage the evolving nature of services. In fact, we created an intelligent Cloud service registry which can automatically deduce the appearance add new Cloud service ond update.
- **Data processing:** In order to get meaningful and clean data we used the natural language processing technique; we will explain it further in the third chapter.
- **Data modeling:**In order to convert the description of the services into a vector space

model we used both TF-IDF and Doc2Vec algorithms.

- **Decision making and Evaluate Algorithms:** In this stage, we unified the existing Cloud services classes proposed by Amazon, Google, IBM and Oracle, then We used K-means algorithm for clustering Cloud services which do not have the same name of class between diverse providers.

Finally, we develop a user friendly platform offering assisted search of Cloud services owing to our Cloud services search engine with the possibility for rating the used services. The figure 1.3 presents the major purpose of our work.

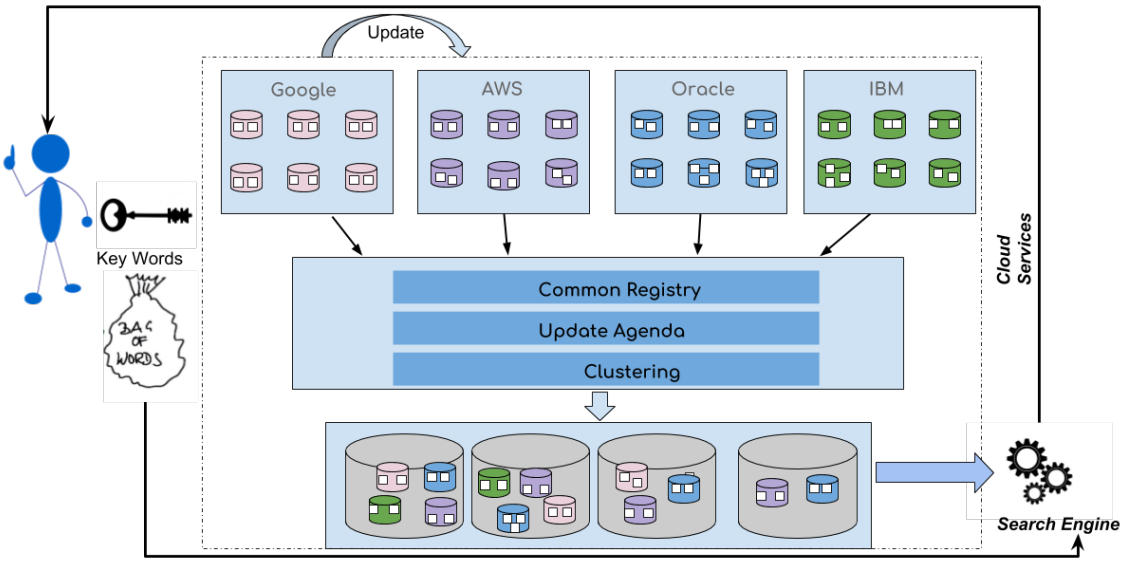


Figure 1.3: Classes Unifying

1.6 Working Methodology

In order to reduce the complexity of our project and share the important tasks, our group members fits the main reason to choose the agile methodology. In this section, we will present the agile methodology as well as the scrum methodology that we chose to organize and limit every week's work and assignments.

1.6.1 Agile Methodology

Agile methodologies are defined as a sort of software development process that is based on four core based outlined in the Agile Manifesto as:

- ✓ **Individuals and interactions** over processes and tools.
- ✓ **Working software** over comprehensive documentation.
- ✓ **Customer collaboration** over contract negotiation.
- ✓ **Responding to change** over following a plan [N1]

The most popular agile methodologies are Scrum and Kanban.

1.6.2 Scrum

In software development scrum is an agile framework for managing work by devising it into actions which is already completed within a time boxed iterations called sprints. Scrum starts with a user story in order to specify actions and requirements. In the figure 1.4, we illustrate a work-flow of scrum methodology.

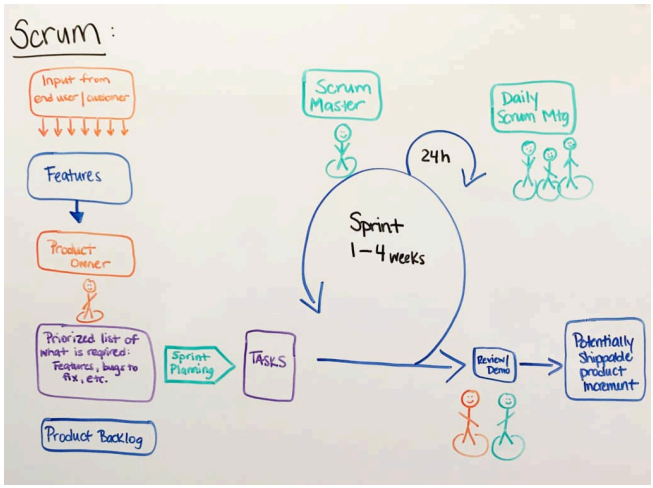


Figure 1.4: Work-flow of scrum methodology [N2]

1.6.3 Kanban

Kanban is a framework that used to implement the agile method. It was developed in 1940 by Toyota in order to optimize its engineering process and increase their productivity. The major difference between scrum and kanban is that scrum is a time based framework unlike kanban which is based solely on priority. Kanban also is specialized by a large prioritized to-do list. requirements in Kanban are tracked by their current stage in the process (to-do, doing , done).

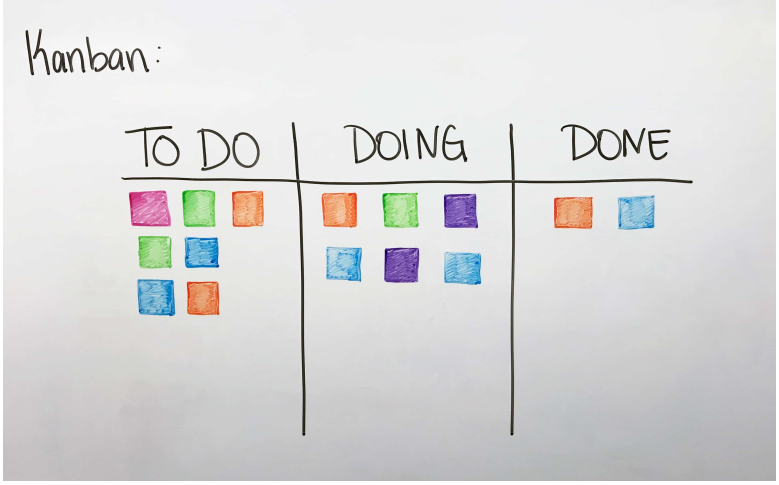


Figure 1.5: kanban to-do list [N2]

1.6.4 Chosen methodology

By matching the amount of work we have and the team capacity, we decide to choose the kanban method, "since there are fewer planning meetings, this approach means the team needs to be extremely close."[N1]. Below the detailed advantages of the kanban method:

- Flexibility in planning: A Kanban team is only focused on the work that's actively in progress. Once the team complete a task, they move on for the next one described in the list. We don't need a fixed-length iterations like scrum.
- Efficiency through focus: Our development team is composed only from three members, so we notice that by focusing on one task the result would be perfect cause multitasking kills efficiency the more work items in flight at any given time , the more context switching, which hinders their path to completion.

Having chosen kanban as a development methodology, we arrange a weekly meeting. During each meeting we understand the project-progress as well as the issues. we discuss tasks that were done those are in progress and what is left to do. We used also an online to-do list provided by a website entitled "Trello" the figure 1.4 present a screen-shot of our list.

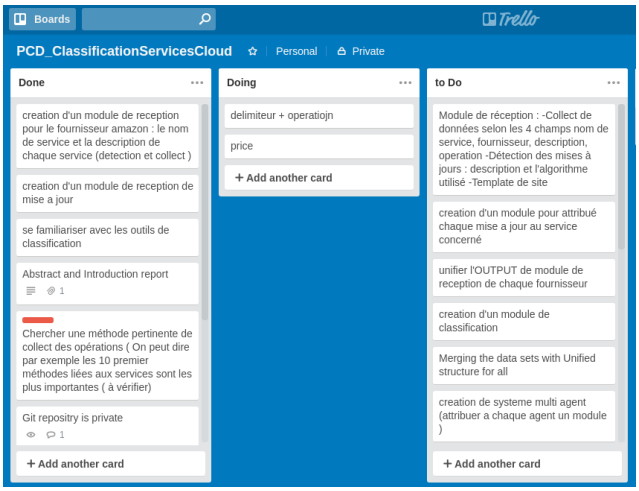


Figure 1.6: An example of to do list of our project

Conclusion

This chapter included a presentation of our project and an overview of the problem setting. Moreover we present the existent solutions and we highlighted their main limits. Finally we present a brief comparison between two main agile methodologies in order to adopt the suitable one for our case.

Chapter 2

Theoretical Study

In this chapter we will start by defining some concepts related to our work as well as the machine learning strategy that we followed.

During this section we will present the used techniques during our work by defining each one of them.

2.1 Text Classification

Text classification is the assignment for each document to a class according to their content using machine learning and deep learning algorithms in order to make this task automated.

In this section we will present the latest text classification recommended techniques. We explain each step in order to justify the chosen algorithm for our case. We distinguish two types of text classification which are **supervised** and **unsupervised** text classification.

2.1.1 Supervised text classification:

In the supervised text classification, the input is a set of a labeled documents from which we will train machine learning model in order to predict and assign the correct label for a new given document. It finally consists of labeling new text documents with their category. The figure 2.1 will present the working pipeline for a supervised text classification problem. In the context of the supervised text classification we need to have a known number of classes , and its based on a training set which is a tagged data. It is used to classify future observation.

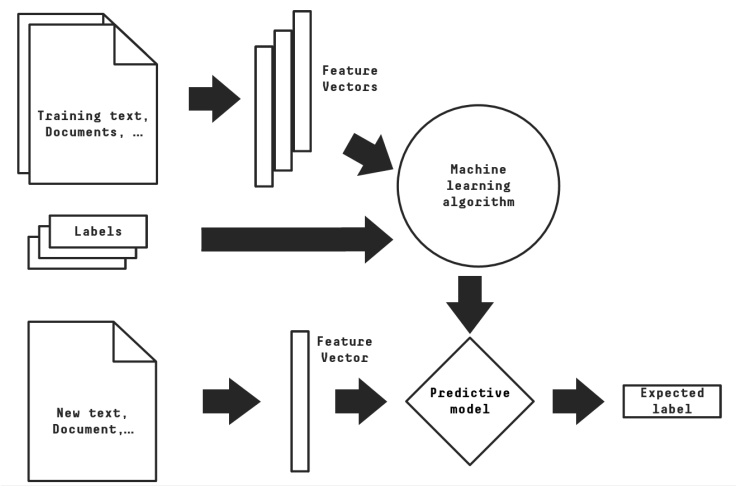


Figure 2.1: Text classification pipeline [B1]

We remind that we will use the text classification in order to regroup the cloud services classes from all the cloud provider based on the classes description, so the supervised text classification does not fit as a good solution for our work that's why we choose the unsupervised type.

2.1.2 Unsupervised text classification:

In the unsupervised technique the number of classes is unknown , we have no prior knowledge and it s used to explore the data we have so in our case we adopted the strategy of the unsupervised text categorization. The figure 2.2 shows the main steps for the unsupervised text classification. We explain in detail each step.

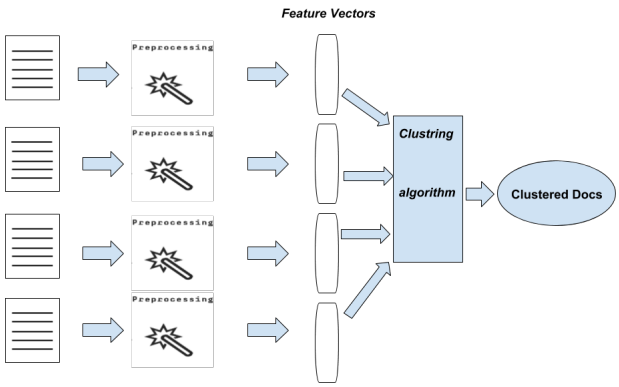


Figure 2.2: Unsupervised Text classification main steps

- **Step1 text preprocessing techniques:** Text preprocessing can be defined by cleaning the data and get the essential part of it. The natural language processing techniques fits in this context.

  - **Tokenizer:** According to the Stanford Natural Processing group,N[5] "A tokenizer divides text into a sequence of tokens, which roughly correspond to words". There is several types of Tokenizer let us quote; Standard Tokenizer, Letter Tokenizer, Lowercase Tokenizer, Whitespace Tokenizer, Classic Tokenizer. N[6]

  - **Stop words:** stop words are words that have no significant meaning in the sentence so we need to remove them from the description. We can also append the list of stop words by adding the words that don't serve to get the best meaning of the description like in our context the word cloud. The figure 2.3 will present the stop words of the English language provided by NLTK [3] which is a python library that provide some natural language processing techniques

```
>>> nltk.corpus.stopwords.words('english')
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "
you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'hi
m', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'i
ts', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which',
 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are',
'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'doe
s', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as',
 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between
', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from
', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',
'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'b
oth', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not'
, 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', '
just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',
've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doe
sn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "
isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'sha
n', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won
', "won't", 'wouldn', "wouldn't"]
```

Figure 2.3: English stop words

  - **Stemming:** For defining stemming, let's begin with an example, the word compute have many variation like computing, computation, computer, etc. But in our work we don't need to differentiate the word and its variation we only need the **root word** that what stemming is made for , "Stemming, in literal terms, is the process of cutting down the branches of a tree to its stem. So effectively, with the use of some basic rules, any token can be cut down to its stem. Stemming is more of a crude rule-based process by which we want to club together different variations of the token. " [N16] the figure 2.4 will present the result of stemming the words

computing , computer and computation.

```
>>> import nltk
>>> sno = nltk.stem.SnowballStemmer('english')
>>> sno.stem('computing')
'comput'
>>> sno.stem('computer')
'comput'
>>> sno.stem('computation')
'comput'
>>>
```

Figure 2.4: Stemming result

- **Step 2 Text modeling techniques:** During this subsection we will present the used technique to transform our text into a feature vector.
  - **Tf-Idf:** Tf-Idf stands for term frequency in-versed document frequency or this algorithm give a weight for each word in sentence, the most important weight is given to the rare words that generally are the most important and for the more often word they have a low weight. The figure 2.5 will present the algorithm's equation

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Figure 2.5: TF-IDF equation N[7]

  - **Word2Vec:** In order to well define wor2vec let begin by presenting some concepts. First Let's remind that we want to have a vector for each document word2vec build a vector for each word which is called a word vector. At one level the word vector was build by the "one hot encoding" just putting one in the position of the word in the vocabulary. For example, suppose our vocabulary has only five words: King, Queen, Man, Woman, and Child. We could encode the word 'Queen' as:

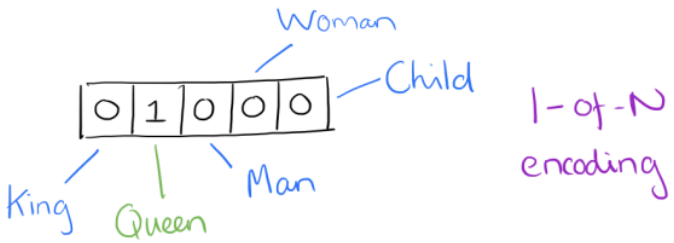


Figure 2.6: one-hot encoding N[8]

This encoding seems too simple because it does not represent any semantic meaning of the word. it only represents the existence of the word in the vocabulary. Unlike Word2Vec that represent the word-vector by a distribution of weights across the vocabulary elements but it requires several hundred words that will represent the dimension of the vector.

Figure 2.7: Word2Vec distributed representation N[8]

Such a vector gives the real representation of the word meaning, it is even possible to notice some relation between vector and get excellent result but let's see first how word2vec build such a vector. In fact word2vec build the vector in two steps. Because of the large number of words the size of the one hot coding vector will be giant so the first step is to reduce the dimension of the vector. The second step is to calculate the probability of appearing in a context given using Softmax function. The context's words are called "the bag of words" and this method is called the continuous back of words. the figure 2.8 explain more the process.

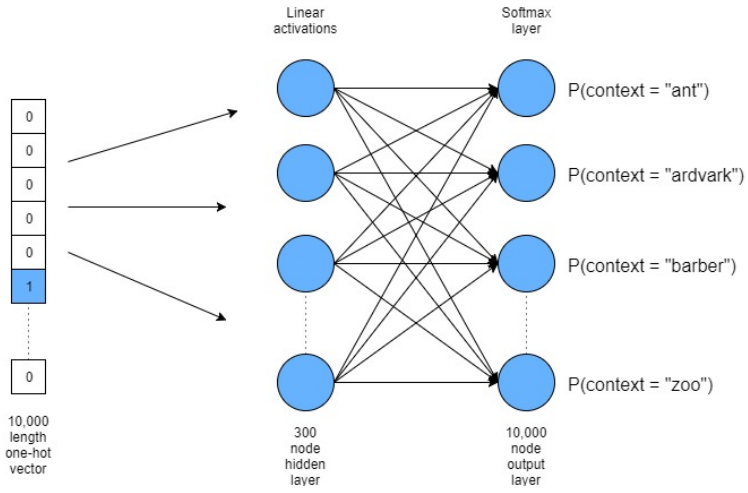


Figure 2.8: Word2Vec distributed representation N[9]

– **Doc2Vec:** After presenting Word2Vec, it will be easier to introduce Doc2Vec, we can simply notice from the name of the algorithms that in this stage we are trying to build a significant vector for the hole document regardless of its length. Doc2Vec uses the same techniques that Word2Vec do but it takes into consideration the document. So, when training the word vectors W, the document vector D is trained as well, and in the end of training, it holds a numeric representation of the whole document[N9]

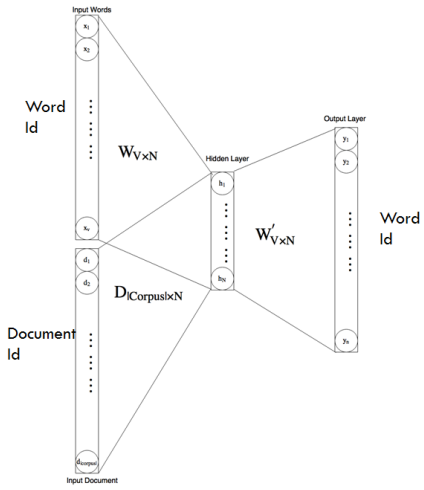


Figure 2.9: Doc2Vec distributed representationN[10]

2.2 Clustering

According to Mr.Bijuraj: "clustering is the task of grouping a set of objects in such a way that objects in the same group (called cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters)." [B2] In this section we will present the most well known clustering algorithm which is K-means.

2.2.1 K-means

K-means is algorithm for clustering a given dataset into k disjoint cluster, where k is a given positive number. We will now analyze the different step of this algorithms

- Firstly, we select the number of classes and we initialize on a random way K point that will represent our data the centroids.
- Secondly, the algorithm calculate and store the distance between every data point and the k centroids usually k-means uses the euclidean distance for this step which is not significant in our case so the solution was to overwrite this method in the algorithm and change the euclidean distance by the cosine similarity.
- In the third step, the k-means algorithm assign each point to its nearest centroid.
- During the fourth step the algorithm update the centroid by calculating the mean of all the point in the group.
- The last step will decide to iterate the algorithm or no based on the update of the centroid position. If the centroid position changed we repeat the process from the second step until the position of the centroid stays the same.
  The figure 2.10 represents the whole process of K-means.

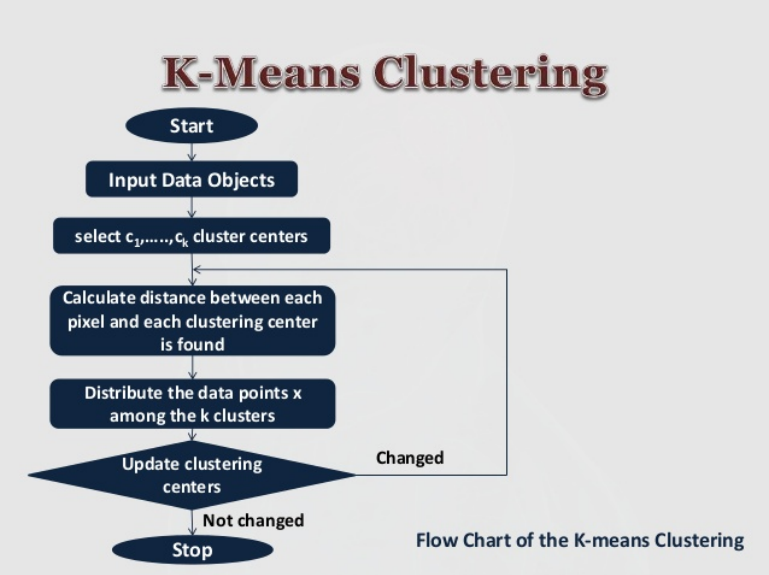


Figure 2.10: k-means process N[11]

2.2.2 Elbow method

The major disadvantage of k-means algorithm is related related to the need to know the number of clusters from the beginning which can be a real problem in our case because we don't know how much clusters we may have. Moreover, our tool will collect cloud services and may collect new classes from the cloud provider catalogs. The Elbow Method is invented to find an optimal k. The idea of this method is to run k-means for different values of k in range then calculate the sum of the square distance between the cluster centroid and the points of the cluster. We call this sum the Sum Square Errors (SSE). We plot the function f(k)=SSE, we pick the k which present the most changing behavior of the function. The two figures 2.11 in which we present a random points and 2.12 that shows the plot of the function will make the elbow method clear.

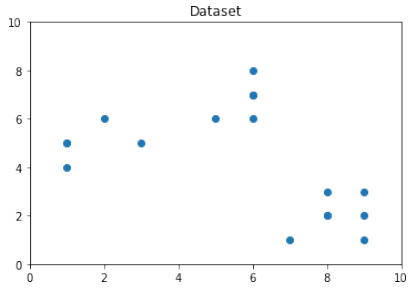


Figure 2.11: Sample of data set

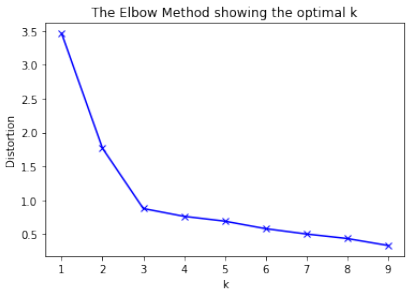


Figure 2.12: Elbow method

We notice a real sudden changing behavior of the function for k equal to three so the optimal

k for this example is three.

Conclusion

This chapter went thought a theoretical study of the algorithms that we used during our project. In the next chapter we will enumerate the different features offered by our search engine.

Chapter 3

Requirements Analysis and Specification

Through this chapter, we aim to analyze and present our project in a formal way. We will begin by presenting our actors then the functional requirements of the project followed by the non functional ones finally we show the project's principal scenarios using system sequence diagrams.

3.1 Requirements Analysis

3.1.1 Actors

The actors of our system are mainly **users of Cloud services** who can use our tool to facilitate the search and selection of Cloud services while staying up to date through automatic notifications of service updates.

3.1.2 Functional Requirements

⋆ **Access Cloud Services Classes**

The platform must give the user the right to :

- ✓ View the description of the Cloud service classes proposed.
- ✓ Add feedback on the classification given by the system.

⋆ **Effect a personalized search for Cloud services**
The platform should enable the user to do these kind of search:

- ✓ Search by Service Class.
- ✓ Search by service quality criteria .
- ✓ Consult the description of the classified services (name, supplier, version, operations, inputs, outputs, etc.).

⋆ **Get notifications**
The system should be able to notify the user when:

- ✓ A new services is added by aa cloud services provider.
- ✓ An update of a service took place.

⋆ **Create a basket of the favorite services**
The system must enable the user to create a basket in which he can :

- ✓ Add a service in the basket.
- ✓ Delete a service from the basket

3.1.3 Non functional requirements

The non functional requirement are the specific criteria that are beyond the purpose of the application , they don't describe specific behaviors and they are used generally to judge the system operation and ensure a high quality of service.

⋆ **Performance** With each update or addition of service by the cloud service provider the system must detect and notify the customer in a time which does not exceed two days.

⋆ **Ergonomy** The application must offer a user-friendly and ergonomic interface that can be used by the user by considering all possible interactions on the screen of the support held.

⋆ **Reliability and Viability** The application must be functional regardless of any circumstances that may surround the user.

⋆ **Maintainability** The application code must be legible and understandable in order to ensure its evolutionary state and extensible in relation to market needs.

3.2 Requirement Specification

3.2.1 Use Case Diagrams

In this part we illustrate the use case diagram which can summerize the user interaction with the system that shows the relationship between the user and the different use cases:

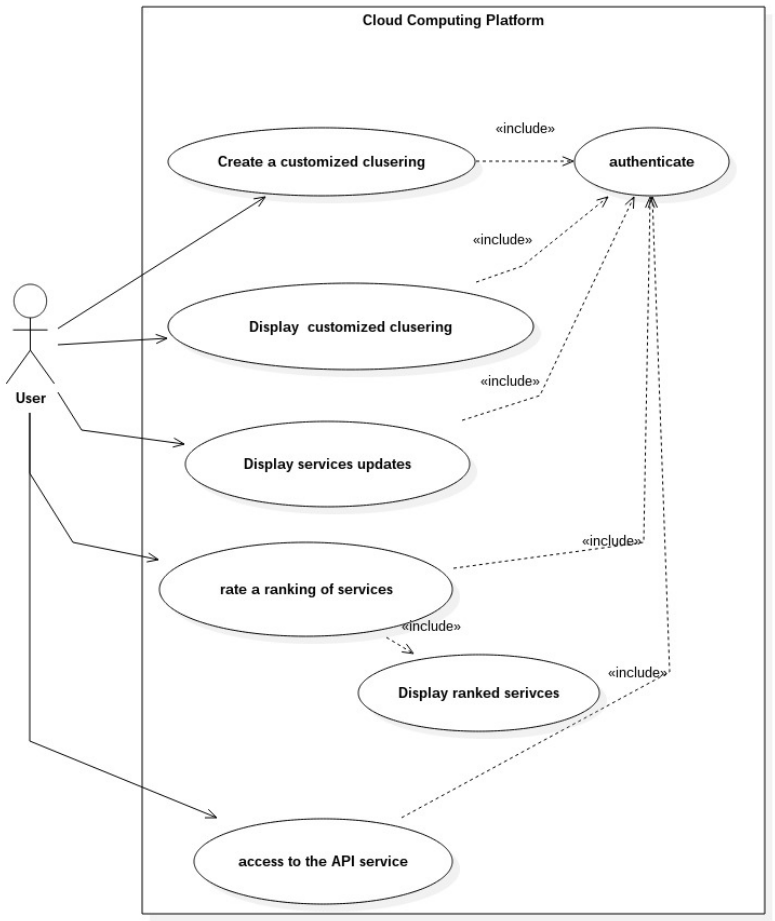


Figure 3.1: Use case diagram

3.2.2 System Sequence Diagram

In this section , we present the nominal sequence diagram that shows the interaction with the user and the different component in our system from a temporal point of view . This scenario is illustrated by Figure 3.2 .

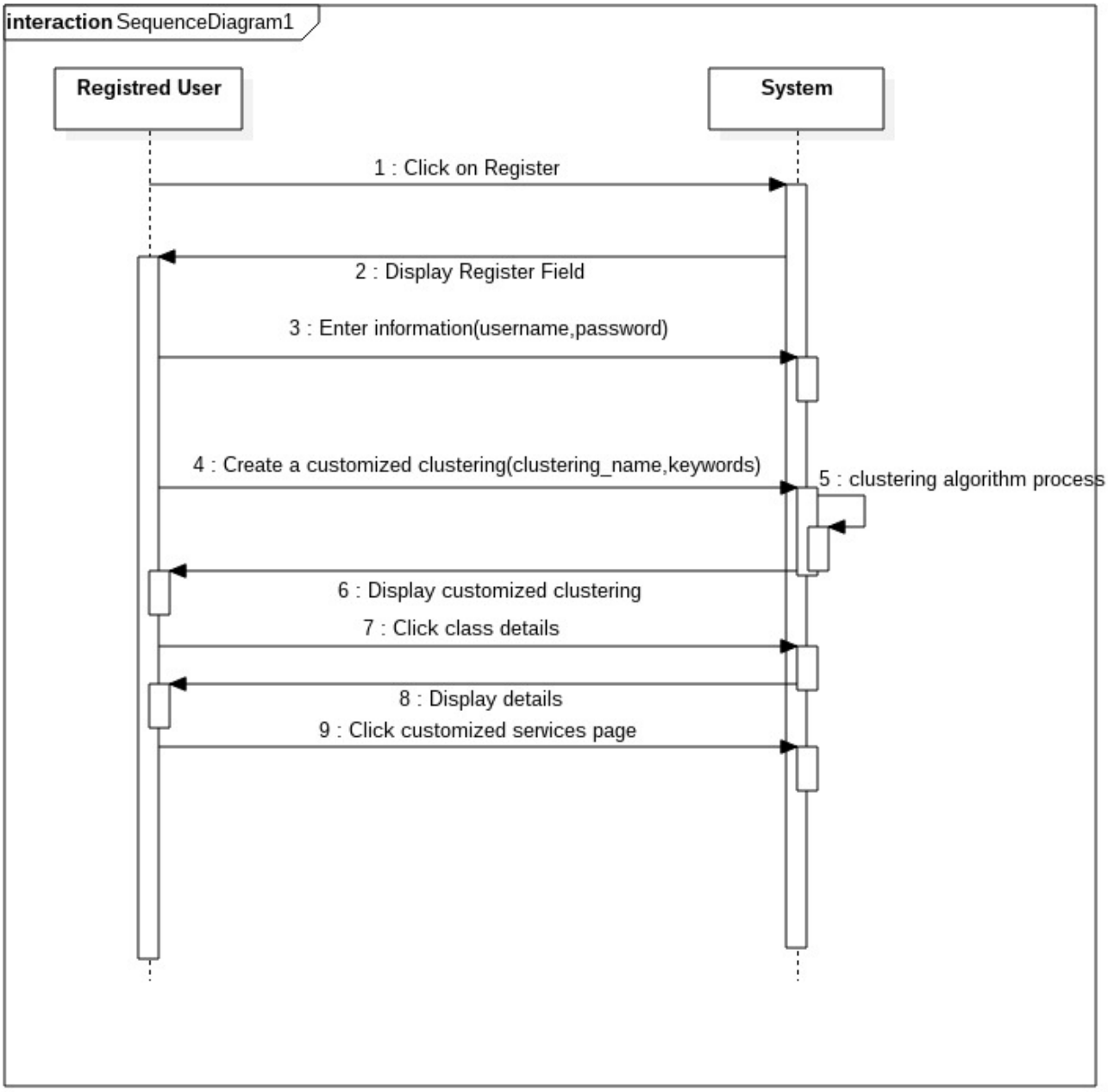


Figure 3.2: Sequence diagram of the use case"Create a customized clustering "

First , the user create an account in order to access to the different services provided by our platform , after registration, the user redirects to the page which allow him to create a customized clustering, he put the keywords of interest in his field of work and which the clustering will be based on. In addition the user select the classes that find interesting and launch the clustering algorithm by clicking "submit" button. After that the user is redirected to the personal classes page where he will find the selected classes.

Conclusion

Throughout this chapter, we identified the actors, specified the requirements of our platform and we gave the use cases and the nominal scenarios of our project .The next chapter aims to go a step further in the process of developing the platform via presenting the project design.

Chapter 4

Design

In order to achieve the appropriate result described in the requirements analysis and specification, In this chapter we will focus on the main architecture and the details of its components.

4.1 Global design

4.1.1 Physical architecture

In this section, we explain the physical architecture of our application. It's the 3-tiers architecture(see figure 4.1). What characterizes this physical architecture is that the three layers of presentation, treatment and data are developed and maintained as independent modules on separate devices. These three layers are respectively client tier, Web server tier, and database server tier. In this context, we should know that the client tier can only send requests to, and receive responses from the web server. The client is not allowed to directly access to database server. The middle-ware HTTP protocol assures this connection between the client and the web server. In the other hand, the web server tier is allowed only to send requests to, and receive responses from the database server. This communication is assured by Django ORM.

What made us opt for this choice is the ability of this architecture to provide multiple access to the application in a secure and safe way. In addition, it makes the system run faster through loading the pages.

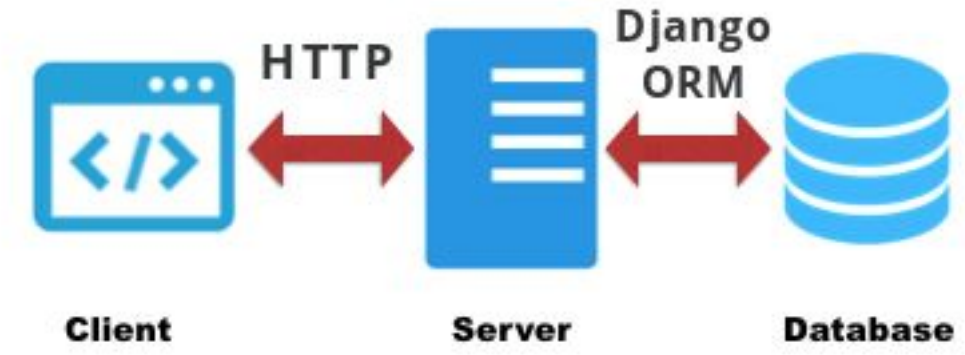


Figure 4.1: 3-tiers architecture

4.1.2 Logical architecture

The framework Django follows MVT which is a design pattern closely build on model-view-controller basics. So first, we have to introduce you to MVC. The MVC design pattern is on top of the logical architecture. It is a widely adopted pattern, across many languages and implementation frameworks, whose purpose is to achieve a clean separation between the three components which are the model, the view and the controller. However, MVT used by Django framework, uses its own logic in the implementation, so there is a difference of the nomenclature of it's components. By analogy with MVC, these three components are respectively the model , the template and the view. So we can observe a nomenclature difference according to each layer and its job because here, the view of MVT is associated to the controller in MVC, while the controller of MVC is associated to the view in the framework Django. Now we are going to describe the role of these three components:

- **Model:** It's the data access layer. Here we describe: Forms the data has, how to access data, how to validate it and the relationships between the data.
- **View:** It's the control logic layer. This layer is responsible for the coordination by containing the logic that accesses the model and manage the appropriate template.
- **Template:** It's the presentation layer, responsible for how something should be displayed on a Web page. .

We should notice that any change in the model may be reflected in the template and changes

done by the user while interacting with the presentation layer may be reflected back on the model. Here, the view layer acts as a coordinator and is responsible for keeping the model and the template connected. This is an over-simplified definition of the MVT shown in the figure 4.2.

In our application, the model contains all the data of: Cloud services, classes of Cloud services, unified classes, and data related to users. The templates contains files used for building the front-end of our application while the view is the coordinator between these to sides. It's responsible for providing the appropriate response to every request related to the application. As shown in the figure, The "External Scripts" part, is also an independent logical component of our logical architecture. It's going to be explained in the third section of this chapter.

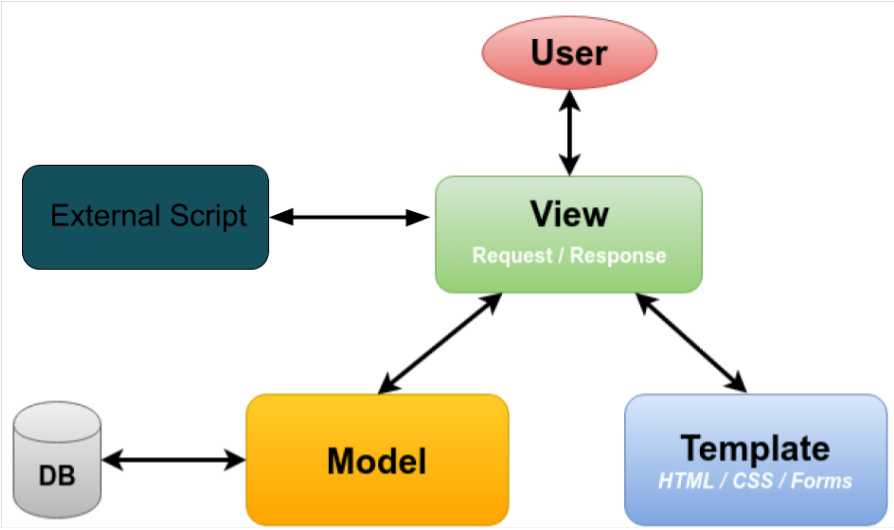


Figure 4.2: MVT architecture

4.2 Detailed design

In this section, we explore more deeply the design of our application. In the first part, we describe the class diagram presenting our database models and relation between classes. While in the second part we show the detailed sequences diagram presenting the whole process of our platform.

4.2.1 Class Diagram

The figure 4.3 is a formal representation of our relational database.

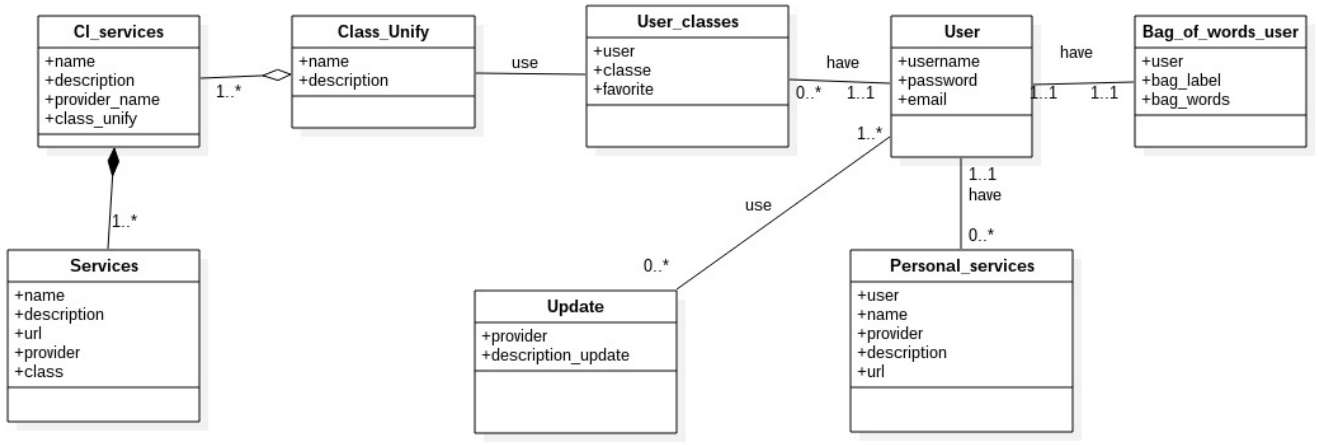


Figure 4.3: Class diagram

4.2.2 Sequences Diagram

This sequences diagram gives a detailed account in words of the communication between different components.

- **Registered user:** It's the simple Cloud user who access to our platform and enjoys its services
- **Website:** It's the front where we have displayed all information and features.
- **Web server:** Manages everything, contains all information, receive requests from Web site and provides responses. Call external scripts
- **External services:** It's the external scripts, including algorithms of modeling and classification, It's called to be executed by the web server .

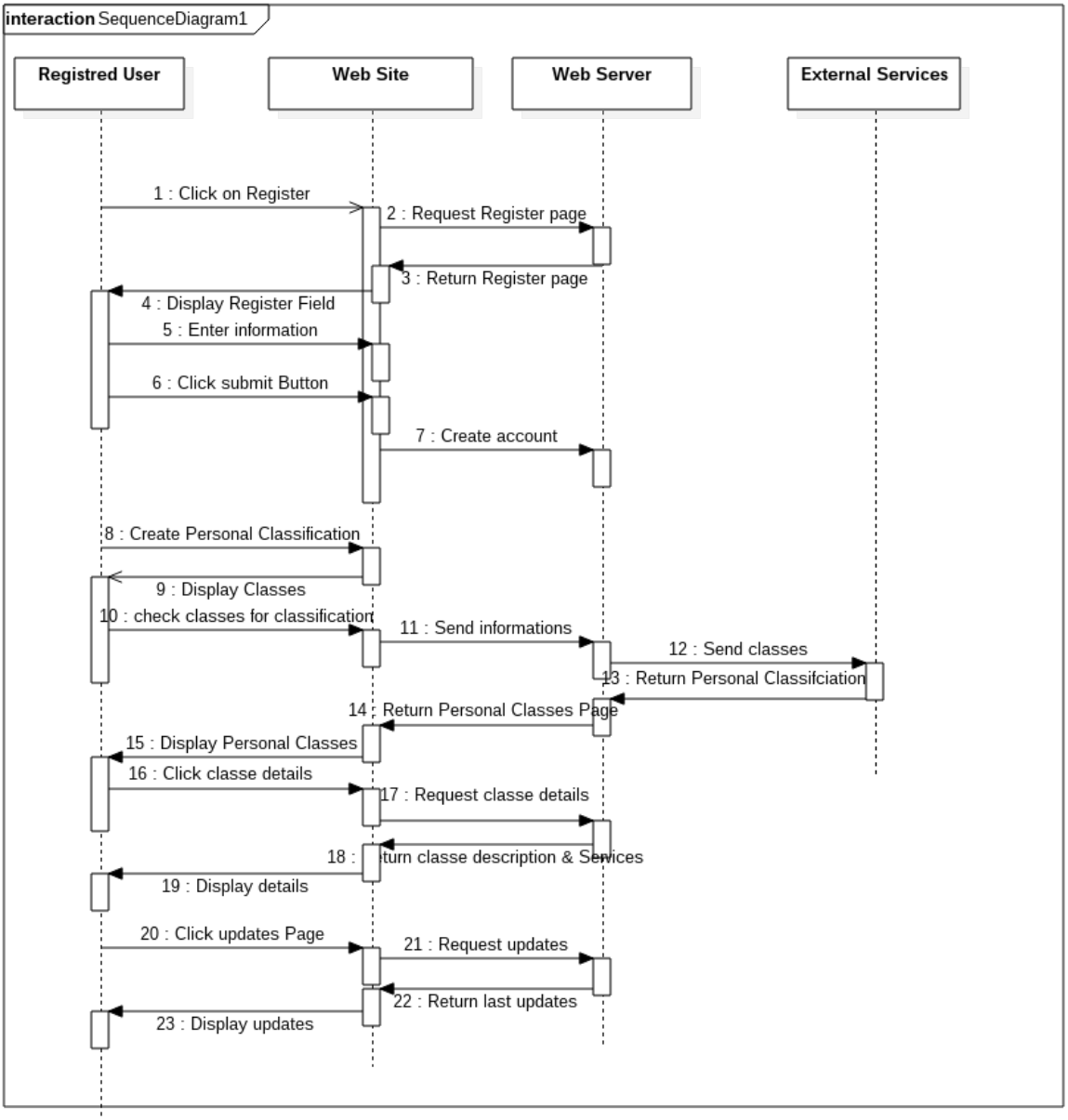


Figure 4.4: Sequences diagram

4.3 External Scripts

First, what we mean by an "External script" is a script created by us, which is not included in MVT design of Django. These scripts are dissociated from the web application, but are in interaction with it. They are called to be executed in order to get output that we need in our application.

Now we have build our web application with Django, and it's perfectly working. But we faced some other problems, we have a lot of data of Cloud services that we must include into our database. We can not do it manually in the shell for two reasons: The first is that we have a lot of data, so it's convinient to do it that way. The second is that we want to automate our system. Another problem we faced is how to call the algorithms of classification written with python using the input from our database to run them and get the appropriate classification, and display the result in the front-end of our application.

To do it, it's quite simple: we have just to redirect the script we want to execute in the shell of Django. It's as easy as the framed command in that capture (figure 4.5). The result of the script is shown in the terminal.

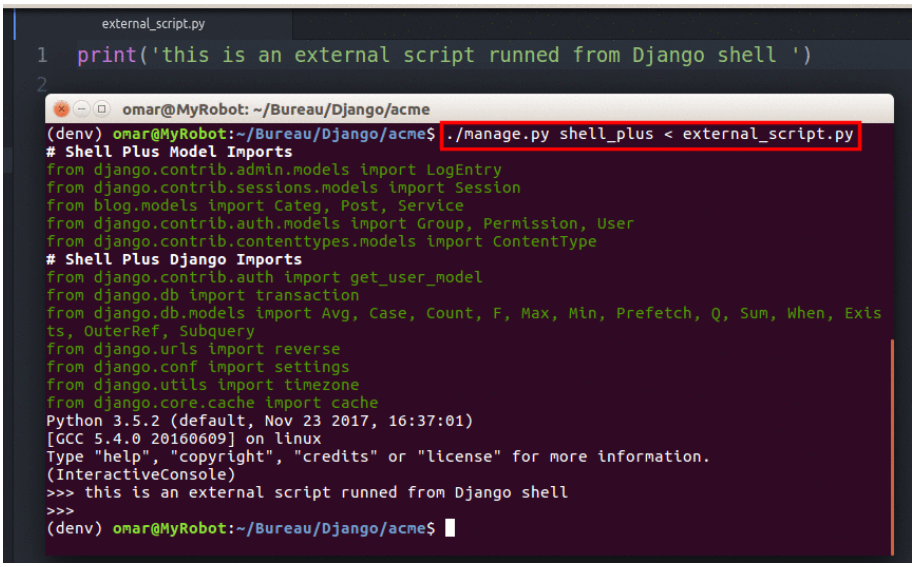


Figure 4.5: Example of running a script

Now we have to clarify some other issues in the capture to make things more understandable in this operation:

- Manage.py: In Django , every operation in the terminal is in fact done by adding an argument to the script manage.py executed in the shell terminal of Django.
- Shell-plus: In this case we use shell-plus in place of shell because it automatically generates all imports we may need for our script used to manage our application. These imports are shown in the capture

In the following part, we will introduce you to all types external scripts called in our Web application with an explanation of why we have used each one. We will just recite the role of them. Details are given in chapter 5.

- Data-Collection scripts: The data provided in different Cloud providers must be collected using python scripts and assembled in a unified csv file. The python script doing this job is written and called to be executed to get this file of data.
- Data-Insertion scripts: Now having our csv file containing all information we need about Cloud services, we should enter the appropriate columns to the data base. This is the main reason for using this script.
- Classification-algorithms scripts: Classes of Cloud services from different providers must be unified to make them well displayed in the front. So we implemented algorithms of classification. The output is unified classes of Cloud services
- Other scripts: We have also needed some scripts to make some changes in the form or the structure of data during the development. Such as adding an attribute to a table or some operations to clean data.

Chapter 5

Achievement

This chapter will detail each sprint made during our work and present during the sprints our implementation and contribution.

5.1 Sprint1: Data collection

The major data science task is how we can get the data and where we can found it. In order to achieve our work we need to build our own database that contain from each Cloud provider its classes , description of each class of services also we need to get the services of each class and the service's description. The newest technique for those kind of problems is "web scraping" so we write for each Cloud provider official website a python-script that extract the data we need and we make these scripts scheduled in order to be executed each period, to notice the if a new services is added by each provider. These figures will present the official websites that we have scraped for each Cloud provider.

The figure 5.1 presents the official website of AWS.

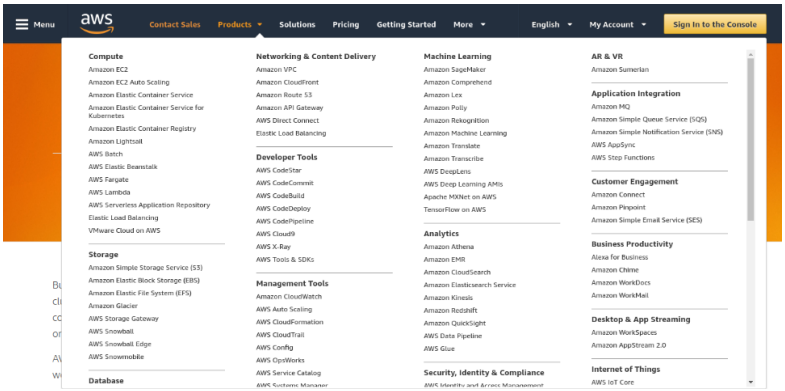


Figure 5.1: AWS products

The figure 5.2 presents the official website of IBM Cloud serivices.

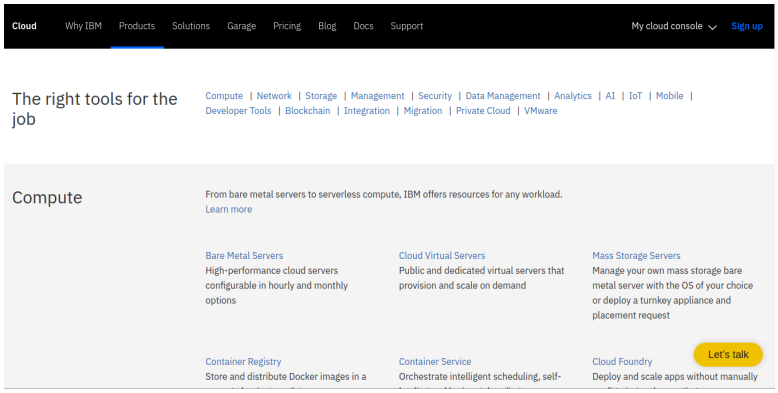


Figure 5.2: IBM products

This figure 5.3 presents the official website Cloud services products for Google.

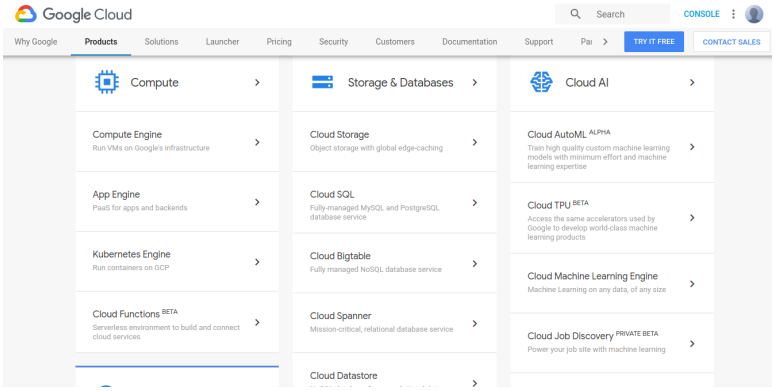


Figure 5.3: Google products

The last website is for oracle and it is presented in the figure 5.4

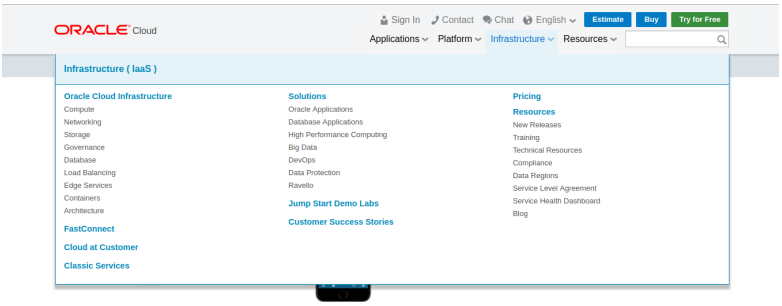
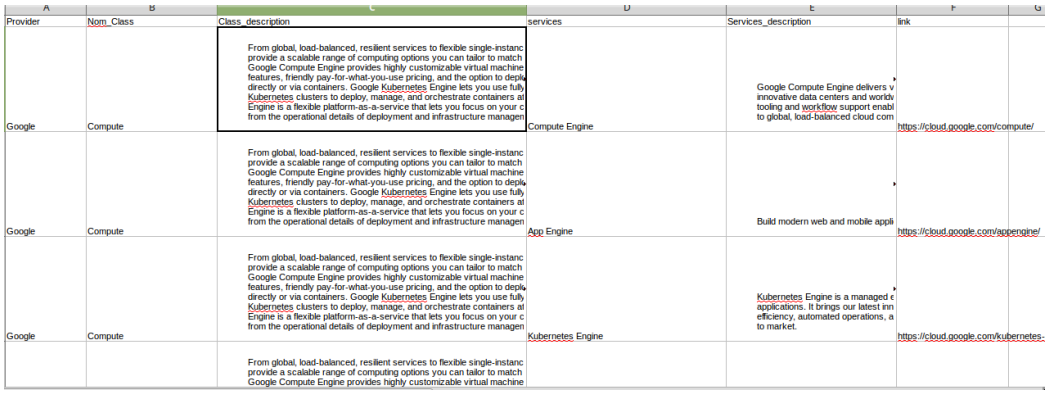


Figure 5.4: Oracle product

After extracting and understanding each website's HTML source code, we finally ended by getting the data we need and we build our own dataset.The picture 5.5 and 5.6 present an overview of our dataset after scraping both google and amazon websites.

Nom_classe	Classe_Description	Services	Service_Description	Lien_desc_Service
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	Amazon EC2	Amazon Elastic Compute Cloud (Amazon EC2) is a web service that pro	https://aws.amazon.com/ec2/?nc2=h_m1
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	Amazon EC2 Auto Scaling	Amazon EC2 Auto Scaling helps you maintain application availability and	https://aws.amazon.com/ec2/autoscaling/?nc2=h
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	Amazon Elastic Container Service	Amazon Elastic Container Service (Amazon ECS) is a highly scalable, hi	https://aws.amazon.com/ecs/?nc2=h_m1
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	Amazon Elastic Container Service for Kubernetes	Amazon Elastic Container Service for Kubernetes (Amazon EKS) is a ma	https://aws.amazon.com/eks/?nc2=h_m1
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	Amazon Elastic Container Registry	Amazon Elastic Container Registry (ECR) is a fully-managed Docker con	https://aws.amazon.com/ecr/?nc2=h_m1
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	Amazon Lightsail	Everything you need to jumpstart your project on AWS?Compute, stora	https://aws.amazon.com/lightsail/?nc2=h_m1
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	AWS Batch	AWS Batch enables developers, scientists, and engineers to easily and e	https://aws.amazon.com/batch/?nc2=h_m1
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	AWS Elastic Beanstalk	AWS Elastic Beanstalk is an easy-to-use service for deploying and scalin	https://aws.amazon.com/elasticbeanstalk/?nc2=h
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	AWS Fargate	AWS Fargate is a technology for Amazon ECS and EKS* that allows you	https://aws.amazon.com/fargate/?nc2=h_m1
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	AWS Lambda	AWS Lambda lets you run code without provisioning or managing servers	https://aws.amazon.com/lambda/?nc2=h_m1
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	AWS Serverless Application Repository	The AWS Serverless Application Repository enables you to quickly deploy	https://aws.amazon.com/serverless/serverlessrep
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	Elastic Load Balancing	Elastic Load Balancing automatically distributes incoming application traff	https://aws.amazon.com/elasticloadbalancing/?nc
Compute	Virtual Server Hosting, Container Management, and Serverless Computing.	VMware Cloud on AWS	VMware Cloud on AWS is an integrated cloud offering jointly developed by	https://aws.amazon.com/vmware/?nc2=h_m1
Storage	A reliable, scalable, and secure place for your data	Amazon Simple Storage Service (S3)	Amazon Elastic Compute Cloud (Amazon EC2) is a web service that pro	https://aws.amazon.com/s3/?nc2=h_m1
Storage	A reliable, scalable, and secure place for your data	Amazon Elastic Block Storage (EBS)	Amazon EC2 Auto Scaling helps you maintain application availability and	https://aws.amazon.com/ebs/?nc2=h_m1

Figure 5.5: Amazon data



Provider	Nom_Class	Class_description	services	Services_description	link
Google	Compute	From global, load-balanced, resilient services to flexible single-instanc provide a scalable range of computing options you can tailor to match Google Compute Engine provides highly customizable virtual machine features, friendly pay-for-what-you-use pricing, and the option to deplo directly or via containers. Google Kubernetes Engine lets you use fully Kubernetes clusters to deploy, manage, and orchestrate containers at Engine is a flexible platform-as-a-service that lets you focus on your c from the operational details of deployment and infrastructure managen	Compute Engine	Google Compute Engine delivers v innovative data centers and worldv tooling and workflow support enabl to global, load-balanced cloud com	https://cloud.google.com/compute/
Google	Compute	From global, load-balanced, resilient services to flexible single-instanc provide a scalable range of computing options you can tailor to match Google Compute Engine provides highly customizable virtual machine features, friendly pay-for-what-you-use pricing, and the option to deplo directly or via containers. Google Kubernetes Engine lets you use fully Kubernetes clusters to deploy, manage, and orchestrate containers at Engine is a flexible platform-as-a-service that lets you focus on your c from the operational details of deployment and infrastructure managen	App Engine	Build modern web and mobile appli	https://cloud.google.com/appengine/
Google	Compute	From global, load-balanced, resilient services to flexible single-instanc provide a scalable range of computing options you can tailor to match Google Compute Engine provides highly customizable virtual machine features, friendly pay-for-what-you-use pricing, and the option to deplo directly or via containers. Google Kubernetes Engine lets you use fully Kubernetes clusters to deploy, manage, and orchestrate containers at Engine is a flexible platform-as-a-service that lets you focus on your c from the operational details of deployment and infrastructure managen	Kubernetes Engine	Kubernetes Engine is a managed e applications. It brings our latest inn efficiency, automated operations, a to market.	https://cloud.google.com/kubernetes-
		From global, load-balanced, resilient services to flexible single-instanc provide a scalable range of computing options you can tailor to match Google Compute Engine provides highly customizable virtual machine			

Figure 5.6: Google data

Now we have the dataset with the following schema; provider name, class name, class description, service name, service description. But we have a data set for each provider still not merged. In order to merge our data especially the classes of the four providers ,we analyze our data from each provider, we notice two major things. On the first hand, there is some classes from each provider that have exactly the same name such as compute, storage, etc... On the other hand there are some classes that seem to be different by name but if we get into their descrption we conclude that they serve the same meaning and they provide the same kind of service.So the idea was to build a model that understand the semantic of the Cloud provider's classes and merge those which are related.

5.2 Sprint2: Data processing

Our main goal now is to merge the classes that have the same meaning together so we need to prepare our data before building the model. It's the step of data processing where we need to clean our data and get the essential part from the class's descriptions. The natural language processing techniques detailed in the chapitre 3 fits in this context and helped us to clean our data by several techniques;

- **Stop words**

  After reading the description of the classes, we conclude that it contains too much words which don't serve to get the meaning of the class.Therefore, so we charged the stop-word list from the nltk library and we added some specific words such as the providers name. Moreover we used a specific regular expression to get only the words and filter out all the

other stuff such as numbers and any words that do not contain letters. The figure 5.7 presents our implementation.

```
In [1]: from copy import deepcopy
        import numpy as np
        import pandas as pd
        import re
        from matplotlib import pyplot as plt
        plt.rcParams['figure.figsize'] = (16, 9)
        plt.style.use('ggplot')
        import csv
        f = open('data.csv','r')
        reader = csv.reader(f)
        class_name=[]
        description=[]
        for i in reader:
            class_name.append(i[0].lstrip())
            description.append(re.sub("[^a-zA-Z]", " ", i[1]))
        import nltk
        stopwords = nltk.corpus.stopwords.words('english')
        stopwords.append('Google')
        stopwords.append('Cloud')
        stopwords.append('cloud')
        stopwords.append('googl')
        stopwords.append('aws')
        stopwords.append('ibm')
        stopwords.append('oracle')
```

Figure 5.7: Stop words implementation

- **Tokenizer and Stemmer**

  The second step of the data process part was to tokenize the description first by sentence, then by word. The figure 5.8 will explain how we did it.

```
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer("english")
def tokenize_and_stem(text):
    tokens = [word for sent in nltk.sent_tokenize(text) for word in nltk.word_tokenize(sent)]
    filtered_tokens = []
    for token in tokens:
        if re.search('[a-zA-Z]', token):
            filtered_tokens.append(token)
    stems = [stemmer.stem(t) for t in filtered_tokens]
    return stems
```

Figure 5.8: Tokenize and stem the descriptions

- **Enriching the descriptions**

  Adding synonym of the words in the description to the description make the model work easier in matching classes together. To do so, we used the babelnet python api to get synonyms of words in our description.The figure 4.9 present some synonyms of the word management.

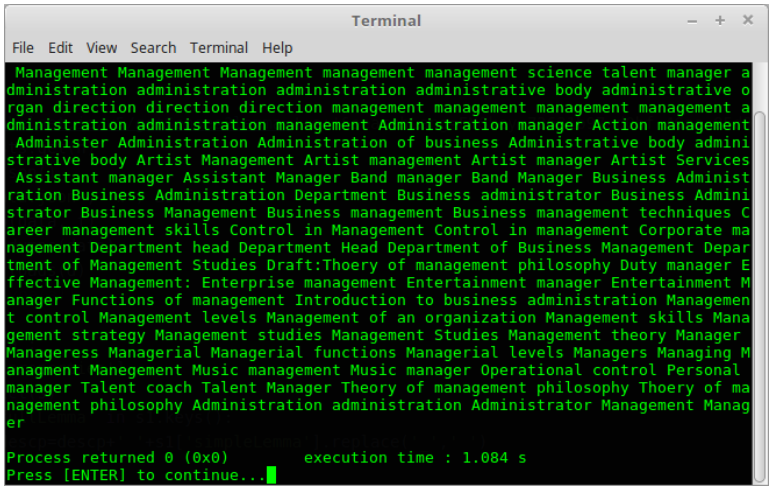


Figure 5.9: Synonyms of management by babelnet

5.3 Sprint3: Data modeling

Now we have the data clean and ready to use we must build a model that classify the descriptions according to their semantic meaning. We introduced in the third chapter the major used algorithms for word embedding through this section we will present the result of each model. The output of the two algorithms will be a Matrix where each column present the vector of a document. We present below the matrix of each algorithms.

$$\begin{bmatrix} -4.44089210e-15 & 9.98702712e-01 & 1.00000000e+00 & \dots & 9.95304468e-01 \\ 9.98702712e-01 & 4.99600361e-15 & 1.00000000e+00 & \dots & 9.75437291e-01 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 9.95304468e-01 & 9.75437291e-01 & 1.00000000e+00 & \dots & 9.99200722e-16 \end{bmatrix}$$

TF-IDF matrix

$$\begin{bmatrix} 3.52390075e+00 & -9.78218198e-01 & -1.18786240e+00 & \dots & -8.79712641e-01 \\ -5.98013774e-02 & -1.39411044e+00 & 1.20198894e+00 & \dots & -9.15879428e-01 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2.00525784e+00 & -3.64801846e-03 & -2.34065913e-02 & \dots & -5.21133393e-02 \end{bmatrix}$$

Doc2Vec matrix

5.4 Sprint4: Decision making and Evaluate Algorithms

At this stage, we have our data modeled by two algorithms; TF-IDF and DOC2VEC. We will now achieve our major goal by clustering and putting classes that they have a similar meaning together using K-means. In order to get an efficient result we proceed by three steps;

- The first step is to implement and automatize the elbow method. We implement an algorithm that run the elbow method on our data then it plot the elbow function and deduce automatically the optimal K. As we mentioned before the optimal K fits with the point that represent the important variation in the elbow function. Our solution was to calculate the angle of each point in the plot using the dot product and the determinant between two vectors, the dot product will present for us the cosine angel and the determinant will introduce for us the direction of the two vectors the vectors in this stage are defined by the points of the plot. The figure 5.10 presents the plot of the elbow method where the data was the class description and detect the optimal k from the plot.

- The second step is to run K-means on the output of the two modeling algorithms but here we notice a problem. K-means used the an euclidean distance to measure the difference between points which is not suitable for our case. We will get deeply into details but let's first define the category of distance measure. According to Mr Jure Leskovec there is two major classes of distance measure;

  ✓ Euclidean: An Euclidean space has some number of real-valued dimensions and "dense" points. There is a notion of "average" of two points. A Euclidean distance is based on the locations of points in such a space.

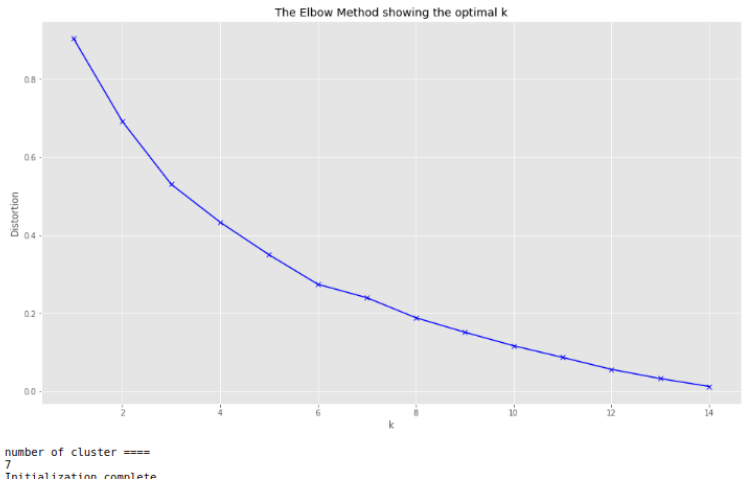


Figure 5.10: Automatic Detection for the optimal K

✓ Non-Euclidean: A Non-Euclidean distance is based on properties of points, but not their "location" in a space.[B3]

K-means uses the euclidean distances which is not significant for our data so we decided to use the cos similarity distance to measure the similarity between docs. It s more significant, The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. Cosine Similarity will generate a metric that says how related are two documents by looking at the angle instead of magnitude, like in the examples below [N13]:

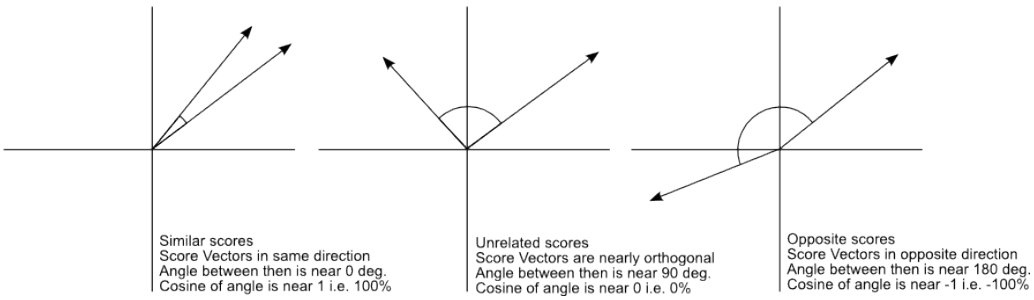


Figure 5.11: Cos similarity presentation

So we redefined the measure distance of k-means and change it from euclidean distance to cos similarity then we plot the data clustered.The figure 5.12 and 5.13 present the result of the clustering after using tf-idf algorithm and doc2vec.

Figure 5.12: Clustering result after TF-IDF

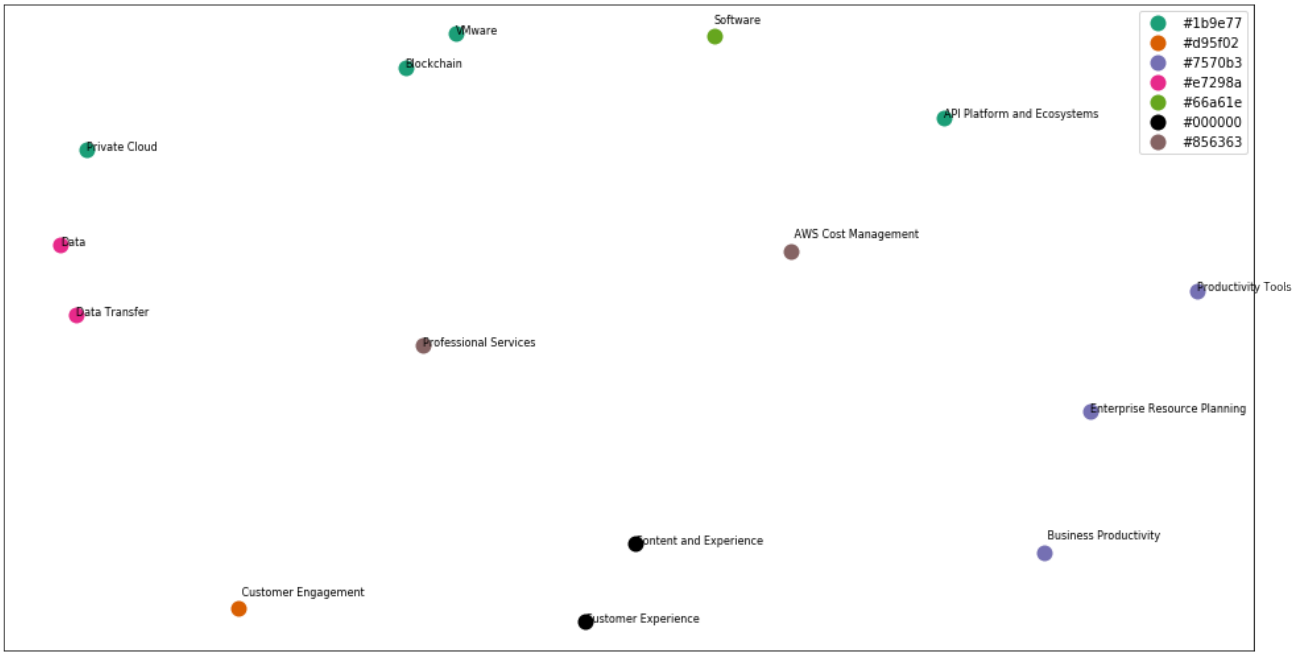


Figure 5.13: Clustering result after doc2vec

5.5 Sprint5: Cloud Services Search Engine

The major purpose of our work was to enable the user to make a meaningful search for the suitable Cloud services by giving some keywords. Our solution was to use a customized search engine for this task. We used Whoosh which is a fast, pure Python search engine library. we specified a schema that fits with our data then we considered the default ranking function which is " Okapi BM25F " [N14] the figure 5.14 will represent the ranking function.

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \left[\frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} + \delta \right]$$

Figure 5.14: whoosh ranking function

5.6 Sprint6: Search engine platform

5.6.1 Development

In this part, we will present the whole operation of building the application with Django. In all files of this framework we write entirely in python. We will explain every phase of this sprint.

- **Object-relational mapper:** In models.py we have defined all our data models with the native programming language python. To access the database, we can use SQL if we need to, But Django offers a dynamic API to access database for free which is very easier to use. The following capture(Figure 5.12) shows all our data models which are mapped to a relational database.

Figure 5.15: Database Models

- **URLs and views :** Focusing on the character of high-quality, we were careful on building significant URLs to each page of the application. In urls.py we build a simple mapping between URL patterns and the views by calling the appropriate function from views.py for every url designed(such as shown in the figure 5.13). This function is the responsible of displaying the related page and it can on its turn call a page from templates as the example shown in the figure where we also notice that we can instantiate objects we are going to use.

Figure 5.16: Relation URLs/Views

- **Templates :** Certainly we need a beautiful front to our application. With Django, we are free to use any front-end languages we need. So we have build our front with HTML, CSS and Bootstrap. In this case, the most important thing offered by Django is that we can write a base page containing features we need to get displayed in all pages and argument it in every case the main content we need. So it's not necessary to write the whole code for every page of the front. This is an example from our work(framed field in figure 5.14)

Figure 5.17: Our Base.html

- **Forms :** Due to the dynamic character of our website, we have to manage forms which are all kind of data submitted by user such as informations related to a new signed up user or key words submitted to get appropriate Cloud services etc.. . For this, Django provides a way to generate object instantiated from our models and containing these submitted informations. .

5.6.2 Interfaces

Now we will describe the interface offered to a Cloud user by showing captures of different pages of our application.

- **Authentication interface:**

  This interface is preliminary, it allows users to access the platform and take advantages of the services provided, the user must enter the username and the password. In the figure 5.18 we illustrate the interface page.

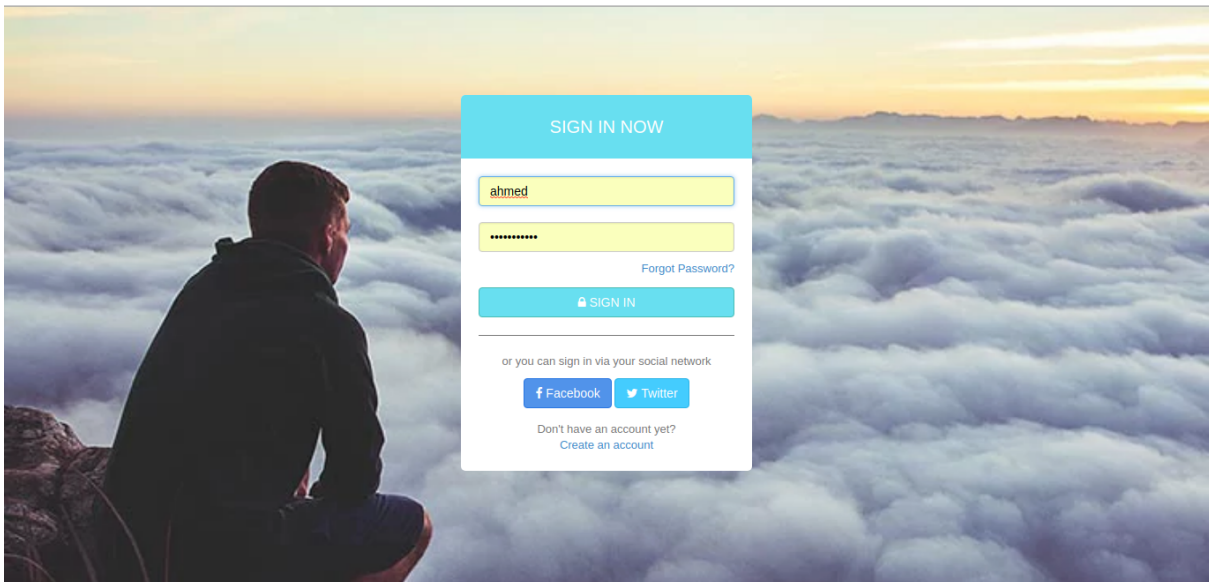


Figure 5.18: Login interface

- **Start page :**

  When the user authenticated the following Start page appears on the browser. Through this page the user creates his customized classification by entering the keywords and selecting the classes that seems interesting and creates a label for this classification. The figure 5.19 shows the layout of this interface.

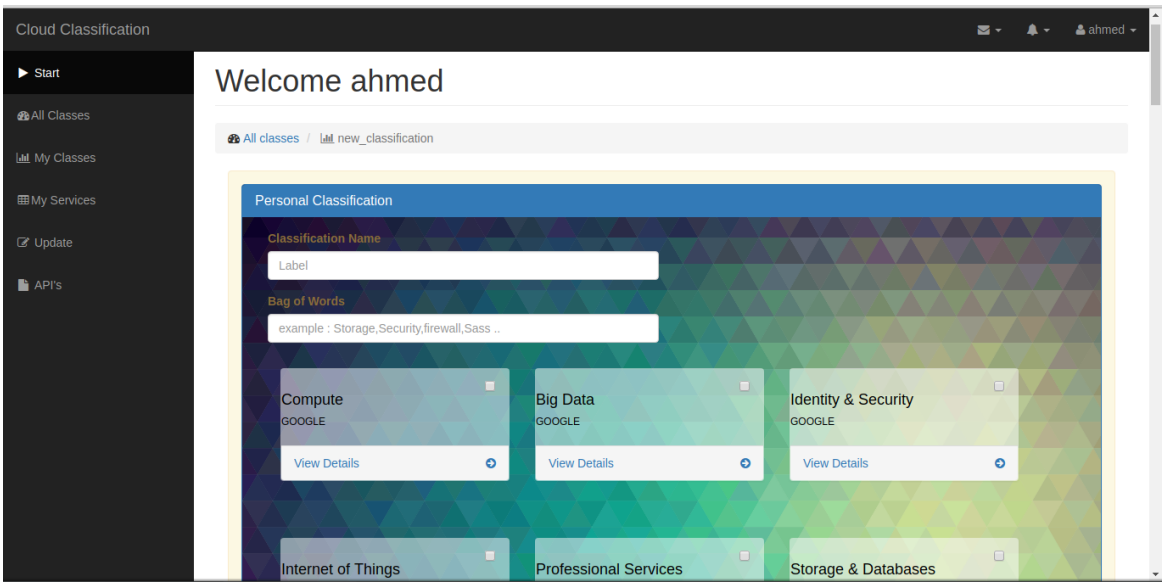


Figure 5.19: Start interface

- **My services page :**

  Once the user submits the chosen classes, he is redirected to this interface. This page generates the services ranked by user interests and allow him to rate this ranking. The figure 5.20 illustrates this page.

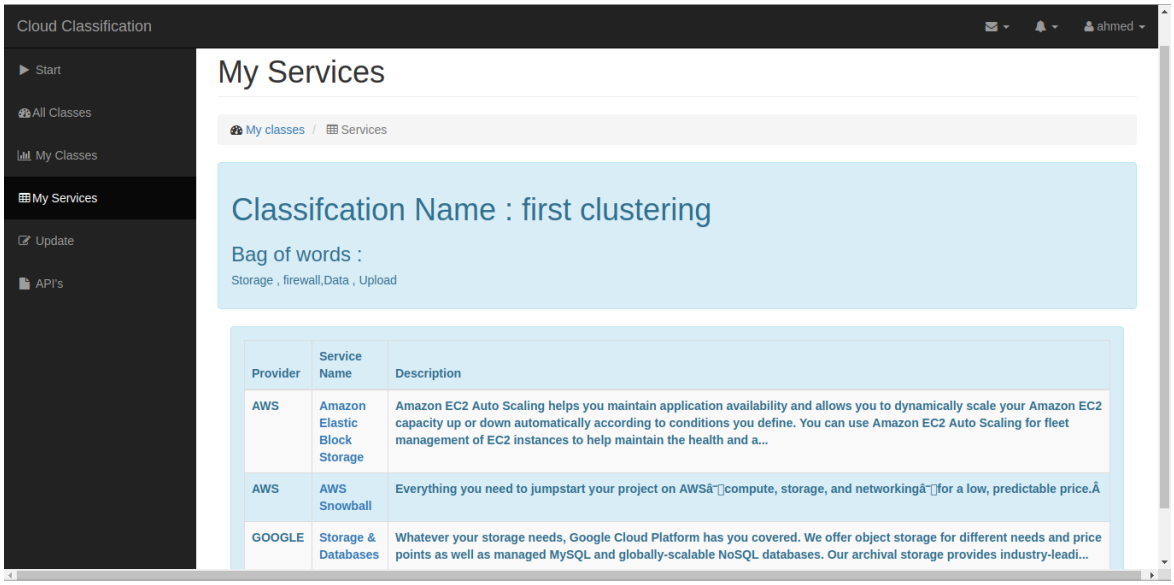


Figure 5.20: My services interface

- **Classes pages :**

On these pages the users can access to all classes in the database in order to know more details about the class selected and to view the services belongs. The figure 5.21 and figure 5.22 shows these interfaces.

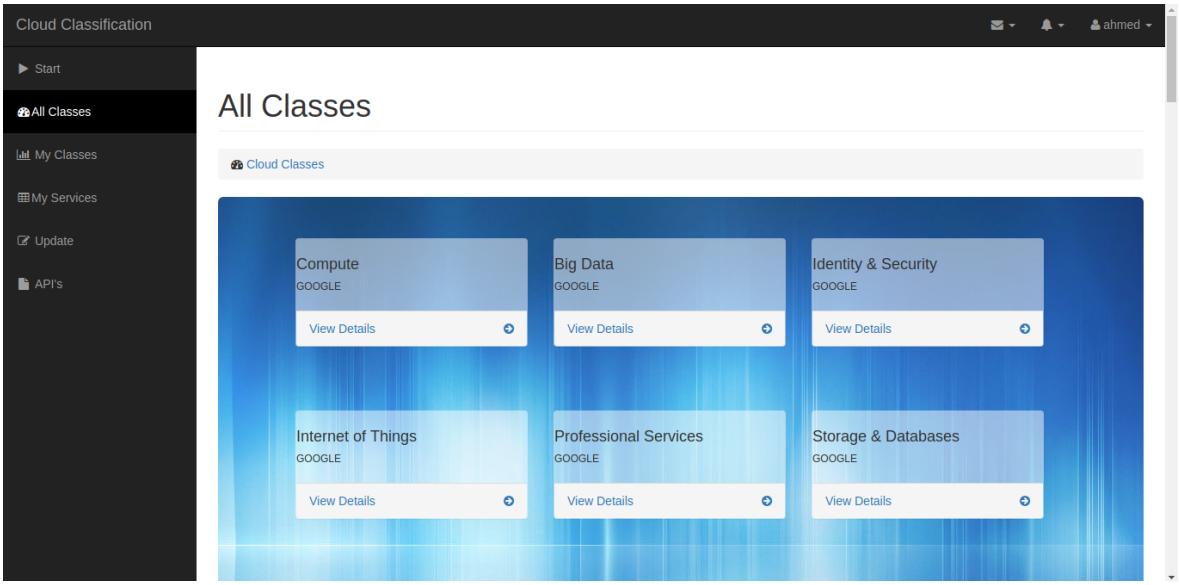


Figure 5.21: Classes interface

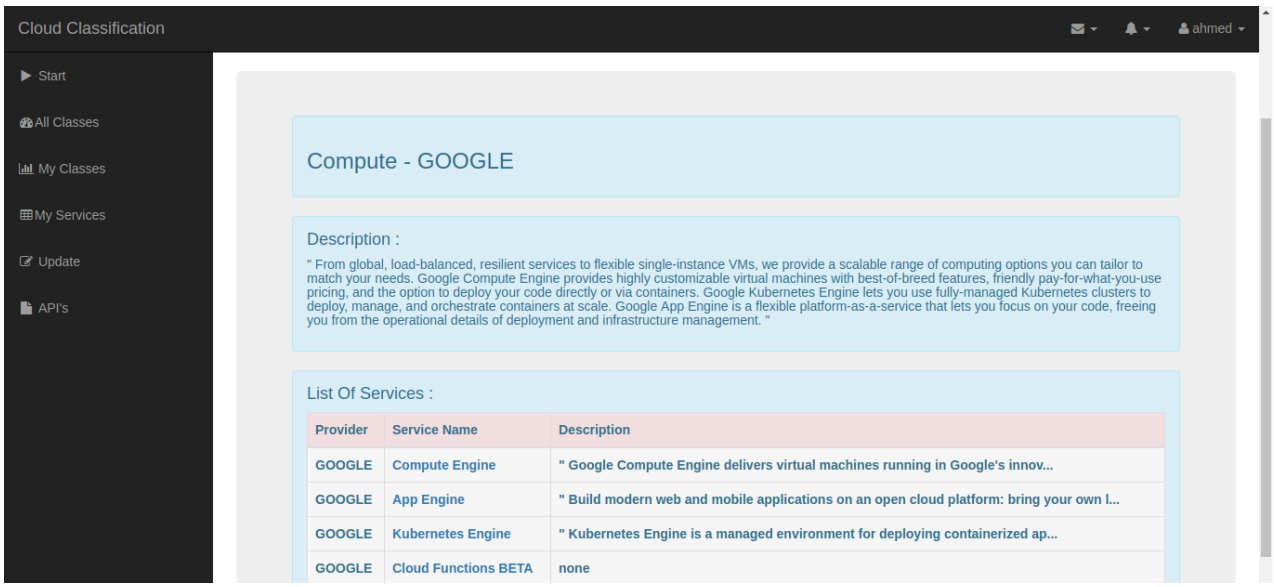


Figure 5.22: Class details interface

- **My classes pages :**

In this interface the user can access the selected classes in the start page, he can also rate it and access the details of these classes (description , services , links to the services pages).

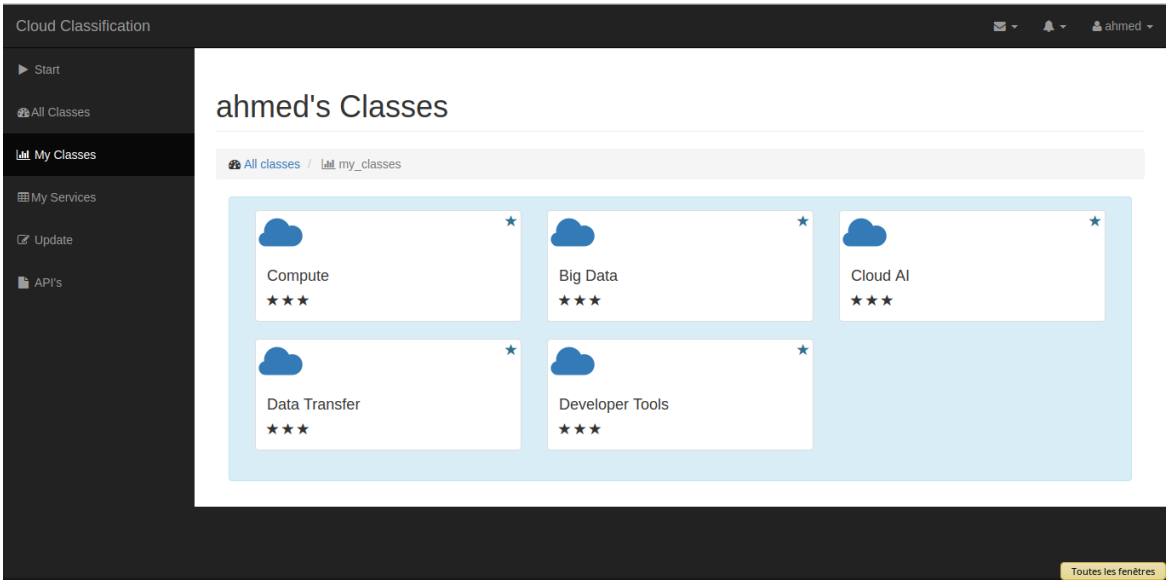


Figure 5.23: My classes interface

- **Updates page :**

  This interface shows the user the latest updates for our providers of Cloud services.

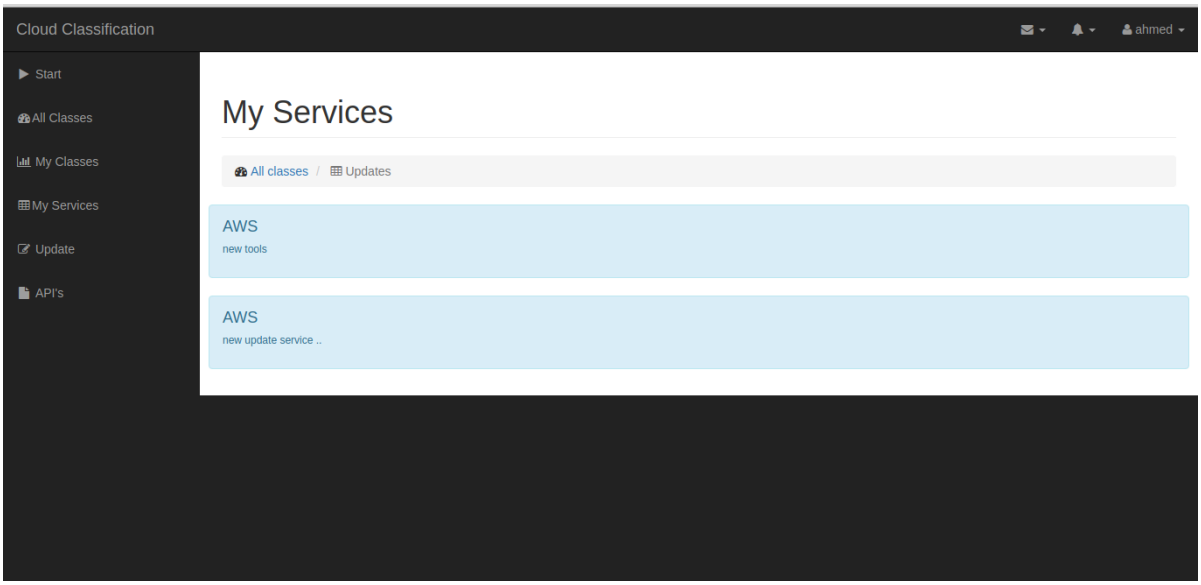


Figure 5.24: Updates interface

- **API's page :**

  In order to offer more flexibility to our features, we have build a REST API for our clients to allow them to access the service classes by repealing a JSON response that contains all services details. They can use it on console mode or for creating another features. The figure 5.25 and 5.26 shows this interface.

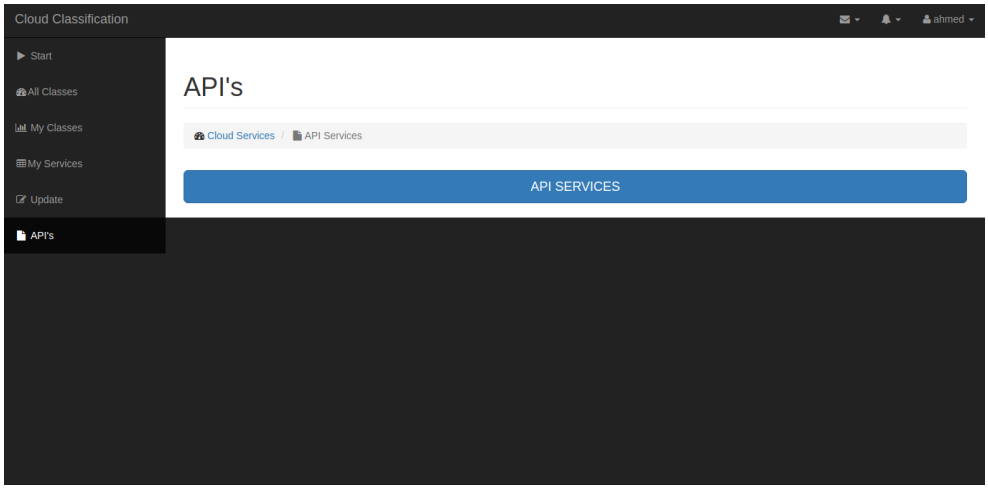


Figure 5.25: API interface

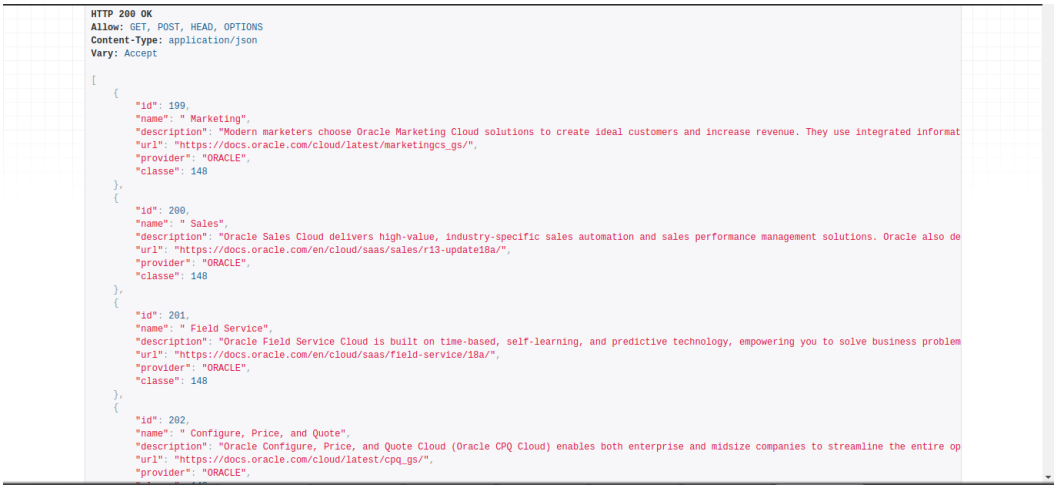


Figure 5.26: API Response

General Conclusion

As it is said, necessity is the mother of invention. In fact, men have devoted themselves through different circumstances into inventing and creating numerous types of accessories and software applications that are beneficial to their own sake. These inventions have always been in pursuance of facilitating and simplifying daily activities. As for our case, we were interested in searching and selecting Cloud services meeting user requirements. As a matter of fact, we implemented a Cloud services search engine that facilitates the searching of suitable Cloud service meeting user requirements. This report was a brief synthesis project which consisted in designing and developing a cloud search engine. To be able to do this, we followed the work plan described as follows: First of all, we studied the existing solutions, and note their major limits . On a second note, we examined the cloud services description of each cloud provider and we decided to proceed with the machine learning strategy, We started with collecting data then, we scraped the cloud product site for each cloud provider. Secondly , we begin the data prepossessing step. We used several techniques in this step such as removing stop words, stemming, enriching the descriptions. The third part is to model our data, so we went deep into studying regarding theoretical aspects of different text classification techniques. We tried two algorithms which are Doc2Vec and TF-IDF. The fourth part is to split and classify the data that we had according to their content, we used as input of k-means algorithm the output of the model in order to cluster the cloud service's classes,the measure unit of k-means algorithm is the cosine similarity . The next step was to implement the cloud search engine which uses the clusters as an input,in order to ameliorate its results and make it more efficient.

It is important to mention that building this project was not that easy, we faced during our works many difficulties and setbacks. However each difficulties helps us to increase our knowledge and ask more questions to improve our work. Becoming acquainted with Cloud services descriptions and classification offered by providers was not that easy and it required an extensive research.

As every project can be enhanced and expanded, various perspectives are imminent for our

project. We can mention the expansion of our data base by adding more Cloud providers. Moreover, in order to enhance the performance of our clustering algorithm we have to try more semantic models , in other words we can think more about deep learning models and how they can be improved. For example, the Word Mover's Distance similarity metric adding to the GloVe vectors helps improving our doc2vec model.

Last but not least we may empower our platform by adding many other features to increase comparisons between Cloud services based on the prices and the QoS etc. Moreover after collecting feed backs from our users we can make our system able to recommend the best cloud service to use according to the user's case.

Bibliography

[B1]: Rehurek (2014)

[B2]: http://www.met.edu/Institutes/ICS/NCNHIT/papers/39.pdf

[B3] : CS246: Mining Massive Datasets Jure Leskovec, Stanford University https://web.stanford.edu/class/cs246/slides/05-clustering.pdf

Netography

[NIST] https://www.nist.gov/programs-projects/nist-cloud-computing-program-nccp (Last visit 30/03/2018)
[N1] http://agilemanifesto.org/ (Last visit 25/04/2018)
[N2] https://www.qasymphony.com/blog/agile-methodology-guide-agile-testing/ (Last visit 25/04/2018)
[N3] https://towardsdatascience.com/automated-text-classification-using-machine-learning-3df4f4f9570b (Last visit 28/04/2018)
[N4] https://www.coursera.org/learn/machine-learning (Last visit 29/04/2018)
[N5] https://nlp.stanford.edu/software/tokenizer.shtml (Last visit 12/04/2018)
[N6] https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html (Last visit 15/04/2018)
[N7] https://deeplearning4j.org (Last visit 18/04/2018)
[N8] https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/ (Last visit 20/04/2018)
[N9] http://adventuresinmachinelearning.com/word2vec-tutorial-tensorflow/ (Last visit 22/04/2018)
[N10] https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e (Last visit 22/04/2018)
[N11] https://www.slideshare.net/DharshikaShreeganesh/brain-tumor-mri-image-segmentation-and-detection-in-image-processing (Last visit 23/04/2018)
[N12] http://www.cloudservicemarket.info/research/researchers.aspx (Last visit 30/04/2018)
[N13] http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/ (Last visit 14/04/2018)
[N14] http://whoosh.readthedocs.io (Last visit 29/04/2018)
[N15] https://www.serchen.com/browse/ (Last visit 2/05/2018)
[N16] https://www.packtpub.com/mapt/book/bigdataandbusinessintelligence/9781787285101/2/ch02lvl1sec17/stemming (Last visit 30/04/2018)