Universitat Oberta de Catalunya

Chatbot implementation as customer service support for an entertainment company

Adrià Riera Llambí
Master in Business Intelligence and Big Data
Area of expertise: Business Analytics

Rafael Luque Ocaña
Joan Melià Seguí

06.2019

To Franzi, for more than the nudging

To Mario, for the experience

To my parents, for (almost) everything else

FINAL WORK'S SHEET

Title:	*Chatbot implementation as customer service support for an entertainment company*
Author's name:	*Adrià Riera Llambí*
Consultant's name:	*Rafael Luque Ocaña*
PRE's name:	*Joan Melià Seguí*
Delivery date (MM/YYYY):	*06/2019*
Title or program:	*Master in Business Intelligence and Big Data*
Area of expertise:	*Business Analytics*
Language:	*English*
Keywords:	*Ruled-based chatbot; learning model; IBM framework; API*

Abstract (250 words or less): *With the purpose, context and applicability, methodology, results and conclusions*

Artificial Intelligence has proven to be a technical field with a big potential impact on the business strategy of a company. Concretely, chatbots could become the next technology to rule the customer services. They offer a gate to capture structured customer data for a posterior postprocessing and analysis, reducing the costs of operation compared to the traditional customer service tools. As well, among other capabilities, chatbots could have a better capacity to capture a loyal customer.

The pursuit of the present thesis was to create a chatbot that interacted with a user in a theoretical entertainment company from the film industry. Such company, having seized the market tendencies, would deploy a chatbot with basic functionalities as trial before a complete integration into the customer service department. Hence, it was initially designed to support the user during the search of product information and basic recommendations.

For such a purpose, a theoretical working environment and a product database for the company were defined. The rule-based bot was developed within IBM Watson framework. Its architecture and language were built considering the criticality of being a direct customer interface.

The final step was to characterize a learning and training model for the bot, as the machine learning engine available within IBM's framework did not cover directly all the bot's needs.

The realization of the thesis concluded with the online deployment of the bot and the identification of further potential developments that would reinforce the impact on both the user’s and company’s side.

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d’aplicació, metodologia, resultats i conclusions del treball*

La intel·ligència artificial ha demostrat ser un camp amb un gran impacte en l’estratègia de negoci d’una empresa. Concretament els chatbots poden ser la propera tecnologia que domini l’atenció al client. Aquests ofereixen una porta per capturar dades d’usuari d’una manera estructurada (per un posterior post processat i anàlisi), reduint els costos d’operació si es comparen amb els de les eines d’atenció al client tradicionals. Entre altres capacitats, els chatbots podrien tenir una millor capacitat per capturar un client lleial.

La present tesi persegueix la creació d’ un chatbot que interaccioni amb un usuari d’una empresa teòrica del camp de l’entreteniment cinematogràfic. Aquesta, després d'haver captat les tendències del mercat, desplegaria un chatbot amb funcionalitats limitades com a prova abans de la seva completa integració als processos de l’empresa. Per tant, aquest es dissenya inicialment per donar suport a l’usuari durant les tasques de cerca d’informació i recomanació de productes.

Així, es defineix un entorn de treball teòric i una base de dades de productes de l’empresa en qüestió. El bot és *rule-based* i es desenvolupa en IBM Watson. La seva arquitectura i llenguatge tenen en compte la criticalitat de ser una interfície directa del client.

L’últim pas és la caracterització d’un model d’aprenentatge i entrenament per al bot, ja que el motor d’aprenentatge disponible al marc d’IBM no cobreix totes les necessitats d’aquest.

La tesi conclou amb el desplegament en línia del bot i la identificació de possibles millores que podrien reforçar l'impacte en l'usuari i la companyia.

Index

1 Introduction 1
  1.1 Background and rationale 1
  1.2 Objectives 4
  1.3 Approach and methodology 4
  1.4 Schedule 4
  1.5 Summary of obtained results 5
  1.6 Short description of the further chapters 5

2 State of the art 7
  2.1 Chatbots 7
  2.2 Reinforcement learning 9
  2.3 Recommendation systems 10

3 Creation of the chatbot and analysis of the results 13
  3.1 Working environment 13
  3.2 Framework 14
  3.3 Architecture description 16
    3.3.1 Bot language and communication with the user 17
    3.3.2 Basic nodes: Welcome, Small Talk and Bye 18
    3.3.3 Error handling nodes: Transfer and Anything Else 19
    3.3.4 Query nodes: Query Mode and Define Product Type 20
    3.3.5 Recommendation nodes 22
      3.3.5.1 By genre and popularity 22
      3.3.5.2 Other recommendation methods 24
  3.4 Learning model 26
    3.4.1 User input product name for query mode 27
    3.4.2 General considerations 29

4 Conclusions 31

5 Lexicon 35

6 Bibliography 37
  6.1 Cited bibliography 37
  6.2 Support bibliography 38

7 Annex 40
  7.1 Flowchart 40
  7.2 Overview of entities and intents used 41
  7.3 Call to the OMDB API (Prod_Film_Info web action) 42
    7.3.1 JSON call to the web action within bot's framework 42
    7.3.2 Web action code 43
    7.3.3 Web action example answer 43
  7.4 Call to the TMDB API (Recommendation web action) 44
    7.4.1 JSON call to the web action within bot's framework 44
    7.4.2 Web action code 44
    7.4.3 Web action example answer 48
  7.5 Access data to the frameworks used 48
  7.5.1 For the IBM Watson framework 48
  7.5.2 For the DialogFlow framework 49
  7.6 DialogFlow vs IBM Watson 49

List of figures

Figure 1: Captions of KLM's assistant Blue Bot; it can be called via Facebook Messenger (left) or Google Assistant (right). Source: KLM. 2

Figure 2: Schematic of a chatbot architecture. 7

Figure 3: An agent takes an action based on the current state and the reward obtained. The action modifies the environment and defines a new state and reward. Source: KD nuggets 10

Figure 4: Content-based filtering vs collaborative filtering. Source: The marketing technologist 11

Figure 5: Conversation flow of a simple three node bot; every node has a condition-action defined. Machine learning matches every user input to an intent (introduced by #). Once a condition is fulfilled, the corresponding node is activated. 15

Figure 6: Schematic architecture of the MovieBot. 16

Figure 7: The bot is fed with 4 different formulations for the Welcome node which are randomly picked up every time the node is active. Blue colored text are clickable options. 18

Figure 8: The bot reacts to positive feedback and tries to induce a further use (left). In case negative feedback is provided (right), the bot reconducts the conversation to its core competences. 18

Figure 9: Example of MovieBot's answers when dialog has to be transferred and when bot cannot identify an intent. 19

Figure 10: Schema of MovieBot's behavior when the query nodes are active. 20

Figure 11: Example of a JSON call from the bot to the web action Prod_Film_Info. 21

Figure 12: Schematic of the web action call for the query function. 21

Figure 13: Results of the search by actor (left) and by movie (right). After a successful search, different options are given to the user 22

Figure 14: Schematic of the web action call for the genre recommendation function 23

Figure 15: Result of a recommendation based on war and drama. 23

Figure 16: Dialog flow pre/after manual training. 27

Figure 17: Schematic of filtering of user input via intents. 28

Figure 18: Time schedule of the project. 40

List of tables

Table 1: Results from TMDB API for different quality criteria. Source: TMDB 24

Table 2: Correct (green) or wrong (red) classification of different user inputs. 27

Table 3: Entity overview. 41

Table 4: Intent overview. 42

Table 5: Classification of different user input by DialogFlow and IBM Watson 50

1 Introduction

1.1 Background and rationale

A chatbot (also conversational bot or chatterbot) is an intelligent computer application capable of simulating the conversational skills from a human.

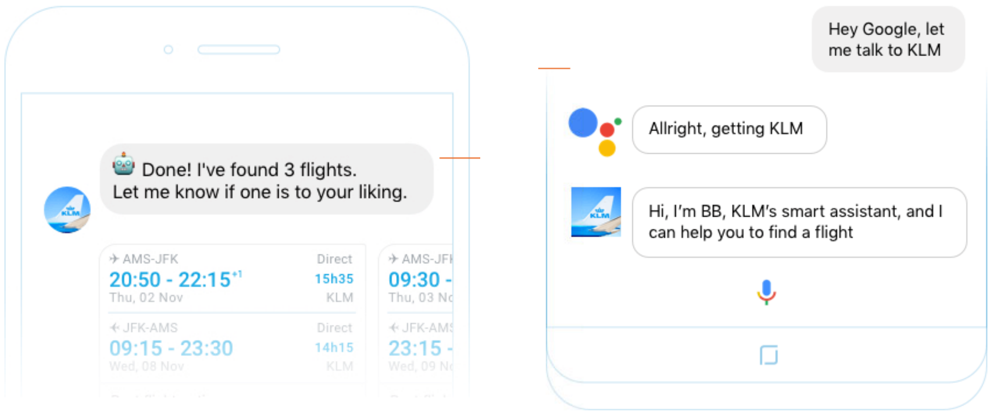
The idea of a computer program that would be not only able to communicate with a human counterpart but to do so in an (at least apparently) intelligent way was already conceived since the beginning of the history of artificial intelligence (AI). In the year 1966 the ELIZA program was developed. ELIZA was capable of mimicking a human conversation: it related the user entries with a series of keywords that accordingly triggered the corresponding answers [1]. Working mainly on the same premises, lots of different and more advanced versions of chatbot have emerged ever since. Thus, the program ELIZA was followed by others like PARRY in 1972, an evolution of ELIZA that simulated to be an individual afflicted with schizophrenia. Its initial version from 1981 (that has been evolving until the present days) was the first chatbot that could speak about general topics; newer versions also learned from the interactions with the different users, building up a dialogue background for the different interlocutors that made the conversations appear as more human alike. In 1995 A.L.I.C.E. was able to fluently hold a normal conversation based on heuristic techniques (evaluating the most suitable answer from a group of different options). Another example is IBM’s Watson which in the year 2010 could win human opponents on a regular basis in the popular TV show Jeopardy! A schematic graphic evolution of the milestones in the development of the technology can be found under [2].

On the other hand, the democratization of computer applications and the appearance of a general computer literacy at the beginning of the 90's have been a catalyst that facilitated the embodiment of the first theoretical works about automatic chats into real life applications. Already in 1996 Microsoft included the Clippy help system as part of its Office Word consumer computing package. While Clippy was strictly speaking not a chat tool (mainly due to its limited discussion topics), it did use some of the new techniques that emerged in parallel to chatbots: based on the behavior that a user had during the work session and the information that was collected in the databases of Microsoft, Clippy detected if help in some specific task was needed and interacted accordingly. Its success was limited (from 2002 onwards the assistant was deactivated by default and from 2007 it stopped being included in the Office software [3]) but also turned out to be one of the first examples of tangible applications of such technologies for a standard consumer.

This line of development has not stopped growing. Currently, chatbot applications integrate new paradigms of artificial intelligence such as machine learning which is the ability of a system to learn from its experience, that is, to modify and optimize its operation with its use (adapted definition of the Oxford dictionary[1]). Here a small remark should be made: nowadays, the concept of *system* is preferred in the different literature rather than other terms (such as a computer or application) because it captures better the range of technologies that today integrate this type of solution, where the couple computer – program is only a part of the possible configurations. Being a system does not only improve the experience and interaction with the user but also makes them almost autonomous. Namely there are solutions where chatbots are integrated to instant messaging tools such as WhatsApp, Telegram or Facebook Messenger. The customer can take these channels to contact a company and get

[1] Available for free online.

automatically answered questions about the features of a specific product, resolve doubts, learn a language[2] or manage reservations. Similar features are available for mobile assistants. In a similar way, chatbots are also integrated as add-ons for navigation on web pages, where visitor assistance is provided as a guide through the page or to solve frequently asked questions. Another field of application is the voice-targeted attendees, such as Siri or Alexa, who are programmed to facilitate the daily work: tasks as organizing the agenda, ordering products (groceries, clothing…) or centrally controlling the devices of the house connected to them can be managed directly chatting with the appliances.


Done! I've found 3 flights.
Let me know if one is to your liking.

Hey Google, let me talk to KLM

Allright, getting KLM

Hi, I'm BB, KLM's smart assistant, and I can help you to find a flight

Figure 1: Captions of KLM's assistant Blue Bot; it can be called via Facebook Messenger (left) or Google Assistant (right). Source: KLM

On a business strategy level, the commitment to tools and resources based on AI is an advantage not only for the value that is extracted from the sale of the product itself but also because of its impact on the company. Specifically, in the case of chatbots, the expected impact can be summarized in the following points (mix from different sources and own contribution, see [4], [5], [6], [7] and [8]):

- Low maintenance, implementation and operation costs
- Customer service can be offered 24/7 with quicker response times
- Customer service can be available in different communication channels
- Technology can coexist with a more individualized customer service whenever automatization does not offer the right answer
- Implementation of chatbots also opens up the possibility to put the focus of human capital efforts on tasks of greater added value
- Possibility of complementing the offer of products with recommendations of similar products or services that are more accurate and adjusted to a specific customer profile
- Improvement in the quality of customer service (as consequence of the previous points)
- Low learning curve for new users
- Ability to manage parallel conversation threads (with customers, intermediaries, internal departments ...)
- Ability to present non-structured information (for example scattered through different databases or files) to the user in a simple and direct way

[2] For further info see, for example, Duolingo's offer.

- Easier collecting of organized customer data that could lead to a richer database for a later in-depth analysis
- Communication interfaces are reduced, as the communication channels are much simpler, immediate and usable on different platforms (mobile, internet, application, web page complement ...)
- Simple introduction in the company's processes

As introduced before, chatbot applications have an impact on both sides of the company processes. In this direction, the technology consultant *Gartner* classifies the current applications of chatbots in three basic categories [9]:

- Virtual Customer Assistants (VCAs), responsible for automating the written and voiced communication of the call centers
- Virtual Enterprise Assistants (VEAs) business assistants that are intended to simplify internal communication and employee access to information
- Virtual Personal Assistants (VPAs) responsible for managing third party services such as emails, social networks or making reservations for business trips

Gartner also estimated that in 2017 only 2% of companies included chatbot applications in customer services, while it is expected that by 2020 the number will have increased up to 25% [10]. This trend will also coincide with the abandonment of the mobile applications that many companies will undertake during the same period. These are allegedly expensive, inefficient and fail to capture a loyal customer [11]. Due to the features listed above, chatbots can occupy part of the technological and business gap that will appear in the future.

It is within this framework that the present work is carried out. Its goal is to create a chatbot that interacts with both, an internal (employee) and external (customer) user. Taking a generic company in the field of the film entertainment industry, the chatbot proposed will:

1) Allow the user (employee or customer) to consult the product database in a simple way (query function)
2) Recommend personalized products adapted to a specific customer taste (recommendation function)

The recommendation function is introduced to support the business strategy of the theoretically defined company. Companies count with recommendation systems to both gain new customers and to retain them. That translates into a revenue growth: some sources speak of an impact between 20% to 30% [12]; according to McKinsey, 35% of consumers' purchases on Amazon and 75% on Netflix come from their recommendation engines [13], the latter engine being allegedly worth 1 billion dollars per year [14]. Further expected impacts on the business are an improved customer knowledge that would derive to a more efficient inventory or logistic processes or a better customer-oriented experience.

The advantages of recommendation systems complement and enhance those of a chatbot. For that reason, a recommendation system is integrated as part of the current chatbot functionalities, boosting the positive customer experience and the added value of the technology. In the present work, it is intended to illustrate in a practical way the applicability of this type of platforms. Concreter objectives, approach, and schedule are detailed below.

1.2 Objectives

This master's final thesis pursues the following objectives:

1) To define a theoretical working environment for the chatbot
2) To define a working database
3) To define a framework to develop the chatbot application
4) To create a chatbot-type solution that works with this database:
    a. To automatically answer questions about it (query function)
    b. Based on the user inputs, to recommend products that are appropriate and adjusted to his or her tastes (recommendation function)
5) To define the parameters and learning model for the bot

1.3 Approach and methodology

The high demand for products and computer applications and their constant change in requirements (in paradigms, technologies, calculation capacities, platforms ...) makes it a common practice for many companies to contact external suppliers of computer solutions, as the option to build internal competences is in many cases impossible: the acquisition of the needed knowledge and its material maintenance is too expensive and, in most cases, far away from the center of the competences of the company in question.

The case of chatbots is no exception: a quick search on the internet results in different models of support for companies that are interested in their implementation. Whether an integral cloud solution or do it yourself solutions are wished, different chatbot models for all levels of complexity are, to a greater or lesser degree, available online.

Within the actual project, due to the limited time allotted and the technical competences of the field of studies, a chatbot will not be completely developed from scratch. Nor is the pursuit to perform a comparative study of pros and cons of the most popular existing solutions. This project aims to take advantage of some of those existing solutions, more concretely frameworks, as base of the chatbot creation (see the chapter 3 Creation of the chatbot and analysis of the results, for more information). Hence, the structure of the bot constituted by the architecture, integration of external knowledge, the understanding of human counterpart, corresponding actions or supporting algorithms will be authored explicitly for the present thesis.

Therefore, first the characteristics of the working environment and the needs for a framework will be defined. Those will be the spine of the bot. After that, the most appropriate strategy to adapt the technical and business sides will be described. This way, the difficulties arise from the definition and resolution of the problem and not vice versa. This aspect is essential, for example, during the learning phase of the chatbot. How to tackle the different steps will be discussed in more detail in subsequent chapters.

1.4 Schedule

The work planning includes both the resolution of the previously defined objectives and the different deliveries and milestones fixed by the UOC.

A time schedule can be found under chapter 7.1 Flowchart. The milestones are summarized as follows:

PEC01: Initial proposal	25.03 – 05.04
PEC02: Evolution and memory of the work	06.04 – 04.06
2.1 Extension of bibliography	06.04 – 12.04
2.2 Compilation of current solutions	06.04 – 28.04
2.3 Definition of the working environment	13.04 – 21.04
2.4 Database preparation	13.04 – 21.04
2.5 Creation and training of query chatbot	21.04 – 15.05
2.6 Creation of recommendation chatbot	15.05 – 04.06
2.7 1st draft report	15.05 – 04.06
PEC03: Final delivery	05.06 – 27.06
3.1 2nd draft report	05.06 – 22.06
3.2 Training of recommendation chatbot	05.06 – 22.06
3.3 PowerPoint presentation	21.06 – 27.06
PEC04: Evaluation committee	03.07 - 10.07
4.1 Evaluation of the thesis	03.07 – 10.07

1.5 Summary of obtained results

With the completion of this work, the following products are expected to be obtained:

1) Definition of a working environment
2) Database that simulates a real working environment
3) Definition and characterization of a learning and training model for the chatbot
4) Definition of the architecture of the chatbot based on the available data, the working environment and the query and recommendation functionalities
5) Functional chatbot
6) Code, files and / or other applications that can support the abovementioned points

1.6 Short description of the further chapters

The work is structured in different blocks. After this introduction, a study of the state of the art regarding chatbot solutions, reinforcement learning methods and recommendation models is provided. In the next step the project framework will be defined and analyzed in terms of:

- Working environment
- Architecture and platform(s) used for the chatbot creation
- Chatbot working principles
- Chatbot actions, such as:
    a. Query
    b. Recommendation
    c. General interactions with the user

- Learning model

The results obtained from the process will give way to the conclusions. The last chapters introduce the lexicon of terms and abbreviations appeared during the text, the bibliography and the annexes.

2 State of the art

After the short introduction to the scope of work, in the present chapter, a deeper technical description and an overview of the state of the art of current chatbot solutions, reinforcement learning systems and recommendation systems are going to be given.

2.1 Chatbots

The online blog Medium [15] differentiates between two main different variants of chatbots: rule-based and self-learning. Rule-based chatbots are basically intended to attach to a set of rules and perform a series of defined actions based on them. Such chatbots are easy and cheap to implement as they are context limited, meaning that they have simple operating principles and cannot interpret the queries of the user if those do not fit to the parameters the chatbot has been designed for. Self-learning chatbots (also called more generically AI-based bots), on the other hand, incorporate machine learning and natural language processing methodologies to technically understand the intents of the different user messages and take the best decisions in an almost autonomous way. Also in [15], two further subtypes of self-learning chatbots are given: retrieval-based, where the chatbot adapts to the previous conversation flow and the knowledge built around the counterpart (e.g. via the username) and selects the better answer from a group of preset options; generative bots, on the contrary, have the capabilities to generate their own unique answers. Of course, the boundaries above stated are only theoretical as the most suitable chatbot for a certain problem usually turns out to be a mix of the above.

Independently from its variant, the basic architecture is common and can be schematized in the following figure:

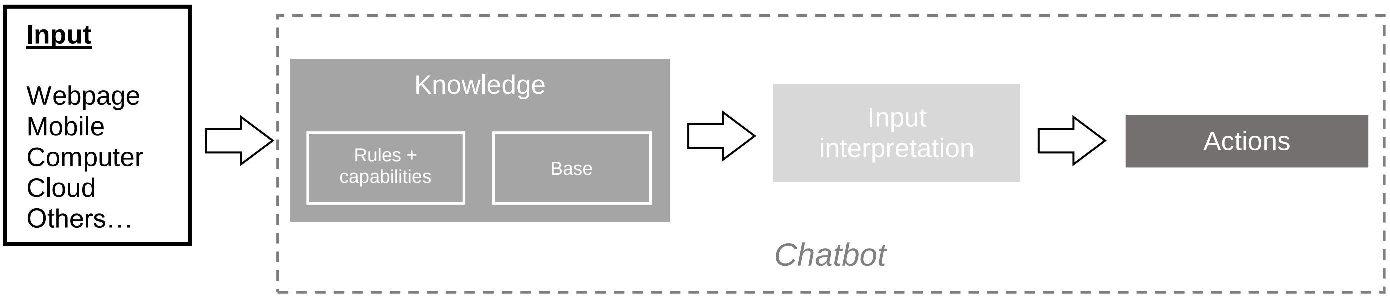


Figure 2: Schematic of a chatbot architecture.

Basically, chatbots process and understand the information a inserted to them based on a set of actions that are written towards the fulfillment of certain objectives. Hence, the first parameter to be considered into the schematic is the way and channel the chatbot can be contacted trough. There are no limitations regarding the possible interfaces as chatbots are deployed in different applications like Telegram, Facebook, web pages, Skype, mobile assistants or as part of an enterprise's computer software. It must be recaptured that the idea of a chatbot is to ease the communication in an enterprise (both externally -to the customers- and internally) and, hence, it is required that such solutions are provided with flexibility to adapt to the different communication channels that a company needs on a daily basis.

The internal part, what comes after the input, can be simplified into three main blocks.
Knowledge can be understood as the information that the chatbot will rely on during its interactions with the user. Knowledge is built on two elements. The set of rules and capabilities defines the bot's technical characteristics, what will limit the way the bot

operates and the level of depth it will handle, manage and understand the needs of the conversation. The set of rules and capabilities is the core of the chatbot development: those will depend on the needs and applications of the chatbot and finally end up defining its level of complexity. Those characteristics will also drive the way the chatbot will work and employ its knowledge. The knowledge is stored via a knowledge base, understood as the repository of the data that the chatbot translates into effective information. Following the current trend in the terminology, the term “knowledge base” is explicitly mentioned instead of “database” as the first is a broader concept that better adapts to the current nature of such repositories, that does not only consider a relationship of simple data but also proper knowledge (practical applications of the data) coming from different sources, even the user itself, and in different forms.[3] As a result, the chatbot will be capable to imbue the different interactions with a meaning and to convert those interactions into an action in response (2nd and 3rd blocks of Figure 2).

There are basically two levels of complexity in the capabilities of the chatbot; whether they will require certain level of AI or not. Simple bots with limited functionalities, as a chatbot that needs to answer to certain questions (best example would be for FAQ support) may not need a complex AI engine as they could mainly work based on a question-answer database. Nevertheless, as customers are increasingly getting accustomed to the use of chatbots, the expectations of the experience with them are also becoming more demanding. It is in most cases necessary that the chatbot reacts to natural language interaction in a natural way. For this reason, most of the solutions need to include natural language processing (NLP) elements, which are a “sub-field of AI that is focused on enabling computers to understand and process human languages” (as per [16]). With that, they escape from the limitations that the strongly structured computer language carries and provide a better “human-alike” interaction.

Chatbots can be written in almost any programming language (Python, Java, Ruby, R, C/C++…). It is therefore relatively easy to find packages or libraries with NLP elements that are often developed for a lot of different spoken tongues.[4] Even with the existing support and tools, programming of a chatbot completely from scratch is a complex undertaking only considered relevant when the final objective is the simulation of intelligence itself and therefore not of the essence in the current scope of study. On a practical note, the level of intelligence can be traded off against an easy (and cheap) development and deployment, as for most applications the actions and responses of the chatbot can be bounded to a limited set of objectives and actions. This way, most of the current solutions are business-oriented and offer a set of tools with which the developer can create highly personalized chatbots, without the need to make the effort of programming most part of the conversational intelligence themselves (what includes the interacting interface) and focus on adapting the bot to a group of objectives. This is the case as well of the current thesis.

Depending on the nature of the set of tools at hand for the creation of a chatbot, three different terms have to be considered. A platform is usually understood as the combination of hardware and software needed for the operation of a certain application (adapted from [17]). Under this concept, one can refer to solutions such as Chatfuel, pandora bots or MobileMonkey. Whereas most of them are a very good option to quickly deploy simple FAQ or chat bots for a dedicated communication channel, they are too limited to be part of complex business needs, which require more flexibility, adaptability and integration with external sources. For the project at hand, a framework

[3] This discussion has been active since the initial phases of the AI, see for example “On Knowledge Base Management Systems” pp 83-86 from Michael L. Brodie and John Mylopoulos (abstract online).
[4] See the following github article for a good overview of the current models and available NLP tongues.

is more convenient as, in the light of the definition of [17] and [18] it provides a basic structure based on libraries on which the developer can work. Most frameworks do not only include simple built-in bot models and coding examples dedicated to specific objectives, but also give the developer a great degree of freedom via in-line coding in Java or C++. Some of them even attach, among other features, code compilators, debuggers for error processing and connection to remote Application Programming Interfaces (API) or knowledge bases. This kind of frameworks are more generally referred to as Software Development Kits (SDK), as, in general terms, allow greater complexity to the application. In the present thesis, as both framework and SDK concepts are synonyms in most contexts, the term framework will be used.

The three most popular SDKs in the frame of chatbots are:

- Microsoft's Bot Framework and Azure Bot Service is an online service based on Visual Studio and C# programming procedure. It provides high flexibility and customized solutions. It includes also its own NLP called Language Understanding Intelligent Service (LUIS). In general terms it requires high coding knowledge to start working on the first bot creations.
- Google's DialogFlow (formerly API.ai) is supposed to overcome some of the difficulties presented by its competitors by making its interfaces and bot construction more intuitive and easier to work with and manage. Its main strength is the support of the Google analytics knowledge as NLP tool. The integration with external functionalities is relatively simple in comparison with other frameworks.
- IBM's Watson incorporates the capabilities developed during the Jeopardy mission (introduced in the paragraph 1.1 Background and rationale) to provide its bots with easier understanding of the language (which in turn leads to a better bot experience). It also facilitates the integration with knowledge bases and other IBM in-line services.

The three companies offer a variety of cloud services hence, bots are only a small part of the catalogue. Regarding the pricing all of them offer different solutions for free, aiming to attract companies to a number of basic functionalities and pull them this way to other costly services as the integration with cloud databases, newer functions, availability of premium communication channels, technical support, no limitations on the communications with the bots or analytical services on the different dialogues to capture potential revenue for the chatbot's owner.

2.2 Reinforcement learning

The latest developments in chatbots have abandoned the field of machine learning to substitute it with the principles of reinforcement learning. Reinforcement learning is about "taking suitable action to maximize reward in a particular situation" (definition taken from [19]). The bot evaluates the possible outcomes of every decision it can take and associates a potential value of reward to it. The final goal is to take the option that, potentially, obtains the maximal accumulated reward, hence the best strategy to get it.

The trained bot or rather agent learns from the environment, from the interactions with the outside and from the reward (and penalty) rules defined for it without direct intervention of a human being. An agent can also be trained almost from a blank initial state, making the potential applications of the algorithms very wide. Two of the most recent applications and best examples of the performance levels that such algorithms can reach come from the DeepMind company.

The first application was a deep learning model that autonomously learned how to play seven different Atari type games, outperforming all the previous attempts and even beating a human expert on some of them.[5] The input for the learning algorithm was the pixel composition and the output a function that estimated the possible punctuation. The learning model can be graphically understood via a YouTube video: the algorithm starts playing randomly and after certain learning time, once it has understood the working principles, finds the best strategy to win.

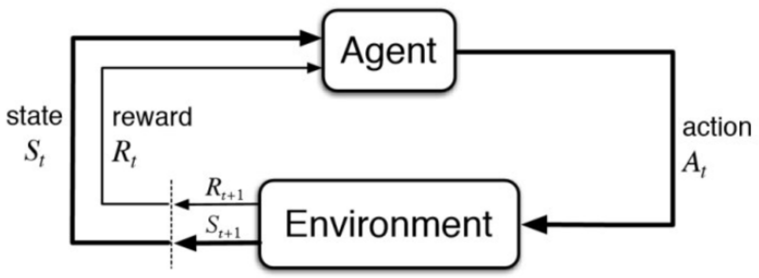


Figure 3: An agent takes an action based on the current state and the reward obtained. The action modifies the environment and defines a new state and reward. Source: KD nuggets

The second one is AlphaGo Zero, an evolution of AlphaGo. Its predecessor was the first computer program to win a human in the game of go, being trained with data from several real human games. AlphaGo Zero defeated AlphaGo by an interrupted 100 win-strike after learning the rules solely by playing against itself[6].

Chatbots are not exempt from the possibilities of reinforcement learning. Several researchers from the Montreal Institute for Learning Algorithms created MILABOT, which can converse with humans on popular small talk topics. For every input, the bot generates a set of answers from different models (template-based, knowledge based, neural networks…) and chooses the best among them: the best is, in this case, the response that represents the optimal trade-off between the user's short term and long term satisfaction The goal is, hence, to keep the counterpart as satisfied as possible for the longest time. After evaluating its performance with humans, MILABOT won over the other participants of the Amazon international university competition.[7]

2.3 Recommendation systems

Recommendation systems (the terms recommendation engine or method can be alternatively found) are one of the natural consequences of the boom of online business. A huge amount of information regarding (almost) every good that is on the market is only a mouse click away from the consumers. Recommendation systems are "information filtering systems that deal with the problem of information overload" (from [20], cited via [21]). So for every "user's preferences, interest, or observed behavior (a) recommender system has the ability to predict whether a particular user would prefer an item or not based on the user's profile" (from [22], also via [21]).

The goal is to provide the product that best suits the taste of a concrete consumer (the economic advantages of that is already explained in 1.1 Background and rationale). Given that taste and preferences are very subjective concepts, a recommendation engine needs to translate those perceptions into a set of rules that can be objectively

[5] Paper available here.
[6] See more info can be found in the project site.
[7] Paper available here.

interpreted. There are three broadly spread methodologies that support a recommendation system: content-based filtering, collaborative filtering and hybrid systems. Further methods such as demographic, knowledge-based or context-ware filtering systems are not discussed as their implementation is, to date, limited.

Content-based systems rely on a set of measurable properties and features for the different products in a catalogue. Whenever customer provide a valuation of a product, they are also indirectly evaluating its properties and features. Based on the knowledge the company has on the properties and features of its own products, every opinion coming from a customer can be extrapolated to the non-evaluated products based on the commonalities in features and properties the different products have. In a simple example, if X bought 10 products whose only commonality was its color, products with that same color would be preferably recommended to him or her. One of the major problems of this model is the need to have objectively defined information for every item in a catalogue; this task is very complex to tackle for highly subjective products such as movies or music, where the product description can neither be categorized nor clustered nor parametrized. Some systems have also the tendency to be "over-specialized", understood as the lack of the capacity to surprise, in other words, to keep recommending "items they (the customers) are already familiar with" [23].

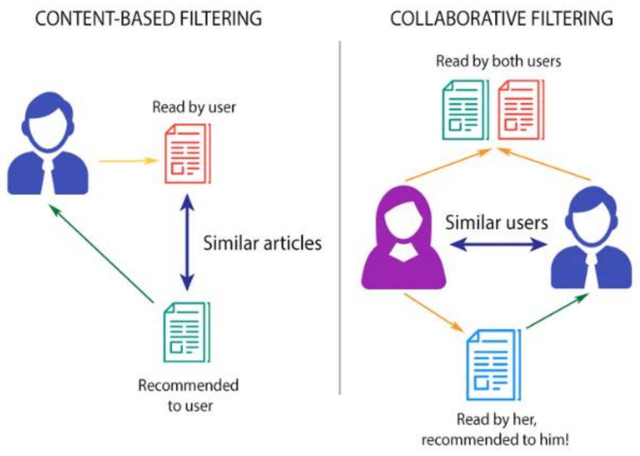


Figure 4: Content-based filtering vs collaborative filtering. Source: The marketing technologist

Collaborative filtering systems, on the other hand, look for similar profiles and match their tastes, following the principle that "items recommended to a user are those preferred by similar users" [24]. A relationship matrix between profiles and products is built, by clustering all the similar profiles in so-called neighborhoods. The user then gets the recommendation of products based on the behavior of the neighboring profiles.

There are numerous techniques to build a recommendation model based on neighboring profiles. Some lines tend to put the efforts into the creation of a strong model itself. Model-based techniques try to extract product features from the user's feedback and ratings, trying to reduce it to a group of relevant parameters that hold the key information. This approach is based on dimensionality reduction techniques, clustering and machine learning algorithms like association rules, neural networks or Bayesian classifiers; further literature can be found in the already mentioned reference [21] or the support bibliography [S1] – [S3] a series of introductory articles written by *Steeve Huang*. Memory based techniques try to find similarities between ratings and profiles. On the basis of a product rated by the target user and the neighbors, the predicted rating of a new product is extracted based on a weighted average of the neighbors' feedback on it. The neighbors that rate the given product more similarly to the target user have more relevance in the predictive model. The similarities can be evaluated via Pearson correlation coefficient, cosine similarity or Sigmoid functions.[8]

Even if collaborative filtering systems present some advantages compared to content-based ones, they also come with several drawbacks as they depend basically on the amount of information available. Such systems are fed with the feedback and profiling from the different users, so not only the profiling must be well oriented from the beginning but, as well, there must be enough data in the system to support the matrix and include all the products in offer. Collaborative filtering systems need also to be capable of handling the new amount of data in an efficient way, while avoiding overburdening the information repository with data that not only would not add any value to the recommendation model but may even add noise to it. Parallel to that, synonymy or abnormal profiles (for example, grey-sheep or gamers, see more in [25]) are complications in the path that also need to be dealt with.

The third group of techniques tries to overcome the problems and limitations of content-based and collaborative filtering systems. Hybrid filtering combines therefore different techniques, conceiving the idea than overlaying different techniques increases the effectivity and power of a single system while reducing its limitations. To cite some of the techniques, weighted hybridization techniques add the results of different recommendation techniques and adjust the corresponding weights given to each of them as the predictions are confirmed or not. Cascade hybridization applies "an iterative refinement process in constructing an order of preference among different items" [21].

Concretely in the film industry, the paradigm of a recommendation engine is Netflix. Netflix model is fed from different factors (input comes directly from the Netflix help page, more details can be found also in the Netflix publication referred in [26]):

- The interaction with the service (viewed titles, ratings, searches, viewing behavior…)
- The tastes or preferences of similar profiles
- Data on the products (title, genre, actors, awards…)
- Metadata (connection time of day, devices used for the view, session length…)

Based on the integration of different models, Netflix shows the recommendations split between different rows, each row usually coming from a different algorithm and ordered by importance. The personalized video ranker orders the complete catalogue available to a user and defines the ordering of the next rows. Top N recommendation locates the best personalized recommendations, focused on his or her profile. One differential characteristic of the Netflix model is the "Trending Now" row, that captures local trends (in time and/or in geographical location) that other models are too slow to detect. Netflix also introduces the similarity between products (in this case, the video-video similarity) that finds for every film in the catalogue a list of films with similar "characteristics".

In the internet a lot of literature with paradigms that try to overperform the actual recommendation engines can be found. Very useful is the one published by PhD. Jekaterina Novikova, see [S6], that includes some of the most popular recommending processes offered by the R package.

[8] The reader is here referred again to the support bibliography [S4] or [S5] for further information.

3 Creation of the chatbot and analysis of the results

In the previous chapter, the state of the art gave the context to the technical framework and the current technologies available for the deployment of chatbots. The purpose of this chapter is to translate the state of the art into a concrete practical solution. For such purpose, first the working environment of the theoretical company that would be deploying the bot, as well as a short introduction to the constituent elements of the chosen framework, are identified. After that, the architecture of the bot will be described in detail and analyzed towards the bot performance understood as its capability to fulfill three tasks: understanding the wishes of the human counterpart, learning from the architecture and fulfilling the query and recommendation functions prescribed.

3.1 Working environment

The bot is going to be deployed in a theoretical service company from the entertainment industry, concretely the movie sector. The company is in a process of capturing market swing and relevance and, having analyzed the potential impact of chatbots in the quality of customer service, has seized the opportunity to try one out with limited features to control the possible errors. The trial will be a test before taking the decision whether to integrate them completely as part of the customer service department. Initially, the chatbot will be supporting the search through different products of the company's catalogue and a basic recommendation of products as per user's taste.

The bigger impact (and hence risk) of the implementation of the chatbot is on the customer side: in the worst-case scenario the company could lose him or her after a bad experience with the bot. The chatbot will be created keeping the customer experience as a central point of the development. Taking that into account, the bot will receive the name of MovieBot.[9] The naming is chosen for two main reasons. On one hand, it defines clearly its field of expertise and by that also its functional limits (e.g. the bot will not talk about books); it will, as well, reduce the complexity of the bot as it can focus only on the functions it was created for. On the other, by clearly stating that the customer is communicating with a bot, he or she may be more prone to showing a higher tolerance for possible misunderstandings in the communication. Nevertheless, as will be explained later in more detail, the bot will keep always a door open to redirect the customer to other sources (a human or an internet site, for example) whenever a conversation is declared to be stuck. The goal is to avoid entering into useless communicative loops which would result in customer's dissatisfaction.

It is not within the objectives of the present thesis to characterize with all the details the theoretical company, hence its product catalogue will not be created but taken from the internet instead. The product catalogue will be divided into two categories: actors, actresses or directors and movies. Whenever a user wants to get information about an actor, actress or director, the bot will send him or her to the corresponding Wikipedia article (in its English version). Information regarding movies will be collected from The Open Movie Database (OMDB) API; for recommendations, The Movie Database (TMDB) API will be called instead. The two resources are for free and need only a registration. More explanation is given in the next chapters.

[9] The name is not unique and is used by other companies for products in the same scope. The name was conceived before this constatation and will be used only in the framework of the current master's thesis. No copyright infringements or similar intended.

As will be discussed later, the unlimited data set hinders to a certain extend the bot's actions; hence, a real product catalogue would be a particular case of the one here presented. In a real-life scenario, a company would have a reduced catalogue rather than the availability of the whole internet. For example, a similar bot for a furniture company would work with a known list of product IDs and relate them to the pdf archives with the mounting instructions or would ease the ordering of missing pieces. In that case the bot, instead of working with an external database or Wikipedia, would call the information directly from internal servers or databases.

3.2 Framework

The bot is launched under the IBM Watson cloud framework and, for simplicity, initially only programmed to understand the English language. The bot is requested to fulfill mainly the two goal functions, so a limited context is the best option as it allows the dialogue to focus on the bot's core competences in an easy way.

IBM's framework offers the most suitable set of tools and capabilities to build a rule-based bot:

- Simple construction of complex dialog flows
- IBM's machine learning engine included
- An easy deployment of the chatbot
- The possibility to integrate added functionalities, written by the developer or third parties, so the MovieBot could be extended if needed (connecting it to external APIs or an external recommendation system)
- A simple interface to try out the bot

Within the Watson framework, a bot is called assistant (for coherence, in the current text the term bot is used instead) and is built after three steps[10]:

1) Skill creation that gathers a set of data, training features and machine learning to understand the interactions from the user and provide an adequate answer
2) Deployment of a bot constituted by a set of skills in a communication channel (via link to be consulted on a browser, Facebook Messenger, slack or other)
3) Analysis and evaluation of the communication logs to improve the bot's structure

A free trial account is enough to cover the objectives of the thesis[11], as it comes with the basic features needed for a functional bot. There are certain limitations in the number of skills than can be created at the same time (five) and in the number of calls to APIs (10000 per month) but both are no limitation for the project's scope.

In this framework, a skill is the core of the bot, that what defines its capabilities. In IBM Watson a skill consists basically of four elements: intents, entities, dialog structure and content catalog. The other elements available (analytics, options and versions) are beta products or not available for free accounts.
Every interaction from the user (e.g. every sentence the it is given to the bot) is considered to have a purpose, to seek a concrete objective; in Watson framework such purposes are called intents. According to the nature, limitations and goals of the bot, its

[10] From https://www.ibm.com/cloud/watson-assistant/, a registration is necessary.
[11] For the creation of the account and the deployment of the functionalities is necessary to use servers located in London, as some capabilities are not available in other countries.

designer will define a set of possible intents. The bot's machine learning engine will be responsible to correctly allocate the different sentences or actions coming into the bot to one of the predefined intents[12]. More on that is explained in the following subchapters regarding the learning model.

There are elements in an answer that contain relevant dialog-intern information (e.g. the name of a product); those pieces of information are captured via entities, groups of values and variables that are necessary for the evolution of the conversation. In the IBM Watson framework there are two entity types at hand: synonym entities and pattern entities. If an entity has several synonyms allocated, the bot can recognize and classify mentions to those synonyms as the original entity itself ("suspense movies" instead of "thriller"). Pattern entities are regular expressions that an entity may follow (for example, an email would always present the structure text@text.text). For the MovieBot, only synonyms will be used for certain entities.

Nodes are *if condition-then action* blocks that constitute and steer the dialog. When the triggering condition of a node is met, it will activate a certain internal action on the bot's side: to wait for an input, activate an internal or external function, give an answer or jump to another node. Every node can have several child nodes and the dialog flow is ruled by a strong hierarchy: nodes are organized vertically, so the above node is preferred to the below one in case the bot cannot identify which one to activate; in the same way, a child node will only be active if the parent node is activated first. An example of a node design and a simple dialogue structure and internal processing is given in the Figure 5. In the IBM framework, intents are introduced by an # and entities by an @ (it is to be appreciated there how the bot captures the entity @Name and uses it in its answer).

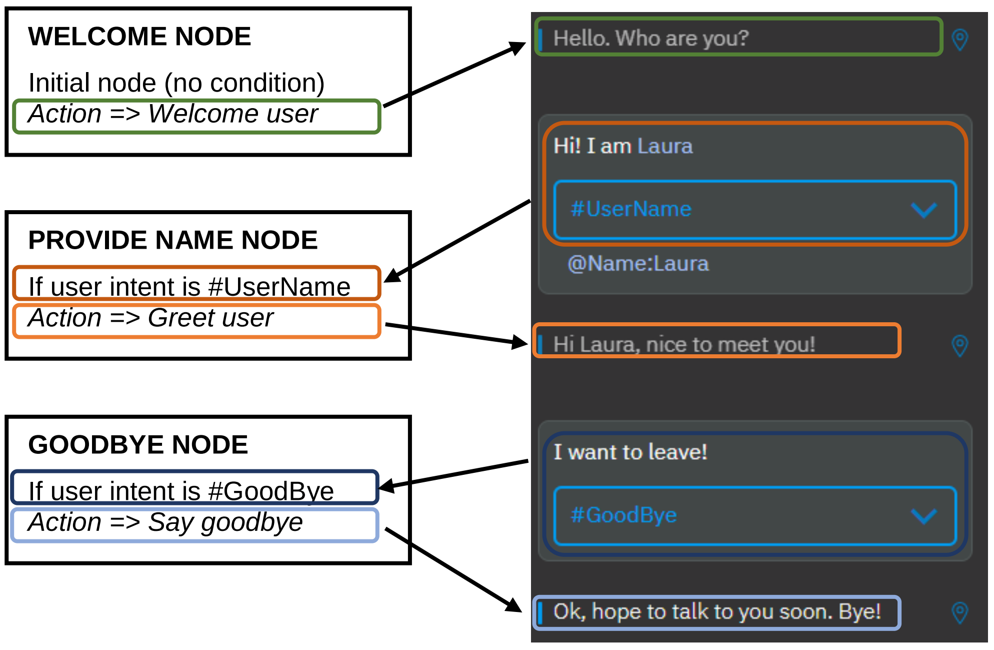


Figure 5: Conversation flow of a simple three node bot; every node has a condition-action defined. Machine learning matches every user input to an intent (introduced by #). Once a condition is fulfilled, the corresponding node is activated.

[12] Further info about the elements can also be found in the IBM help site, see for example intents or entities.

The fourth constitutive element of a skill is the content catalog. It is a library of existing typical intents already created by IBM that can be added to a skip to speed up its creation; those can be tailored as well for the concrete use case.

Regarding the deployment of the bot, a version of the MovieBot is accessible online here. Integration with other communication channels is possible but not planned. Analysis properties of the framework are not used due to the lack of real data and are substituted by trials via “Try it out” panel.

3.3 Architecture description

Once the basic constitutive elements of a bot within the IBM Watson framework have been described, the concrete structure of the MovieBot case can be treated in detail.

Even when the potential user of the bot is mainly a digital native, the MovieBot dialog architecture is built with the main objective to offer a quality service independently of the user’s profile or knowledge. Therefore, a single skill is created that will oversee all the functions and actions needed, mainly because it is easier to manage, interact and work with a unique skill.

A schema of the architecture appears in the Figure 6. The different boxes represent a simplified view of the nodes. The different nodes are triggered (mainly) by intents; to ease the understanding, the nodes are named after them. The hierarchy follows the rules previously described. White arrows show actions done and/or leading outside the bot perimeter; black arrows show the dialogue flow evolution induced by the chatbot. The active control of the conversation is introduced to limit the possibilities of the conversation landing on topics outside the core competences of the bot (for example, discussing about politics, for what the bot is not designed). Hence, as the schema shows, the conversation tends basically to bring the discussion to the two main functions of the bot.

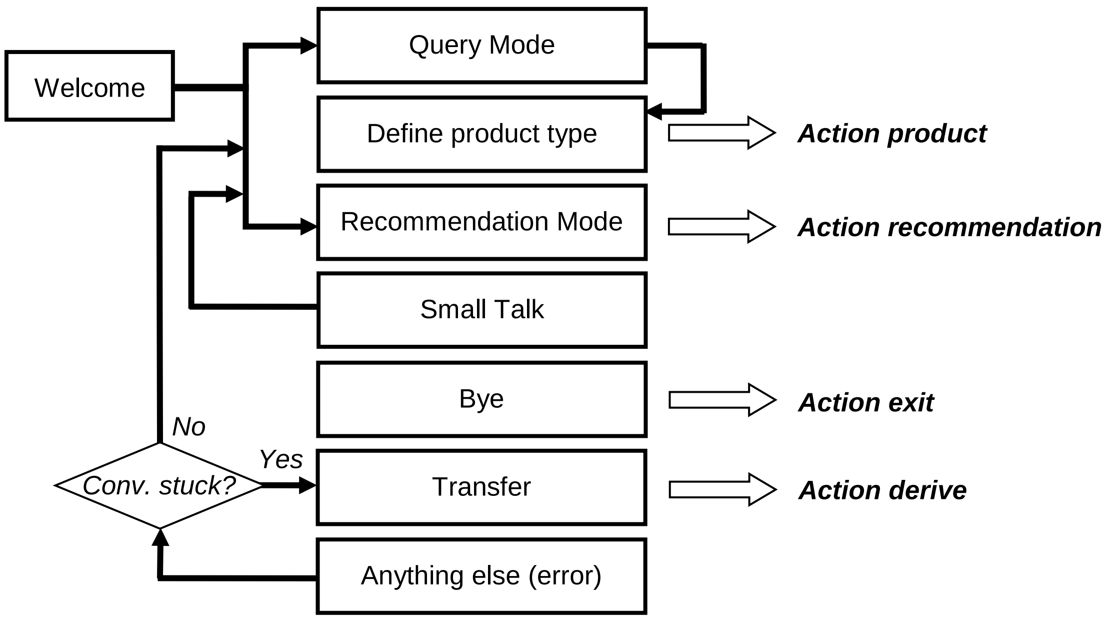


Figure 6: Schematic architecture of the MovieBot.

The explanations and structure here provided are kept simple and do not include all the details and aspects present in the final bot design for the sake of concision; the reader is invited to try the bot in its online deployment for a practical test of the bot's capabilities; its internal working processes can be checked directly in the framework.

Support data is also included. All the associated code, bot core architecture and short explanation are stored in a github repository, free of access. Support information can be found in the appendixes as well regarding the following:

- 7.2 Overview of entities and intents used
- 7.3 Call to the OMDB API (Prod_Film_Info web action)
- 7.4 Call to the TMDB API (Recommendation web action)
- 7.5 Access data to the frameworks used

3.3.1 Bot language and communication with the user

As the bot is an interface with the customer, several aspects will need to be taken into account regarding the language and communication means used by the MovieBot.

As the rest of the bot, the dialog construction will be rule-based; there is no need to simulate interaction with a totally intelligent human counterpart. Some criteria will be used as a baseline for the bot's language and understanding:

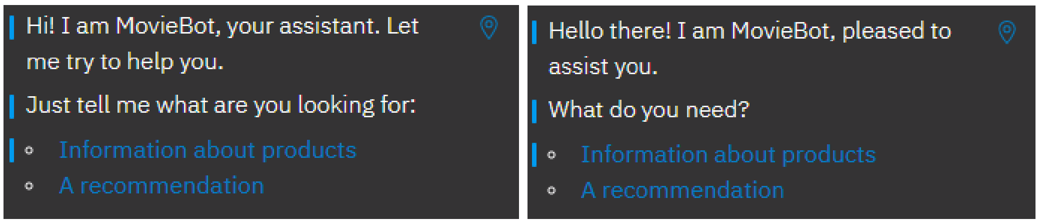
- Whenever possible, the bot will randomly choose between up to 4 different pre-set answers. That reduces the feeling of a repetitive and cold conversation.
- During the conversation, the bot will try to show understanding (revealing an intelligent counterpart), by reusing user inputs and confirming decisions taken. Reusing inputs helps to know where the conversation is at.
- Answers are formulated using a friendly tone and simple constructions to make the interaction more relaxed and easier.
- Keeping the cordiality, excusing after making a mistake or approaching possible problems from the positive side are implied characteristics of the bot.
- Being part of the entertainment industry, the MovieBot will be allowed to have a certain mocking and ironic touch in its expressions. Not only it will reveal a certain character and personality (hence making it more human), but it will also strengthen the company's image towards the customer.
- The bot will state clearly to the customer what he or she can expect from the bot from the beginning. The derived impact on the conversation is:
    a. On the one hand, to reduce the chances that user employs the bot for other purposes than the ones it is created for (very important as MovieBot bot has a limited functionality).
    b. On the other hand, to make the conversation path clear from the start.
    c. Both a. and b. help to keep the user expectations low and reduce the frustration from the limited interaction.
- The bot aims to simplify decision-making. That is why clickable options will be preferred in most of the points of decision. It is also a way to nudge the user towards the desired direction and to delimitate the scope of the conversation. Clickable answers will also help making the conversation more fluid.
- In every point, even when clickable options are offered, typed answers from the customer will also be accepted.

- Sentences will be concise and avoid long blocks of text, as they could overwhelm or distract the user, even more so in case the bot is going to be deployed in channels accessible via mobile phone.
- The user is allowed to change his or her mind during the conversation (it is done via "intent filtering", see more in the Figure 17). This feature will be available to a certain extent only, so the activation is not possible for every single step as it would make the discussion with the bot slower.
- To avoid the conversation to get in unproductive and never-ending loops, the bot shall detect when a conversation is stuck and, in such cases, try to direct it to a human agent or another information source.
- Whenever the user shows doubt or does not provide clear inputs the dialog should be redirected to clear decision gates.

All the constituent nodes are built keeping in mind the features mentioned before; the details or how it is translated into the actual architecture are addressed next.

3.3.2 Basic nodes: Welcome, Small Talk and Bye

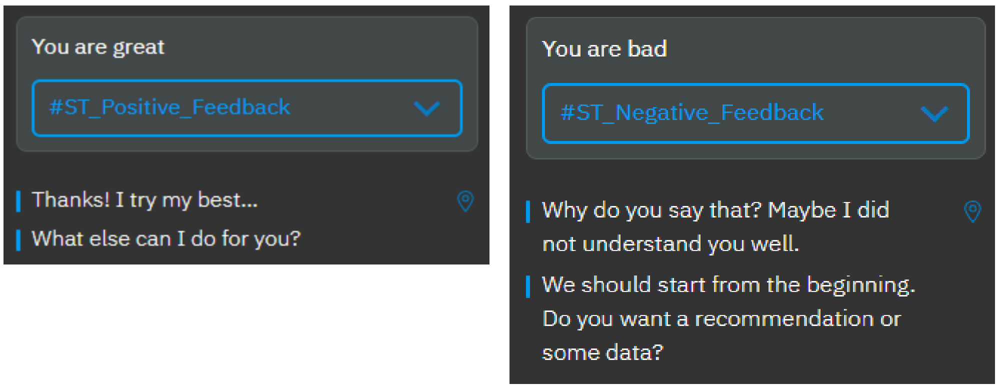
The Welcome node introduces MovieBot's capabilities and main functions. This first node is active per default, triggered before any interaction with the bot has been made. That gives the control of the conversation from the beginning to the chatbot, that takes the chance to introduce itself and, implicitly, its limitations.


Hi! I am MovieBot, your assistant. Let me try to help you.
Just tell me what are you looking for:
- Information about products
- A recommendation

Hello there! I am MovieBot, pleased to assist you.
What do you need?
- Information about products
- A recommendation

Figure 7: The bot is fed with 4 different formulations for the Welcome node which are randomly picked up every time the node is active. Blue colored text are clickable options.

The MovieBot will also be equipped with some small talk functions. Those are included to give a certain flexibility to the interaction with the bot, enrich it and to simulate certain social intelligence: the final goal is to enhance the customer satisfaction, so reducing the interaction with the bot to a "Yes/No" relationship would be counterproductive. Small talk functions will be also a good opportunity to reinforce the bot's personality and company's image.


You are great
#ST_Positive_Feedback
Thanks! I try my best...
What else can I do for you?

You are bad
#ST_Negative_Feedback
Why do you say that? Maybe I did not understand you well.
We should start from the beginning. Do you want a recommendation or some data?

Figure 8: The bot reacts to positive feedback and tries to induce a further use (left). In case negative feedback is provided (right), the bot reconducts the conversation to its core competences.

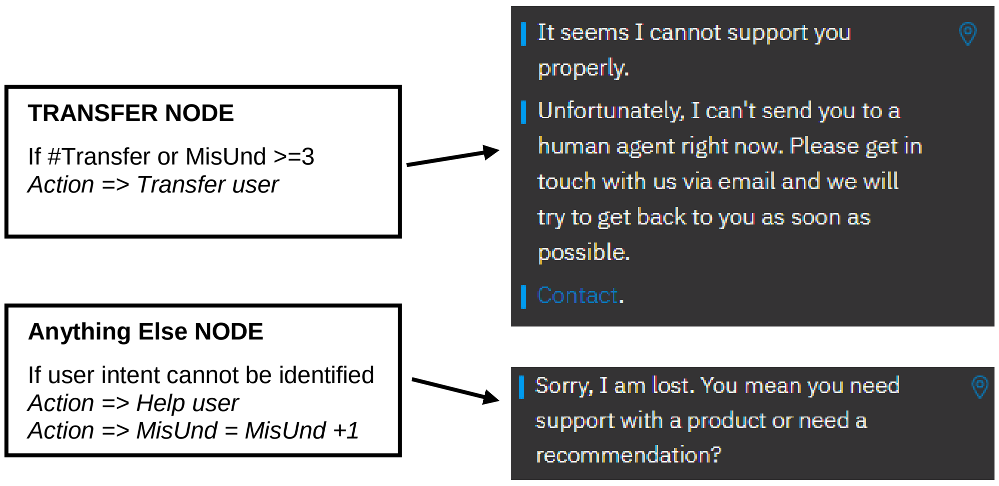
The bot is equipped with 9 different small talk intents: About MovieBot, Capabilities, General, Greetings, Jokes, Negative or Positive Feedback, Human or Bot and Quote being most of the intents reused from the content catalog and self-explanatory. Without avoiding questions as “Who are you?”, “How old are you?” or “What can you do?”, the bot provides a short info about itself and ends up mentioning its capacities, pushing (again) indirectly the conversation to its two main functions. Being a movie bot, if the user wants to hear some jokes, the MovieBot provides some quotes from movies instead. This node is also released when the bot appreciates that the user is not sure about what to do.

Finally, the “Bye” node is active if the user intents to leave the conversation. The bot closes then the communication and provides a farewell sentence. The user is offered the opportunity to jump to this node (finish the interaction) after every successful search of information or recommendation.

3.3.3 Error handling nodes: Transfer and Anything Else

The second group of nodes cover an important aspect of the bot, the error handling. They are intended to provide an adequate answer to the customer even when the interaction with the bot did not run as expected; both nodes serve as escape in case the conversation is not evolving properly, before the human agent gets fed up with the bot or the conversation lands in a never-ending vicious circle.

The node “Anything Else” is created by default and collects all the inputs from the user that the bot cannot allocate to any of the other nodes or to a concrete intent. In such cases, MovieBot is prepared to apologize for not understanding the interlocutor and tries to nudge him or her, again, to switch to any of the recommendation or information modes.


TRANSFER NODE
If #Transfer or MisUnd >=3
Action => Transfer user
Anything Else NODE
If user intent cannot be identified
Action => Help user
Action => MisUnd = MisUnd +1

It seems I cannot support you properly.

Unfortunately, I can't send you to a human agent right now. Please get in touch with us via email and we will try to get back to you as soon as possible.

Contact.

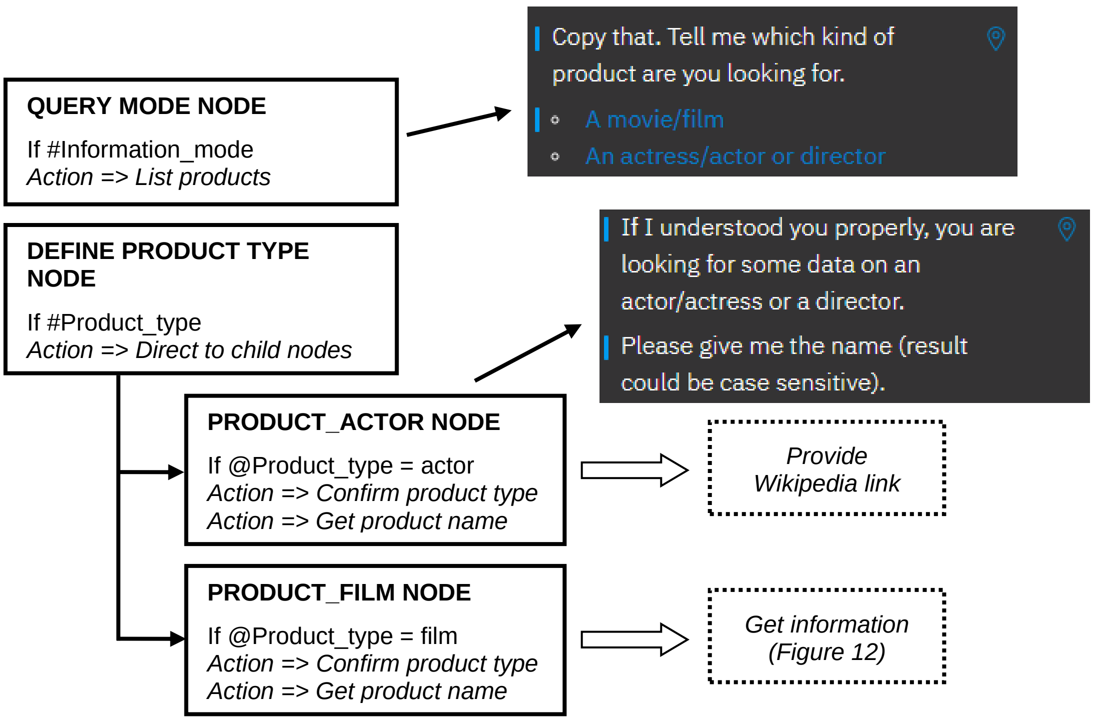
Sorry, I am lost. You mean you need support with a product or need a recommendation?

Figure 9: Example of MovieBot’s answers when dialog has to be transferred and when bot cannot identify an intent.

“Transfer” is activated under two circumstances. First, whenever the bot understands that the user is seeking for a new counterpart or contact point, the bot passes him or her to further support. Second, whenever the “Anything Else” node has been called more than three times: in that case, the conversation is declared to be stuck and the user is as well transferred to another instance. The control of the loops is done via the entity “MisUnd” that is internally augmented every time the “Anything Else” node is called; “MisUnd>=3” is set then as an alternative triggering condition of the “Transfer” node. Every time the chatbot starts a new conversation, the “MisUnd” entity is set to zero. Due to the limitations of the scope of the present thesis, instead of physically deriving the user to a human resource, the chatbot provides a link to a contact email. It is left to the reader to check the operation of these two nodes typing in a sentence like “I have a brown cat” in the dialog field of the bot.

3.3.4 Query nodes: Query Mode and Define Product Type

The group of query nodes execute the information function of the bot. If it detects the intent to use the information function (either directed explicitly from other nodes or thanks to typed interactions), the Query Mode node asks the user which product type he or she is interested in. Whenever the user chooses a product type, via the Define Product Type node, the bot asks and captures the name of the product, stores it in an entity and runs the different information gathering actions. Please be aware, the Define Product Type node is not a child of the Query Mode one. This way, while inputs like “I want information” would trigger “Please select product”-type of bot answers, sentences such as “I want information *about a movie*” would pass directly to the name capture node (as the user already selected the product type), increasing the sensation of interacting with an intelligent and understanding counterpart.



Copy that. Tell me which kind of product are you looking for.

- A movie/film
- An actress/actor or director

If I understood you properly, you are looking for some data on an actor/actress or a director.

Please give me the name (result could be case sensitive).

Figure 10: Schema of MovieBot’s behavior when the query nodes are active.

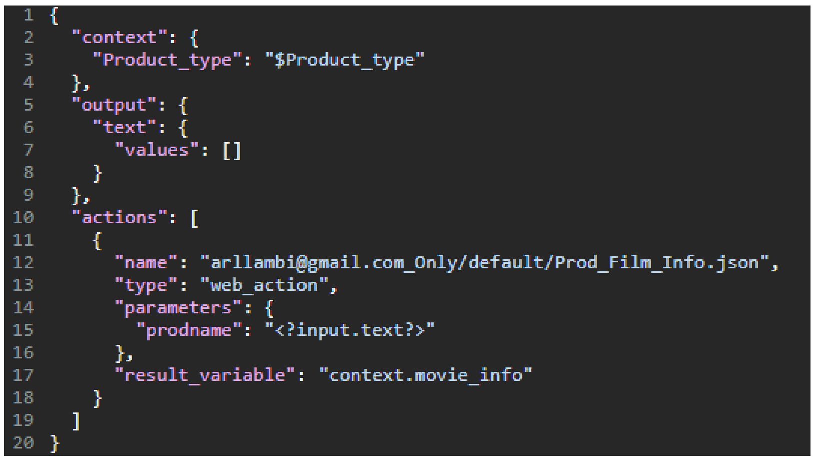
If the product is an actor/actress or movie director, the bot inserts the captured product name into a Wikipedia link (the user is advised to be careful with the capital letters, as Wikipedia addresses are in some cases case sensitive). If the product chosen is a film, the bot calls a web action (WA) to connect to the external OMDB API.

Within the IBM Watson framework, web actions are part of the programmatic calls (in some environments they may be referred to as webhooks instead). These are functions that a programmer can use to call a web application from a third party[13] (as would be the case of an API). Such programmatic calls can send requests to external

[13] More information is available in the IBM help site.

applications and get actions or answers back from them, as the communication process is usually done in a JSON data format.

The bot itself is not capable of directly calling an external application, so it must contact first a web action via a specific JSON call and pass the relevant information for the API on to it. The web action will be here in charge to call the external OMDB API, send to it the movie name captured by the bot and the access keys to the API, process the information that the API answers back (concretely, the request is done via promise, the reader is kindly asked to refer to the appendixes and/or the repository for more information), and prepare an answer in a form that the bot can understand and work with. The web action Prod_Film_Info is created within the IBM Watson framework specifically for such purpose and stored in the cloud. It can be reached by the bot via JSON call, directly written in the corresponding bot's node via the JSON editor available.



```
{
  "context": {
    "Product_type": "$Product_type"
  },
  "output": {
    "text": {
      "values": []
    }
  },
  "actions": [
    {
      "name": "arllambi@gmail.com_Only/default/Prod_Film_Info.json",
      "type": "web_action",
      "parameters": {
        "prodname": "<?input.text?>"
      },
      "result_variable": "context.movie_info"
    }
  ]
}
```

Figure 11: Example of a JSON call from the bot to the web action Prod_Film_Info.

As the API answer is in JSON format, an "undefined" type for the movie_info variable would mean that the prodname is not available in the database. In order to avoid the provision of wrong information, the bot reads the availability of the movie title first before delivering the information to the user. If the film is not available, the bot asks the user to do a new search or exit the bot; if the film is indeed available, the bot creates an answer based on the information provided by the API via the movie_info variable: a short resume of the plot is given.

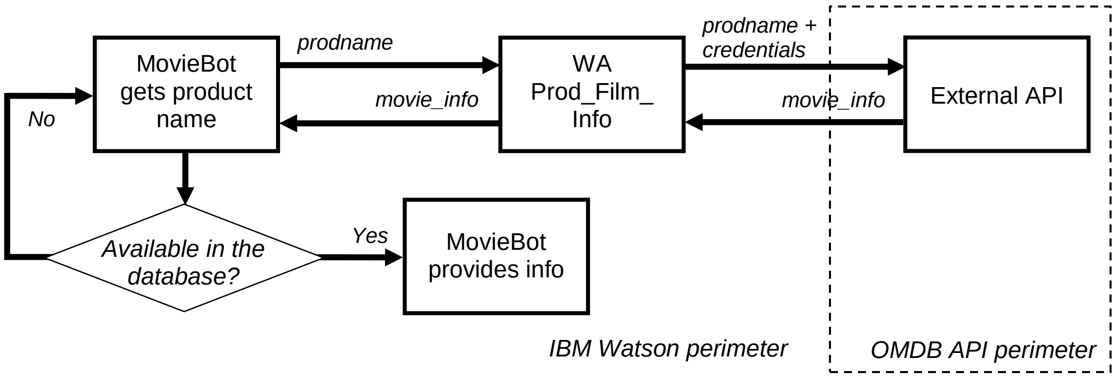
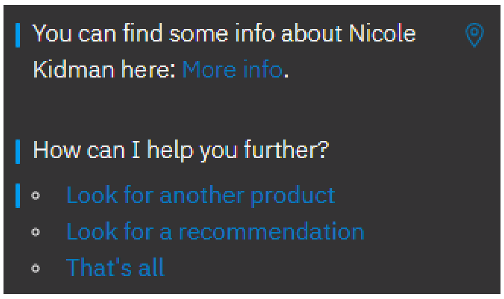


Figure 12: Schematic of the web action call for the query function.

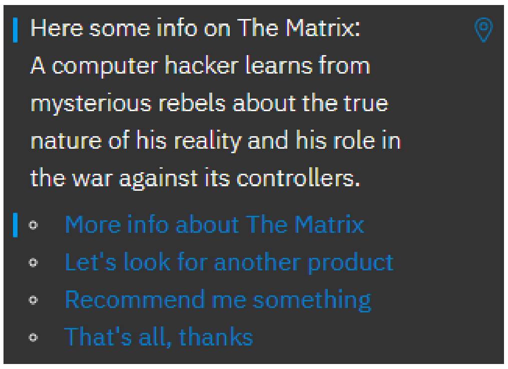
After the plot is sent, to keep the amount of information manageable by the user, only when further information is selected the bot provides the second set of data about the movie: the movie poster, the cast, movie genre, duration, year and a link to the homepage. Here, the bot takes a couple of seconds to answer; again, this leaves time to digest all the information provided.

Once the search is successfully completed, it can be selected whether to keep on searching for other products, to get a recommendation or to leave the bot.


You can find some info about Nicole Kidman here: More info.

How can I help you further?

- Look for another product
- Look for a recommendation
- That's all


Here some info on The Matrix:
A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.

- More info about The Matrix
- Let's look for another product
- Recommend me something
- That's all, thanks

Figure 13: Results of the search by actor (left) and by movie (right). After a successful search, different options are given to the user.

As a side note, the call to the API can find similarities between the database and the product searched, so in most cases it is not necessary to give the complete movie title to get results back. For instance, “James Caan” would call information on the movie “The Day Robert DeNiro Met James Caan”.

3.3.5 Recommendation nodes

The MovieBot offers the different recommendation models it has included whenever the recommendation intent is detected.

3.3.5.1 By genre and popularity

The first option recommends a movie based on two genres. The information from the OMDB can only be called after movie title, so randomly calling films until the one obtained has the two desired genres appears to be a non-optimal solution. TMDB on the other hand does allow filtered search, so the user can easily obtain a list of movie titles depending on directors, actors or genres. As the information from the OMDB for every film is considerably more complete, OMDB will be used to call the information from the recommended movie once a suitable title is obtained from TMBD. It must be noticed to the reader that the approach to use TMBD is freely based on the solution from Kissa Eric for Facebook Messenger in [27]; the tasks that will be followingly described, such as the specific code, integration to the IBM framework, architecture or characterization of the different quality models are tasks explicitly created by the hand of the author of the current thesis.

The genre information is captured via entity; TMBD only recognizes nineteen genres (action, adventure, animation, comedy, crime, documentary, drama, family, fantasy, history, music, mystery, romance, science fiction, tv movie, thriller, war and western). The movie genre conception is very vague, some genres may be overlapping (as war and action), and some may fall into different categories (as biopic, in the MovieBot case considered to be a documentary). To reduce the complexity, synonyms are provided for most of the entities. For example, “dystopic” is used as an alternative to science fiction, so the entry “I like dystopias” is accordingly classified by the bot under science fiction; the reader can refer to Table 3 in the annexes for more details.

Using two genres for the selection helps to refine the search and avoid some of the misunderstandings. After a genre is introduced, the first action by MovieBot will be to check if the genres can be allocated to one of the nineteen predefined ones.

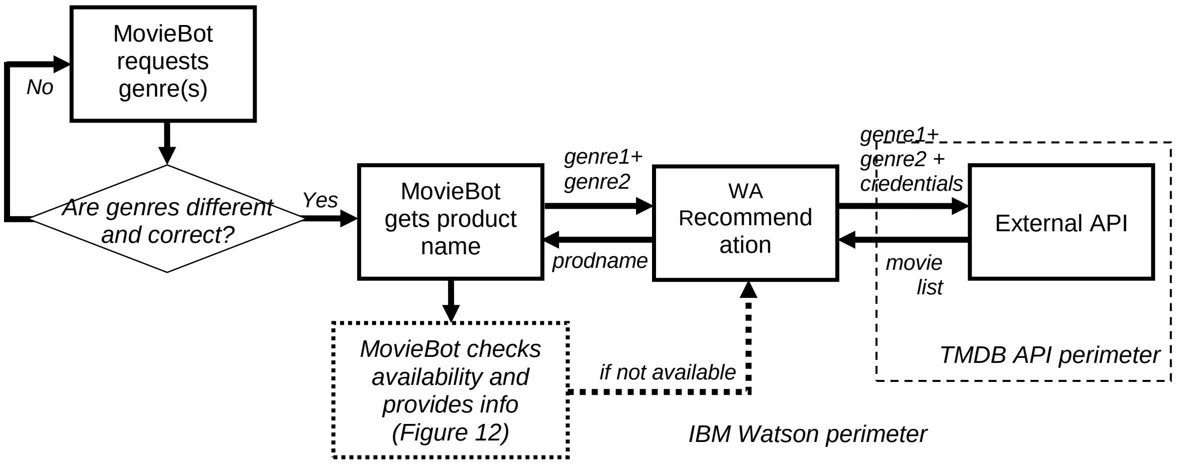
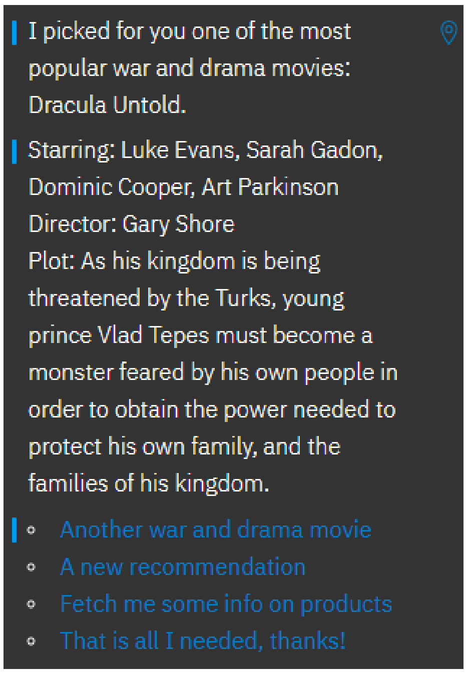

MovieBot requests genre(s)
No
Are genres different and correct?
Yes
MovieBot gets product name
genre1+ genre2
prodname
WA Recommendation
genre1+ genre2 + credentials
External API
movie list
MovieBot checks availability and provides info (Figure 12)
if not available
TMDB API perimeter
IBM Watson perimeter

Figure 14: Schematic of the web action call for the genre recommendation function.

Only when the user has entered three times a genre that cannot be allocated, the bot will list them all in order to allow the user to pick a correct one (the count is restarted after every new conversation start). The list of genres is avoided at the beginning to strengthen the image of intelligence of the bot (the user thinks he or she is able to enter anything, and the bot will understand it in most of the cases) and also to reduce, again, the overwhelming of the counterpart with "cold" formulations and long text chains.


I picked for you one of the most popular war and drama movies: Dracula Untold.

Starring: Luke Evans, Sarah Gadon, Dominic Cooper, Art Parkinson
Director: Gary Shore
Plot: As his kingdom is being threatened by the Turks, young prince Vlad Tepes must become a monster feared by his own people in order to obtain the power needed to protect his own family, and the families of his kingdom.

- Another war and drama movie
- A new recommendation
- Fetch me some info on products
- That is all I needed, thanks!

Figure 15: Result of a recommendation based on war and drama.

Once two different recognizable genres are captured by MovieBot, it will send them to the Recommendation web action that will call the TMDB API. The API will provide a list of movies ordered by descending popularity that fulfill the genre criteria. The web action will randomly take one of the first twenty titles and feed it to the MovieBot who will pass it on to the Prod_Film_Info web action. If the movie is available, the MovieBot provides some information related to the recommendation as shown in the left; if not, another title from the recommendation web action is automatically picked up. The availability check is done here for consistency, to avoid misalignments between the two different external API sources.

When the recommendation is finished, the user can get another recommendation based on the same genres, ask for a completely new recommendation, information about products or leave the bot.

The recommendation model is then a content-based one that will filter the results based on simple product characteristics captured by the API database and work only with a subset of products that explicitly fulfill the selection criteria. The products that will be recommended are those from the subset with a highest score regarding three parameters: popularity of the title (that captures the temporal evaluation of the film with parameters such as views or votes for the day, and previous day score[14]), vote count (number of people voted for a film) and vote average (the mean note). The preferable differentiation parameter between products seems to be the popularity as it is ruled by the temporal tendencies of the moment and weighs the newest films (more probably not seen yet by the user) higher, but also captures older movies, while vote average tends to highlight movies with 10 points but only reviewed by a very few users (hence not statistically conclusive) and vote count does not represent the final translation of the vote into a tangible result.

TOP 5 crime films from 30.05 – 05.06 ordered by		
Popularity	**Vote average**	**Vote count**
John Wick: Chapter 3	According to Matthew	The Dark Knight
Cold Pursuit	Dangerous Odds	Pulp Fiction
The Poison Rose	The Field	The Dark Knight Rises
John Wick: Chapter 2	Workers Con	Suicide Squad
Trainspotting	The Things We've Seen	The Shawshank Redemption

Table 1: Results from TMDB API for different quality criteria. Source: TMDB

The reader must take into account that the model could be easily refined adding another filter (for example, including the option to filter also by actor/actress or director). In this case, the call to the database would need to include the cast ID parameter. In databases it is common to work with IDs instead of the complete name strings (as they reduce processing time and avoid typos, among other benefits). TMDB is not an exception; see, for example, how the different genres are converted to the corresponding IDs in the appendix 7.4.2 Web action code before the call to the web API. To introduce a cast filter, the translation from the name to an ID should be done prior to the consultation of the database. As the complete database with the coupling cast – ID is not available and as the added value of that extension is limited and would follow exactly the same principle and architecture already shown in a practical way for the genre, this feature is not incorporated in the current release of the MovieBot.

3.3.5.2 Other recommendation methods

The recommendation based on genre is a methodology that does not exploit most of its business possibilities. For that reason, the MovieBot offers the possibility to get a recommendation following other methods; if that option is
selected, MovieBot answers that this action is left open for further developments.

The recommendation method via genre has a couple of limitations. On one side, the connection to the APIs is very inflexible regarding the call method itself and the shape of the results obtained (no user profiling possible, only few characteristics of the film given to build the recommendation…): there is small room left to implement any complex data postprocessing, at least in an efficient way, via web action. The only feasible alternative would be to use an external application to write a recommendation model and integrate it to the body of the bot. That alternative carries more restrictions. Leaving aside the nature of the coding itself (far from the technical scope of work), the connection to the APIs, data capturing and translation into a usable format by the corresponding language cannot be deployed straight forwardly. The integration of the code with the bot body itself turns out to be more complex than in the web action case.

[14] A complete list of parameters can be consulted here.

All in all, the integration challenge combined with the time limitation, hindered a satisfactory advance in this direction.

As a result of all the above an alternative method was partially outlined during the development of this thesis. Being the present work the conclusion of a study in data analytics, the alternative recommendation method is referred here in case the present work is retaken in the future.

The basic idea is to use offline databases instead of API connections (like the one from MovieLens, free of registration); that would allow the function to be easily written in any language without the need of calling APIs nor translating the data in an appropriate format and would resemble more a real life application. The MovieLens database is also richer in content compared to the external APIs, so the recommendation model could be fed with a greater number of parameters; there is even some individual feedback from different users, so the possibility to count with customer profiling in the analysis would also remain open.

The information available is scattered between different sources. The first step suggested is to integrate the needed data sources in a single working data set via the common element, the film ID, and keep the features from the product “movie” on which the model will be built (short code in R to build the data set is included in github). The following information would be kept:

- Title
- Adult (Y/N)
- Genres
- Overview (plot)
- Popularity
- Runtime
- Vote_average
- Vote_count
- Year
- Cast (actors)
- Crew (directors, producers…)
- Keywords

From the total amount of information compiled by MovieLens, those are the characteristics of a movie that, potentially, best define its acceptance by a viewer. Only the data regarding the movie awards would be needed to have a sound model. This could be taken easily from other sources.

Once the database would be gathered together, the user would be asked to provide three of his or her favorite movies, whose availability would be checked and, in affirmative case, the corresponding features for each of them loaded. The recommendation engine would then convert the different features into qualifiable values:

- Adult (Y/N) into 0/1
- Genres to a character string
- Overview (plot) to a character string

- Popularity normalized with the population
- Runtime normalized with the population
- Vote_average normalized with the population
- Vote_count normalized with the population
- Year normalized with the population
- Cast (actors) to a character string
- Crew (directors, producers…) to a character string
- Keywords to a character string
- Prizes normalized with the population

Even some of the features could be split into reduced fields: for example, prizes could be classified per "prestige" or actors as per role type (main, secondary…). That would be done for the three movies.

The next action would be to analyze the similarity between features (for example, a text comparison for plots) between the three films. To accelerate the process, an analysis of the relevance of the set of features could be included, discarding those that from the beginning don't appear to be driving the user's taste (for example, if the genres were completely different, or the years of release…).

Once the action is finished, in the first loop, a random weight would be given to each of the remaining features; based on it, a punctuation for every of the three movies could be calculated. So given a set of *N* final features, the punctuation of a film *i* may be defined as something in the shape of:

$$punct_i = (f1_{i,i+1})*w1 + (f2_{i,i+1})*w2 + \dots + (fN_{i,i+1})*wN$$

For i = [1, 3], $fN_{i,i+1}$ = similarity of feature N for film i against $i+1$ (were $fN_{i,i+1} = fN_{i+1,i}$) and wN the weight of the feature N.

At this point is where the reinforcement learning would take the lead: the objective would be to find a w1…wN that would obtain the maximum punctuation for the three movies at the same time. Once the final weighing would be obtained, the rest of the movies in the database could be evaluated and punctuated; the one with the higher punctuation would be the one recommended.

3.4 Learning model

Here the bot's working principles explained before are retaken: every user interaction is associated to an intent, intents that could trigger different actions from the bot. In order to correctly classify each and every user interaction to an intent, IBM's machine learning engine available in the framework is employed. For such a purpose, the bot programmer provides a set of typical user sentences for each intent. The bot is trained on them and after training, machine learning makes the bot able to correctly recognize (the concept *understand* could be here indistinctly used) further expressions outside the pregiven ones that might have the same intention behind.

In the next table it is shown a sample overview of the bot's learning capability. A welcome intent is trained with two different training sets: one and three sentences. Different further welcoming expressions are used to check whether the bot correctly identifies them (marked in green) or not (red) as a welcome intent.

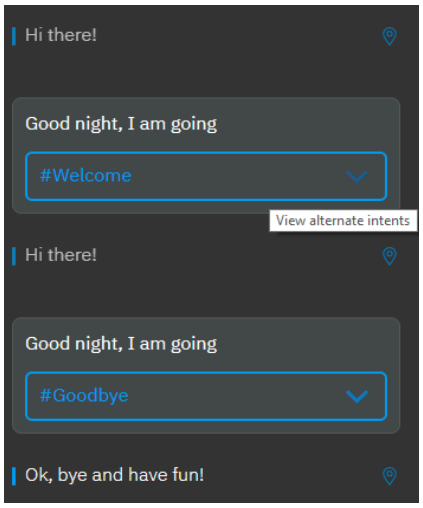
A first insight of both the understanding power and working principles of the engine can be already extracted from the example under. After training with a single sentence, the bot is capable of automatically understanding short interactions in different languages. It also uses some keywords to capture longer sentences (“I may need help” is not understood, but “Hi, I may need help” is). It is also worth mentioning that the classification performance can be very precise, so very similar sentences like “Are you there?” and “Is anybody there?” could be understood differently. As soon as the bot is trained with a new training set only slightly wider (the training takes little more than 30 seconds), its flexibility and classification skills also expand. Not only the bot understands the new training sentences, but also more complex constructions, even some typos that with the initial training set were not captured (see “Hy there” or “How#dy” for instance).

Welcome intent trained with single user example “Hi”		**Welcome intent trained with user example “Hi”, “How are you?” and “What’s up?”**	
🟩 Hello	🟥 Are you there?	🟩 What’s up?	🟩 What’s new?
🟩 Hola	🟥 I need some help	🟩 How are you?	🟩 What are you up to?
🟩 Hallo	🟥 What’s up?	🟩 How is it going?	🟥 Who is there?
🟩 Ciao	🟥 I may need help	🟩 What’s on?	🟥 Anybody out there?
🟩 Hey there!	🟩 Hi, I may need help	🟩 Are you there?	🟩 hw r u?
🟩 Howdy?	🟥 I am looking for somebody	🟩 What's going on?	🟩 How#dy?
🟥 Hy there!	🟩 Hello, I am looking for somebody	🟩 How’ve you been?	🟥 How’dy?
🟥 How is it going?	🟩 Is anybody there?	🟩 Where’ve you been?	🟥 ¿Qué tal estás?
🟥 What’s on?	🟥 Long time no see	🟩 Hy there!	🟩 Looking good

Table 2: Correct (green) or wrong (red) classification of different user inputs.

It can be extracted that the learning model can be easily refined feeding an intent with further examples of user’s formulations, making the knowledge base for the machine learning system wider. A quick possibility is to import intents directly from a .csv file.

The second option is manually correcting the bot whenever it is associating user’s input to the wrong intent. This can be done via the “Try it out” panel (see right). After initial training, machine learning captures the sentence “Good night, I am going” as a Welcome intent and answers accordingly. Via the “View alternate intents” this association can be manually changed to the correct intent (Goodbye in this case); after correction, giving the same input ends up with the classification to the right intent and consequently, the bot behaves as expected. In a similar way, a bot can also be “untrained” in case there are expressions that should not be allocated to any intent in particular; in this case, the alternate intent would be “irrelevant”.


Hi there!

Good night, I am going

#Welcome

View alternate intents

Hi there!

Good night, I am going

#Goodbye

Ok, bye and have fun!

Figure 16: Dialog flow pre/after manual training.

3.4.1 User input product name for query mode

Machine learning performs well enough to cover most of the intents needed for the MovieBot. The major difficulty of understanding comes with the user intent to provide a product name.

Once the MovieBot knows which kind product the user is interested into, it asks back regarding the name or title (go to Figure 10). From the user's answer, the MovieBot extracts the value of the entity *prodname*. Exactly capturing the correct content from the user answer is a very tricky step, as the structure of *prodname* is very variable. A movie title, for example, can be a single word ("pi"), a complete or incomplete sentence ("One flew over the cuckoo's nest" or "I, Tonya"), a number ("13"), the name of a public person or place ("Lincoln" or "Manhattan"), even nonexistent words ("AntZ" or "Fargo") or a combination of them. To that, the user may be providing the product name as part of a sentence (as in "I want information about the movie Titanic"). That makes it for the frameworks' machine learning engine very difficult to extract the correct value of *prodname* from the user input.

The first approach was to define a couple of synonyms of the *prodname* entity and train the bot with a set of sentences in the shape of: "Get me data on *prodname*" for each of the given synonyms. It was expected that the engine would be able to recognize the structure and correctly identify and extract the product name from user inputs outside the training set. Nevertheless, the engine was not capable of correctly understanding simple variations from the data. Not only similar formulations as "Fetch me data on" or "I want data on" but also simple variations from the given entities are not processed correctly nor new product names are recognized and passed to the corresponding action. There is no improvement in the performance if personal names are used instead. In this aspect, it can be stated that Google's DialogFlow engine is stronger; some more information on that topic can be found in the chapter 7.6 DialogFlow vs IBM Watson from the annex.

The second idea of using an intent trained only with movie titles was also discarded after few trials. The background logic was, that as the machine learning engine proved to be very capable of extracting knowledge from intents and of using patterns, it might be as well capable of finding hidden relationships in the movie titles if those were treated directly as intents. This capacity indeed increases when there is a rich variety of training examples but, nevertheless, the data needs to be organized to a certain extend. As stated above, that is not the case of a movie title (reason why using pattern entities is not working either). Simply giving a list of typical movie titles would never cover the complete range of possible movie title structures, so most titles would not be detected as such (chances of mistake will grow if other languages than English are included). Because of the limitations mentioned in the previous case, the translation of the intent to an actual entity is no trivial issue that would need to be handled as well.

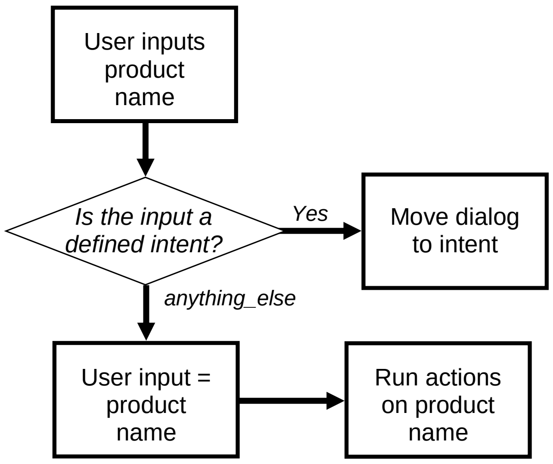

User inputs product name
Is the input a defined intent?
Yes
Move dialog to intent
anything_else
User input = product name
Run actions on product name

Figure 17: Schematic of filtering of user input via intents.

The described complication is solved via architecture by literally using any input from the user as the information to be passed to the following nodes. This way, any format of the product name would be processed. This strategy has a couple of derived problems. First of all, the bot would be taking explicitly *any* user input as the product name. Cases where the user would be changing his or her mind (e.g. "Sorry, I want a recommendation") would be treated as potential titles and not associated to the correspondent intents. Similar would happen with formulations like "Give me information about Titanic", where the complete sentence would be processed.

To avoid the first point, some filtering logic via intents can be implemented, logic that will check if the user input corresponds to certain known intents and if not, suppose that the user is providing a product name type movie. This "if not" condition is covered with an "anything_else" intent. With the defined structure, some product names could contain words or expressions that may wrongly identify them as intents (e.g. "Scary Movie" could trigger the action to select movie as product type). This can be tackled via the creation of a "Product_name" intent trained based on several problematic title expressions. After short training, IBM's Watson engine is powerful enough to differentiate between very similar expressions, hence to correctly read when user is calling to another intent or really providing a movie title. A similar result would be achieved if actual movie titles were "untrained", associated to the "irrelevant" intent.

The second point is substantially reduced by explicitly asking the user to only enter the movie title. It should not remain forgotten the fact that digital native users (the big part of the target audience of the bot) are used to similar platforms and to searching directly the name or title of the desired object, so in most uses, complex interactions are not expected. Anyhow, for films, a content availability check is already integrated (the availability of the title is checked against the API database, refer to Figure 12); the same check will declare typos or long sentences not corresponding to a film title as inconsistent and ask for a new title, after advising of the title requirements. This second step is not incorporated for actors, actresses or movie directors because there is no database available for the check. Hence, in case the user provided a wrong actor name, the Wikipedia link would lead to an empty page.

This issue is considered to have a low impact in the bot performance and is due to the fact that there is no real database available; in case there was one, a similar construction than the one for the movie title could be implemented. It must not be forgotten that the case here treated is developed inside a theoretical framework and with almost any input being valid as possible product name. In most real-life applications, the product name would be handled via a numerical ID with a fix structure that could be easily extracted from the user input and checked for validity against an organized and reduced data base.

3.4.2 General considerations

As a resume, the following points were taken into consideration when building the learning architecture of the MovieBot:

- Bot needs to be trained before being able to correctly allocate user inputs to intents.
- IBM Watson's logic is capable or differentiating and classifying very similar user expressions.
- As a consequence from the point above, frontier cases (similar expressions that correspond to two completely different intents) should be treated carefully. One

idea to reinforce the detailed understanding is the use of keywords unique to an intent.

- The bigger the set of training examples, the more precise are the model and the capacity to correctly classify user inputs.
- Training examples can be created and exported in .csv format.
- Format of the training examples should be rich, including different sentence lengths, expressions or vocabulary and flexible grammatical constructions.
- Most typos are understood after little training.
- Independently from the point above, the model shall be trained with typical typos (e.g. “recommendation” instead of “recommendation”) to make the understanding stronger.
- Model can be trained and untrained manually.
- Training sentences should be unique (the same sentence shall not trigger different nodes or actions).
- Intents can be used to filter out non-desired user inputs.
- Literally taking a user’s answer is a good procedure to facilitate the caption of non-standardized inputs (inputs that may have an undefined format, length, construction). This procedure needs to be combined with data filtering and checking processes.
- IBM Watson’s engine is not capable of extracting new entities from the ones given in the training set.
- IBM Watson’s engine is not capable of locating new entities in a sentence.
- Synonyms help the understanding and capture of important user information but do not counteract the two points above.
- Even when the bot interpretation engine is not case sensitive, the external applications could be.
- Node architecture affects bot’s comprehension. In the example, user input “war” will be understood as genre if the conversation is under recommendation function nodes and as a movie title if it is under query function nodes.

4 Conclusions

At this point of the document, once the chatbot is active and deployed, is time for a recapitulation to compare the initial goals of the thesis stated in 1.2 Objectives against the actual results obtained.

Along the chapter 3.1 Working environment, the context of the entertainment company, an application field for the MovieBot and a database were defined and with that the two first objectives. The internet will act as information database of the bot, as no product catalogue was available, nor one was going to be created (the later not being in the frame of the thesis); two products will be disposed for consultation: actors, actresses and directors or films. Information on the first products will be taken from the Wikipedia; for the films, two free of access APIs will be used: OMDB and TMDB. The first considerations can be extracted:

1) The bigger risk of the implementation of the chatbot is on customer side, so keeping user experience will vertebrate the complete bot architecture.
2) The theoretical environment is a general case that introduces restrictions to the bot architecture. Any real environment will be a reduced case of the defined one.

Being the implications of the second point explained along the next subchapters.

The bot is developed within the IBM Watson framework. IBM Watson framework provides a set of tools, learning engine, and easy training of a rule-based dialogue, what avoids creating the bot from scratch. The framework showed several advantages when compared to its concurrence: the free version allows an easy integration of external functionalities (very relevant for the communication with the external APIs) and the building up of complex dialogue flows, while being its deployment uncomplicated and compatible with different communication channels. This points together with the architectural elements that build the dialogue are extensively described in 3.2 Framework.

In 3.3 Architecture description is treated the fourth objective, the creation of the chatbot itself. Following the considerations previously extracted, the subobjective of offering a quality service to the user is also embraced during the process. The conversation of the bot consists of an overlapping of condition-action couples, grouped into nodes. A simplification of the resulting architecture is given in the Figure 6. A practical interaction with the bot can be run via its online deployment or directly in the framework.

Among others, the architecture of the conversation successfully confronted several issues:

3) The bot needed to show a certain personality in its answers, aligned with the image of the company it represents. To strengthen the image with a certain social intelligence, the bot was enhanced with some small talk capabilities and with a transfer function to another communication source.
4) The bot had to simulate a certain level of intelligence in terms of capacity to correctly understand the user, to control the conversation flow and to know its own limits. That translated into a language structure and architecture that:
    a. Uses concrete and clear phrasing and vocabulary, avoiding long sentences and formulations that may sound strict, cold or rude.

b. Has always an active control of the discussion flow in order to bring it to the main competences of the bot, to avoid getting in repetition loops or to manage unclear inputs from the user.

c. Tends to nudge the user to the core competences of the bot.

d. Is able to correctly classify the intent of a user input what includes having a certain flexibility to react to the change of mind of the user.

e. Keeps a simple and understandable dialogue evolution as the bot is planned to interact with different profiles and skills.

f. Inserts information extracted by the bot from the conversation in the answers to show understanding and reinforce the simulation of intelligence.

g. Analyzes the validity of certain user inputs.

5) Per se, the bot is incapable to connect with the external APIs. In order to fulfill the recommendation and query functions, the bot is integrated with two cloud functions that are the active responsible to call the information from OMDB and TMDB and pass it back to the bot.

The Figure 13 resumes the results obtained after the successful call of the query functions; the Figure 15 the ones from the recommendation function. The recommendation method introduced is content-based, that primarily filters out the data (in this case a list of films) based on two genres chosen by the customer. The recommendation system was driven by the technical and temporal limitations. Being the thesis developed within a data analytics framework, the opportunity to schematize in a very global way a second recommendation method based on reinforcement learning was not dribbled. The functions were described in detail in 3.3.4 Query nodes: Query Mode and Define Product Type and 3.3.5 Recommendation nodes.

All the previous conveys to the fifth and final objective, the characterization of the learning model for the bot. The objective is covered in 3.4 Learning model. The understanding of the bot is simulated via its capacity to correctly allocate a user input to the corresponding defined intent. Each of the intents is associated to a set of triggering sentences, words or parameters; after that, the bot is automatically trained to understand deviations from the triggering data via machine learning. In case the automatic training is not working properly, the framework also includes the possibility to manually train the bot; an example is given in the Figure 16. For the MovieBot case, the complexity arises when the bot has to locate the product name inside the user's answer. Product names are non-standardized inputs (they do not have a unified format, length or structure) nor are pared to any IDs, so the engine is not capable of correctly capturing them. As a side note, in this direction Google's DialogFlow proved to have a better performance in this aspect; as these concrete points are outside the scope, the comparison between both frameworks is attached in the chapter 7.6 DialogFlow vs IBM Watson from the annex.

The limitation is solved via architecture: idea is to accept any input from the user as a potential product name, validate the intent of the input via filtering (see Figure 17) and then, for movies, check against the API if the title exists. As for actors, actresses and directors there is no database, the final check is not done for that kind of products. Being most of real-life applications a particular case of the MovieBot, and as a solution is implemented and working for the movies, the issue is considered to have a low impact in the final performance of the bot, as a real case would allow an immediate correction.

3.4.2 General considerations contains the set of key elements for the training of the MovieBot, being the most important of them:

6) The bigger the set of training examples, the more precise are the model and the capacity to correctly classify user inputs; that includes a rich formatting of training sentences in terms of length, structure or vocabulary, also including typos.

7) IBM Watson's engine is not capable of extracting new entities from a training set, nor to segregate them from a sentence.

8) Using synonyms for entities improves the bot's performance when capturing important user information; this does not counteract the previous drawback.

9) The use of keywords in the training set (concepts that can be indistinctly associated to an only intent) reinforces the understanding capabilities of the bot.

In the introduction to the work, it was justified that there is a market potential in the deployment of chatbots and that those chatbots can be the spark to a long-term business strategy. Chatbots could be initially used as customer interface and as entry to future business developments or other areas of the company. That approach placed the biggest risk on the customer, so the bot was built to have certain social skills and to use easy language and conversation flow, to adapt to different user profiles. The recommendation function was included to increase the impact of its deployment on the company's revenue; the same impact in the revenue could potentially subsidize part of the investment needed for the test phase of the deployment.

The final pursuit of the thesis was to show in a practical and cost-effective way how a chatbot could be deployed and support a theoretical company. It can be concluded from the above points that the pursuit is satisfactorily achieved.

For that, some modifications in the initial scope of work and project planning were introduced to adapt them to the evolution of the project and as the first results were obtained. The biggest redirection was in the project objectives. The first idea was to center the work around reinforcement learning methods. After some research in the area, the approach turned out to be too wide and technically far from the core competences of the field of studies (business analytics). That justified a change of focus towards more practical applications of derived methods as machine learning supported chatbot applications.

The difficulties arose from the limitations in technical knowledge were also palpable after the first bot models were created and the need to enhance their basic functionalities via external APIs was faced. Some learning in JSON and Java coding was necessary to cope with that phase, as well as in the internal working methods of the different frameworks. That derived into the change from DialogFlow to IBM Watson treated in the annex what also impacted the scheduled plan.

Even after the new project objectives, it was considered that the influence of reinforcement learning in the bot development was still slightly present along the bot training phase. Reinforcement learning methods showed also a big potential in a future improvement of the bot. For those reasons, a short resume in that aspect has been kept in the chapter 2.2 Reinforcement learning. That fact is a good link to step into the anteroom of the end of the conclusions, the open points.

On a technical level, the following points were not completely developed:

i) Rule-based conversation always has the counterpoint that reduces an infinite problem to a set of limited condition-reaction pairs; that could be improved using reinforcement learning techniques that would pick up between different answer options.

ii) For the same reasons as stated above, in some cases, bot's answers may appear to be a little bit cold and out of the blue. That could be improved by working on a better formulation of bot's answers and including more nodes outside the core competences.

iii) A better recommendation model can be implemented following the explanation given in 3.3.5.1 By genre and popularity and 3.3.5.2 Other recommendation methods.

iv) A user-oriented recommendation model or even bot interaction could be also implemented. The bot would adapt its answers, language or even the conversation style to the user profile.

v) If the product name corresponds to a movie saga, the API provides directly the first of the list. To avoid these cases, a disambiguation capability (similar to "Did you mean…?") would be necessary. That could also redirect misunderstood user intents.

The potential of the bot could be enhanced with further functionalities. The ones considered to have more relevance are:

vi) The bot could be used as a gate to gather consumer information for further offline customer profiling. That would be useful to study consumer tendencies, common doubts or to feed a better recommendation system.

vii) Actively integrate a FAQ mode to tackle most basic questions regarding the bot functionalities, products and also regarding the company.

viii) Being an international company, the bot should incorporate other languages than English.

ix) Bot should be able to react to voice interaction as well.

x) Bot should be deployed in all the communication channels from the company it represents.

xi) Bot should be integrated with human-driven user attendance.

Even when the real impact of the bot on the quality of customer services or its cost would need to be specifically validated (interviewing customers, defining test and control groups…), the current design of the MovieBot has a degree of maturity and of consideration of the customer needs that makes it ready for a first trial deployment without the need to include any of the open points listed.

5 Lexicon

AI / Artificial Intelligence: is the capability of machines (robots, informatic applications, computer programs…) to act in an intelligent matter. Artificially intelligent machines learn, understand, use knowledge or solve problems with a minimal (or no) support of a human counterpart.

API / Application Programming Interfaces: given a software system, is a group of built in tools, methods and routines that ease the data gathering from external sources after only small programmatic effort (see OMDB and TMBD).

Chatbot or chatterbot: a computer application capable of simulating the conversational skills from a human.

Collaborative filtering systems: within recommendation systems, collaborative filtering systems recommend products based on the user profile.

Content-based filtering systems: within recommendation systems, content-based filtering systems recommend products based on the properties and features of the products in a catalogue.

Database: organized collection of data part of a computer system or application.

Entity: constitutive element of a bot within the IBM Watson framework. An entity can be understood as dialogue-relevant information.

Framework: provides a basic structure based on libraries on which a developer. Related to the creation of bots, most frameworks include simple built-in bot models that can be adapted via in-line coding in Java or C++.

Hybrid systems: within recommendation systems, systems that offered a combined answer from collaborative and content-based filtering systems.

Intent: constitutive element of a bot within the IBM Watson framework. An intent can be understood as the intention behind every user input to the bot.

JSON / JavaScript Object Notation: readable data format used in the exchange, storage and transport of information between applications. Its application appears mainly in server – web page connections.

Knowledge base: information repository that stores data from different sources and forms. Knowledge base add to the data a set of rules, metainformation and access policies that allow a practical application of the stored information (a real knowledge in contraposition to databases).

LUIS / Language Understanding Intelligent Service: Microsoft's NLP.

Machine learning: ability of a computer system to automatically learn (in terms of modifying and optimizing its operation) from its own experience and use.

NLP / Natural Language Processing: set of elements that enable a computer to understand and process human language.

Node: constitutive element of a bot within the IBM Watson framework. Nodes are *if condition-then action* blocks that steer the dialogue.

OMDB / Open Movie Database: free of access movie repository API. Given a movie title, the API provides back a set of characteristics in a JSON format (actors, plot, poster…).

Platform: combination of hardware and software needed for the operation of a certain computer application (adapted from [17]).

Recommendation system: via information filtering, recommendation systems manage information about products. The goal is to provide the product that best suits the taste of a concrete user. The most broadly implemented methods are content-based filtering, collaborative filtering and hybrid systems.

Reinforcement learning: an area of machine learning where bots (called agents) are trained to take decisions in order to maximize the accumulated future reward in a defined environment.

Rule-based chatbot: in contraposition to a self-learning chatbot, rule-based chatbots are basically intended to follow a set of predefined rules and perform a series of actions. Such chatbots are easy and cheap to be implemented.

Self-learning chatbot: Self-learning chatbots (also called AI-based bots), on the other hand, use machine learning and natural language processing methodologies to technically understand the intents of the different user messages and take the best decisions in an almost autonomous way. Self-learning chatbots can be split between retrieval-based and generative bots.

Skill: within the IBM Watson framework, set of capabilities that, together, constitute a bot. A skill is built on entities, intents and nodes.

SDK / Software Development Kits: a framework that, in general terms, allows the application to have a greater complexity as they may add other features to those of frameworks such as code compilers, integration with APIs or similar.

TMDB / The Movie Database: free of access movie repository API. Given a movie title, the API provides back a set of characteristics in a JSON format (actors, plot, poster…). Answers from TMDB can be filtered via different parameters (actor, genre, year of release). In this case, a list of movies that fulfill the filtering criteria is delivered instead.

VCA / Virtual Customer Assistants: as per Gartner's classification [9], chatbot responsible for automating the written and voiced communication of the call centers.

VEA / Virtual Enterprise Assistants: as per Gartner's classification [9], business assistants that are intended to simplify communication and access to information.

VPA / Virtual Personal Assistants: as per Gartner's classification [9], chatbots responsible for managing third party services.

WA / Web Action: within the IBM Watson framework, function deployed in the cloud that can be called from other applications.

6 Bibliography

6.1 Cited bibliography

[1] “Paper Review 1: ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine”; Fatih Çağatay Akyön; NLP Chatbot Survey; 11.10.2018; online source; last consulted on 31.03.2019

[2] “The history of chatbots - Infographic”; no author; futurism; no release date; online source; last consulted on 31.03.2019

[3] “Clippy and the 50-Year History of Chatbots“; Run Dexter (blog); no release date; online source; last consulted on 31.03.2019

[4] “*Chatbots: lo que debes saber para incluirlos en tu estrategia de contenidos”*; Alvaro Rodríguez; 40 de fiebre; no release date; online source; last consulted on 31.03.2019

[5] “What are the benefits of using a Chatbot?”; Maruti Techlabs; no release date; online source; last consulted on 31.03.2019

[6] “Top 7 Benefits of Chatbots for Your Business”; Asena Atilla Saunders; Digital Doughnut; 08.11.2017; online source; last consulted on 31.03.2019

[7] “4 Uses for Chatbots in the Enterprise”; Susan Moore; Gartner; 06.07.2017; online source; last consulted on 31.03.2019

[8] “5 ways chatbots are revolutionizing knowledge management”; Matt Wade; AtBot.io; 12.02.2018; online source; last consulted on 05.04.2019

[9] “Market Guide for Conversational Platforms”; Magnus Revang, Van Baker, Brian Manusama and Anthony Mullen; Gartner; 20.06.2018; online source; last consulted on 02.04.2019

[10] “Gartner Says 25 Percent of Customer Service Operations Will Use Virtual Customer Assistants by 2020“; Susan Moore**;** Gartner; 19.02.2018; online source; last consulted on 02.04.2019

[11] “Gartner Predicts a Virtual World of Exponential Change”; Heather Pemberton Levy; Gartner; 18.10.2016; online source; last consulted on 01.04.2019

[12] “Amazon Recommendation System”; Ding Ding; Medium; 06.12.2017; online resource; last consulted on 15.04.2019

[13] “How retailers can keep up with consumers”; Ian MacKenzie, Chris Meyer, and Steve Noble; McKinsey; 10.2013; online resource; last consulted on 01.04.2019

[14] “Why Netflix thinks its personalized recommendation engine is worth $1 billion per year”; Nathan McAlone; Business Insider; 14.06.2016; online resource; last consulted on 01.04.2019

[15] “Building a Simple Chatbot from Scratch in Python (using NLTK)”; Parul Pandey; Medium; 17.09.2018; online resource; last consulted on 03.04.2019

[16] “An easy introduction to Natural Language Processing”; George Seif; Towards Data Science; 02.10.2018; online resource; last consulted on 28.04.2019

[17] “Framework vs. Platform”; Stephanie Wittmann; Commerce Tools; 14.08.2018; online resource; last consulted on 20.04.2019

[18] “Difference between IDE, Library, Framework, API, and SDK”; Jainil Vachhani; codoholic confessions; 13.06.2017; online resource; last consulted on 01.05.2019

[19] “Reinforcement Learning”; Prateek Bajaj; Geeks for Geeks; no release date; online resource; last consulted on 05.2019

[20] “Recommender systems: from algorithms to user experience”; Joseph A. Konstantin and John Riedl; User Modeling and User-Adapted Interaction; Springer; 04.2012, volume 22, Issue 1-2, pp 101–123; online extract (mentioned via [21]); last consulted on 13.05.2019

[21] “Recommendation systems: Principles, methods and evaluation”; F.O.Isinkaye, Y.O.Folajimi and B.A.Ojokoh; Egiptian Informatics Journal Volume 16, Issue 3; 11.2015; online resource; last consulted on 13.05.2019

[22] “Research paper recommendation with topic analysis”; Chenguang Pan and Wenxin Li; 2010 International Conference On Computer Design and Applications; IEEE Explorer; 09.08.2010; online extract (mentioned via [21]); last consulted on 13.05.2019

[23] “Trends, Problems And Solutions of Recommender System”; Dr. Sarika Jain, Anjali Grover, Praveen Singh Thakur and Sourabh Kumar Choudhary; National Institute of Technology Kurukshetra; 05.2015, International Conference on Computing, Communication and Automation (ICCCA2015); online resource; last consulted on 07.06.2019

[24] “Recommendation Systems”; Jure Leskovec, Anand Rajarama and Jeffrey David Ullman; Mining of Massive Datasets; Cambridge University Press; 12.2014, chapter 9, pp 292-324; online resource; last consulted on 28.05.2019

[25] “Challenges in building recommendation systems”; Rabin Poudyal; Medium; 09.07.2018; online resource; last consulted on 01.06.2019

[26] “The Netflix Recommender System: Algorithms, Business Value, and Innovation”; Carlos A. Gomez-Uribe and Neil Hunt; Netflix Inc; 12.2015, ACM Trans. Manage. Inf. Syst. 6, 4; online resource; last consulted on 11.06.2019

[27] “Building a Messenger Movie Recommendations Chatbot in 20 Minutes Or Less”; Kissa Eric; Chatbotslife; Medium; 31.12.2018; online resource; last consulted on 11.05.2019

6.2 Support bibliography

[S1] “Paper Review: Neural Collaborative Filtering Explanation & Implementation”; Steeve Huang; Towards Data Science; 03.09.2018; online resource; last consulted on 10.06.2019

[S2] “Introduction to Recommender System. Part 1 (Collaborative Filtering, Singular Value Decomposition)”; Steeve Huang; Hackernoon; 28.01.2018; online resource; last consulted on 10.06.2019

[S3] “Introduction to Recommender System. Part 2 (Neural Network Approach)”; Steeve Huang; Towards Data Science; 16.02.2018; online resource; last consulted on 10.06.2019

[S4] “An improved memory-based collaborative filtering method based on the TOPSIS technique”; Hael Al-bashiri, Mansoor Abdullateef Abdulgabber, Awanis Romli and Hasan Kahtan; Plos One; 04.10.2018; online resource; last consulted on 09.06.2019

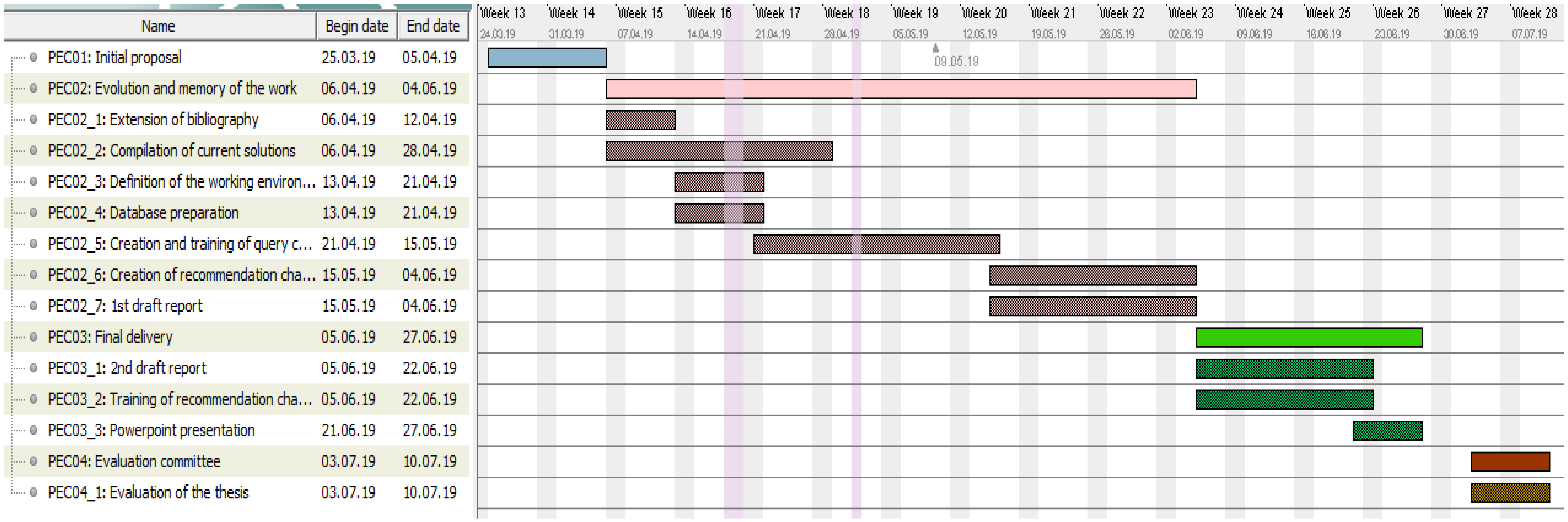
[S5] “A survey of memory based methods for collaborative filtering based techniques for online recommender system”; Anuj Verma and Kishore Bhamidipati; Academia; 04.2013, International Journal of Computer Engineering & Technology (IJCET), Volume 4, Issue 2, pp 366-372; online resource; last consulted on 09.06.2019

[S6] “Building a Movie Recommendation System”; PhD. Jekaterina Novikova; RPubs; 06.2016; online resource; last consulted on 14.06.2019

[S7] “Creating a NodeJS based Webhook for Intelligent Bots”; Siddarth Ajmera; Chatbotslife; Medium; 07.10.2017; online resource; last consulted on 20.04.2019

7 Annex

7.1 Flowchart



Name	Begin date	End date
PEC01: Initial proposal	25.03.19	05.04.19
PEC02: Evolution and memory of the work	06.04.19	04.06.19
PEC02_1: Extension of bibliography	06.04.19	12.04.19
PEC02_2: Compilation of current solutions	06.04.19	28.04.19
PEC02_3: Definition of the working environ...	13.04.19	21.04.19
PEC02_4: Database preparation	13.04.19	21.04.19
PEC02_5: Creation and training of query c...	21.04.19	15.05.19
PEC02_6: Creation of recommendation cha...	15.05.19	04.06.19
PEC02_7: 1st draft report	15.05.19	04.06.19
PEC03: Final delivery	05.06.19	27.06.19
PEC03_1: 2nd draft report	05.06.19	22.06.19
PEC03_2: Training of recommendation cha...	05.06.19	22.06.19
PEC03_3: Powerpoint presentation	21.06.19	27.06.19
PEC04: Evaluation committee	03.07.19	10.07.19
PEC04_1: Evaluation of the thesis	03.07.19	10.07.19

Figure 18: Time schedule of the project.

7.2 Overview of entities and intents used

As the schematic given in Figure 6 is reduced for an easier understanding, a more detailed overview of the intents and entities created follows.

Entities capture information necessary for the bot to work. In some cases, synonyms are used. For example, when the bot is inquiring the type of product a user is looking for, the bot will categorize the synonyms of both entities as the original ones; so "actress" will be categorized as "actor" and "movie" as "film". Be aware that some typos are also included as synonyms: that is done on purpose to enhance the understanding capabilities of the bot.

Entity	**Function**
Availability	Check if a certain film is available in the information database.
Genre	Capture the genre types the user wants for the recommended movie. *Nineteen categories:* - **action**; synonyms: karate, explosions, film with action, violent movie, action - **adventure**; synonyms: epic, tale, aventure, venture, journey, adventures - **animation**; synonyms: illustrated, graphic, cartoon, stop motion, animated, illustration - **comedy**; synonyms: comedic, comic, humour, funny film, funny movie, laugh film, comedia - **crime**; synonyms: film noir, cops, killing, police, murder, fraud, homicides, murders, felony - **documentary**; synonyms: biopic, mockumentaries, mockumentary, docu, docudrama - **drama**; synonyms: tragic, dramatic, tragedy, dramedy, melodrama, dramas - **family**; synonyms: kids, children, family - **fantasy**; synonyms: magical, fantastical, fantasia, imaginary, imagination - **history**; synonyms: true story, true facts, historic, culture, past, historical - **horror**; synonyms: gore, slasher, fear, scary, terror - **music**; synonyms: musical - **mystery**; synonyms: unsolved, puzzling, enigma, mysterious - **romance**; synonyms: romances, loved, romantic, love - **science fiction**; synonyms: dystopic, futuristic, syfy, sci fi, scifi, cyberpunk, sci-fi - **thriller**; synonyms: suspense - **tv movie** - **war**; synonyms: bellicose, confrontation, conflicts, hostilities, wartime, conflict, battle, wars - **western**; synonyms: cow boys, cowboy
MisUnd	Check the number of times a misunderstanding happens.
Product_name	Capture the name of the product the user is looking for.
Product_type	Capture which product is the user interested. *Two categories:* - **actor**; synonyms: actress, actors, actor/actress, performers, starring, starlet, protagonist, main character, girl, guy, Actor, Director - **film**; synonyms: films, movies, cinema, Movie

Table 3: Entity overview.

Intents on the other hand, interpret what is the intention of a user, what is the user goal whenever he or she is giving a concrete sentence.

Intent	Triggered when
AnotherMovieGenre	User wants another recommendation based on same genres
Ending	User wants to exit the chatbot
Information_mode	User wants to get information about a product
More_info	User wants to get further information about a product
Product_name	Used to differentiate product names from other intents
Product_type	User provides the type of the product wanted
Recommendation_mode	User wants to get a recommendation
Recommendation_mode_genre	User wants to get a recommendation based on movie genre
Recommendation_mode_other	User wants to get a recommendation not based on genre
ST_About_You	User wants to know general information about the bot
ST_Agent_Capabilities	User wants to know the capabilities of the bot
ST_General	User wants to start an informal chat
ST_Greetings	User greets the bot
ST_Human_or_Bot	User doubts about the nature of the bot
ST_Jokes	User wants to hear a joke
ST_Negative_Feedback	User shows dissatisfaction with the bot
ST_Positive_Feedback	User shows satisfaction with the bot
ST_Quote	User asks for a quote from a film
Transfer	User wants to get in touch with a human counterpart

Table 4: Intent overview.

7.3 Call to the OMDB API (Prod_Film_Info web action)

In the following subchapter, the code regarding the call to the query web action (Prod_Film_Info) within the bot's frame, the web action itself and an example of the results obtained are provided.

For information regarding the context of application, go to 3.3.4 Query nodes: Query Mode and Define Product Type.

7.3.1 JSON call to the web action within bot's framework

```
{
  "context": {
    "Product_type": "$Product_type"
  },
  "output": {
    "text": {
      "values": []
    }
  },
  "actions": [
    {
      "name": "arllambi@gmail.com_Only/default/Prod_Film_Info.json",
      "type": "web_action",
      "parameters": {
        "prodname": "<?input.text?>"
      },
      "result_variable": "context.movie_info"
    }
  ]
}
```

7.3.2 Web action code

```
function main(msg){

  const http = require('http');
  const API_KEY = '85324cac';
  const prodname = msg.prodname;
  var reqUrl = 
encodeURI(`http://www.omdbapi.com/?t=${prodname}&apikey=${API_KEY}`);

return new Promise(function(resolve, reject) {
    http.get(reqUrl, (responseFromAPI) => {
        let completeResponse = '';
        responseFromAPI.on('data', (chunk) => {
                completeResponse += chunk;
                let movie_info = JSON.parse(completeResponse);
                console.log(movie_info);
                resolve({movie_info});
        })
        responseFromAPI.on('error', (error) => {
            console.log(error);
            reject(error);
        });
    });
});

}
```

7.3.3 Web action example answer

The JSON answer from the API (via web action) when it is called with “prodname = Titanic” is as follows:

```
{
  "movie_info": {
    "Actors":  "Leonardo  DiCaprio,  Kate  Winslet,  Billy  Zane,  Kathy 
Bates",
    "Awards": "Won 11 Oscars. Another 111 wins & 77 nominations.",
    "BoxOffice": "N/A",
    "Country": "USA",
    "DVD": "10 Sep 2012",
    "Director": "James Cameron",
    "Genre": "Drama, Romance",
    "Language": "English, Swedish, Italian",
    "Metascore": "75",
    "Plot": "A seventeen-year-old aristocrat falls in love with a kind 
but poor artist aboard the luxurious, ill-fated R.M.S. Titanic.",
    "Poster":                                          "https://m.media-
amazon.com/images/M/MV5BMDdmZGU3NDQtY2E5My00ZTliLWIzOTUtMTY4ZGI1YjdiNj
k3XkEyXkFqcGdeQXVyNTA4NzY1MzY@._V1_SX300.jpg",
    "Production": "Paramount Pictures",
    "Rated": "PG-13",
    "Ratings": [
      {
        "Source": "Internet Movie Database",
        "Value": "7.8/10"
      },
      {
        "Source": "Rotten Tomatoes",
        "Value": "89%"
      },
      {
        "Source": "Metacritic",
        "Value": "75/100"
      }
    ],
    "Released": "19 Dec 1997",
    "Response": "True",
    "Runtime": "194 min",
    "Title": "Titanic",
    "Type": "movie",
    "Website": "http://www.titanicmovie.com/",
    "Writer": "James Cameron",
    "Year": "1997",
    "imdbID": "tt0120338",
    "imdbRating": "7.8",
    "imdbVotes": "951,902"
  }
}
```

7.4 Call to the TMDB API (Recommendation web action)

In the following subchapter, the code regarding the call to the recommendation web action (Recommendation) within the bot’s frame, the web action itself and an example of the results obtained are provided.

For information regarding the context of application, go to 3.3.5 *Recommendation nodes*.

7.4.1 JSON call to the web action within bot’s framework

```
{
  "context": {},
  "output": {
    "text": {
      "values": []
    }
  },
  "actions": [
    {
      "name": "arllambi@gmail.com_Only/default/Recommendation.json",
      "type": "web_action",
      "parameters": {
        "genre1": "$genre1",
        "genre2": "$genre2"
      },
      "result_variable": "context.movie_info"
    }
  ]
}
```

7.4.2 Web action code

```
function main(msg){

  const https = require('https');
  const API_KEY = '71ce1d1aba104714900df8bcd1b0a610';
  const genre1 = msg.genre1;
  const genre2 = msg.genre2;
  var reqUrl =
encodeURI(`https://api.themoviedb.org/3/discover/movie?api_key=${API_K
EY}&page=1&with_original_language=en&with_genres=`);

//We convert the two genres to the corresponding ids

switch (genre1){
    case "action":
    genre1_id = 28;
    break;

    case "adventure":
    genre1_id = 12;
    break;

    case "animation":
    genre1_id = 16;
    break;

    case "comedy":
    genre1_id = 35;
    break;

    case "crime":
    genre1_id = 80;
    break;

    case "documentary":
    genre1_id = 99;
    break;

    case "drama":
    genre1_id = 18;
    break;

    case "family":
    genre1_id = 10751;
    break;

    case "fantasy":
    genre1_id = 14;
    break;

    case "history":
    genre1_id = 36;
    break;

    case "horror":
    genre1_id = 27;
    break;

    case "music":
    genre1_id = 10402;
    break;

    case "mystery":
    genre1_id = 9648;
    break;

    case "romance":
    genre1_id = 10749;
    break;

    case "science fiction":
    genre1_id = 878;
    break;

    case "tv movie":
    genre1_id = 10770;
    break;

    case "thriller":
    genre1_id = 53;
    break;

    case "war":
    genre1_id = 10752;
    break;

    case "western":
    genre1_id = 37;
    break;

    default:
    genre1_id = 'no_genre';
    }

switch (genre2){
    case "action":
    genre2_id = 28;
    break;

    case "adventure":
    genre2_id = 12;
    break;

    case "animation":
    genre2_id = 16;
    break;

    case "comedy":
    genre2_id = 35;
    break;

    case "crime":
    genre2_id = 80;
    break;

    case "documentary":
    genre2_id = 99;
    break;

    case "drama":
    genre2_id = 18;
    break;

    case "family":
    genre2_id = 10751;
    break;

    case "fantasy":
    genre2_id = 14;
    break;

    case "history":
    genre2_id = 36;
    break;

    case "horror":
    genre2_id = 27;
    break;

    case "music":
    genre2_id = 10402;
    break;

    case "mystery":
    genre2_id = 9648;
    break;

    case "romance":
    genre2_id = 10749;
    break;

    case "science fiction":
    genre2_id = 878;
    break;

    case "tv movie":
    genre2_id = 10770;
    break;

    case "thriller":
    genre2_id = 53;
    break;

    case "war":
    genre2_id = 10752;
    break;

    case "western":
    genre2_id = 37;
    break;

    default:
    genre2_id = 'no_genre';
    }

var reqUrl = reqUrl+genre1_id+`,`+genre2_id;
let i =Math.floor(Math.random() * 19); //we take a random result from
the first 20 (0-19) most popular movies of the genre
return new Promise(function(resolve, reject) {
    https.get(reqUrl, (responseFromAPI) => {
       const chunks = [];
        responseFromAPI
          .on('data', (chunk) => {
                chunks.push(chunk);
          })
         .on('end', _=> {
               let movie_info = JSON.parse(Buffer.concat(chunks));
               movie_info = movie_info.results[i];
               console.log(movie_info);
               resolve({movie_info});
        })
        responseFromAPI.on('error', (error) => {
            console.log(error);
            reject(error);
        });
    });
});

}
```

7.4.3 Web action example answer

The JSON answer from the API (via web action) when it is called with genre1 = “comedy” (id 35) and genre2 = “thriller” (id 53) is as follows:

```
"movie_info": {
    "adult": false,
    "backdrop_path": "/3KdQh3EsjXBSojVydtBQEh5TdAG.jpg",
    "genre_ids": [
      27,
      9648,
      53,
      878,
      35
    ],
    "id": 512196,
    "original_language": "en",
    "original_title": "Happy Death Day 2U",
    "overview": "Collegian Tree Gelbman wakes up in horror to learn
that she's stuck in a parallel universe. Her boyfriend Carter is now
with someone else, and her friends and fellow students seem to be
completely different versions of themselves. When Tree discovers that
Carter's roommate has been altering time, she finds herself once again
the target of a masked killer. When the psychopath starts to go after
her inner circle, Tree soon realizes that she must die over and over
again to save everyone.",
    "popularity": 36.836,
    "poster_path": "/4tdnePOkOOzwuGPEOAHp8UA4vqx.jpg",
    "release_date": "2019-02-13",
    "title": "Happy Death Day 2U",
    "video": false,
    "vote_average": 6.1,
    "vote_count": 917
  }
}
```

7.5 Access data to the frameworks used

7.5.1 For the IBM Watson framework

An overview of the skill can be found here: link.

An overview of the web actions can be found here: link.

The access data for the IBM Watson framework is:

- IBMid: UOCMovieBot@gmail.com
- Password: UOC_MovieBot_2019

7.5.2 For the DialogFlow framework

An overview to the bot can be found here: link.

The access data for the DialogFlow framework is:

- Account: UOCMovieBotDF@gmail.com
- Password: UOC_MovieBot_2019

7.6 DialogFlow vs IBM Watson

For the first attempt of building up a bot, Google's DialogFlow framework was used. This framework is constituted by very similar elements than IBM Watson (meaning also entities, intents, and a similar approach to the dialogue structure, see 3.3 Architecture description). In this case, the connection is done via a webhook created in Visual Studio and deployed via Node.js and Heroku (see [S7] for more info). It is worth mentioning that the solution provided in [S7] is for the version V1 of DialogFlow; the author of this document had to reformulate the code of the webhook to V2 standards and adapt the bot architecture for such purpose.

Due to the difficulties regarding the construction of a complex language flow, bot training and connection to external APIs and functionalities, the IBM Watson framework was used instead for the creation of the final version of the MovieBot. A copy of the DialogFlow bot is included in the github repository for information.

During the decision process, a short analysis between engines was done. Even when the comparison between frameworks was not an objective of the project, the information is provided for qualitative purposes and outside the main body of the document. For the analysis, one bot from DialogFlow and one from IBM Watson were trained with the same following data:

- A set of user examples:
  - Fetch me data on Antonio Banderas
  - Fetch me data on Leonardo di Caprio
  - Fetch me data on Nicole Kidman
  - Fetch me data on Olivia de Havilland
  - Fetch me data on Peter O'Toole
- A set of product names:
  - Nicole Kidman
  - Peter O'Toole
  - J. J. Abrams
  - Olivia de Havilland
  - Joseph Gordon-Levitt
  - Jennifer Jason Leigh
  - Tom Cruise
  - Tom Selleck
  - Daniel Day-Lewis

The set of examples puts together a fixed sentence structure with a set of product names. Idea is to see the capability to capture and identify new product name entities outside the training data; in other words, if a user entered a sentence like “Fetch me data on Santiago Segura” to correctly identify “Santiago Segura” as product name.

The table with the results is included above, green meaning input name was correctly captured, red not captured, and yellow only partially (in this case, the product name understood by the bot is given in italics). The reader must take into account that different structures of names (name and surname, name and two surnames, composed names…) are used to give a widener range of learning data.

Input name	DialogFlow	IBM Watson
James Bond	Correctly captured	Not captured
Antonio Banderas	Not captured	Not captured
Ana de Armas	*On ana de armas*	Not captured
Leonardo Sbaraglia	Correctly captured	Not captured
Santiago Segura	*data on santiago segura*	Not captured
j. j. abrahms	*j*	Correctly captured
nicle kidman	Correctly captured	Correctly captured
john c. reily	*John*	Not captured
josehp gordon	Correctly captured	Correctly captured
josph gordon-levtt	*Josph*	Correctly captured
Joseph Fiennes	Correctly captured	Not captured
Jamie lee curtis	Correctly captured	Not captured
francis ford coppola	Correctly captured	Not captured

Table 5: Classification of different user input by DialogFlow and IBM Watson.

Not being the number of trials enough to consider the test statistically valuable, it is at first sight quite clear that DialogFlow outperforms in this feature IBM Watson, the latter being only capable of detecting as names similar than the ones given as example (“nicle kidman” vs “nicole kidman”), while DialogFlow is capable of using entities to extract knowledge as well.