Université de Blida 1 – SAAD DAHLAB

Faculté des sciences

Département d'Informatique

Mémoire présenté par :

Mrs.

BENMOUFFOK SAID Oualid

LOUNAS Mohamed Sofiene

Pour l'obtention du diplôme de Master

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : Traitement Automatique de la Langue

Sujet :

Creation of a Chatbot for video games recommendation with NLP

Defended on: 06/07/2020, in front of the members of the thesis committee:

Mrs. S. OUKID	Professor	University of Blida 1	President
Mr. S. FERFERA	Associate Professor	University of Blida 1	Examinator
Mrs. M. MEZZI	Associate professor	University of Blida 1	Supervisor

Abstract

Chatbots are software-based systems designed to interact with humans using text-based natural language. They first appeared in the 1960s, many years passed till the world became ready for their implementation into the real life, as a result of the rapid progress in Natural Language Processing, Machine Learning, and the global presence of text messaging applications.

Chatbots are so trendy nowadays, they are being exploited in many aspects, like products recommendation for instance, but their main weakness is that their domain of knowledge is narrow, and it is nearly impossible to create a Chatbot that can tackle multiple domains with today's knowledge, and thus each Chatbot is created differently to fit the requirements of the specific aspects it is put in.

This thesis represents a practical case of a Chatbot deployed for video games recommendation, as supposedly, the field of video games has not been fully exploited yet. The Chatbot uses a combination of traditional Natural Language Processing mixed with other modern techniques such as Neural Networks.

The thesis also shows how it is possible to reduce the amount of text data needed to obtain a relatively good intent classification model, by using the semantic similarity of words, or in other words, multi-dimensional meaning representations of words.

Keywords: Conversational Agents, Chatbots, Recommender Systems, Context-management, Machine Learning, Natural Language Processing.

Résumé

Les agents conversationnels sont des logiciels basés système, désigné pour interagir avec des humains en utilisant le langage naturel textuel. Leur première apparition remonte aux années 60, plusieurs années se sont écoulées pour qu'enfin le monde soit prêt pour leur incorporation dans la vie réelle, due à l'évolution rapide concernant le domaine du Traitement Automatique de la Langue, l'Intelligence Artificielle, et la présence d'applications de messagerie textuelle.

Les agents conversationnels sont devenus quasi-omniprésent, exploités dans plusieurs aspects comme la recommandation de produits par exemple, mais leur principale faiblesse réside dans le fait que leur domaine de connaissance est limité. Avec la technologie actuelle c'est presque impossible de créer un agent conversationnel qui puisse gérer plusieurs domaines de connaissances en même temps ce qui explique donc que chaque agent conversationnel doit être conçu pour un domaine bien précis.

Cette thèse représente une approche pratique dédiée à un agent conversationnel déployé pour la recommandation de jeux vidéo, étant donné que le domaine des jeux vidéo n'a, à notre connaissance, pas encore été exploité. L'agent conversationnel utilise une combinaison d'approches traditionnelles du traitement automatique de la langue ainsi que d'autres plus modernes comme les réseaux de neurones.

Cette thèse démontre comment il est possible de résoudre certains problèmes comme le manque de données spécifique au domaine d'études utilisant certaines techniques qui réduisent la quantité de données nécessaires pour obtenir des résultats acceptables.

Mots clés : Agents Conversationnel, Chatbots, Systèmes de Recommandation, Gestion du Contexte, Apprentissage Automatique, Traitement Automatique de la Langue.

ملخص

روبوتات المحادثة عبارة عن أنظمة برمجية مصممة للتفاعل مع البشر باستخدام لغة نصية طبيعية. يعود ظهورها الأول إلى الستينيات، وقد مرت عدة سنوات حتى أصبح العالم جاهزًا لإدماجها في الحياة الواقعية، بسبب التطور السريع فيما يتعلق بمجال المعالجة التلقائية للغات، والذكاء الاصطناعي، وانتشار تطبيقات الرسائل النصية.

أصبحت روبوتات المحادثة حاضرة في كل مكان تقريبًا، ويتم استغلالها في عدة مجالات كالتوصية التلقائية للمنتوجات على سبيل المثال، ولكن نقطة ضعفها الرئيسي تكمن في حقيقة أن مجال معرفتها محدود. باستخدام التكنولوجيا الحالية، يكاد يكون من المستحيل إنشاء روبوت محادثة يمكنه إدارة العديد من مجالات المعرفة في نفس الوقت، لذلك نجد أن كل بوت محادثة مصمم بطريقة مختلفة ليستطيع تلبية متطلبات المجال الموجود فيه.

تمثل هذه الأطروحة طريقة عملية لإنشاء روبوت محادثة موجه للتوصية التلقائية لألعاب الفيديو، نظرًا لأن مجال ألعاب الفيديو، على حد علمنا ، لم يتم استغلاله جيدا بعد. روبوت المحادثة يستعمل مزيجا من تقنيات تقليدية لمعالجة اللغة الطبيعية مع تقنيات اخرى أكثر حداثة كالشبكات العصبونية.

الأطروحة تبين أيضا كيف يمكن تقليل كمية البيانات النصية اللازمة لتدريب نموذج جيد لتصنيف الرغبات باستعمال تقارب معاني الكلمات، أو بمعنى آخر، تمثيلات نقطية للكلمات ذات معنى في معلم متعدد الأبعاد.

الكلمات المفتاحية: روبوتات المحادثة، Chatbots، أنظمة التوصيات، إدارة السياق، التعلم الآلي، معالجة اللغة الطبيعية.

Dedications

We would like to dedicate this thesis as the final statement to our beloved parents who helped us in many ways during this journey, may god bless them.

To our brothers, sisters, supervisor and friends who shared their words of advice and encouragement to finish this work.

And finally, to ourselves for being in this long journey and for making it to the very end.

Oualid and Sofiene.

Acknowledgments

First of all, we would like to thank Allah who gave us the strength and courage to finish this work, and our parents for being our biggest support.

We would also like to thank everyone who helped us to make this work, our supervisor Mrs. MEZZI in particular for being always supportive no matter the situation.

Table of content

List of figures

List of tables

General Introduction

1.1 Research background

Both Artificial Intelligence (AI) researchers and industry are increasingly recognizing the importance of Chatbot systems. Chatbots are embedded in every day's life, performing various roles even serving as assistants for end-users in a variety of industries.

Chatbots evolved in the last years from a simple conversation perspective to being nearly everywhere, exploiting every aspect of human's life: news – sport – medicine and many others, even considered as personal assistants for some kinds of Chatbots.

Despite this evolution, some aspects have not been exploited by Chatbots; Video games are one of them, which is understandable, knowing how people see video games. Many see them as a source of entertainment only, others see them as a source of nuisance and violence, even though nowadays many video games are used for pedagogical purposes such as teaching English and History.

Video games are more than just an entertainment business; they are reshaping the way we interact with the world. Right now, there is a lot of games and gaming platforms that may satisfy everyone in the society. Despite the many available games and platforms, users sometimes feel lost when trying to find a game to play.

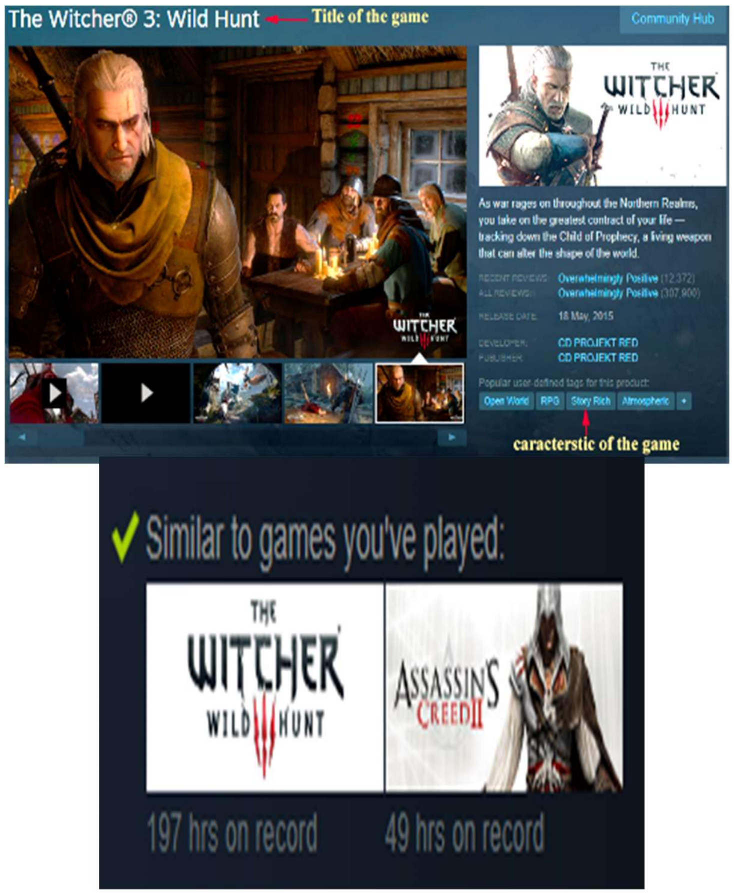
Some video games platforms today have built recommendation systems to help their users, for instance: **Steam**[1], which is the most common distribution platform for PC with around 75% of the market share in 2013 [1], offers the users recommendations based on their previous purchases in the store, offering them games of the same category of the ones they own ("Content Based Recommendations"), and recommendations based on other users' reviews ("Collaborative Recommendations"), we will talk more about these methods in Section I.3 . This kind of system has been proved to be effective in most cases.

Our goal is to deploy a Chatbot that is capable of having coherent conversations with its users, understanding their needs, and providing personalized recommendation for each user. That will be beneficial for both the users, and for the sellers.

[1] https://store.steampowered.com/

1.3 Research challenges

As we mentioned before in our context introduction, most video games distributors recommend games to the users depending on their previous purchases or by analyzing their preferences list, supposing that the user has already played games on the platform in the first place, that is how "Steam" recommendations work for example as shown below (Figure 0.I.1) :



As war rages on throughout the Northern Realms, you take on the greatest contract of your life — tracking down the Child of Prophecy, a living weapon that can alter the shape of the world.

RECENT REVIEWS: Overwhelmingly Positive (12,372)
ALL REVIEWS: Overwhelmingly Positive (307,900)
RELEASE DATE: 18 May, 2015
DEVELOPER: CD PROJEKT RED
PUBLISHER: CD PROJEKT RED
Popular user-defined tags for this product:
Open World RPG Story Rich Atmospheric +

Figure 0.I.1 : Steam platform interface.

Our focus is to elaborate a Chabot, which will help to understand the gamers, by being a virtual friend, and which can help the gamer choose games to play depending on his preferences.

The Chabot will analyze the users throughout the course of a conversation, which means the Chabot will be having a conversation with the user, and gathering information about him, such as the games and types of games he likes or dislikes. This information will be used then to suggest games to that user, and to other users.

For example: when suggesting a game, the Chabot will wait for the user's feedback and readapt the search to try to avoid that particular game or the games similar to it.

1.4 Research problems

After some research, we concluded the problems that we are going to face in order to fulfill our objective are summarized in the following points:

- **Ambiguity related to the user's intentions**
  A maze is the first image that we can attribute to human nature, complex, unpredictable. This unpredictability is illustrated in the case of a discussion with the Chabot, the user tends to switch from a subject to another, making it difficult for the Chabot to finish any unfinished task, or to insure a relevant conversation.
- **Entity extraction: which entities to use, and how to extract them?**
  Entities are pieces of information that are needed to accomplish some tasks in various domains. There's a lot of libraries and APIs (Application Programming Interfaces) that offer entity extraction from a given text, but most of them focus on general case entities like times, dates, places etc. and some of them offer the possibility to create custom entities. Games names and types are the only entities we need, so we are going to either use one of the available methods, or create or own customized entity extractor.
- **The small amount of data concerning the subject**
  That means we are obliged to create our own datasets, for most of the project's parts, and that will cost a large amount of time. In a production environment, there is the possibility to retrain the models with data from real users on the go, through an alpha release, but in our case, we are going to work with the datasets we create.

1.5 Thesis outline

This portion of the thesis serves as a "foretaste" to what is coming next as we introduced the context of our thesis and the purpose of choosing this subject, finally we specified the different problems that we are going to face in our journey.

The thesis is like a "banquet" where each following chapter represents a dish filled with more details about the main subject decomposed as follow:

- Chapter I: The State of the art can be considered as the "entry" which is going to introduce the notions related to Chatbots, their evolution, followed by, the context managing aspects in Chatbots and the different techniques available and finally, a view on recommendation systems and Chatbot.
- Chapter II: The modelization chapter serves as our "main dish" where we are going to introduce the different approaches we took in the modelization of our chatbot and the different aspects of it.
- Chapter III: The implementation chapter is our "dessert". It is going to focus on representing the final iteration of our Chatbot in addition to the tools and technologies used in its conception together with some screen shots illustrating examples of conversations that our Chatbot is able to carry.

Chapter I:
State of the art

I.1 **Introduction**

This chapter represents our state of the art and is going to focus on introducing the different domains of research regarding our work, separated in three sections.

The first section is an introduction to Conversational Agents, including the definition I.2.1, as well as a brief historical overview I.2.2, followed by a deep peek into the functionalities of Chatbots I.2.3 and finally the drawbacks of some Chatbots I.2.5.

The second section talks about Context management in Chatbots, and includes a definition of Context I.3.1, an explanation of Context-aware Computing I.3.2 and finally the different approaches regarding context-aware computing I.3.3.

The third and last section is dedicated to the relation between Chatbots and Recommendation Systems, and includes a definition of Recommendation Systems I.4.1 and the different techniques used in Recommendation Systems I.4.2

I.2 **Conversational Agents**

This section is devoted to the Conversational Agents.

I.2.1 **Definition**

A Conversational Agent was defined by (Prevost et al., 2000) as "*Computer interfaces that can hold up their end of the conversation, interfaces that realize conversational behaviors as a function of the demands of dialogue and also as a function of emotion, personality, and social conversation.*" [2]. Alternatively, it can be defined as a software-based system designed to interact with humans via text based natural language.

The underlying principle of every Chatbot is to interact with a human user (in most cases) via text messages and behave as though it were capable of understanding the conversation and reply to the user appropriately. The origin of computers conversing with humans is as old as the field of computer Science itself. Indeed, Alan Turing[2] defined a simple test referred to now

[2] Alan Mathison Turing (23 June 1912 – 7 June 1954) was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist. Turing was highly influential in the development of theoretical computer science, providing a formalization of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general-purpose computer.

as the Turing test back in 1950 where a human judge would have to predict if the entity they are communicating with via text is a computer program or not. However, this test's ambition is much greater than the usual use case of Chatbots; the main difference being that the domain knowledge of a Chatbot is narrow whereas the Turing test assumes one can talk about any topic with the agent. This helps during the design of conversational agents as they are not required to have a (potentially) infinite domain knowledge and can, as such, focus on certain very specific topics such as for instance helping users book a table at a restaurant [3].

Furthermore, another general assumption Chatbot designer's bear in mind is that users typically have a goal they want to achieve by the end of the conversation when they initiate an interaction with a Chatbot. This then influences the conversation's flow and topics in order to achieve the chosen goal. This can be exploited by developers since certain patterns of behavior tend to arise as a result [3].

I.2.2 History of Chatbots

In this section, we will give some examples on how Chatbots evolved in the past 50 years from ELIZA bot to the current so-called personal assistant such as Apple's Siri or Microsoft's Cortana.

I.2.2.1 Eliza-Bot

The first instance of a conversational agent was born in 1966: ELIZA was a computer program that simulated a psychiatrist and rephrased user input using basic (by today's standards) Natural Language Processing (NLP) techniques [4]. Despite being relatively simple, the program managed to give the illusion of understanding the user's problems and successfully fooled a great many people. Its creator, Joseph Weizenbaum, even mentioned that his secretary would ask him to leave her so she could have a private conversation with ELIZA.

I.2.2.2 Smarter Child

During several decades, Chatbots heavily followed ELIZA's approach with minor additions brought into the field like speech synthesis and emotions management. Then in 2001 came SmarterChild, a conversational agent developed by ActiveBuddy, Inc. (now Colloquis) that operated on AOL Instant Messenger and MSN Messenger. Inspired by the rise of instant messaging platforms such as SMS, SmarterChild was created to provide quick access to news, weather forecasts, sports results, etc. The main innovation was that Smarter Child was connected to a knowledge base and detained useful information for its users. Unfortunately, the technical limitations of Natural Language processing caught up with bots on these platforms at the time and they were forgotten by History.

I.2.2.3 **Watson AI**

The next advancement for conversational agents was made by a team at IBM through the Watson AI project that has been in development since 2006. The agent was designed with the sole purpose of winning the American TV show *Jeopardy!* Which it did in 2011 when competing against two of the show's former champions. *Jeopardy!* Is interesting from an NLP point of view since the questions involve a lot of play on words and require fast information retrieval in vast knowledge bases. Unfortunately, this AI in its past form could only answer to one-liner questions and was unable to carry on a proper conversation with someone else.

I.2.2.4 **A New Generation of Chatbots**

In the early 2010's came the rise of virtual assistants such as Apple's Siri, Microsoft's Cortana, Google's Google assistant, Amazon's Alexa and others. Those agents brought in the field the concept of conversation as well as goal-oriented dialog. Another major event in the field of Chatbots was the release of the Messenger Platform for Facebook Messenger in 2016 and allowed the creation of conversational agents for non-AI related companies (Figure I.1).

As shown in this brief summary of the field of conversational agents, a lot of progress has been made since the early days of NLP. This does not imply however, that current solutions are without flaw as will be highlighted in the next section.

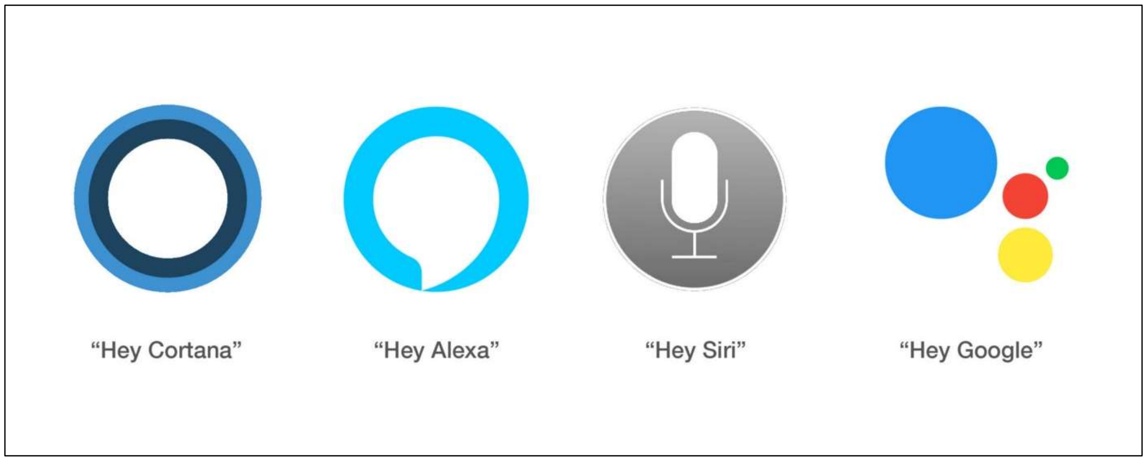


Figure I.1 : The logos of the big four: Cortana, Alexa, Google Assistant and Siri.

I.2.3 Chatbots under the hood

Considered as machines, **Chatbots** do not learn or adapt like humans where the human's mind can jump from a point to another instinctively, whereas **Chatbots** need a whole lot of adjustments to be deployed in performing any task.

Certain **Chatbots** based on regular expressions, use a sequence of characters that define a search pattern [5], which **ELIZA** was the first to use in **Chatbots**.

Input sentences are analyzed based on decomposition rules, triggered by key words appearing in the input text, responses were generated by reassembly rules associated with selected decomposition rules [4], which means that **ELIZA** was defragmenting the user sentences following specific rules to create another sentence depending on the previous input. For a better understanding, table I.2-1 shows a small conversation with ELIZA:

Table I.2-1 : Example of a discussion with ELIZA.

User: hello	ELIZA: HI. What seems to be your problem?
User: I don't have any problems	ELIZA: Don't you really have any problems?
User: yes	ELIZA: I understand

Some other Chatbots are referred to as Question/Answer (QA) bots, like the case of **WATSON** which was built using 200 million pages of content, both structured and unstructured, including dictionaries and encyclopedias. When asked a question, Watson initially analyzes it using more than 100 algorithms, identifying any names, dates, geographic locations or other entities. It also examines the phrase structure and the grammar of the question to better gauge what's being asked. In all, it uses millions of logic rules to determine the best answers [6].

Watson is based on QA technology and the difference between QA and document search is that document search takes a keyword query and returns a list of documents, ranked in order of relevance to the query (often based on popularity and page ranking), While QA technology takes a question expressed in natural language, seeks to understand it in more detail, and returns a precise answer to the question [7]. As shown in the following example (Table I.2-2):

Table I.2-2 : Examples of questions asked to Watson.

Category: Before and After Goes to the Movies *Clue:* Film of a typical day in the life of the Beatles, which includes running from bloodthirsty zombie fans in a Romero classic. *Sub clue 2:* Film of a typical day in the life of the Beatles. *Answer 1:* (*A Hard Day's Night*) *Sub clue 2:* Running from bloodthirsty zombie fans in a Romero classic. *Answer 2:* (*Night of the Living Dead*) *Answer: A Hard Day's Night of the Living Dead*
Category: General Science *Clue:* When hit by electrons, a phosphor gives off electromagnetic energy in this form. *Answer:* Light (or Photons)
Category: "Rap" Sheet *Clue:* This archaic term for a mischievous or annoying child can also mean a rogue or scamp. *Sub clue 1:* This archaic term for a mischievous or annoying child. *Sub clue 2:* This term can also mean a rogue or scamp. *Answer:* Rapscallion

At some point, with the propagation of the internet, appeared some Chatbots that were created without prior training, where the Chatbot was meant to learn on its own, from interaction with people.

CLEVERBOT was one of them, like a human learning appropriate behavior by studying the actions of members of his or her social group, **CLEVERBOT** "learns" from its conversations with internet users. It stores them all in a huge database, and in every future conversation, its responses to questions and comments mimic past human responses to those same questions and comments [8].

Through the years, the approaches of the creation of Chatbots multiplied going from using simple regular expressions (ELIZA), to using techniques to learn from live interactions with the users. In the next subsection, we are going to talk more about the different Types and architectures of Chatbots.

I.2.4 Types and architectures of Chatbots

When it comes to creating a Chatbot, the only thing agreed on is that it should be able to communicate with the user via natural language, but the existence of any Chatbot is defined by the reason it was created for, and as so, it is difficult to group them in different categories as sometimes a Chatbot can be a built on mix of different architectures, but in this subsection, we will try to talk about some of the most common types of Chatbots.

The next figure shows the different types of Chatbots.

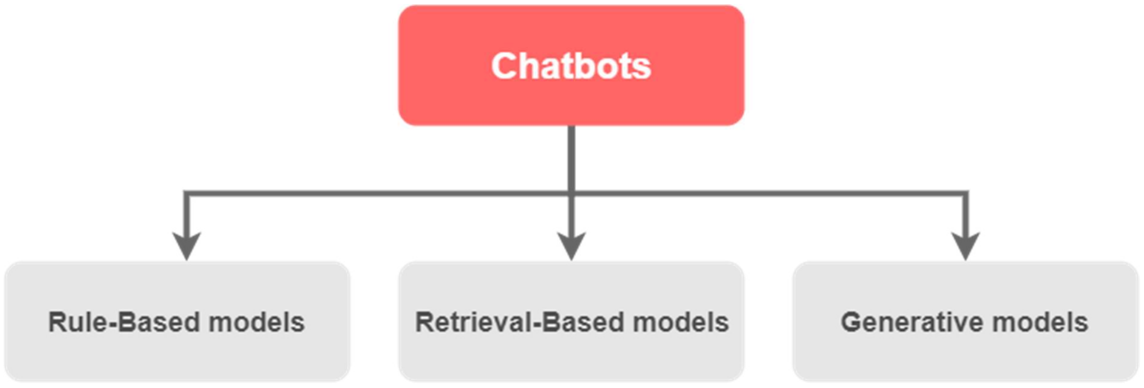


Figure I.2 : illustration representation the different types of Chatbots

I.2.4.1 Rule-Based models

The principle idea underlying rule-based RG is that a bot contains a knowledge base with documents, where each document contains a <pattern> and <template>. When the bot receives an input that matches the <pattern>, it sends the message stored in the <template> as a response. The<pattern> could either be a phrase **"What's your name?"** or a pattern **"My name is *"**, where star is a regular expression. Typically, these <pattern> <template> pairs are hand-crafted. [9]

The programmer's main challenge is to choose an algorithm to use for pattern matching between the input sentence and the documents in the data corpus. We can think of this as a nearest- neighbor problem, where our goal is to define the distance function, and retrieve the closest document to the input sentence. Below, we are going to give an example about it.

ELIZA used an incremental parsing or "direct match" approach for pattern matching.

ELIZA parsed the input text word by word from left to right, looking each word up in the dictionary, giving it a rank based of importance, and storing it on a keyword stack. The keyword with the highest rank of importance would be treated as the <pattern> and pattern matched against the documents in the corpus to find the appropriate template response. If no document existed corresponding to the input, ELIZA would say “I see” or “Please go on” in the DOCTOR script. ELIZA was created with multiple “scripts” which indicate different <template> responses to <pattern> inputs. [10]

I.2.4.2 Retrieval-Based models

The main problem with rule-based systems is that the programmer must specify all “rules” or <Pattern> <template> pairs. With the advent of large datasets such as dialogues on Reddit and Twitter, this no longer became necessary. Instead, developers could analyze millions of conversations and store these conversations in documents as pairs of <statuses> (first speaker) and <Responses>. The principle behind retrieval-based systems is, given an input sentence, to pattern match this against the set of <status> <response> pairs and select a response.

The main challenge in Information Retrieval (IR) algorithms is determining how to conduct pattern matching. First and foremost, there is the consideration of, given an input sentence, whether to pattern match against the <status> or <response>. The former seems more intuitive, and would involve finding the most similar <status> in the data corpus. The latter, however, is used more often in practice because it is more effective, and involves comparing the input sentence against the <responses> in the data corpus. The reason matching against <responses> is more effective is that if words co-occur between the input sentence and a <response>, it is likely an appropriate response. By contrast, just because words co-occur in the input sentence and <status>, if it is not an exact match, we don’t know if the <response> will be appropriate. Likewise, if the system pattern matches against responses, not inputs, it can simply use search engines to query for a set of responses that are similar to the input phrase (e.g. search engines retrieve response sets). [11]

Second, regardless of whether we are pattern matching against <status> or <response>, we must choose a pattern matching algorithm. So, the programmer needs to find the best way possible on how to handle the pattern matching aspect to have proper results, let us give an example of a chatbot who used retrieval-based aspect.

WATSON: known as the first Chatbot to be able to win the JEOPARDY! Tv show, Watson by itself was just the front face on what actually caused that victory without forgetting the architecture behind that success.

DEEPQA Architecture is the software architecture for the massively parallel deep content analysis and evidence-based reasoning that is be- hind the Watson program which appeared as a Jeopardy! Contestant.

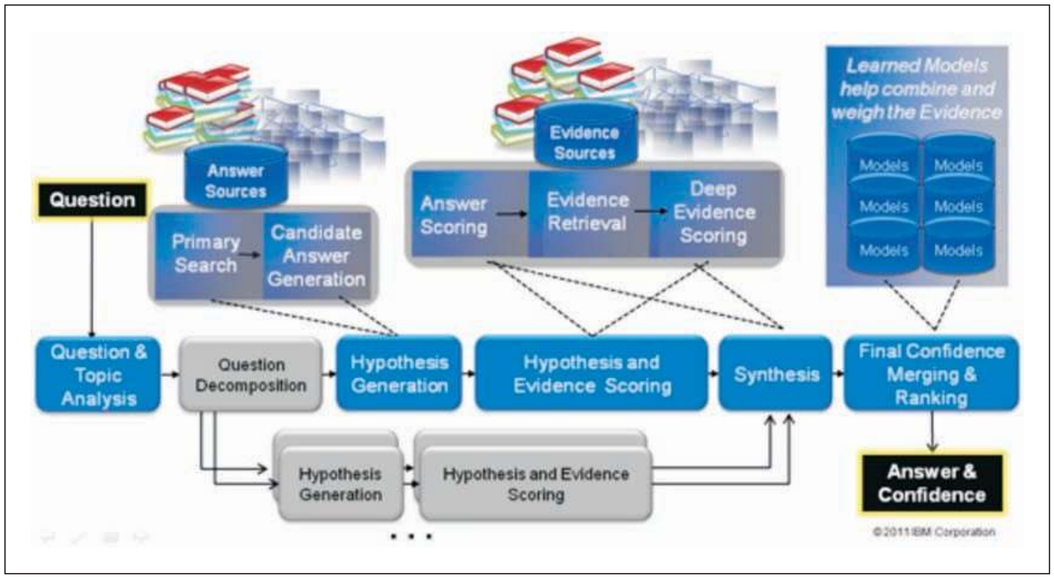


Figure I.3 : Overview of DeepQA's architecture

A question analysis step then determines how the system will process each question. Here a number of components extract information at various levels, ranging from:

- Category classification: is the question a puzzle, or a pun, etc.?
- Answer type detection: what is the answer expected to be? A Person, place, sport, etc. ?
- Relation detection: are there relations we can identify between the entities in a question?
- Question decomposition: should questions be broken into sub- questions and if so, how?

The DeepQA architecture allows for task-specific components to be plugged in, to handle specific aspects. Here are some examples of components specific to the Jeopardy! Task.

There are a number of issues specific to playing Jeopardy! That are not encountered in normal question-answering tasks, such as trying to decode category names. As we have seen above, category names range from the prosaic descriptive ones such as *Astronomy,* which are useful, to ones with embedded puns, which are harder to interpret in a meaningful way. Following the strategy of using many different sources of information, Watson had a component to learn from the category name if it could. [12]

After the clue was revealed, the system made use of query classification. Some questions can be answered with factual information; for example, the question "*These are the two US states that are rectangular in shape."* This requires Watson to determine what is required in the answer, and to develop a plan to get to the relevant part of this factual information. [12]

Some questions have nested sub-questions, which require question decomposition, for example: "*Which is the southern-most landlocked country in Africa?"* This requires a list of landlocked countries in Africa to be known, and the southernmost of these identified.

It also helps to figure out the expected entity class, or topic, of the answer: whether it is a person's name, the name of a place, a date, etc., for that constrains the set of possible answers. So, if the category is "*16th Century People"*, knowing that the answer is a person or persons can be very useful.

I.2.4.3 **Generative models**

Generative Chatbots are the future, but building and maintaining such advanced Chatbots is really difficult as they are AI based. There are various ways of how to create such Chatbots, but the underlying principle behind them is being able to decompose the user's message, understand it, and generate answers on the go without having a predefined set of answers.

Building this kind of Chatbots usually requires a lot of data. Some of them are built to fully learn from their interaction with the users without any supervision like the case of **Tay** (but that did not work well as we will see in the next subsection), while other Chatbots are oriented into learning to create their own personalities from interacting with users like the case of **Replika** [3], which -in our personal opinion- brought the legacy of **ELIZA** into the modern world.

[3] https://replika.ai/

In the next section we are going to talk about the drawbacks of the techniques used in the creation of certain Chatbots.

I.2.5 **Drawbacks**

The Chatbots distinguish themselves from each other in their respective categories and purposes; each one is based on a different architecture, a unique model that makes them differ from each other. In this section, we are going to talk about the drawbacks of some well-known Chatbots.

ELIZA, the predecessor of all the Chatbots we know, has many technical problems including the identification of key word [13] , the discovery of minimal context [14], the choice of appropriate transformation [15], generation of responses in the absence of keywords [16] and provision of an editing capability [17].

Other modern Chatbots like **CleverBot** or **TAY** proved that learning from previous interactions could cause major controversies due to miss use by the public. Internet trolls on Twitter targeted TAY, Microsoft's AI Chatbot. 16 hours only after its launch, Microsoft had to shut it down after it started sending extremely offensive Tweets to users [18]. CleverBot's developers, in the other hand, put a warning message in their webpage regarding the inappropriate answers the bot may give, we can sum up these points in (Table I.2-3):

Table I.2-3 : The drawbacks of both TAY and Cleverbot.

TAY	• Misusage due to the direct interaction with the users. • No supervision over the information collected by the bot. • Mimics the previous interactions with the users.
cleverbot ***Cleverbot***	• Mimics the previous interactions with the users. • Warning concerning the interactions it may causes with users.

The next section is going to focus on the context management aspect required for the functioning of Chatbots.

I.3 Managing Context in Conversational Agents

A conversation as simple as it can be cannot happen without a subject, a reason to converse for the two (or more) sides that are interacting. As vast as the subject can be, the conversation needs to be precise even for humans who tend to jump from one subject to another, it can be quite difficult to keep the flow of the conversation going.

Context management is the first priority to handle when we are having a conversation so we can focus on the main concern directly, because the vaster the subject is, the more complex the context can be extracted, knowing the human nature, which tends to see every meaning and aspect of words.

The human mind by its complexity can distinguish between contexts because of its capability to differentiate the meanings of words, but machines are different. Machines cannot treat context, as easily as humans do. Solving this problem was the main concern of many Chatbots in the recent years.

In this section, we are going to present some techniques used to manage context in conversational agents.

I.3.1 Definition of context

The word "context" by itself does not have a proper or universal definition, each researcher defines it according to their own work, so it depends on how we interpret the word, for everyone can have their own understanding of it but no one can give it proper definition.

"Context-aware" term was first introduced by (Schilit and Theimer) [19] , who referred to context as location, identities of nearby people and objects, and changes to those objects. These types of definitions that define context by example are difficult to apply, as the context may not be composed of multiple entities, including entities that are not cited in the definition.

Other definitions have simply provided synonyms for Context; for example, referring to context as the environment or situation [20, 21]. As with the definitions by example, definitions that simply use synonyms for context are extremely difficult to apply in practice. The definitions by Schilit et al. [22] and Pascoe [23] are closest in spirit to the operational definition we desire. Schilit et al. claim that the important aspects of context are: *where you are*, *who you are with*, and *what resources are nearby*. Pascoe defines context to be the subset of physical and conceptual states of interest to a particular entity. These definitions are too specific. Context is all about the whole situation relevant to an application and its set of users. We cannot enumerate which aspects of all situations are important, as these will change from situation to situation. For this reason, we could not use these provided definitions.

Concerning our definition of the word **Context**, we see it as: "**any information describing a precise situation, which helps us define a proper way of interaction with the information we got**. The following example (Table I.3-1) will clarify this definition:

Table I.3-1 : Difference between two sentences in term of context interpretation.

I want to buy an apple.	I want to buy an apple phone.

At first, we do not see a big difference between the sentences except the addition of the word "phone", but that is the thing that makes the two sentences different from each other. In the first example, the machine cannot figure out if the user is talking about "Apple" the American company or "Apple" the fruit.

As for the second one, the addition of the word "phone" precise exactly in which context the machine is going to focus on to give an answer to the user, which should be about any phone product from the "Apple" brand.

That means that Chatbots can be considered as "context-aware" since the root to create any Chatbot is the "*on what the bot is going to focus on to interact with the users?*" question but what is a context-aware application first?

I.3.2 Context-aware computing

Context-aware computing was first discussed by Schilit and Theimer [19] in 1994 as software that ''*adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time''*. Since then, numerous attempts defined context-aware computing, most of which have been too specific [24].

Our definition of context-aware follows the same trail, as we define it as ***"software, which can tell precisely the context in which it is in, and is capable of providing relevant answers to the user's tasks."***

The importance of context awareness is without doubt the first thing we need to focus on during the creation of a Chatbot since it determines if the Chatbot is able to accomplish any task given by the user , but context-awareness by itself has many approaches on how to model it efficiently especially in multi-contextual Chatbots.

I.3.3 Context-awareness approaches models

In the following, we will introduce two of the most interesting Contextual Models in our sole point of view.

I.3.3.1 Spatial context modeling

Context information is important for most context-aware applications. Places were used in different context definitions [25, 26] to show that spatial information is a core component of developing context-aware applications. Moreover, most spatial context models are fact-based models that organize context information by physical location. There are two different representation forms of spatial (location) based attempts. Therefore, spatial information is divided into symbolic and geometric coordinates.

Symbolic coordinates are often represented by an identifier such as a room number or the ID of a cell or access point in a wireless local network area [27]. Symbolic coordinates are used in applications where an explicit positioning of an object is not necessary for the context modelling. The negative aspect of using symbolic coordinates is the lack of spatial relations between pre-defined spatial coordinates, thus to avoid this lack, we need to specify the relationship between pairs of the symbolic coordinates.

Let us take an example, a game for instance, can be represented with symbolic coordinates (name of the game) and geometric coordinates (type of the game), this modeling will allow certain spatial relations between the components and thus a more context-aware model, As shown in (Figure I.4):

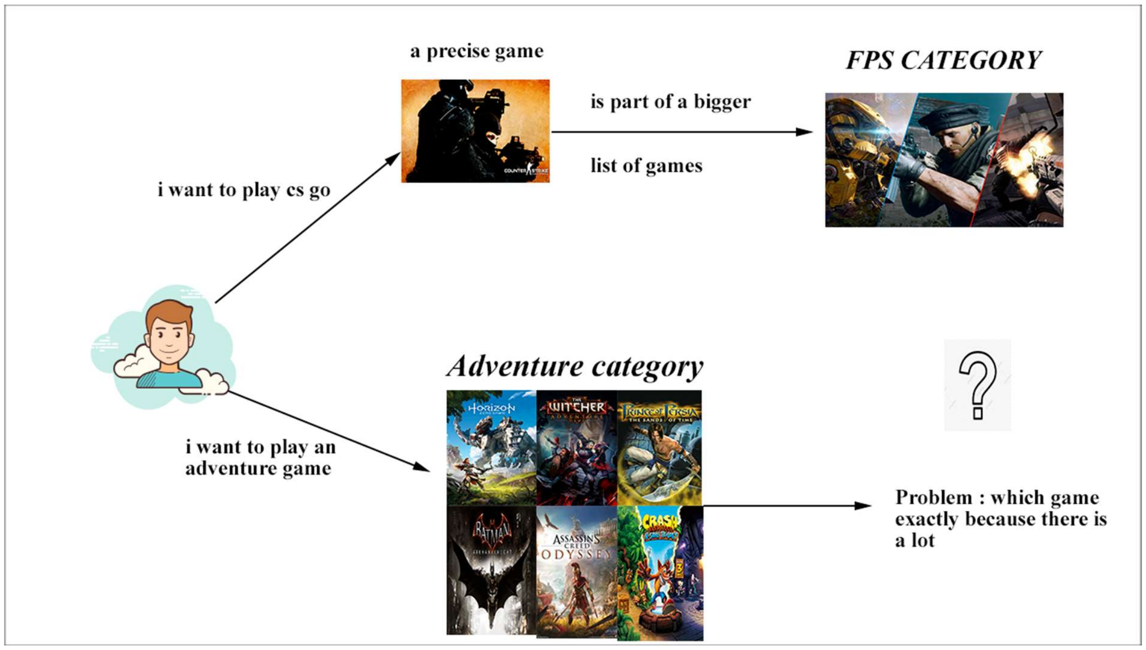


Figure I.4 : Illustration of the spatial modeling approach in choosing a video game.

In the case of the user choosing a specific game (like in the example), the spatial area would be restricted to the game itself even if it appears in a larger category; the focus is on the game itself.

Whereas the case of the user choosing a category without specifying the game, the spatial area would be larger than the previous case, which leads sometimes to the unsatisfaction concerning the game recommended by the Chatbot.

I.3.3.2 Ontology based context modeling

Ontology based approaches represent knowledge, concepts and relationships of a given domain and describes specific situations in it [28, 27, 29]. There are different areas where researchers are using ontologies to describe various types of applications for specific issues. The underlying goal of ontology development is to provide common terminologies and rich semantics to enable knowledge sharing and reuse between different systems [28, 30].

Such general vocabulary is an important feature to share information among different omnipresent computing systems. Now, we understand that ontology-based context modelling is a generally used attempt to provide knowledge over different domains (systems) in order to get a suitable infrastructure for complex context-aware systems.

Ontologies being as rich as they can be, they may cause trouble concerning answer generating especially if the word itself has different aspects to it, an example will make it clearer as shown in (Figure I.5):

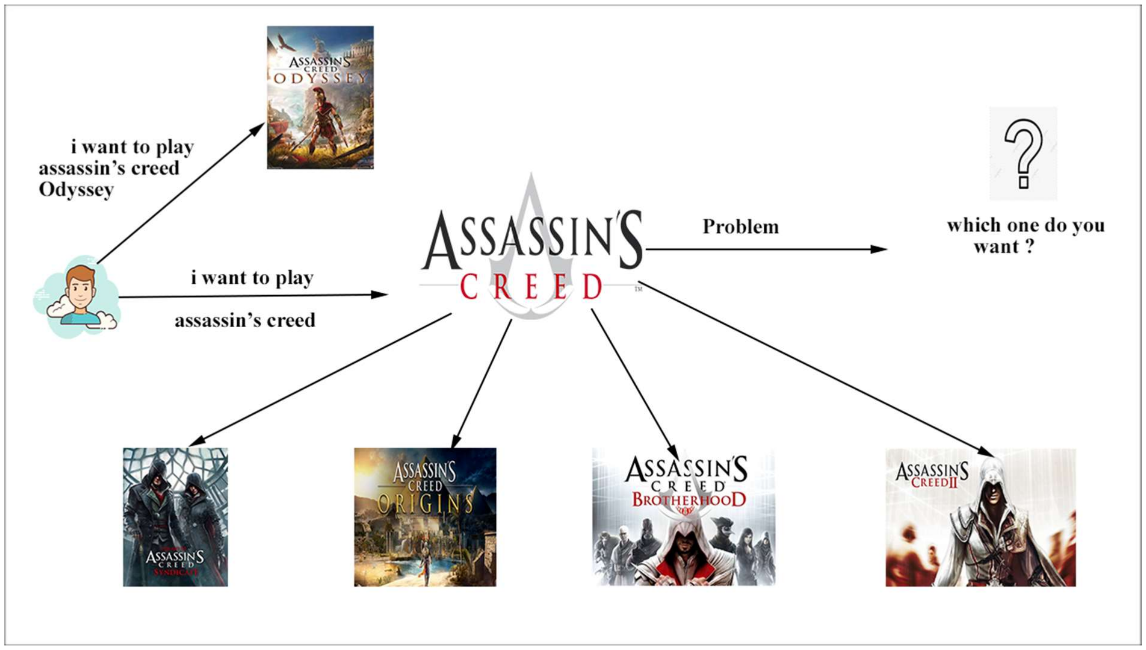


Figure I.5 : Illustration of Ontology approach on choosing a video game.

In the case seen before in Figure I.4, the user chose a specific game so there was no ambiguities, but in this (Figure I.5), the user chose a game which a lot of variation: Assassin's Creed is a series of games; such problem can be modelized using an ontology to help the Chatbot specify the context.

I.4 Chatbots and Recommendation Systems

For many years recommendation systems had been a part of many online shopping systems. But in recent years they are being integrated in many other systems like portals, search engines, blogs, news, Webpages and in our case, Chatbots.

In the Internet age, the biggest problem for a person who wants to buy something online is not how to get enough information about the product, but how to take the right decision with that enormous amount of information. Nowadays, people always search the Internet to find out the proper products and services that they need. Consciously or unconsciously, they depend on the recommendations they get to overcome the information overload. Recommendation systems proved to be a good solution for these problems, by providing more proactive and personalized information services to the users.

I.4.1 Definition

The definition of a recommendation system according to (Schafer et al., 1999) is "*a Recommendation system gives a piece of advice about the products, information or services that the user wanted to know. It is an intelligent application to assist the user in a decision-making process where they want to choose one item amongst the potentially overwhelming ET of alternative products or services. It is also one of the most prominent applications having a substantial impact on the performance of e-commerce sites and the sectors in general.*" [31]

Recommendations systems are used in a large number of e-commerce applications to personalize the content given to their customers. Although recommendation systems suggest items that are based on the person's taste, they can also be used in a more general way to make each application more customer-centric. When people have to make a choice without any personal knowledge of the alternatives, the natural course of action is to rely on the experience and opinions of others. Recommendation systems gather and save recommendations from people who are familiar with these choices, and value these people's perspectives and recognize them as the experts.

I.4.2 Recommendation Systems Techniques

Two principal paradigms for computing recommendations have emerged: Content-Based and Collaborative Filtering. We are going to give more details about these paradigms in the next subsection.

I.4.2.1 **Content-Based Filtering Technique**

This is a conventional technique which is used when dealing with the information overload problems. This filtering technique recommends items for the user based on the descriptions of the items he has previously evaluated. A Recommendation system uses this technique and recommends items which are similar to the ones preferred by the user in the past [32, 33, 34]. In this regard, items (products, services or people) are defined by their associated features and user preferences (stored in user profile) [35] appear considering those associated features in items already rated by users. This is why we associate this kind of systems to *Item's content* rather than *other users' ratings*, an example would clarify the idea (Figure I.6)

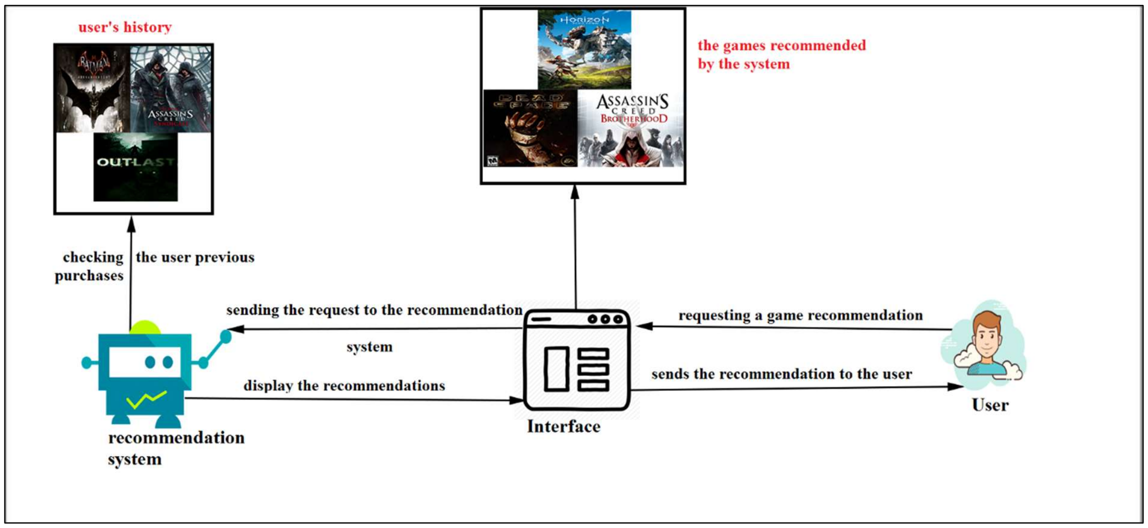

user's history
the games recommended by the system
checking the user previous purchases
sending the request to the recommendation system
display the recommendations
requesting a game recommendation
sends the recommendation to the user
recommendation system
Interface
User

Figure I.6 : Illustration of content-based filtering in the recommendation of a game.

As we can see in (Figure I.6), the recommendation system checked the user's history of purchases, and recommended the closest games (or the most similar ones) to the ones the user purchased.

According to their scores, the items are ranked and presented to the user in a specific order.

However, Content-based approaches have the following weaknesses in recommending good items:

- A user's selection is often based on subjective attributes (quality, style) whereas content-based approaches are based on the objective attributes of the items (content in a document, description of a product etc.).

- Formulating taste and quality is not any easy task.
- These systems can only work when there are at least some items in the user's profile. This is known as the cold start problem.

I.4.2.2 **Collaborative Filtering Technique**

A Complementary technique widely used is collaborative filtering. The basic idea of collaborative filtering is people recommending items to one another. This technique essentially automates the processes of "word of mouth" recommendations. In this technique items are recommended to a user based upon values assigned by other people with similar interests [31]. In this technique the user expresses his/her preferences by rating items that the system presents to them. Then these ratings serve as an approximate representation of their taste in that domain. The system, then, matches these ratings against ratings submitted by other users of the system. The result is the set of that user's "nearest neighbors", this formalizes the concept of people with similar taste that will be used then to make recommendations [31] as (Figure I.7) shows:

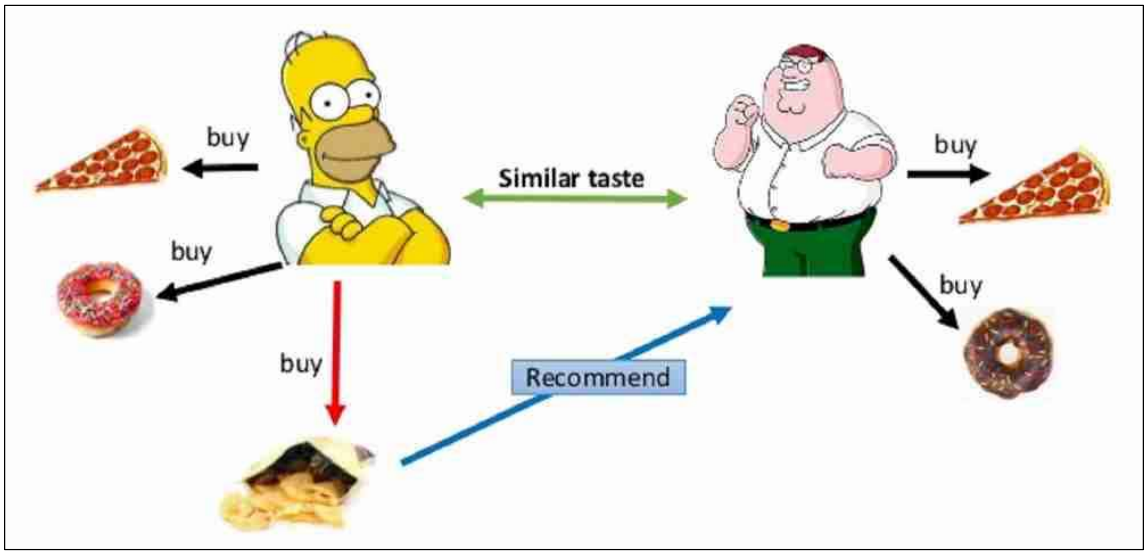


Figure I.7 : Illustration of the collaborative filtering technique.

However, collaborative filtering approaches also have some drawbacks, such as:

- Large number of people must participate to increase the likelihood of finding other users with similar interests.
- It may give poor recommendations, when users share very few information with the system.

- When a new item appears in the database, there would not be any other option except recommending it to random users in order to gather information about the item.

I.4.2.3 **Hybrid systems**

Hybrid recommendation systems combine the two above mentioned recommendation techniques to gain better performance with fewer of the drawbacks of any individual one. A hybrid recommendation system would compute the results of the techniques present in the system using some internal algorithms, before combining them in a single metric to allow consistent ranking. This allows pertinent recommendations from the beginning with a continuous improvement over time by gathering and using more users' information. In a hybrid system, both objective and subjective properties of an item are taken into account in predicting its quality to recommend this item to the recommend seeker [34].

While hybrid system overcomes the shortcomings of pure content-based and pure collaborative system with respect to the objective and subjective properties of recommendations, they do so in a rigid and predetermined manner. Specifically, such systems try to use one of the recommendation properties (either objectiveness or subjectiveness) as complement for the weaknesses of the other, when the latter does not work effectively [34].

Steam uses this approach for its recommendations as shown in (Figure I.8) and (Figure I.9).

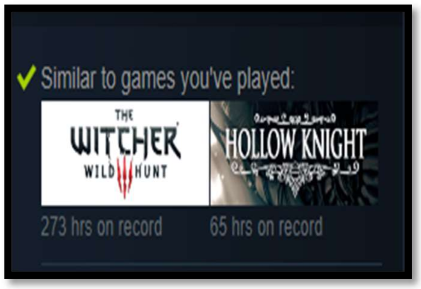
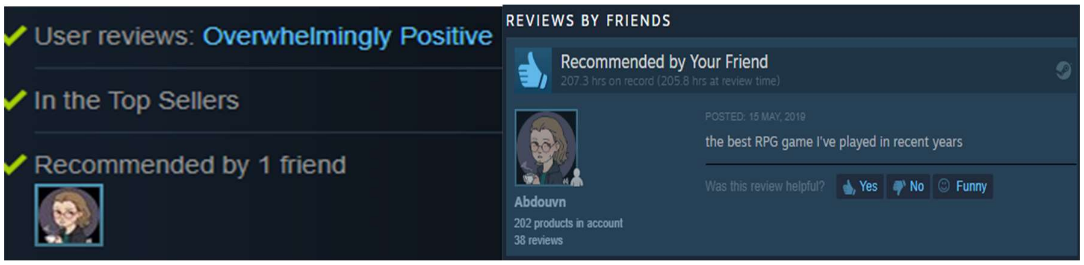


Figure I.8 : Example of the content-based approach on steam.

As shown in (Figure I.8), **Steam** uses content-based approach to recommend games based on previous purchases, where the game "Sekiro" is similar to "The Witcher 3" and "Hollow Knight" which are owned by the user.



User reviews: Overwhelmingly Positive

In the Top Sellers

Recommended by 1 friend

REVIEWS BY FRIENDS

Recommended by Your Friend

207.3 hrs on record (205.8 hrs at review time)

POSTED: 15 MAY, 2019

the best RPG game I've played in recent years

Was this review helpful? Yes No Funny

Abdouvn

202 products in account

38 reviews

Figure I.9 : Example of the Collaborative approach on steam.

(Figure I.9) demonstrates **Steam**'s way in using the collaborative approach, where the user is going to check other users' recommendations and reviews regarding the game he wants to buy.

However, there is no automated way of determining in what circumstances which kind of properties (objective, subjective, or both) are relevant to a particular user in their current context, which means on what points are the other users' reviews and recommendations based on, since every user has a different point of view on the item, making it difficult to distinguish between good and bad recommendations.

In this section, we introduced 3 techniques (or 2 actually) in which we are interested, however, there are other techniques that are most commonly combined with collaborative filtering, but their study is beyond the scope of this thesis.

I.5 **Conclusion**

In this chapter, we introduced the different domains of research regarding our subject, from Conversational Agent and their different types and architectures to Context Management in Chatbots and finally the relation between Chatbots and Recommendation Systems.

In the next chapter, we are going to explain the modeling of our Chatbot, going through the difference between its different versions, and the details about the Chatbot's components.

Chapter II:

Modeling of the Chatbot

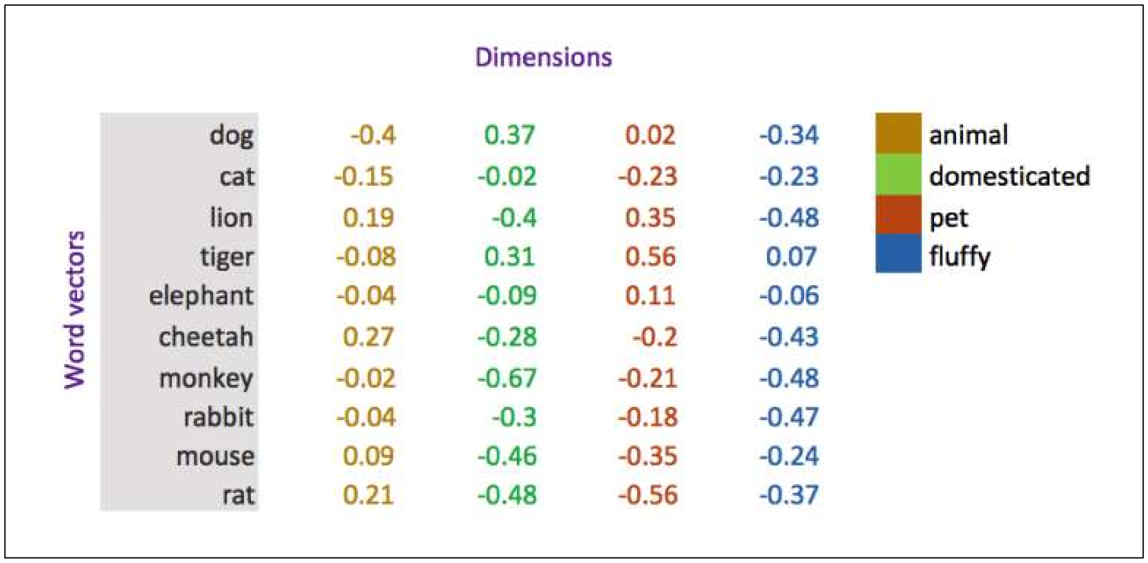
II.1 Introduction

After introducing the state of the art of our project, in this chapter will focus on presenting the conception of our Chabot, which describes the different states of evolution concerning our vision of the Chabot from the first to the final version. But first we will introduce some ML (Machine Learning) and NLP (Natural Language Processing) techniques used in practice in our Chatbot.

II.2 Machine Learning and Natural Language Processing techniques

II.2.1 Word Embedding

Word embeddings are vector representations of words, meaning each word is converted to a dense numeric vector that can be fed to a machine learning algorithm such as Neural Networks. There are multiple ways of performing this operation, from simple BOW (Bag-of-Words) that focus on counting words, to more advanced techniques like Word2Vec and GloVe that focus on capturing the semantic and syntactic aspects of words, where similar words are represented by similar vectors. The following figure illustrates an example of word embedding based on semantically comprehensible parameters.



Word vectors	Dimensions			
dog	-0.4	0.37	0.02	-0.34
cat	-0.15	-0.02	-0.23	-0.23
lion	0.19	-0.4	0.35	-0.48
tiger	-0.08	0.31	0.56	0.07
elephant	-0.04	-0.09	0.11	-0.06
cheetah	0.27	-0.28	-0.2	-0.43
monkey	-0.02	-0.67	-0.21	-0.48
rabbit	-0.04	-0.3	-0.18	-0.47
mouse	0.09	-0.46	-0.35	-0.24
rat	0.21	-0.48	-0.56	-0.37

🟫 animal
🟩 domesticated
🟥 pet
🟦 fluffy

Figure II.1 : Word embedding by example

Each word in (Figure II.1) is represented with a vector, the word “cat” for example is represented with the vector [-0.15, -0.02, -0.23, -0.23], and each value in each dimension (or case) has its semantic meaning.

II.2.2 Neural Networks

Neural networks are a set of Machine Learning algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

The main use of Neural Networks is clustering and classification, they help to predict the labels of the input data using the knowledge gathered from the labeled dataset used for the training.

The following figure is an illustration of a Neural Network.

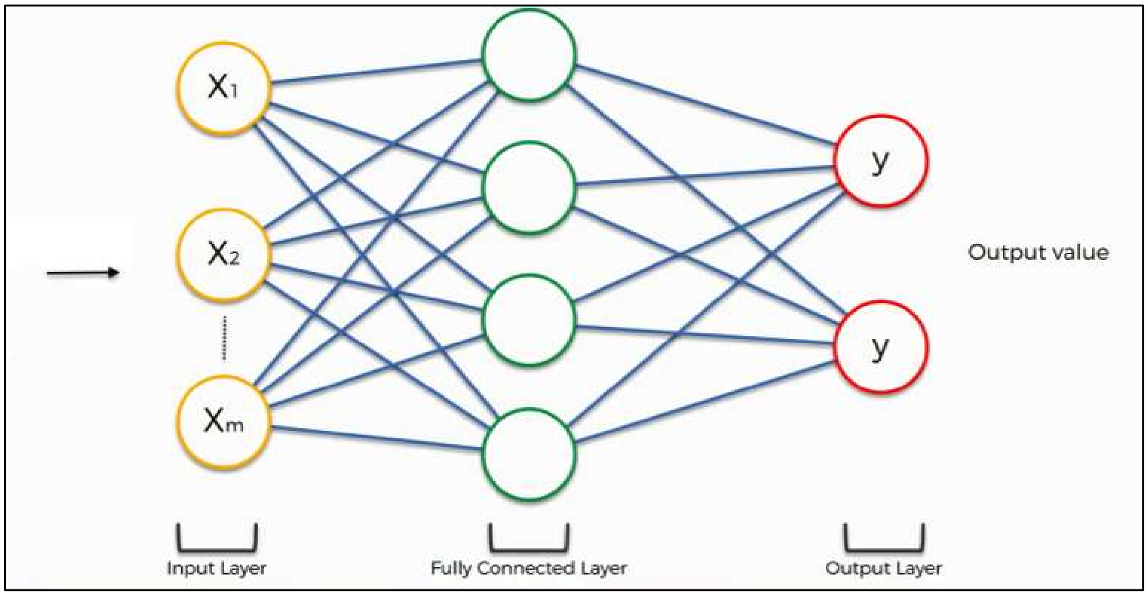


Figure II.2 : Illustration of a Neural Network.

In the figure II.2, the values X1 to Xm represent the values of the input vector, and the y values represent the predicted output vector.

The values of the Ys depend on the use case of the Neural Network, and the functions used in it. In our case, the Ys are going to be the probabilities of the input belonging to a certain class.

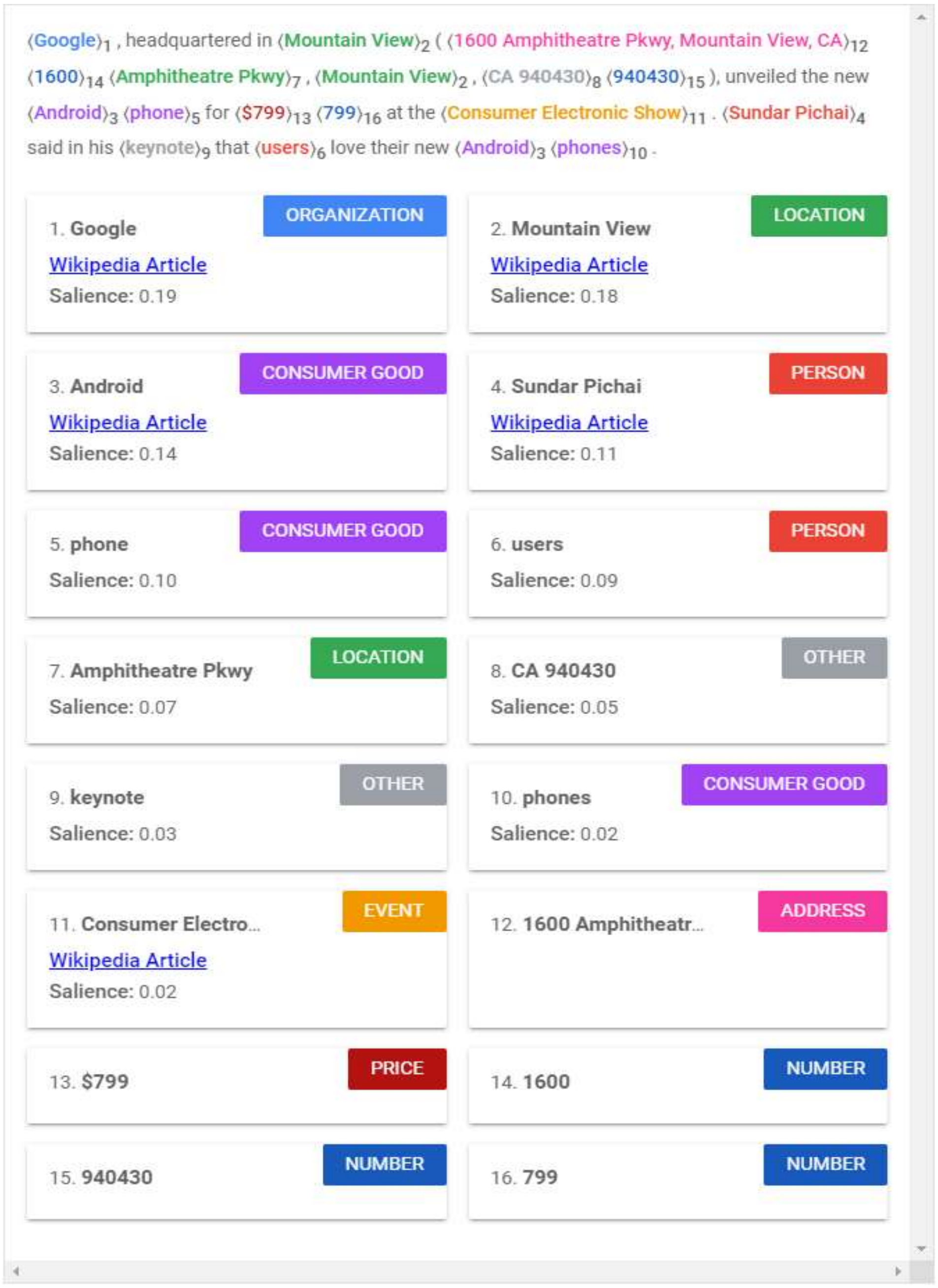
The neurons of the layer in the middle (called hidden layer) are used to calculate the weighted sum of inputs and weights, execute some functions that are predefined by creator of the Neural Network, and pass the result to the next layer.

The arcs represent weights, which are used to calculate the values of the neurons, these weights are calculated during the training phase of the Neural Network using a loss function and with the help of some optimization function, and again, these functions are going to be chosen based on the use case.

II.2.3 Named Entity Recognition

Named Entity Recognition (NER) is the process of extracting the entities from a given text, an entity in text may be a business, location, person name, time, a game or a genre in our case etc. An object that has a meaning in the query, and will have further meaning in the bot's logic. These entities are then used as inputs for some functionalities in the Chatbot.

(Figure II.3) represents some entities extracted from a text using Google's Natural Language API.



⟨Google⟩$_{1}$, headquartered in ⟨Mountain View⟩$_{2}$ (⟨1600 Amphitheatre Pkwy, Mountain View, CA⟩$_{12}$ ⟨1600⟩$_{14}$ ⟨Amphitheatre Pkwy⟩$_{7}$, ⟨Mountain View⟩$_{2}$, ⟨CA 940430⟩$_{8}$ ⟨940430⟩$_{15}$), unveiled the new ⟨Android⟩$_{3}$ ⟨phone⟩$_{5}$ for ⟨$799⟩$_{13}$ ⟨799⟩$_{16}$ at the ⟨Consumer Electronic Show⟩$_{11}$. ⟨Sundar Pichai⟩$_{4}$ said in his ⟨keynote⟩$_{9}$ that ⟨users⟩$_{6}$ love their new ⟨Android⟩$_{3}$ ⟨phones⟩$_{10}$.

1. Google — ORGANIZATION
Wikipedia Article
Salience: 0.19

2. Mountain View — LOCATION
Wikipedia Article
Salience: 0.18

3. Android — CONSUMER GOOD
Wikipedia Article
Salience: 0.14

4. Sundar Pichai — PERSON
Wikipedia Article
Salience: 0.11

5. phone — CONSUMER GOOD
Salience: 0.10

6. users — PERSON
Salience: 0.09

7. Amphitheatre Pkwy — LOCATION
Salience: 0.07

8. CA 940430 — OTHER
Salience: 0.05

9. keynote — OTHER
Salience: 0.03

10. phones — CONSUMER GOOD
Salience: 0.02

11. Consumer Electro... — EVENT
Wikipedia Article
Salience: 0.02

12. 1600 Amphitheatr... — ADDRESS

13. $799 — PRICE

14. 1600 — NUMBER

15. 940430 — NUMBER

16. 799 — NUMBER

Figure II.3 : Screenshot from Google's Natural Language Api's NER example

II.3 First version of the Chatbot

The first iteration focused on the main task of any Chabot which is chatting with the user and responding according to a distinctive pattern as shown in (Figure II.4):

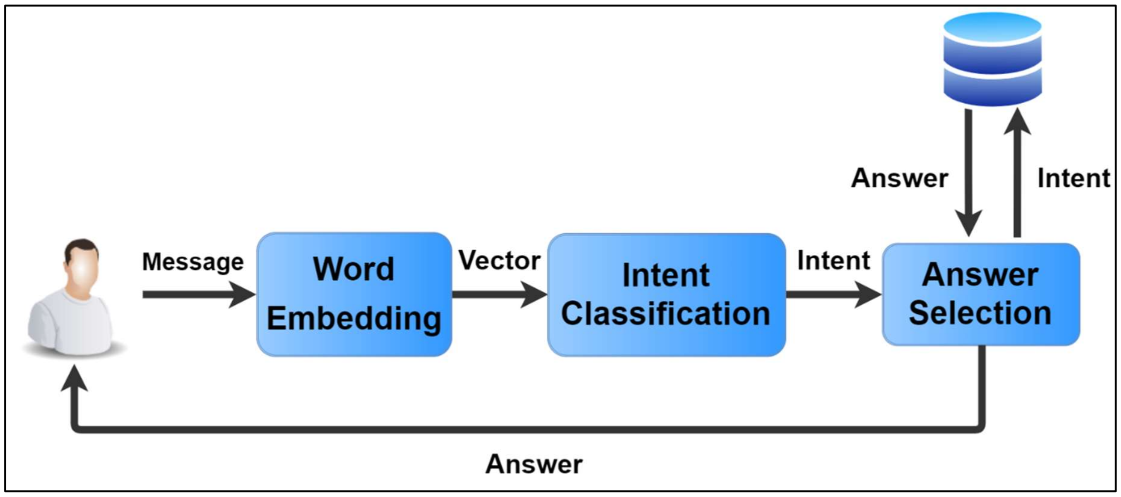


Figure II.4 : Illustration of our first approach.

As shown in (Figure II.4), this approach was composed of three parts that treat the user's input:

II.3.1 Word embedding

At this point the word embedding technique we were using was the Bag-of-Words technique, which means neither the semantic relations between the words nor the words order were taken into consideration, and only phrases with the same words density were considered similar.

The next figure (Figure II.5) represents two simple cases of word embedding using BoW method.

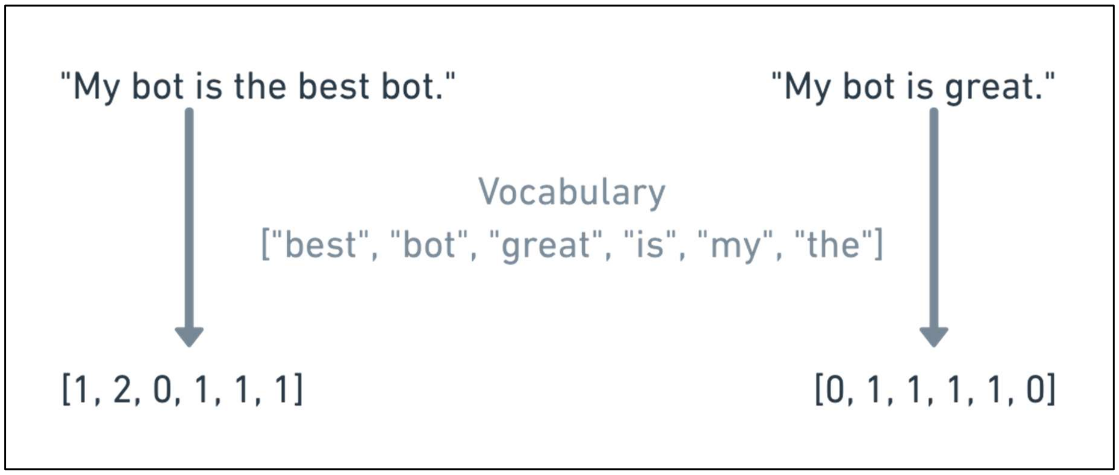

"My bot is the best bot."
"My bot is great."
Vocabulary
["best", "bot", "great", "is", "my", "the"]
[1, 2, 0, 1, 1, 1]
[0, 1, 1, 1, 1, 0]

Figure II.5 : Word embedding using the BOW method

As we can see in the figure, first a vocabulary set is formed. Then, the word embedding technique consists of generating vectors having the same size as the vocabulary vector where the items represents the frequency of the vocabulary's token in the case (sentence).

II.3.2 **Intent classification**

This part is composed of a Neural Network that was trained with a list of labeled word vectors, where each vector (representing a message) was labeled with an intent (i.e. greeting, goodbye). That Neural Network is then used to predict the intent of the user's input.

(Figure II.6) illustrates how the Neural Network is trained, and how it is used to classify the intents, and (Figure II.7) represents a -close to real world- example of the intent classification process.

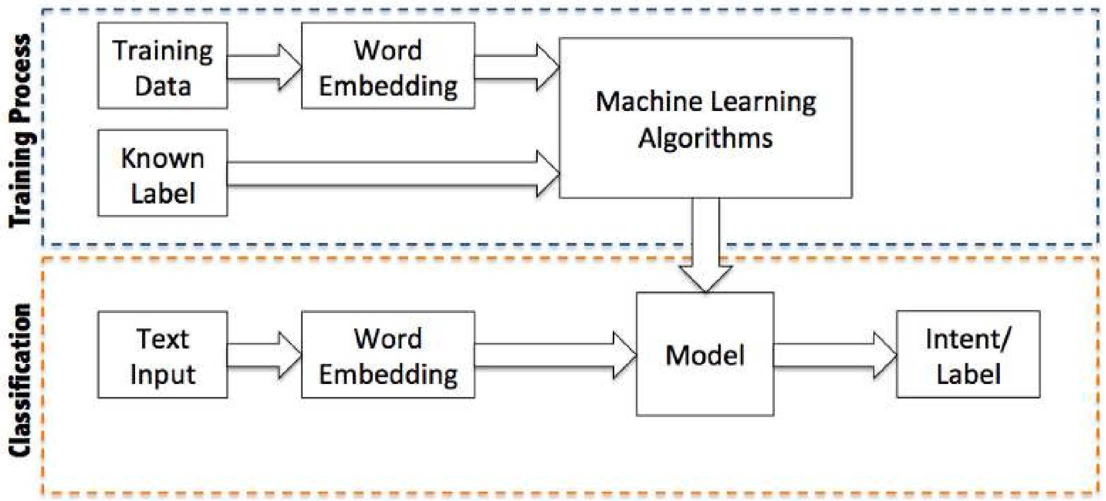


Figure II.6 : Illustration of intent classification using supervised machine learning.

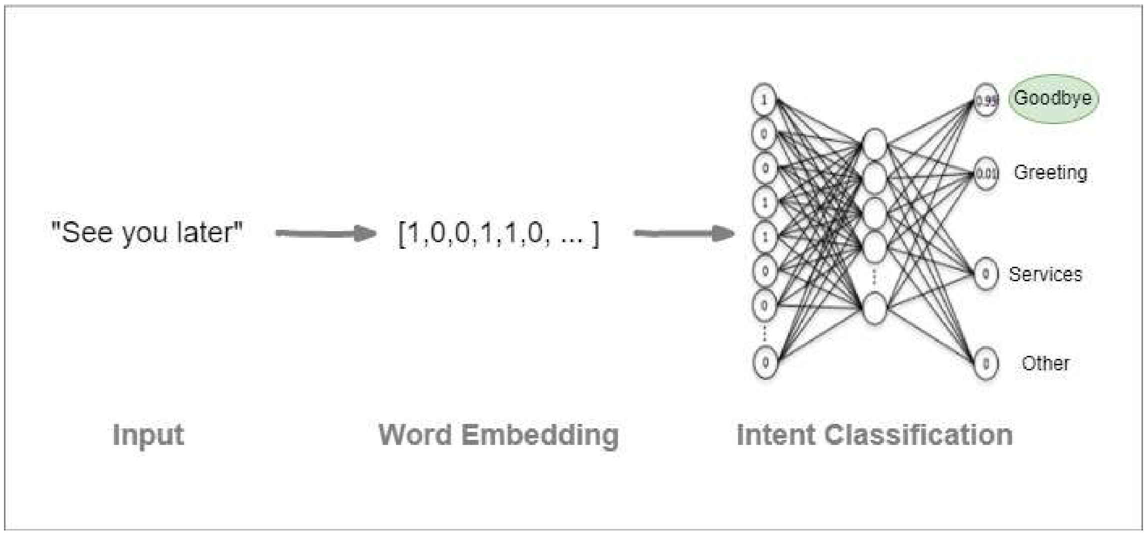
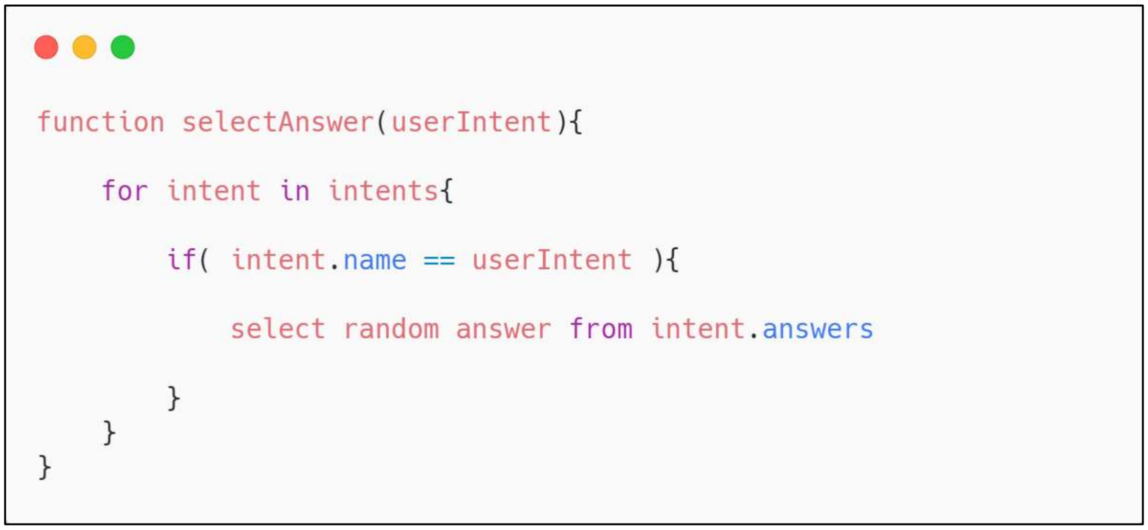


Figure II.7 : Example of intent classification

II.3.3 Answer selection

This is the simplest part, when receiving the user's intent, this component was responsible of selecting an answer for that intent from the predefined answers (each intent was given one or many possible answers).



```
function selectAnswer(userIntent){

    for intent in intents{

        if( intent.name == userIntent ){

            select random answer from intent.answers

        }
    }
}
```

Figure II.8 : Pseudo-code explaining the answer selection process.

This first version was very limited, one of its main flaws was that the intents predicted were not accurate due to the simplicity of the word embedding technique, but it was a stepping stone to make a better version of the Chatbot. The newer version is an improvement of this first one.

II.4 Second version

The illustration in (Figure II.9), represents the two main Chatbots architectures (or aspects), which are: task-oriented dialogs and data-driven dialogs.

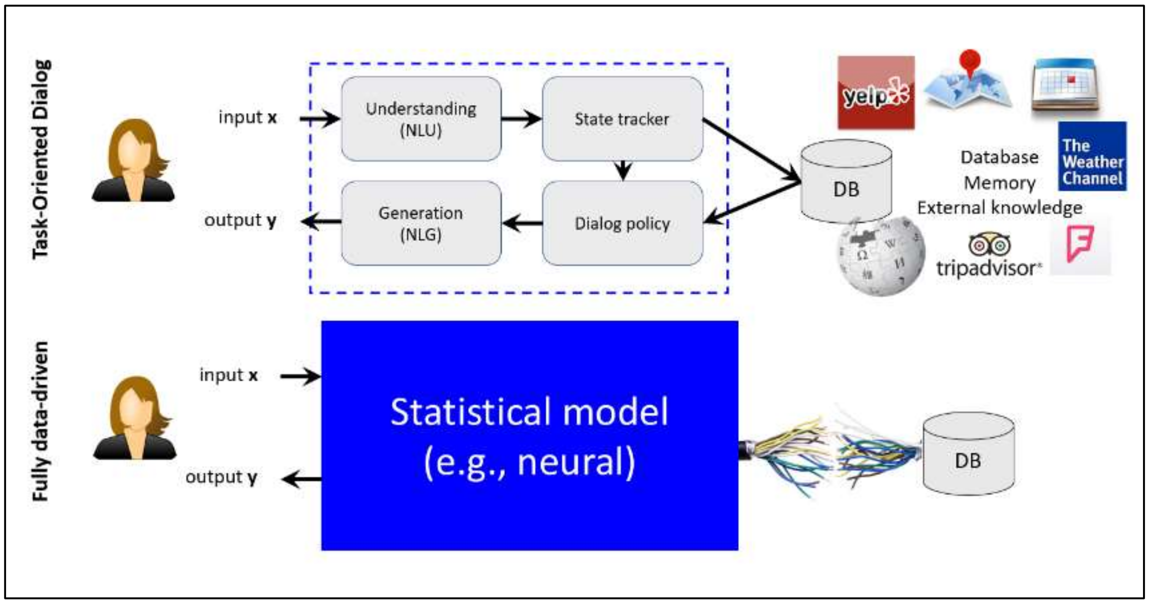


Figure II.9 : Fully data driven VS task-oriented architectures.

In what follows, we will talk about each architecture separately.

II.4.1 Task-Oriented Dialog

This architecture focuses on the tasks management aspect (like booking a trip, setting a reminder, quick searching etc.), where a call to the database or to an API is needed, and where the answer is not predefined but generated according to the situation.

II.4.2 Fully Data-driven

This architecture is used for simple chatting, where the answers are generated by a pre-trained Neural Network or any other technique. It uses data gathered from real human-human conversation to generate answers.

Choosing an architecture depends on the goals set for the Chatbot, sometimes both architectures are mixed together to give more flexibility to the Chatbot. In our case, since we could not get real data of conversation between humans in the field of Gaming, we needed to turn our focus to the task-oriented dialogs, without ignoring the simple small talk aspect.

II.5 Conceptual model

For the second version of the Chatbot, we made a lot of changes, we improved what was existing, and we added more functionalities like the entity extraction and the user creation, as shown in (Figure II.10)

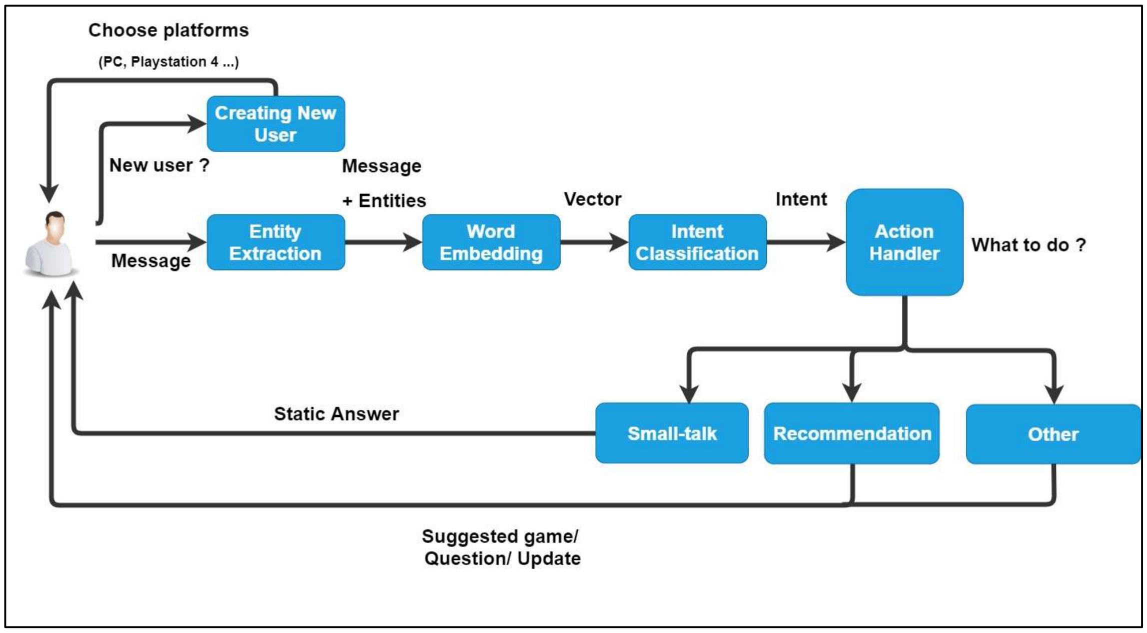


Figure II.10 : The new conception of the Chatbot.

We will now discuss each component of (Figure II.10)

II.5.1 Entity Extraction

The first phase of the text treatment which aims to recognize the different entities in the user's message (Named Entity Recognition), which in our case are either names of games, or types (genres), the entities will be then replaced with special words producing a mutated message. The mutation technique helps to extremely reduce the amount of training data required for the intent classification model, and to improve the classification of the new messages, the next figure will explain the difference between the initial and the mutated message:

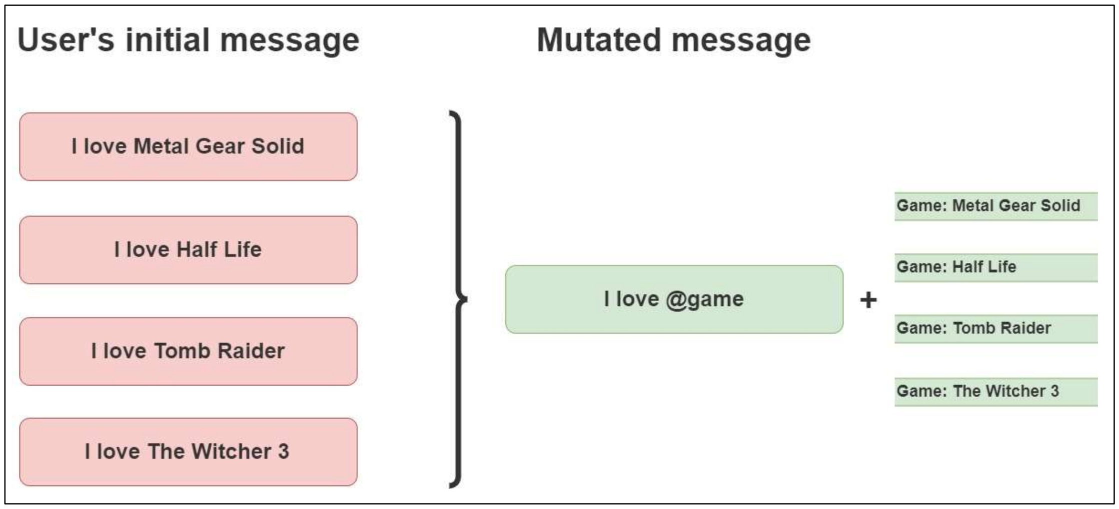


Figure II.11 : Illustration of the entity extraction phase.

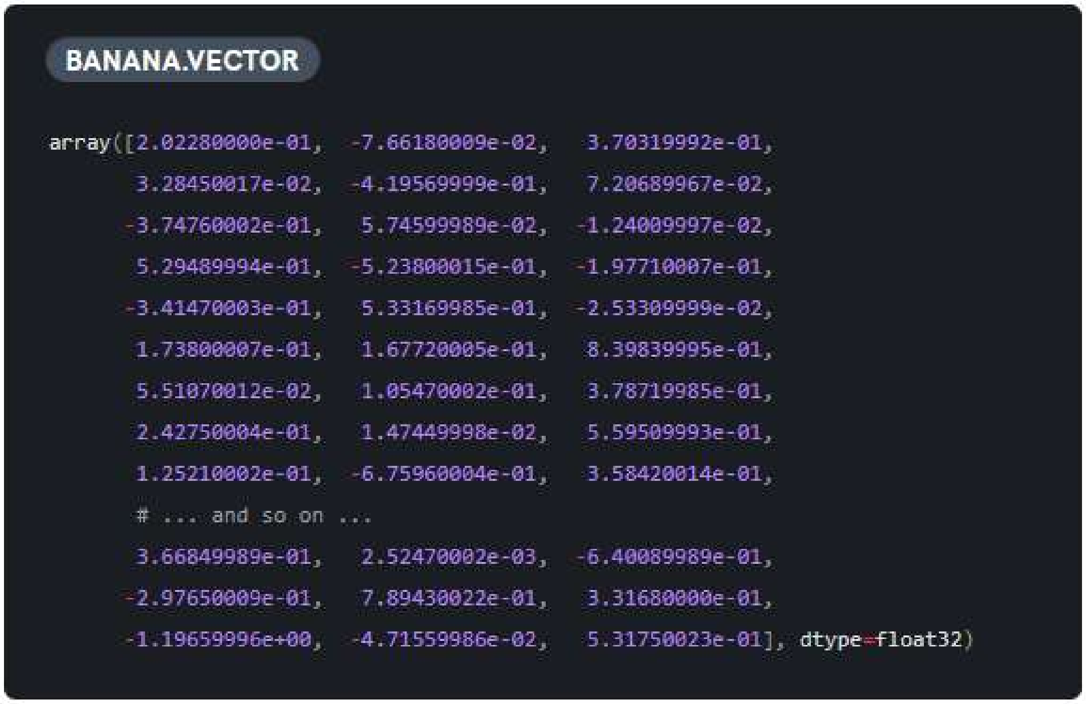
As shown in the previous figure, the entities which are names of games in this case, are replaced with special words, and the real entities are stored separately and passed to the next step of the treatment.

II.5.2 Word Embedding

The BoW word embedding technique was clearly insufficient, so we decided to use a better word embedding technique, and thus came the idea to use spaCy[4] library which generates word vectors for each word using a word2vec algorithm, the special distance between these vectors represents the semantic similarity between words, and so intent classification process became much more accurate and able to identify words that were not even in the training data, the luxury that was not available in our first version.

The following figure represents the word vector generated by spacy for the word banana from their official website.

[4] https://spacy.io/



```
BANANA.VECTOR

array([2.02280000e-01,  -7.66180009e-02,   3.70319992e-01,
        3.28450017e-02,  -4.19569999e-01,   7.20689967e-02,
       -3.74760002e-01,   5.74599989e-02,  -1.24009997e-02,
        5.29489994e-01,  -5.23800015e-01,  -1.97710007e-01,
       -3.41470003e-01,   5.33169985e-01,  -2.53309999e-02,
        1.73800007e-01,   1.67720005e-01,   8.39839995e-01,
        5.51070012e-02,   1.05470002e-01,   3.78719985e-01,
        2.42750004e-01,   1.47449998e-02,   5.59509993e-01,
        1.25210002e-01,  -6.75960004e-01,   3.58420014e-01,
        # ... and so on ...
        3.66849989e-01,   2.52470002e-03,  -6.40089989e-01,
       -2.97650009e-01,   7.89430022e-01,   3.31680000e-01,
       -1.19659996e+00,  -4.71559986e-02,   5.31750023e-01], dtype=float32)
```

Figure II.12 : spaCy vector of the word Banana.

II.5.3 Intent Classification

This part is constructed of a Neural Network that classifies the intent of the user's message, same as the previous version, the only difference this time is the form of the input. More technical details are to be introduced in the next chapter.

II.5.4 Action Handler

This is the core of the whole project, it is the part that is responsible for deciding what to do depending on the situation, which includes the incoming user's message, the state of the conversation, and the state of the database. The possible outcomes are recommended games, questions to complete an unfinished task and static answers in the case of a simple conversation.

Behind the scenes, this part tracks the changes in the context which includes the current or unfinished task and the game or genre that the conversation is based on, it also interacts with the database, from adding games to the user's list of liked or disliked games, to retrieving the recommended games depending on the recommendation method. An example of how the action handler works is when the current unfinished task is "Recommend" and the incoming message's intent is "Another", the Chatbot will suggest another game with the same parameters, whereas if the incoming message represents a “Recommend” intent and has new parameters (game or genre), the Chatbot will readapt the recommendation based on the new parameters.

II.6 **Requirement specification and analysis**

In this section, we are going to present our modeling of the **Chatbot** with a more straightforward approach principally focused on the different scenarios that can occur during the interactions between the user and the **Chatbot.**

We are going to introduce the different scenarios in form of sequence diagrams each one representing a case of our **Chatbot**, but first we are going to present the general model of the possible cases that can occur as shown in the following use case diagram (Figure II.13.)

II.6.1 **Use-Case Diagram**

Figure II.13 hereafter, represents our general use case. It details all the actions the users are able to perform in the platform.

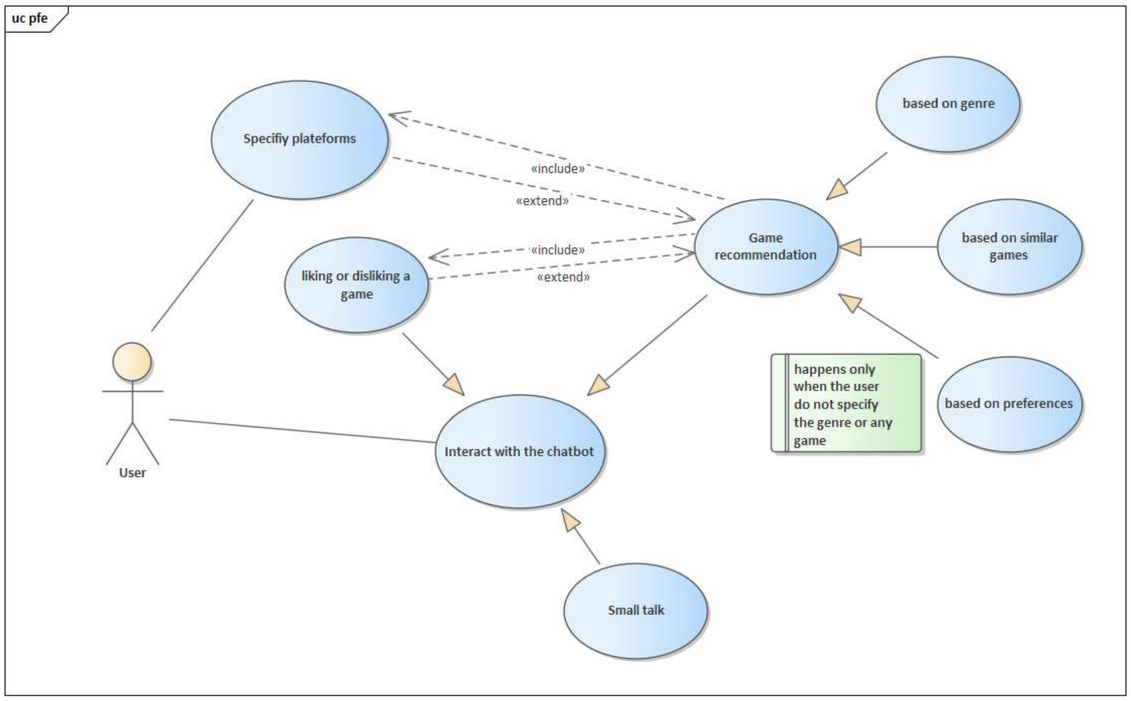


Figure II.13 : Use case diagram representing the different cases of the Chatbot

As shown in the previous diagram (Figure II.13), we can distinguish different entities composing our diagram which are: **Actors and Cases.**

Concerning the actors, we only have one actor who is the **User** since the different interactions are related to the user's needs.

On the other hand, there are different cases which can occur depending on certain conditions, we can summarize them in the following table):

Table II.6-1 : Table summarizing the cases of the use case diagram

Cases	Description
Specify platforms	Each new user is required to specify the video games platforms he is playing on, helping the Chatbot to create a specific profile for that user. As for the platforms, we are going to use the most common ones which are: PS3-PS4-PC-Xbox360 and Xbox One
Interact with the Chatbot	Or in other words, conversing with the chatbot, it regroups the different functionalities of the chatbot which are: • Small talk • Game recommendation • Liking/disliking a game • Other
Small talk	This case concerns the "casual chat" aspect of our Chatbot where the user is going to have normal conversation with the Chatbot.
Liking or disliking a game	In this case, the user informs the Chatbot about his liked and disliked games, this case extends another case which is Game recommendation and more precisely the one Based on preferences.
Game recommendation	This case which is a generalization, concerns the user asking for a game recommendation and it is divided into three different cases which are cited below. This case includes the **specify platforms** case since the bot cannot suggest games without the user specifying the platforms he is playing on and the **Liking or disliking a game** case since in case the user asks for recommendation based on preferences, the bot needs to check the user's likes and dislikes lists.
Based on genre	This is the case of the user asking for a recommendation based on a given genre.
Based on another game	This case happens when the user asks for recommendation based on a given game, which means asking for similar games to that given one.
Based on preferences	This case happens only when the user does not specify a genre or a game, in this case the Chatbot uses the lists of liked and disliked games gathered from the user to find similar games.

II.6.2 **Sequence diagram**

In this subsection, we are going to present for each case a sequence diagram to show the course of events related to it.

II.6.2.1 **Specify platforms**

Occurs only one time per user who wants to interact with the bot, as the platforms specification is required for further interactions with the bot.

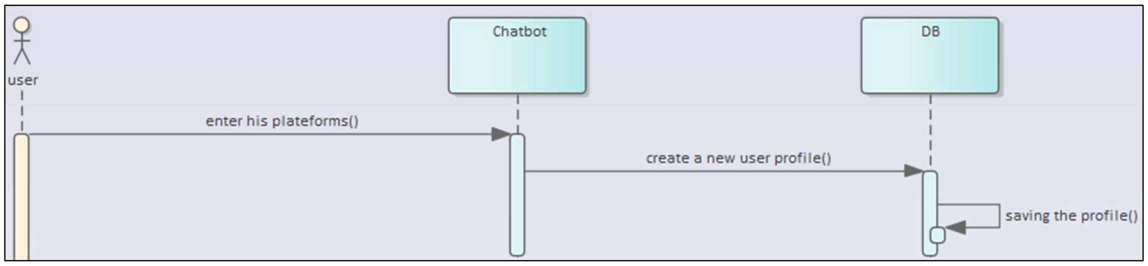


Figure II.14 : Sequence diagram of the "specify platforms" case

As shown in (Figure II.14), the "specify platforms" case includes the user, the Chatbot and the database, this operation is going to happen only one time per user so if a new user wants to interact with the Chabot, he needs to specify his platforms first so the bot can create a new user profile, as it extends another case as shown in (Figure II.13***Figure II.13***) which is **"Game recommendation".**

II.6.2.2 **Game recommendation**

Since the "**interact with the Chatbot**" case is just a generalization notion, we are going to focus on the precise cases of that generalization starting by the case of **Game recommendation**, as seen in (Figure II.13), this case includes the **specify platforms** case as the recommendation cannot happen without specifying the platforms that the user uses, and also includes the **Liking or disliking a game** case since the recommendation **based on preferences** is based on the user's games lists.

Game recommendation itself is also a generalization, and thus we will talk about each of its cases separately.

1) Based on Genre

As suggested by the name, this recommendation aspect is based on the game's genre asked by the user and the different genres available in the bot's database.

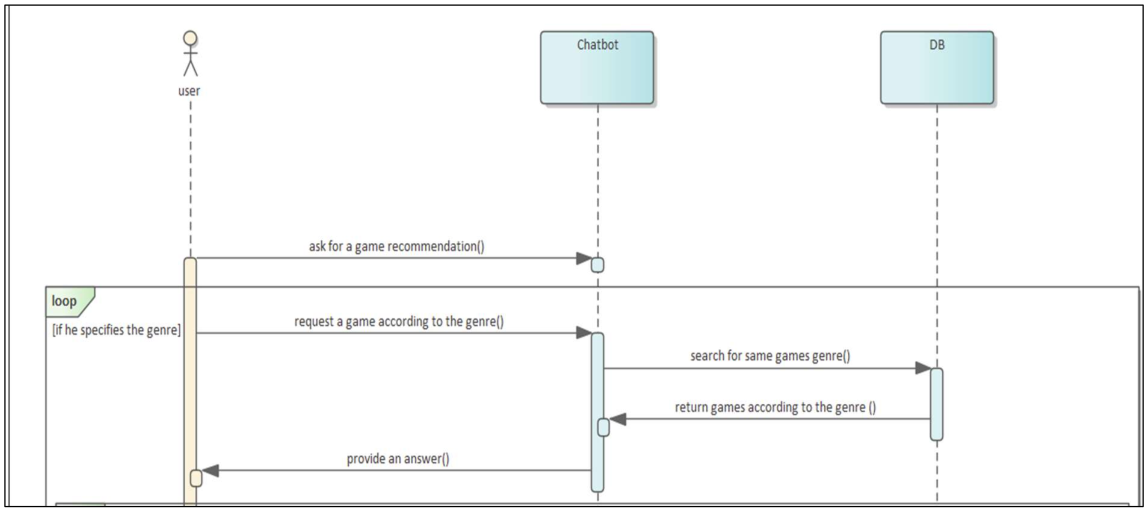

user
Chatbot
DB
ask for a game recommendation()
loop
[if he specifies the genre]
request a game according to the genre()
search for same games genre()
return games according to the genre ()
provide an answer()

Figure II.15 : Sequence diagram of game recommendation based on genre.

As shown in (Figure II.15), the game recommendation based on genre is when the user asks for recommendations in a specific genre (adventure-horror...etc.) and so the Chatbot will start suggesting games in that genre **which do not already exist in the user's liked and disliked lists,** and according to his reaction, multiple scenarios may occur, note that these scenarios are the same for the other two recommendation cases (Based on another game, Based on preferences).

a) "Already played" scenario

This scenario occurs when the user specify that he already played the game proposed by the bot.

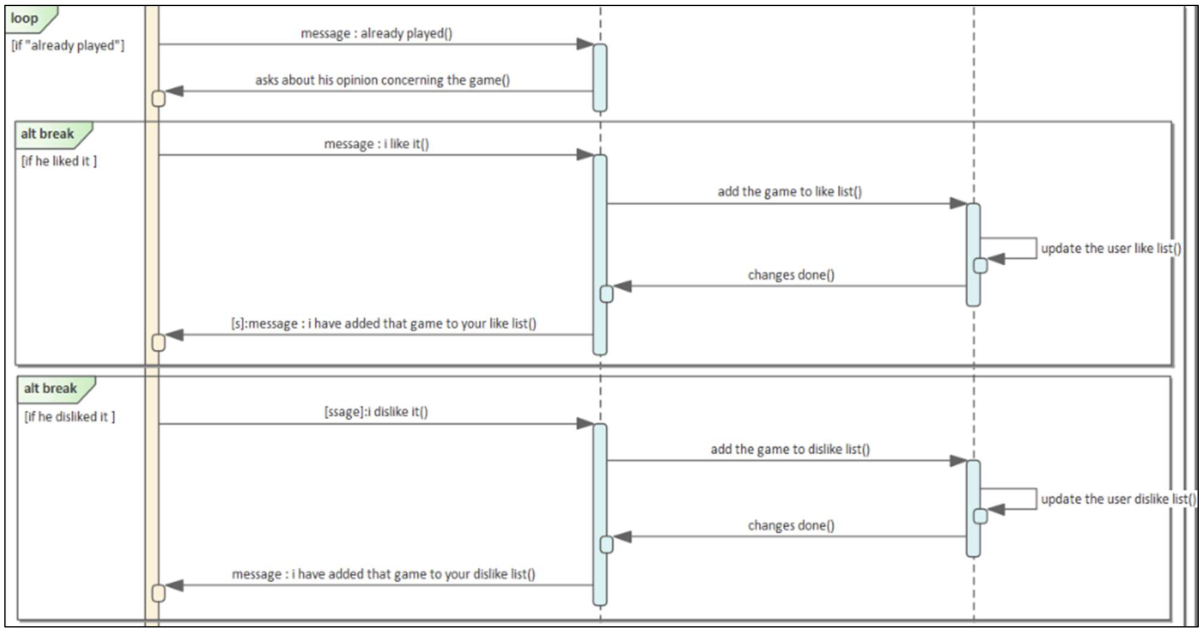

loop
[if "already played"]
message : already played()
asks about his opinion concerning the game()
alt break
[if he liked it]
message : i like it()
add the game to like list()
update the user like list()
changes done()
[s]:message : i have added that game to your like list()
alt break
[if he disliked it]
[ssage]:i dislike it()
add the game to dislike list()
update the user dislike list()
changes done()
message : i have added that game to your dislike list()

Figure II.16 : Sequence diagram of the "already played" scenario

As seen in (Figure II.16), when the user says he already played the suggested game, the **Chatbot** is going to ask him about his opinion regarding that game which in our case is going to be either liked or disliked, this will help to know more about his likes and dislikes.

In case, the user did like the game proposed by the bot, the bot is going to add that game to the user "liked list".	In case, the user did not like the game proposed by the bot, the bot is going to add that game to the user "dislike list" and update his profile in the database.

b) "Another One" scenario

This scenario occurs when the user wants another game rather than the one proposed.

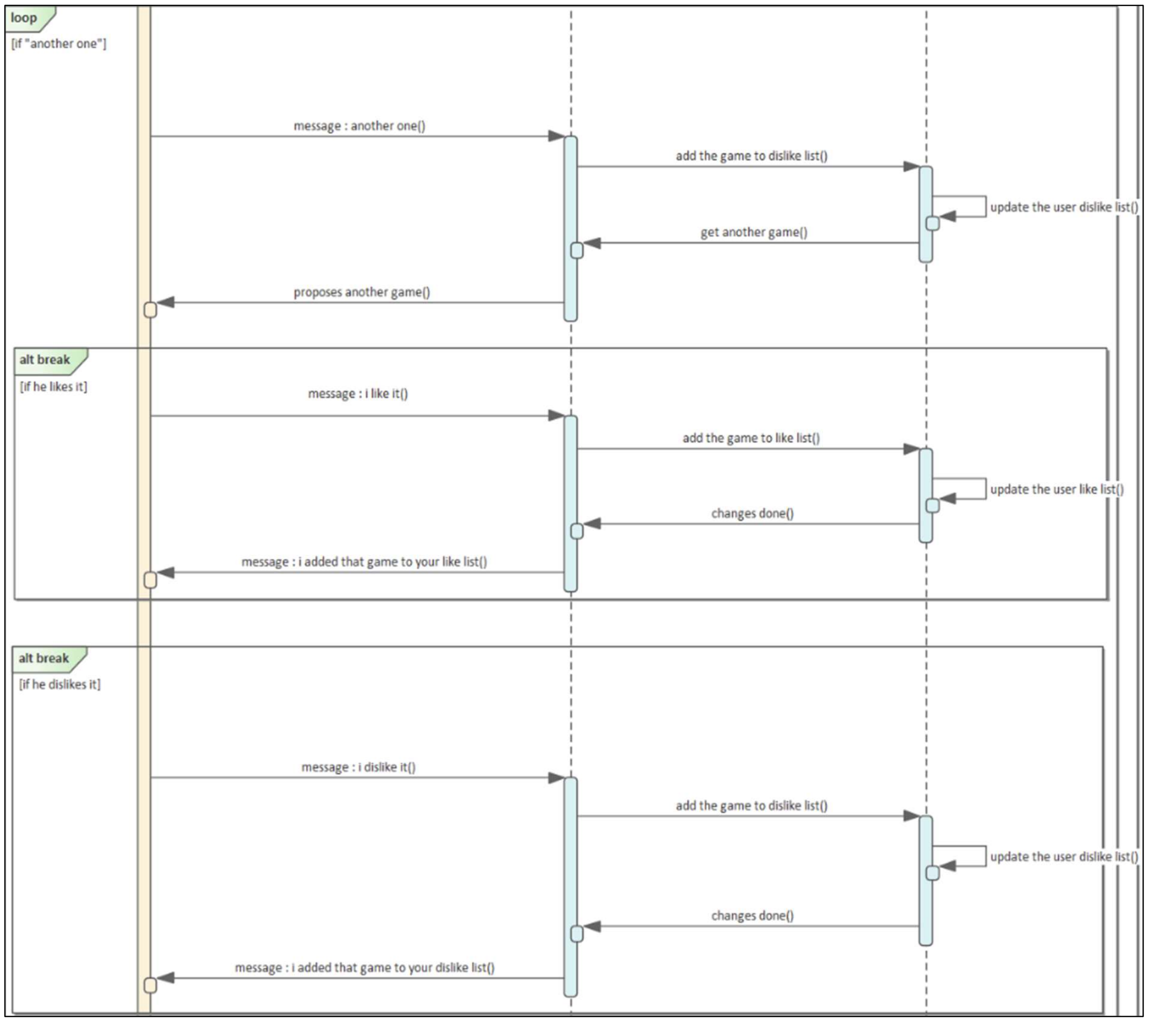

loop
[if "another one"]
message : another one()
add the game to dislike list()
update the user dislike list()
get another game()
proposes another game()
alt break
[if he likes it]
message : i like it()
add the game to like list()
update the user like list()
changes done()
message : i added that game to your like list()
alt break
[if he dislikes it]
message : i dislike it()
add the game to dislike list()
update the user dislike list()
changes done()
message : i added that game to your dislike list()

Figure II.17: Sequence diagram of the "Another one" scenario

As seen in (Figure II.*17*), when the user says he wants another game than the one recommended is going to be added to his disliked games, in order not to suggest it again, and the next game is going to be proposed to him.

Another possibility is the user explicitly saying he liked or did not like the suggested game; in that case it is going to be added to his liked or disliked list accordingly.

The recommendation loop breaks with either the user saying he will try the suggested game, or by him changing the subject of the conversation.

2) Based on game similarity

This concerns the case which the user is going to ask about a game recommendation based on a game similarity between the game proposed by the user and the games similar in the bot's database.

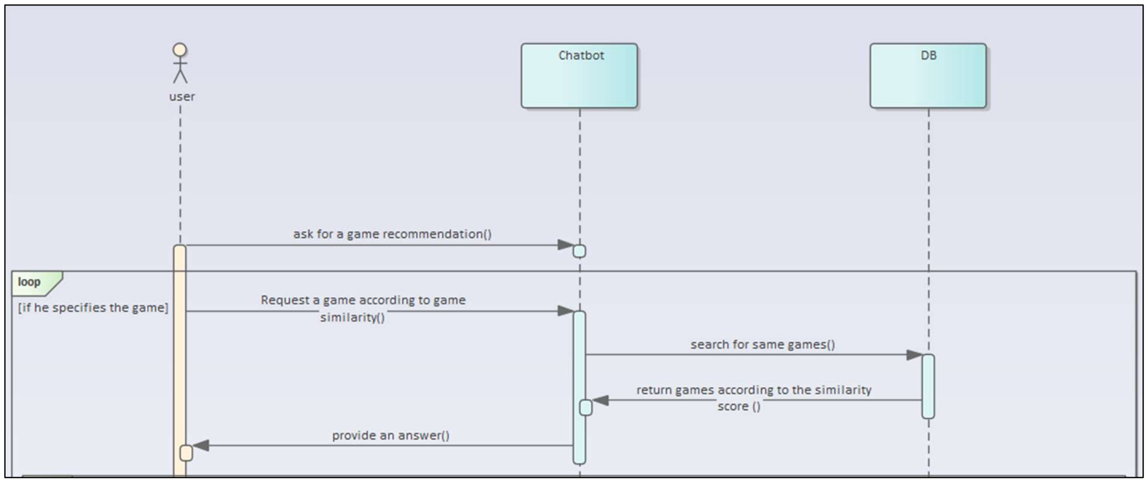

user
Chatbot
DB
ask for a game recommendation()
loop
[if he specifies the game]
Request a game according to game similarity()
search for same games()
return games according to the similarity score ()
provide an answer()

Figure II.18 : Sequence diagram of game recommendation based on game similarity

As shown in (Figure II.18), the game recommendation based on game similarity is going to recommend games similar to the one given by the user, for example if he asks for games similar to "**The Witcher 3**", the Chatbot will look for similar games which are either in the same genre or from the same publisher, we are going to give more detail about it in the implementation section.

3) Based on preferences

Case which happens when the user does not specify any genre or any game similar task , the bot is going to recommend games according to the user's preferences.

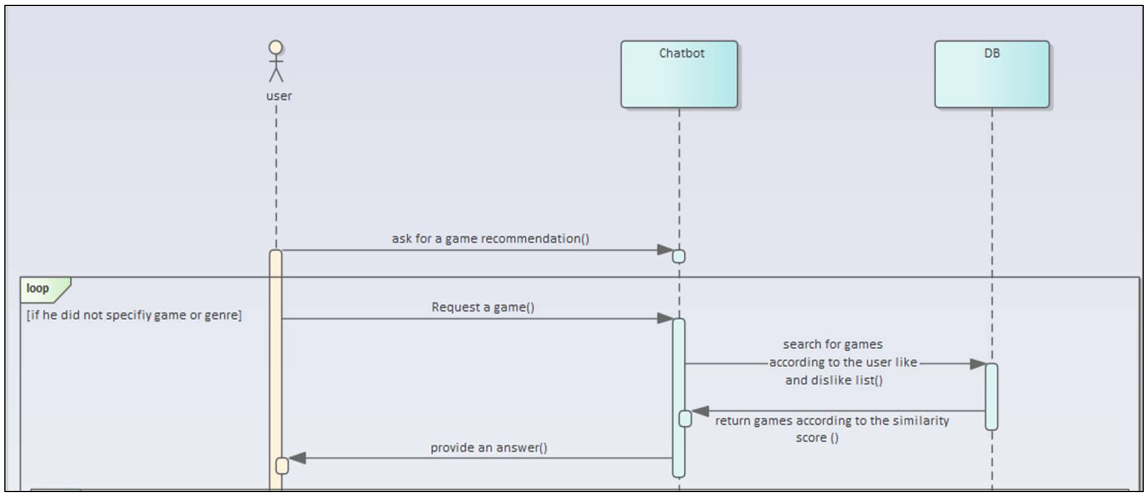

user
Chatbot
DB
ask for a game recommendation()
loop
[if he did not specifiy game or genre]
Request a game()
search for games according to the user like and dislike list()
return games according to the similarity score ()
provide an answer()

Figure II.19 : Sequence diagram of game recommendation based on user's preferences

As shown in (Figure II.19), the game recommendation based on user's preferences focuses on recommending games according to the user's like and dislike lists, this type of recommendation does not exclude any genre or any publisher, but instead it runs on the whole list of existing games, and attributes extra score to the games that are similar to his liked games, and less score to the ones similar to his disliked games. With this method, the Chatbot will be able to suggest games that have high critic scores but are not necessarily similar to his liked ones.

II.7 **Conclusion**

In this chapter, we presented our vision regarding the conception of our Chatbot going from a simple perspective model to a more detailed and complex one. We tried to present an explanation on every aspect of our approaches and their different versions as well as a UML representation to simplify that complex vision introduced at first and clarify what are the Chatbot's purposes and capabilities.

In the next chapter, we are going to talk about the technical details of the solution that we came up with and the implementation of the Chatbot.

Chapter III:
Solution Implementation and Evaluation

III.1 Introduction

In the previous chapters, we mostly talked theoretically about Chatbots in general, and our Chatbot in particular. In this chapter, we are going to present the implementation of our system, from the development environment to the details about each step of development.

III.2 Tools and libraries

This section we be dedicated to the libraries and tools that we used to develop our solution.

III.2.1 Python

Python[5] is a general purpose and high-level programming language that can be used for a wide variety of applications. It was created by Guido van Rossum, and first released on February 20, 1991.

Python is one of the most popular programming languages used by developers today[6], and it is by far the most used language in the field of Artificial Intelligence. Most of the NLP and Machine Learning libraries are available in Python, and thus it is the best choice for our case.

Figure III.1 : Python logo.

[5] https://www.python.org/

[6] Number 1 in demand programming languages to learn in 2020 according to https://towardsdatascience.com/

III.2.2 spaCy

spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python.

spaCy is designed specifically for production use and helps building applications that process and “understand” large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning [36]**.**

Figure III.2 : spaCy logo.

III.2.3 TensorFlow

TensorFlow [7] is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well [37].

Figure III.3 : TensorFlow logo.

[7] https://www.tensorflow.org/

III.2.4 **Flask**

Flask is a lightweight WSGI (Web Server Gateway Interface) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug[8] and Jinja[9] and has become one of the most popular Python web application frameworks [38].

Figure III.4 : Flask logo.

III.2.5 **MongoDB**

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by **MonGoDBInc**[10] and licensed under the Server-Side Public License (SSPL).

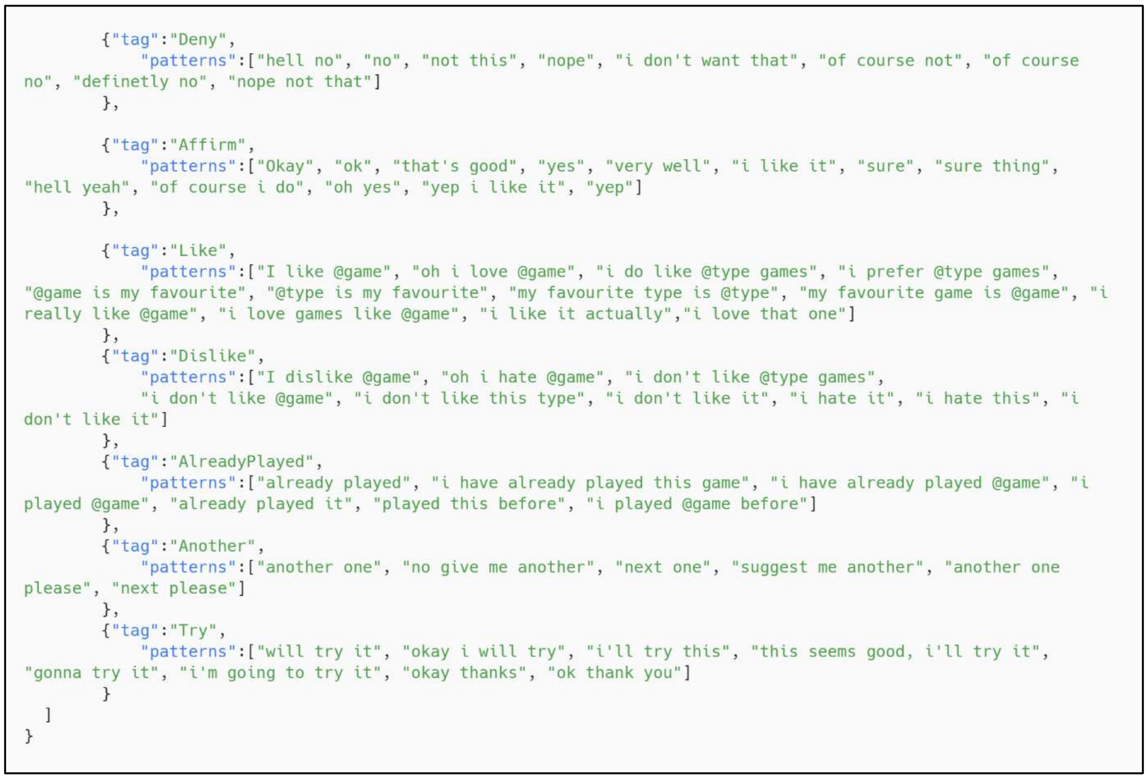
Figure III.5 : MongoDB logo.

[8] https://palletsprojects.com/p/werkzeug/
[9] https://jinja.palletsprojects.com/en/2.11.x/
[10] https://www.mongodb.com/company

III.3 **Preprocessing**

III.3.1 **Intents dataset**

As referred to in the Research problems subsection (Section 1.4), we were obliged to manually create our own dataset for this phase. The dataset is composed of the tag or the intent's label, and some phrases that refer to that intent, the dataset is used to train the Neural Network responsible for the Intent Recognition.



```
        {"tag":"Deny",
            "patterns":["hell no", "no", "not this", "nope", "i don't want that", "of course not", "of course
no", "definetly no", "nope not that"]
        },

        {"tag":"Affirm",
            "patterns":["Okay", "ok", "that's good", "yes", "very well", "i like it", "sure", "sure thing",
"hell yeah", "of course i do", "oh yes", "yep i like it", "yep"]
        },

        {"tag":"Like",
            "patterns":["I like @game", "oh i love @game", "i do like @type games", "i prefer @type games",
"@game is my favourite", "@type is my favourite", "my favourite type is @type", "my favourite game is @game", "i
really like @game", "i love games like @game", "i like it actually","i love that one"]
        },
        {"tag":"Dislike",
            "patterns":["I dislike @game", "oh i hate @game", "i don't like @type games",
            "i don't like @game", "i don't like this type", "i don't like it", "i hate it", "i hate this", "i
don't like it"]
        },
        {"tag":"AlreadyPlayed",
            "patterns":["already played", "i have already played this game", "i have already played @game", "i
played @game", "already played it", "played this before", "i played @game before"]
        },
        {"tag":"Another",
            "patterns":["another one", "no give me another", "next one", "suggest me another", "another one
please", "next please"]
        },
        {"tag":"Try",
            "patterns":["will try it", "okay i will try", "i'll try this", "this seems good, i'll try it",
"gonna try it", "i'm going to try it", "okay thanks", "ok thank you"]
        }
   ]
}
```

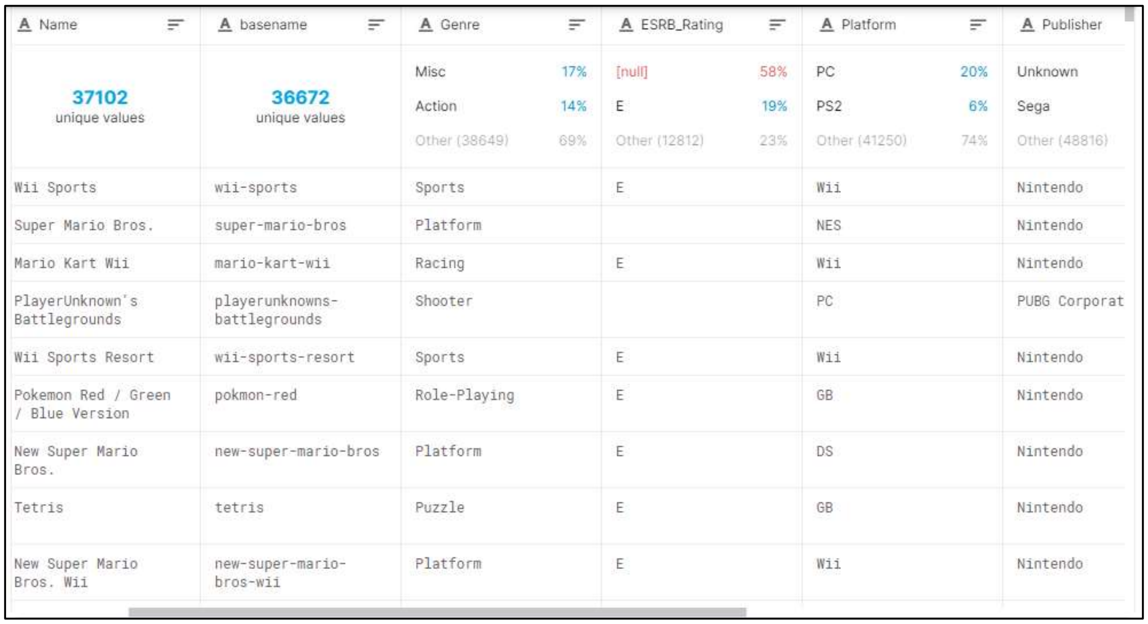
Figure III.6 : Intents dataset.

The data was manually created and has roughly around 150 phrases in 15 categories (intents). This means approximately 10 phrases per category[11].

[11] It really is hard (if not impossible) to give a 'one size fits all' size in this kind of practices. For some classification problems, we need training data on the order of million examples per class. For other problems, only few examples can be enough. It is agreed that in order to get a proper result, data must be for at-least 10 times the degree of freedom. In our case, 15 categories implied 150 data points (phrases).

III.3.2 Games dataset

The games dataset contains 55,792 records generated by a scrape of vgchartz.com as of April 12th, 2019, found on Kaggle.com[12], the dataset contains information such as the publisher, the year of release, the genre and the critic score.



Name	basename	Genre	ESRB_Rating	Platform	Publisher
37102 unique values	36672 unique values	Misc 17% Action 14% Other (38649) 69%	[null] 58% E 19% Other (12812) 23%	PC 20% PS2 6% Other (41250) 74%	Unknown Sega Other (48816)
Wii Sports	wii-sports	Sports	E	Wii	Nintendo
Super Mario Bros.	super-mario-bros	Platform		NES	Nintendo
Mario Kart Wii	mario-kart-wii	Racing	E	Wii	Nintendo
PlayerUnknown's Battlegrounds	playerunknowns-battlegrounds	Shooter		PC	PUBG Corporat
Wii Sports Resort	wii-sports-resort	Sports	E	Wii	Nintendo
Pokemon Red / Green / Blue Version	pokmon-red	Role-Playing	E	GB	Nintendo
New Super Mario Bros.	new-super-mario-bros	Platform	E	DS	Nintendo
Tetris	tetris	Puzzle	E	GB	Nintendo
New Super Mario Bros. Wii	new-super-mario-bros-wii	Platform	E	Wii	Nintendo

Figure III.7 : The original games dataset.

The first thing we had to do was to ignore the unnecessary columns, and only keep the ones we need, which are: the genre, the publisher, the critic score, the platform, and of course the name.

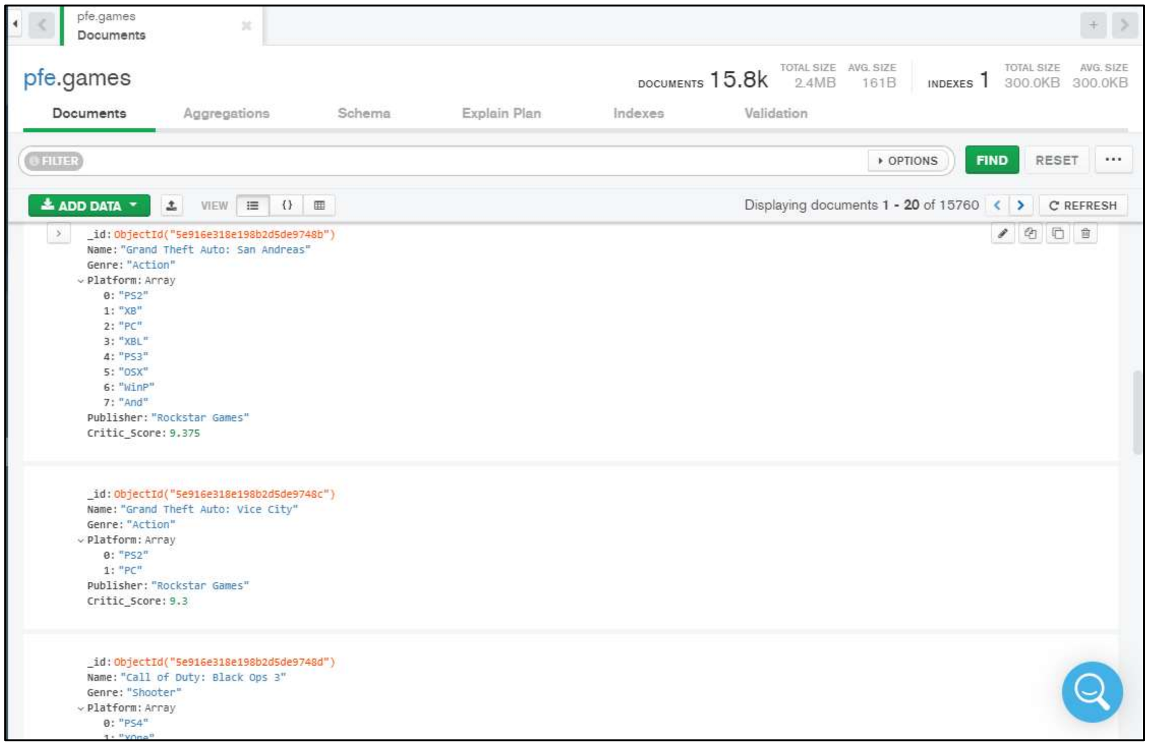
The second thing we had to deal with was the repeated games, where the same game has multiple entries for different platforms, due to the fact that some of them had different release dates for each platform, amongst other differentiations. So, we regrouped the games by name and recalculated the average critic score from different platforms.

With all the reduction we did, the dataset was still too big, and filled with unnecessary games (reduced from around 55K to around 37K), so the next step was to remove the games that are not at least in one of the most used platforms, which are: "PC", "PlayStation 4", "Xbox

[12] https://www.kaggle.com/ashaheedq/video-games-sales-2019

One", "PlayStation 3" and "Xbox 360". This step reduced the number of games to around 15K, which was fairly acceptable.

Next, we saved the games in our MongoDB database for further use.

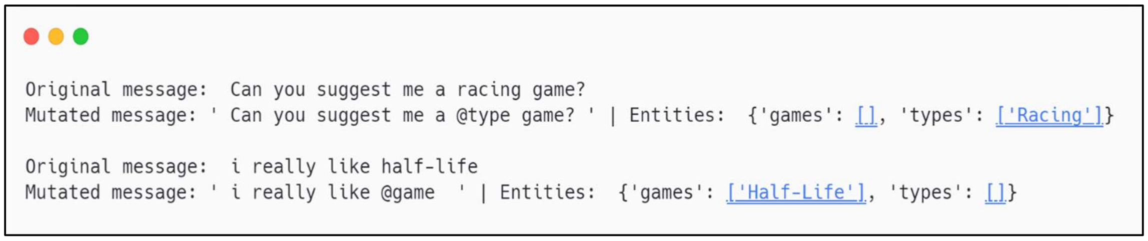

pfe.games
Documents
pfe.games
DOCUMENTS 15.8k TOTAL SIZE 2.4MB AVG. SIZE 161B
INDEXES 1 TOTAL SIZE 300.0KB AVG. SIZE 300.0KB
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
OPTIONS
FIND
RESET
ADD DATA
VIEW
Displaying documents 1 - 20 of 15760
REFRESH

```
_id: ObjectId("5e916e318e198b2d5de9748b")
Name: "Grand Theft Auto: San Andreas"
Genre: "Action"
Platform: Array
    0: "PS2"
    1: "XB"
    2: "PC"
    3: "XBL"
    4: "PS3"
    5: "OSX"
    6: "WinP"
    7: "And"
Publisher: "Rockstar Games"
Critic_Score: 9.375

_id: ObjectId("5e916e318e198b2d5de9748c")
Name: "Grand Theft Auto: Vice City"
Genre: "Action"
Platform: Array
    0: "PS2"
    1: "PC"
Publisher: "Rockstar Games"
Critic_Score: 9.3

_id: ObjectId("5e916e318e198b2d5de9748d")
Name: "Call of Duty: Black Ops 3"
Genre: "Shooter"
Platform: Array
    0: "PS4"
    1: "XOne"
```

Figure III.8 : Games database.

III.4 **Entity Extraction**

As explained in the previous chapter (II.5.1), this component is responsible for extracting the games and types from the incoming user message, the component loads the list of games (and types) from the database, and then for each incoming message loops through all the games and types and looks for matches in the message using regular expressions, an example of the result is in figure III.4.



```
Original message:  Can you suggest me a racing game?
Mutated message: ' Can you suggest me a @type game? ' | Entities:  {'games': [], 'types': ['Racing']}

Original message:  i really like half-life
Mutated message: ' i really like @game  ' | Entities:  {'games': ['Half-Life'], 'types': []}
```

Figure III.9 : Example of the output of the Entity Extraction component.

III.5 **Text treatment and Word Embedding**

spaCy takes care of tokenization and word embedding very efficiently, with its improved tokenization algorithm and its pre-trained language models, but we needed to extend the vectors obtained to support the two special words “@game” and “@type” that were added in the previous step, so we added two dimensions to the vectors (the new total is 302), each of them taking the value 0 or 1 for the presence or absence of the special words.

III.6 **Intent classification**

As we previously explained in sections II.3.2 and II.5.3 , we used a Neural Network to predict the intents of the incoming messages, the Neural Network is created using TensorFlow, and it is composed of an input layer, two hidden dense (fully connected) layers of size 128, and an output layer. The Neural Network uses Categorical Cross-entropy loss function, and the Adam[13] optimizer.

- ***Categorical Cross-entropy loss function:*** Also called *Softmax* Loss. It is a Softmax activation[14] plus a Cross-Entropy loss[15]. Using this loss function, the CNN is trained to output a probability over the C classes for each phrase. It is used for multi-class classification. The principle of this function can be summed up in what follows:

[13] https://arxiv.org/abs/1412.6980
[14] In mathematics, the softmax function, also known as softargmax or normalized exponential function, is a function that takes as input a vector *z* of *K* real numbers, and normalizes it into a probability distribution consisting of *K* probabilities proportional to the exponentials of the input numbers.
[15] Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

$$CE = -\log\left(\frac{e^{S_P}}{\sum_j^C e^{S_j}}\right)$$

Where S_p is the CNN score for the positive class.

- ***Adam optimizer***: Adam is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks. The algorithms leverages the power of adaptive learning rates methods to find individual learning rates for each parameter.

The next figure is a minimized graphical representation of the Neural Network we used.

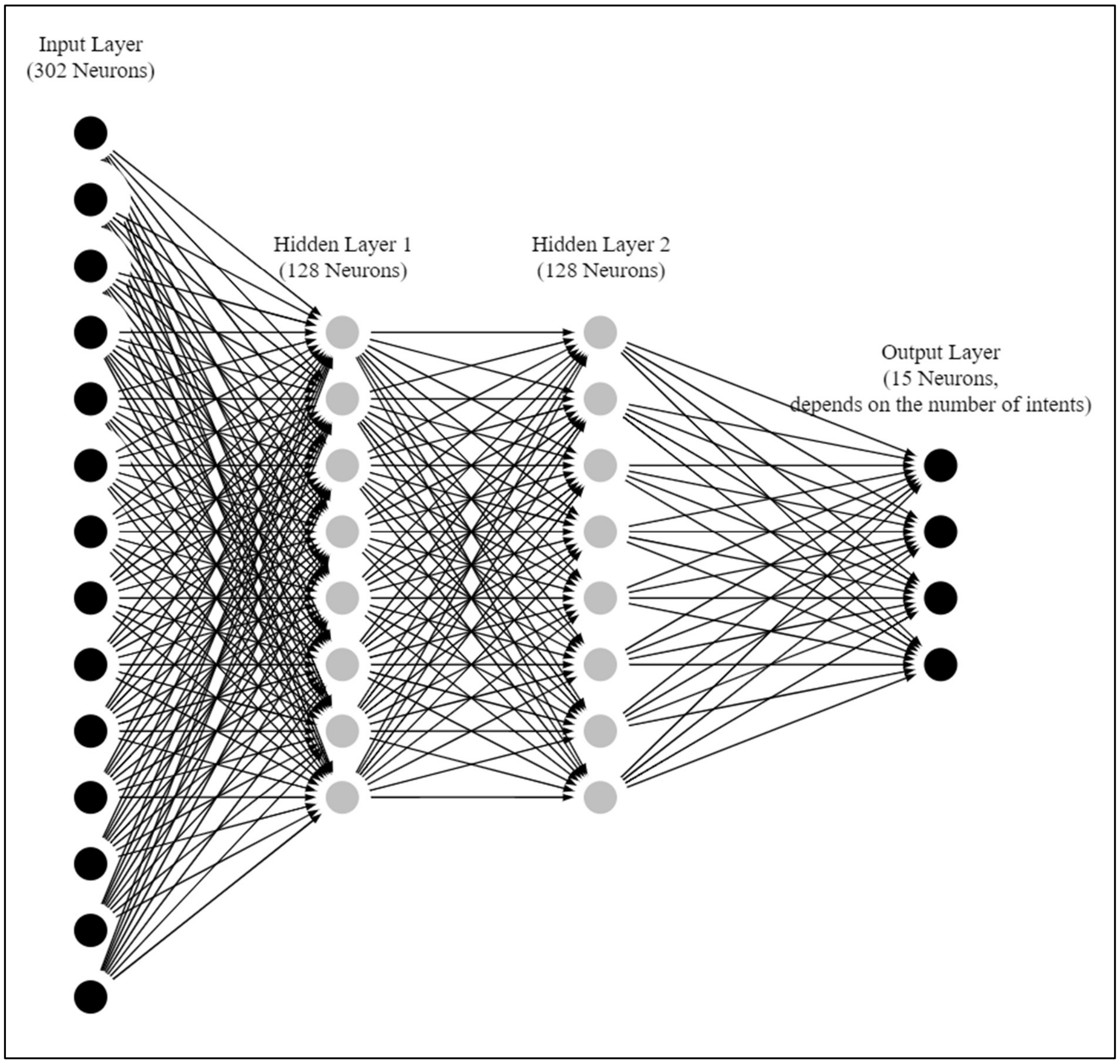


Figure III.10 : Illustration of our Neural Network (reduced size).

The results we obtained from the training and validation of our Neural Network are presented in figure III-11.

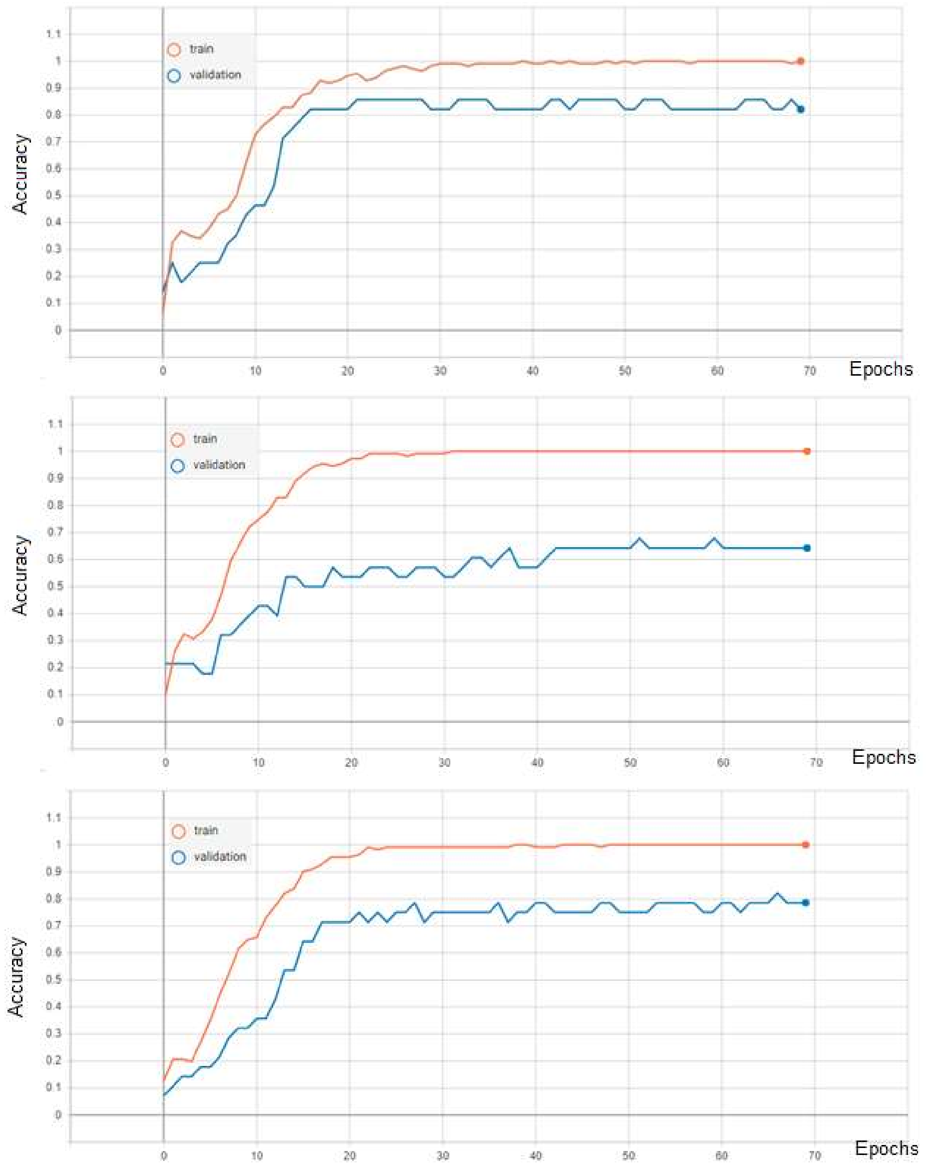
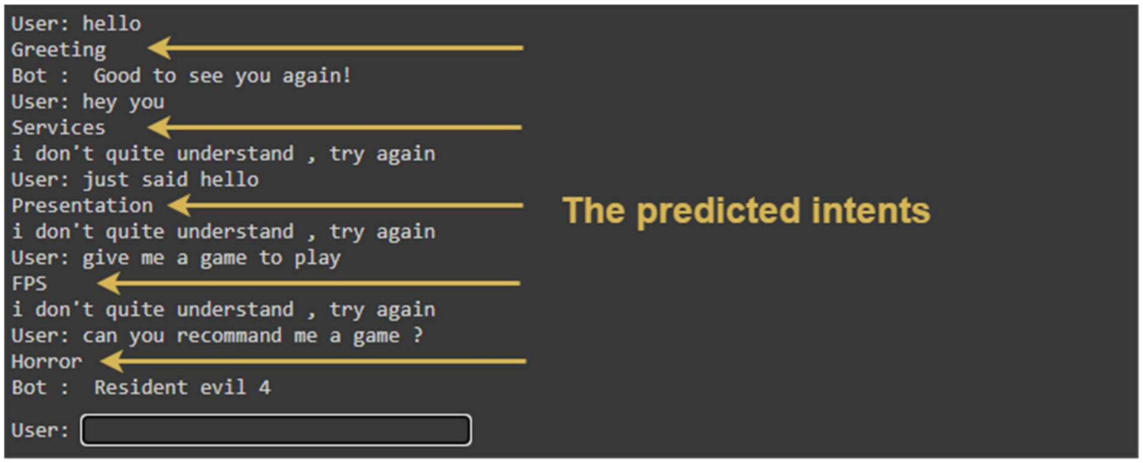


Figure III.11 : Neural Network's accuracy

As seen in the figure above, we tested the accuracy of the Neural Network and the results varied from 64% to 86% during the evaluation for each time we shuffled the data, this large gap is due to the small amount of training and validation data.

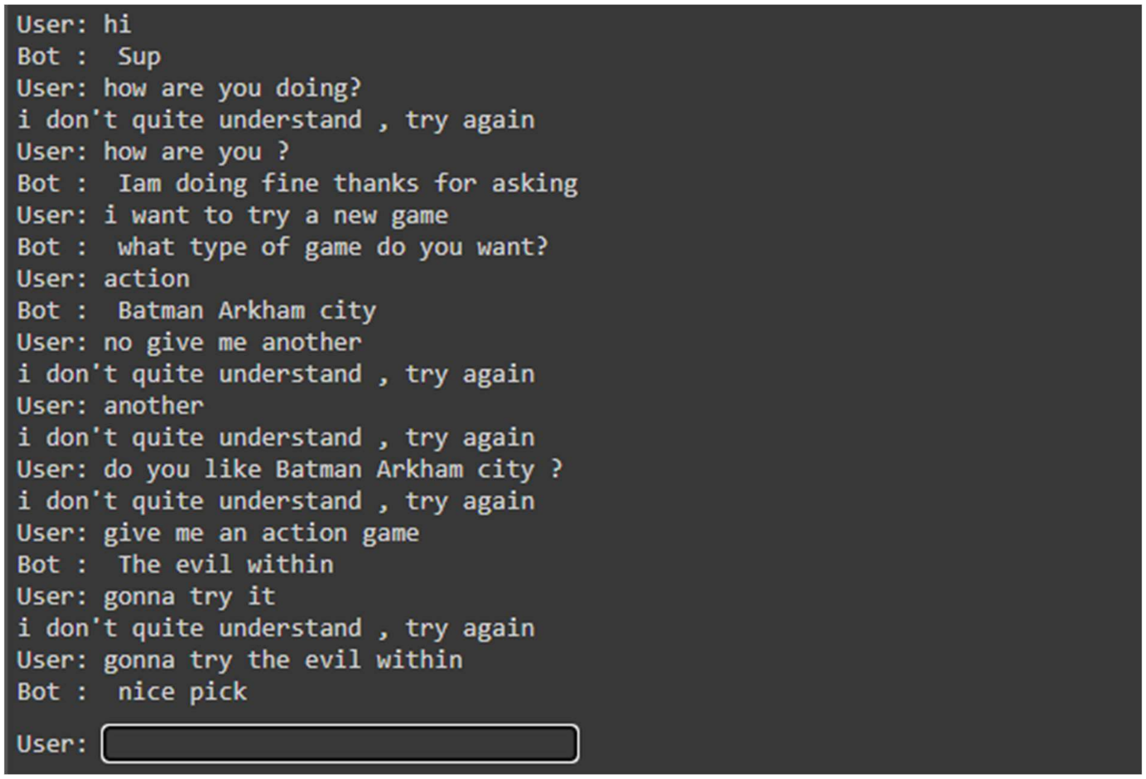
III.7 **Testing the Chatbot**

For this final section, we are going to show some examples of conversations that we had with the two versions of the Chatbot, some went well, and others did not.



```
User: hello
Greeting
Bot :  Good to see you again!
User: hey you
Services
i don't quite understand , try again
User: just said hello
Presentation
i don't quite understand , try again
User: give me a game to play
FPS
i don't quite understand , try again
User: can you recommand me a game ?
Horror
Bot :  Resident evil 4

User:
```

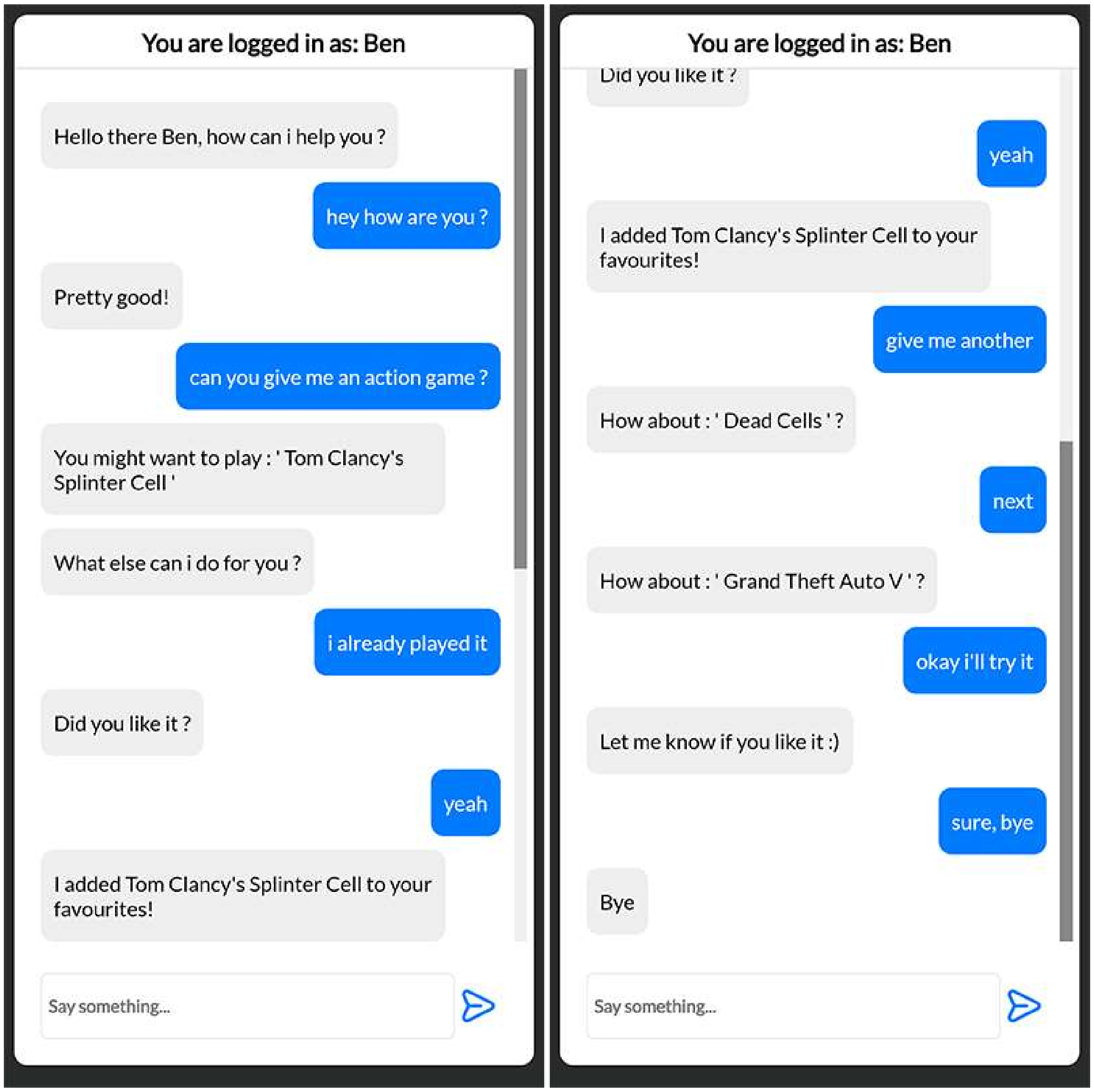
Figure III.12 : Conversation Example N°1 with Bot Version 1.



```
User: hi
Bot :  Sup
User: how are you doing?
i don't quite understand , try again
User: how are you ?
Bot :  Iam doing fine thanks for asking
User: i want to try a new game
Bot :  what type of game do you want?
User: action
Bot :  Batman Arkham city
User: no give me another
i don't quite understand , try again
User: another
i don't quite understand , try again
User: do you like Batman Arkham city ?
i don't quite understand , try again
User: give me an action game
Bot :  The evil within
User: gonna try it
i don't quite understand , try again
User: gonna try the evil within
Bot :  nice pick

User:
```

Figure III.13 : Conversation Example N°2 with Bot version 1.

The last 2 figures were examples of conversations with the first version of the Chatbot, the next ones will be conversations with the second version.


You are logged in as: Ben

Hello there Ben, how can i help you ?

hey how are you ?

Pretty good!

can you give me an action game ?

You might want to play : ' Tom Clancy's Splinter Cell '

What else can i do for you ?

i already played it

Did you like it ?

yeah

I added Tom Clancy's Splinter Cell to your favourites!

You are logged in as: Ben

Did you like it ?

yeah

I added Tom Clancy's Splinter Cell to your favourites!

give me another

How about : ' Dead Cells ' ?

next

How about : ' Grand Theft Auto V ' ?

okay i'll try it

Let me know if you like it :)

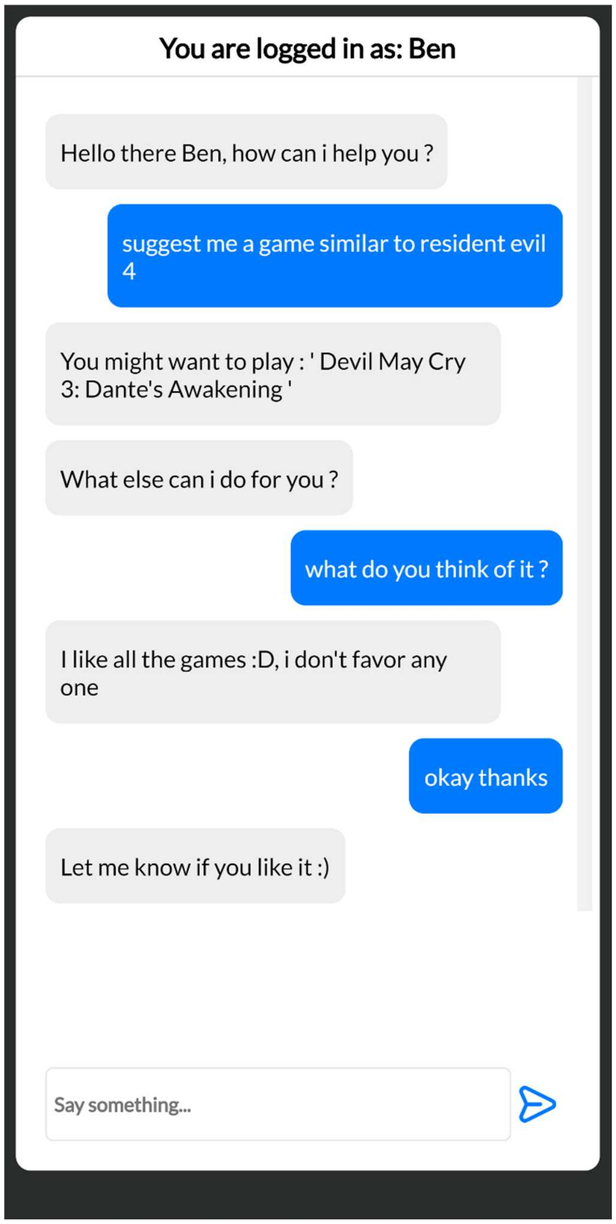
sure, bye

Bye

Figure III.14 : Conversation Example N°1 with Bot Version 2.

As shown in (Figure III.14), the type of scenario represented is **the recommendation aspect based on genre** (Figure II.**15**), where the user is going to ask for a recommendation in a specific genre.

Depending on the user's answers regarding the recommendation, different scenarios may occur, depending on whether he did like the game or not, did play if before(Figure II.16) or just wants another game(Figure II.17).


You are logged in as: Ben

Hello there Ben, how can i help you ?

suggest me a game similar to resident evil 4

You might want to play : ' Devil May Cry 3: Dante's Awakening '

What else can i do for you ?

what do you think of it ?

I like all the games :D, i don't favor any one

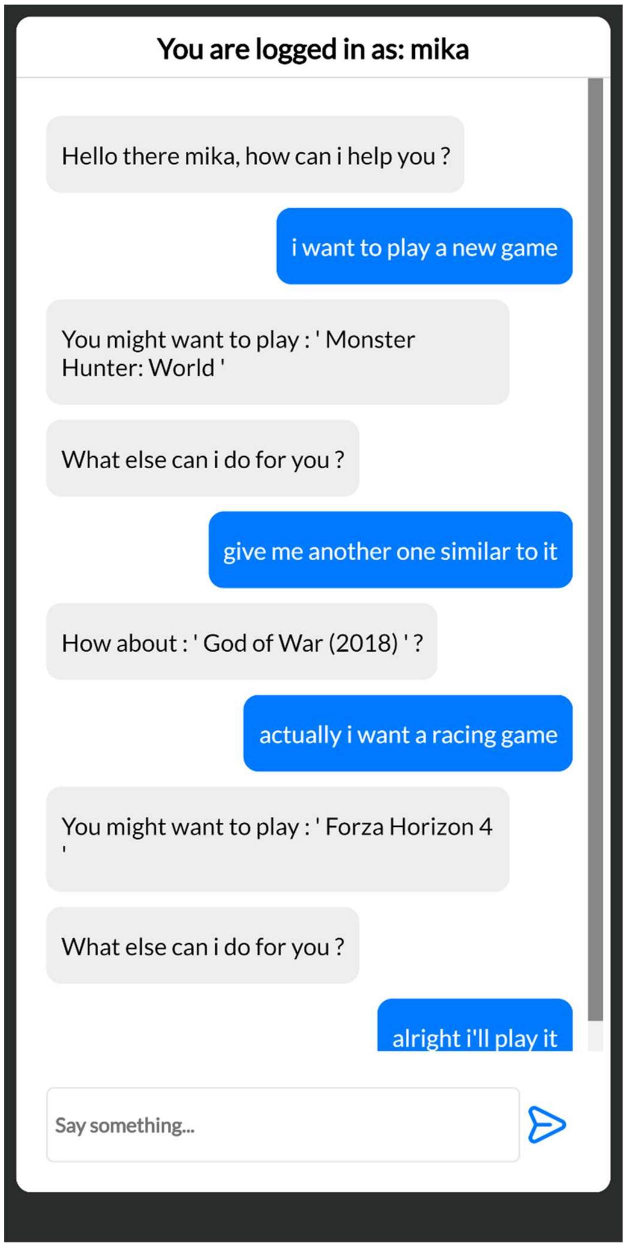
okay thanks

Let me know if you like it :)

Say something...

Figure III.15 : Conversation Example N°2 with Bot version 2.

(Figure III.15), demonstrates the case of **recommendation based on game similarity** **(**Figure II.18), in which; the user is going ask about a recommendation based on a given game, which includes the similarity in the genre and publishers (Figure II.18).



You are logged in as: mika

Hello there mika, how can i help you ?

i want to play a new game

You might want to play : ' Monster Hunter: World '

What else can i do for you ?

give me another one similar to it

How about : ' God of War (2018) ' ?

actually i want a racing game

You might want to play : ' Forza Horizon 4 '

What else can i do for you ?

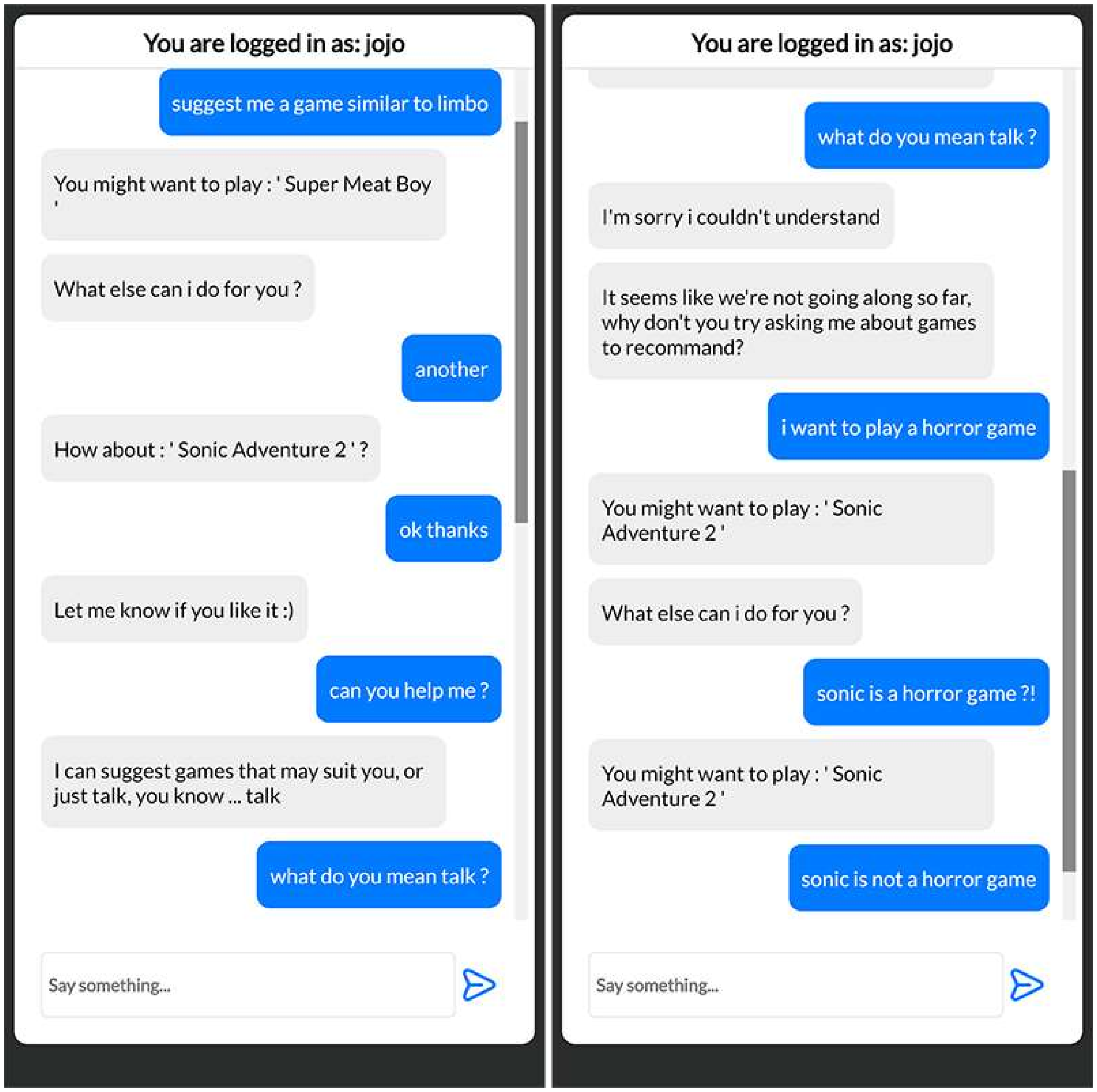
alright i'll play it

Figure III.16 : Conversation Example N°3 with Bot version 2.

(Figure III.16) demonstrates the case where the user wants a **game recommendation based on his preferences** (Figure II.19).

We started by giving good examples in which the bot is going to work smoothly according to what the user is asking in terms of game recommendation.

Next, we are going to give bad examples of a scenario where the bot is going to have problems answering the user's demands.



You are logged in as: jojo

suggest me a game similar to limbo

You might want to play : ' Super Meat Boy '

What else can i do for you ?

another

How about : ' Sonic Adventure 2 ' ?

ok thanks

Let me know if you like it :)

can you help me ?

I can suggest games that may suit you, or just talk, you know ... talk

what do you mean talk ?

Say something...

You are logged in as: jojo

what do you mean talk ?

I'm sorry i couldn't understand

It seems like we're not going along so far, why don't you try asking me about games to recommand?

i want to play a horror game

You might want to play : ' Sonic Adventure 2 '

What else can i do for you ?

sonic is a horror game ?!

You might want to play : ' Sonic Adventure 2 '

sonic is not a horror game

Say something...

Figure III.17 : Conversation Example N°4 with Bot version 2.

As shown in (Figure III.**17**), sometimes the bot does not answer properly or does not understand the user’s message and just replies with “I’m sorry couldn’t understand”, this also happens when the Chatbot is expecting a specific scenario but it does not happen.

Also, sometimes in the recommendation aspect, the Chabot can give games which are not similar to the genre or category asked by the user, this may occur when the context gets mixed up, and because the categories in the games dataset we used are not well specified.

III.8 **Conclusion**

In this chapter, we presented our implementation regarding the solution we proposed, including the tools that we used, also the different steps of the implementation from cleaning the dataset to the final form of the chatbot.

What is coming up next, is a general conclusion about our retrospective of what we have achieved in this lap of time and what the next step to make it even better.

General Conclusion

Summary

During the last few years, the need for Chatbots has significantly increased, and it will increase even more in the upcoming years. And that is because Chatbots provide a fast, automated and cheap way of communication between companies and users, which is beneficial for both.

In this project, our goal was to create a functional Chatbot that is capable of having coherent conversations with the "Gamers", and to provide some useful utilities for them. For that, we had to see many examples of the available Chatbots, and to try to understand how they work, and how we can implement our own version from scratch for our specific field which is gaming or games recommendations, without using any Chatbots creation platform, for that would not help us understand how the core of the Chatbot works.

During our research, we found out that contrary to what we initially thought and to what most people think, the field of Chatbots and Natural Language Processing in general suffers from many deficiencies, too much work still has to be done even for the pioneers like Google or Amazon.

The goals we set initially were partially achieved, we managed to create a functional Chatbot that can have a few conversations and can provide the core functionality of games recommendation, although we initially hoped for something more than that, it can be said that what we achieved is acceptable considering the situation the world has been through during the creation of this work.

We encountered many difficulties and we managed to overcome some of them, but others forced us to change the way we were going to do things and to readapt, again and again. The main problem we encountered was the lack of data, things would have been much better with more data at hand, "**Data is the new oil"** as said by Clive Humby. Some of the other difficulties were related to the unpredictable nature of humans, a conversation has an infinity of possible scenarios, and it is impossible to cover all of them.

Outlooks

Many improvements can be made in order to make our Chatbot more accurate, useful and scalable. The first thing that needs to be done is to make a better structure that allows for more scalability and easier integrity of new features.

Another thing to be improved is the execution time of some components, mainly the entity extraction part which takes an average of 2.4 seconds to process an incoming message, which is a little bit too much, time is money, and it should not be wasted especially with Chatbots.

There are many things that could be added, like the multi-labeled classification for when a single message may contain multiple intents, or adding the collaborative filtering option for recommendations when the users' data becomes sufficient for that.

Most importantly during this work, we learned a lot of things, both theorical and practical, it was an introduction for us to the field of Machine Learning, and it will certainly open doors for us to explore.

Bibliography

[1] Cliff.Edwards, ""Valve Lines Up Console Partners in Challenge to Microsoft, Sony,","
Bloomberg, 2013.

[2] Cassell.Justine, Sullivan.Joseph, Prevost.Scott and Churchill.Elizabeth, Embodied Conversational Agents, 2000.

[3] Alan.Mathison.Turing, Computing Machinery and Intelligence". In: Mind 49 (1950), pp. 433–460., 1950.

[4] Weizenbaum.Joseph, ELIZA—a computer program for the study of natural language communication between man and machine., Communications of the ACM, 1966.

[5] Jan.Goyvaerts, Regular expression, 2006.

[6] Noyes.Katherine, "It's (not) elementary: How Watson works," pcworld, 7 October 2016. [Online]. Available: https://www.pcworld.com/article/3128401/its-not-elementary-how-watson-works.html.

[7] Rhinehart.Craig, "10 Things You Need to Know About the Technology Behind Watson," 2011.

[8] Wolchover.Natalie, "How the Cleverbot Computer Chats Like a Human," 2011.

[9] Wallace.RS, "The Anatomy of A.L.I.C.E.," 2007. [Online]. Available: https://alicebot.org/.

[10] Lokman.Abbas.S and Zain.Jasni.M, "One-Match and All-Match Categories for Keywords Matching in Chatbot," *American Journal of Applied Sciences,* 2010.

[11] Zhou.Jianshe, Yan.Zhao, Duan.Nan, Bao.Junwei, Chen.Peng, Zhou.Ming and A. Li.Zhoujun, "DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[12] Chandrasekar.Raman, "Elementary? Question Answering, IBM's Watson, and the Jeopardy! Challenge," *Resonance,* 2014.

[13] P.Abelson.Robert and Carroll.J.Douglas, Computer simulation of individual belief systems, 1965.

[14] GOAN.S, "Semiotic relationships in ambiguously stratified bmguage systems," 164.

[15] *Natural language input for a computer problem solving system.,* 1964.

[16] Weizenbaum.Joseph, Symmetric list processor, 1963.

[17] R.Rogers.Carl, Client Centered Therapy: Current Practice, Implications and Theory, Boston, 1951.

[18] Alex.Hern, "Microsoft scrambles to limit PR damage over abusive AI bot Tay," The Guardian, 24 March 2016. [Online]. Available: https://www.theguardian.com/technology/2016/mar/24/microsoft-scrambles-limit-pr-damage-over-abusive-ai-bot-tay.

[19] Schilit.Bill and Theimer.Martin, Disseminating active map information to mobile hosts, 1994.

[20] Peter.J.Brown, The Stick-e document: a framework for creating, 1996.

[21] Ward.Andy, Jones.Alan and Hopper.Andy, A new location technique for the active office., 1997.

[22] Bill.Schilit, Norman.Adams and Roy.Want, "Context-aware computing applications. In: First International Workshop on Mobile Computing Systems and Applications," in *First International Workshop on Mobile Computing Systems and Applications*, 1994.

[23] Jason.Pascoe, "Adding generic contextual capabilities to wearable computers. In: Proceedings of 2nd International Symposium on Wearable Computers," in *Proceedings of 2nd International Symposium on Wearable Computers*, 1998.

[24] Dey.Anind.K and Abowd.Gregory.D, "Towards a better understanding of context and context-awareness. CHI'2000 Workshop on the What, Who, Where, When, and How of Context-Awareness," in *CHI'2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*, 2000.

[25] Dey.Anind.K, Understanding and using context," Personal and Ubiquitous Computing, 2001.

[26] Schilit.Bill, Adams.Norman and Want.Roy, Context-aware computing ap-plications,in Mobile Computing Systems and Applications, 1994.

[27] Bettini.Claudio, Brdiczka.Oliver, Henricksen.Karen, Indulska.Jadwiga, Nicklas.Daniela, Ranganathan.Anand and Riboni.Daniele, "A survey of context modelling and reasoning techniques," Pervasive and Mobile Computing," 2010.

[28] Ying.Xu and Fu-yuan.Xu, "Research on context modeling based on ontology," in Computational Intelligence for Modelling, Control and Automation," 2006.

[29] Uschold.Michael and Gruninger.Michael, Ontologies: principles, methods and applications," The Knowledge Engineering Review, 1996.

[30] Ye.Juan, Coyle.Lorcan, Dobson.Simon and Nixon.Paddy, Ontology-based models in pervasive computing systems," The Knowledge Engineering Review, 2007.

[31] Schafer.J.Ben, Konstan.Joseph and Riedl.John, Recommender systems in e-commerce, New York, 1999.

[32] Adomavicius.Gediminas and Tuzhilin.Alexander, "Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," 2005.

[33] Perugini.Saverio, Gonçalves.Marcos.André and Fox.Edward.A, "Recommender systems research: A connection-centric survey. Journal of Intelligent Information Systems," 2004.

[34] Burke.Robin, "Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction," 2002.

[35] Ken.Lang, Newsweeder: learning to filter netnews, 1995.

[36] [Online]. Available: https://spacy.io/usage/spacy-101. [Accessed June 2020].

[37] [Online]. Available: https://github.com/tensorflow/tensorflow. [Accessed June 2020].

[38] [Online]. Available: https://palletsprojects.com/p/flask/. [Accessed June 2020].

[39] J. Worland, "Microsoft takes chatbot offline after it starts tweeting racist messages," *Time,* 2016.

[40] C. Edwards, "Valve Lines Up Console Partners in Challenge to Microsoft, Sony," *Bloomberg,* 2013.

[41] [Online]. Available: https://spacy.io/usage/spacy-101. [Accessed June 2020].

[42] [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. [Accessed June 2020].