

MedBot: Chatting for Healthcare Services

Pedro Manuel Simões de Sousa Ferreira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. André Ferreira Ferrão Couto e Vasconcelos

Examination Committee

Chairperson: Prof. Francisco João Duarte Cordeiro Correia dos Santos

Supervisor: Prof. André Ferreira Ferrão Couto e Vasconcelos

Members of the Committee: Prof. António Manuel Ferreira Rito da Silva

October 2019

Abstract

The heavy usage of chat platforms by users, allied with developments in natural language understanding, offer a favourable scenario for organizations to implement use cases in chatbots. This research identifies factors that influence the suitability of use cases for conversational user interfaces, enabling organizations to make more informed decisions regarding chatbots implementations. The factors identified are grouped in three categories: (i) general factors; (ii) factors to be considered to implement a chatbot over a human operator; and (iii) factors that should be considered when implementing a chatbot over a traditional GUI application. A use case selection process is created that uses the factors gathered and enables to prioritize the more appropriate use cases between a set of potential use cases for chatbot implementation. The selection process is applied to several use cases in the health care domain, yielding the use case of scheduling a medical appointment with highest priority. A reference architecture of a chatbot for scheduling an appointment is defined and implemented. User tests are conducted, comparing the interactions with the chatbot with a traditional graphical user interface (website). User tests indicate the chatbot as more efficient than the website, and most testers indicate it as a preferable method for scheduling an appointment, when comparing to the website.

Resumo

A intensa utilização de plataformas de chat pelos utilizadores, conjugada com os desenvolvimentos em processamento de língua natural, criam um cenário favorável às organizações para a implementação de *chatbots*. Esta tese identifica as características dos casos de uso que influenciam a sua adequabilidade para *chatbots*, permitindo às organizações tomarem decisões mais informadas no que respeita a implementações em *chatbots*. Os fatores identificados estão divididos em três grupos: (i) fatores gerais; (ii) fatores a considerar para comparação com operadores humanos; e (iii) fatores a considerar para comparação com aplicações gráficas tradicionais (GUI). É definido um processo de seleção de casos de uso que, usando os fatores identificados, permite avaliar casos de uso e ordenar um conjunto de casos de uso para priorização de implementação em *chatbot*. Este processo é aplicado a um conjunto de casos de uso no domínio da saúde, o que resulta na identificação “marcação de consulta médica” como o mais adequado a implementar dentro do conjunto de casos de uso avaliados. É definida e implementada uma arquitetura de referência para um *bot* que implementa este caso de uso. São realizados testes com utilizadores, onde é comparada a interação com o *chatbot* e com uma aplicação gráfica tradicional (website) para marcação de consultas. Os resultados dos testes indicam o *chatbot* como mais eficiente do que o website, e a maioria dos utilizadores indica o *chatbot* como método preferencial para marcação de consultas, quando comparado com o *website*.

Acknowledgments

I would like to thank my family for their support and encouragement throughout all my academic path.

I would also like to thank my thesis advisor, professor André Vasconcelos, for the guidance and feedback offered during the elaboration of this dissertation. I also want to thank Medclick and Rui Cruz, for the support and for the opportunity to develop this work in the context of Medclick.

Last, but not least, I would like to express my gratitude to my friends, that through their presence and support were also of great importance.

Table of Contents

Abstract.....	ii
Resumo	iii
Acknowledgments.....	iv
Table of Contents	v
List of Figures.....	vii
List of Tables	ix
List of Acronyms	x
1. Introduction.....	12
1.1 Research Problem	12
1.2 Thesis Outline.....	13
2. Related work	14
2.1 Dialog Systems.....	14
2.2 Uses of Chatbots	15
2.3 Chatbots In Healthcare	18
2.4 Chatbots, Humans and GUI Applications.....	20
2.5 Natural Language Processing	21
2.6 Natural Language Understanding in Dialog Agents	23
2.7 Existing Chatbot Services.....	24
2.7.1 Dialogflow	25
2.7.2 Microsoft Bot Framework	26
2.7.3 Watson Assistant	27
2.7.4 Performance Comparison	28
2.7.5 Comparison Summary	29
3. Chatbot Use Case Evaluator.....	31
3.1 Use Case Evaluation Factors	31
3.2 Use Case Selection Process	33

3.3	Use Case Evaluator Application.....	36
3.3.1	Scheduling an Appointment	36
3.3.2	Paying for an Appointment.....	39
3.3.3	Performing a Medical Diagnosis	40
3.3.4	Use Cases Selection.....	42
4.	Chatbot Implementation.....	44
4.1	Components and Integrations Overview.....	44
4.2	Dialog Management.....	48
4.2.1	State Machine: first iteration.....	48
4.2.2	State Machine: final version	51
4.3	Natural Language Understanding	54
4.3.1	Entity Recognition and Mapping	57
4.4	Active Learning.....	66
4.5	Login.....	69
4.6	Appointment Reminder.....	72
5.	Evaluation.....	75
5.1	Evaluation Process.....	75
5.1.1	Website and Chatbot Scheduling.....	75
5.1.2	Questionnaire.....	76
5.1.3	Evaluation Process Summary	77
5.2	Evaluation Results	79
5.2.1	Chatbot Results	80
5.2.2	Website (traditional GUI) and Chatbot Results Comparison	80
5.2.3	Questionnaire Results.....	81
5.3	Evaluation Results Conclusions.....	83
6.	Conclusion	84
6.1	Contributions	84
6.2	Future work.....	85
7.	References	86

List of Figures

Figure 1 - Florence bot medication reminder – in [9]	19
Figure 2 - Watson Assistant Overall Architecture – source [27]	27
Figure 3 - F-scores for the different NLU services, grouped by corpus – source [29].....	28
Figure 4 - Use Case Evaluator Factors	33
Figure 5 - Use Case Selection Process: overview.....	34
Figure 6 - Assess Evaluation Factors subprocess	35
Figure 7 - Scheduling appointment business process	36
Figure 8 - Gather Appointment Information subprocess	37
Figure 9 - Chatbot Architecture.....	45
Figure 10 - Medclick domain objects	46
Figure 11 - Schedule dialog first state machine	49
Figure 12 - Chatbot first iteration in skype	50
Figure 13 - Schedule dialog final state machine	52
Figure 14 - Chatbot final version on Facebook Messenger	53
Figure 15 - Intent and entities modelling.....	54
Figure 16 - Training LUIS app with labelled utterances	55
Figure 17 - LUIS response to query "marcar consulta com doutor bernardo amanhã"	57
Figure 18 - Mapping user utterances to domain objects.....	58
Figure 19 - Mapping department list entity to business entity	60
Figure 20 - Algorithm used to calculate distance of the doctor mentioned name to a database doctor name	63
Figure 21 - LUIS resolution for "amanhã à tarde" datetimeV2 entity detected.....	64
Figure 22 - Department Entity NLU recognition failure	67
Figure 23 - Reviewing LUIS NLU predictions	68
Figure 24 - Login with social token and Medclick token flux	70
Figure 25 - Appointment reminder sent to user	72

Figure 26 - Medclick backend extensions for notifications.....	73
Figure 27 - Appointment notification communication sequence.....	74
Figure 28 – Chatbot user experience questions asked in the questionnaire	76
Figure 29 - Website interface	77
Figure 30 - Chatbot interface on Facebook Messenger	78
Figure 31 - Evaluation with users process.....	78
Figure 32 - Age distribution of test users	79
Figure 33 - Sex distribution of test users	79
Figure 34 - Comparison of faster method per user test	81
Figure 35 - User preferred scheduling method results	82

List of Tables

Table 1 - Bot examples for each use case.....	17
Table 2 - Comparison summary of studied platforms	29
Table 3 - General factors assessment for scheduling appointment UC.	37
Table 4 - Factors over GUI application assessment for scheduling appointment UC.	38
Table 5 - Factors over Humans assessment for scheduling appointment UC.	38
Table 6 - General factors assessment for paying appointment UC.....	39
Table 7 - Factors over GUI application assessment for paying appointment UC.....	39
Table 8 - Factors over Humans assessment for paying appointment UC	40
Table 9 - Use cases individual assessment summary (step 1)	42
Table 10 - Use cases priority rank and conclusions.....	43
Table 11 - LUIS entities types	56
Table 12 - F-measure scores for different Personal Name Matching Techniques.....	62
Table 13 - Entity Recognition and Mapping Summary	65
Table 14 - Chatbot user testing results	80
Table 15 - Task completion and efficiency comparison (chatbot and website).....	81
Table 16 - Short UEQ results.....	82

List of Acronyms

API	Application Programming Interface
GUI	Graphical User Interface
NLP	Natural Language Processing
NLU	Natural Language Understanding
UC	Use Case
UI	User Interface

1. Introduction

The developments in the artificial intelligence field have been responsible to the rise of new, more intelligent systems. Specifically, the developments in the Natural Language Processing, drive the development of chatbots. Chatbots are systems that interact with the user using natural language, as if the user were talking with another human. Today people are using chat platforms as one of the main channel of communication, using applications such as Facebook Messenger or WhatsApp. The heavy usage of chat platforms allied with the developments in natural language process (NLP) create a favourable scenario to organizations to offer their services using conversational user interfaces. Services can be accessed directly from the chat platforms the users already use, in a more natural way, instead of requiring the users to install a specific app or access the organization website.

Health Care is one field that can benefit from the implementation of chatbots, both in customer service, enabling, for instance, users to schedule appointments or even in the clinical field, by assisting the decision process of healthcare professionals.

Although chatbots can offer the beforementioned advantages, it is important to assess if a use case is suitable for a conversational interface. Moreover, organizations should be able to select the more appropriate use cases between several use case candidates, in order to understand the more suitable ones and prioritize them.

This work is realized in the context of Medclick¹. Medclick is a company that will provide a one-stop platform to book a medical appointment in a fast and user-friendly way, across multiple medical service providers.

1.1 Research Problem

This research aims to understand what use cases are adequate to expose via a conversational user interface. This raises the following questions:

- How can suitable use cases be identified in a systematic way?
- What use case factors influence its suitability to be implemented in a dialog agent?
- How can organizations prioritize use cases for a chatbot implementation?

It is therefore a goal of this research to find factors that impact the suitability of a use case for a chatbot, and use such factors to create an *use case evaluation process*, that can be used by any organization to evaluate the suitability of its use cases for a conversational user interfaces, and prioritize the more suitable ones.

¹ <https://www.medclick.pt/>

Furthermore, this research aims to create a reference architecture of a chatbot in the healthcare domain. The *use case evaluation process* is used as the starting point to answer the following question:

- What healthcare use cases are more appropriate to be exposed via a chatbot?

After answering this question, it is possible to select a suitable healthcare use case and create a reference chatbot architecture.

1.2 Thesis Outline

The thesis is organized in five sections. In the next section, **section 2**, it is performed a review of the related work and background information useful to address the research problem. In **section 3**, factors that should be considered when evaluating the suitability of a use case for a chatbot are defined, and a *use case selection process* that uses the factors defined is proposed. This process is applied to use cases in the healthcare domain and yields the use case of *scheduling a medical appointment* as the more appropriate for chatbot. Later, in **section 4**, it is explored the architecture and implementation of the bot that supports this use case. In **section 5**, the evaluation of the chatbot created and of the *use case selection process* is discussed. Finally, in **section 6**, the conclusions of this research are presented.

2. Related work

In this section, the background and related work useful to address the research problem are reviewed. It is defined what are dialog systems and its subcategorizations. The chatbots general uses and specific to the healthcare domain are explored. It is also explored the differences between the interaction of users with bots, humans and traditional GUI applications. An overview of natural language processing is performed. Finally, it is analysed the existing chatbot services that facilitate the development of chatbots nowadays.

2.1 Dialog Systems

Conversational agents or *dialog systems* are programs that communicate with users in natural language. This kind of systems can be classified in two categories [1]:

- **Task-oriented dialog agents:** are designed for a particular task and set up to have short conversations to get information from the user to help complete the task. These include the digital assistants that can give travel directions, control home appliances, find restaurants, or help make phone calls or send texts.
- **Chatbots:** Chatbots are systems designed for extended conversations, set up to mimic the unstructured conversational characteristic of human-human interaction, rather than focused on a particular task. These systems often have an entertainment value. Chatbots are also often attempts to pass the Turing test [2]. Chatbots can also have some practical uses such as testing theories of psychological counselling.

The word “chatbot” is often used in the media and in industry as a synonym for conversational agent [1]. In this document the term “chatbot” is used in that same more general sense. In reality, the kind of systems explored in this thesis are typically task-oriented dialog agents, even though we may refer them using the word “chatbot” instead of “task-oriented dialog agent”.

It is also important to notice that even though dialog systems communicate with users in natural language, other form of GUI elements are often used such as pre-defined quick replies that the user can click in order to make the interaction faster and easier.

2.2 Uses of Chatbots

Today there are several tasks that can be performed by chatbots. This set of tasks make possible a panoply of use cases that can be supported by bot interaction. The main tasks performed by a chatbot are: send alerts; take action; pull information and answer questions [3].

It is possible to identify some categories of uses cases that were already being implemented by some companies taking advantage of the previously mentioned set of tasks, [4] identifies the following major use cases.

Conversational Commerce. Conversational bots offer the ability to order and explore services or goods directly by a conversational interface. This has the advantage of removing the need to install the specific app of the business, calling services directly from the preferred conversational channel.

Bots for business. Bots are not only being used for external customers but also internally by businesses, in order to support their internal business processes. This can improve productivity by facilitating short, contextual and actionable tasks.

Notification Bots. Notification through chat platform are being used as an alternative to more traditional notifications such as email. One of the advantages of this type of use is that traditional notifications usually redirect the user to another platform, while chatbots, taking advantage of its “take action” capability, can perform some related action directly from the conversational UI. For instance, a client can receive some appointment reminder, and can confirm/cancel it directly from the chat platform.

Bots as Routers between humans. Bots can also help connecting humans to humans. By using chatbots we can by identifying the intent of the user and connect him or her to the most suitable human. This can act as a replacement of the traditional IVR systems, providing a more natural experience until a human takes the place.

Customer Service and FAQ bots. Bots helping answering questions to clients are a very common use case. The fact that most questions are usually asked several times, and the answer is standard, make bots a great way to replace humans in this repetitive task, reducing costs and usually being faster.

Productivity and Coaching. There are also various examples of bots focused on reminders, to-do lists, personal and team task management completion, that help people be more productive. Bots can also

be used for personal coaching to assist in areas such as weight loss and finance. By providing a more personal experience, bot users are more willing to provide information to the bot than to fill forms in an app.

Third-Party integration Bots. Third party integrations make possible to bring external apps used in someone workflow to some chat app like slack. This avoids that users have to context-switch between apps to gather the information needed to their workflow.

Games and Entertainment Bots. Bots are also being used in the entertainment area by using a conversation as a fun activity. One of the advantages of using bots in this area, is that bots can reengage with the users and encourage them back to the service in a less intrusive and more customizable and friendly way than app notifications, for example.

Brand Bots. Business are using the chat channels to create brand awareness and engagement. One of the major incentives to use bots in this area is app fatigue, creating this way a new way to engage with users.

Examples of bots available in each category of use case can be seen in Table 1.

Table 1 - Bot examples for each use case

Use Case	Bot Example	Bot Description
Conversational Commerce	Kip ²	AI assistant for team shopping that saves time coordinating office purchases.
Bots for Business	LawGeex ³	LawGeex automates the contract review process, so that lawyers are free for other tasks.
Notification Bots	CNN	Instant breaking news alerts and the most talked about stories.
Bots as Routers between humans	Sensay ⁴	Sensay is a bot that connects you to real people to chat
Customer Service and FAQ bots	Chatbotler ⁵	Automates responses to commonly asked questions
Productivity and Coaching.	Lark ⁶	Helps users monitor their food habits, helping with weight loss
Third-Party integration Bots	Statsbot ⁷	Connects to your database, generates data relationships from it automatically, and allows to get first insights
Games and Entertainment Bots	Swelly ⁸	It is a polling chatbot that enables sharing your opinion with a worldwide community and get feedback on your own questions within seconds.
Brand Bots	Whole Food Market ⁹	Lets users search for recipes, as well as find stores and contact the company over Facebook Messenger

² <https://botlist.co/bots/kip-2>

³ <https://www.lawgeex.com/>

⁴ <https://www.kik.com/bots/sensay/>

⁵ <https://www.chatbotler.com>

⁶ <https://www.lark.com>

⁷ <https://statsbot.co/statsbot>

⁸ <https://www.swelly.ai/>

⁹ <https://www.chatbotguide.org/whole-foods-bot>

2.3 Chatbots In Healthcare

In this section we focus on use cases specific to the healthcare domain.

Chatbots in healthcare have impact on multiple actors of the healthcare domain. They are present in hospitals and clinics context to be used both by professionals and patients.

Clinical Context: Used by Health Care Professionals. Conversational agents can be used to support the medical professional's decision-making process, by helping them retrieve information in a fast and practical way. One example is the "SafedrugBot", a messaging app that helps doctors who need appropriate information about the use of drugs during breastfeeding, offering readily accessible drug information [5]. There are also efforts made to support medical students' education, [6] created a bot that can answer medical students queries by using the Unified Medical Language System (UMLS) as domain knowledge and converting the student natural language to SQL queries. The SQL queries are then run against the knowledge base and results returned to the user in natural dialog. Based on a survey to medical students, it was concluded that the most required type of question was of the form "what is <concept>?". The system supports the following type of queries: what is; what is the type of; what are the causes of; what are the symptoms of.

Clinical Context: Used by Patients. Bots can also be used by patients, [7] introduces a chatbot system that provides conversational service for mental health care based on emotion recognition methods. This service conducts psychiatric counselling to its users. To understand the dialogues and recognize user's emotion, the service applies various emotional intelligence techniques such as multi-modal emotion recognition from conversation content, intonation, and facial expression, enabling continuous observation of user's emotional changes.

Furthermore, chatbots can be used to help triage patients. England NHS experimented triage NHS 111 inquiries with "Babylon", a chatbot drive by clinically-based algorithms that can triage patients based on the symptoms reported, without the intervention of humans. Based on the symptoms and its own algorithms, the app could refer the patient to hospital or recommend a GP appointment [8].

Customer Service. In the medical provider's customer service context, they can be used to help patients find doctors and scheduling their appointments. Bots can also be used to notify patients of incoming appointments.

Used by Regular Users. There are also solutions on the consumer side that facilitate the day-to-day of users regarding to healthcare. Chatbots can be used to remind patients to take their medication, search for diseases information and find close pharmacies and doctors' offices. One example of a bot that provides all the referred functionalities is "Florence" (see Figure 1).

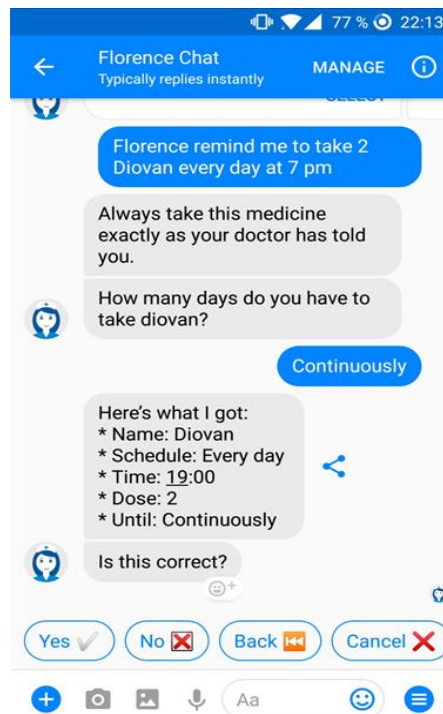


Figure 1 - Florence bot medication reminder – in [9]

2.4 Chatbots, Humans and GUI Applications

This section explores what are the chatbots benefits, when replacing human interaction and traditional applications (GUI). It is also explored the intrinsic differences of the interactions.

Human-Human vs Human-Chatbot Interaction. There are some differences in the way that people interact with a bot compared to a human. A study [10] concluded that people communicate with the chatbot for longer durations, using shorter messages, than they did with another human. Additionally, human–chatbot communication used simpler vocabulary than what is found in conversations among people and exhibited greater profanity. Factors such as number of words per conversation, shorthand terms, and emoticons were found to have no statistically significant differences.

The usage of chatbots in some scenarios bring advantages over humans, namely [11]:

- *Consistency:* chatbots can be consistent in services, which is important in certain sectors and may be hard to achieve with human operators.
- *Scalability:* chatbots can easily scale up to handle periods of unregular increased traffic, which is much harder with human operators.

With good design and implementation, Accenture [12] reports more than 80% of chat sessions resolved by a chatbot, that would otherwise been a human in a chat session or call.

Human-Traditional GUI vs Human-Chatbot Interaction. A report [13] by Forrest identifies the following factors that foster chatbots adoptability over traditional applications:

- *Chatbots promise a more convenient and natural user interface:* Typically, users must go to the process of discover, download, and install apps. Then, apps provide touch graphical interfaces to help consumers perform tasks. The experience isn't natural, but it is effective. Conversations offer are more natural experience.
- *Mobile moment ownership is plateauing for enterprises:* Mobile is the first screen for consumers; however, consumers use only 25 to 30 apps on average each month and spend 88% of their time in just five downloaded apps.
- *Heavy use of instant messaging platforms:* Consumers spend 78% of their time on smartphones within apps. The median usage of instant messaging apps is 21.47 minutes per day among users of those apps and the pace of adoption is accelerating.

The fact that we are living an app fatigue moment, allied with the heavy usage of messaging apps, present an opportunity to replace traditional applications with chatbots available on the messaging applications that users are already using.

2.5 Natural Language Processing

Natural language processing (NLP) is a subfield of computer science concerned with using computational techniques to learn, understand, and produce human language content [14]. Some applications of NLP include: *information extraction*, transforming unstructured data found in texts into structured data [1]; *conversational agents*, that aid human-machine communication [14]; or *machine translation*, the use of computers to automate the process of translating from one language to another, aiding human-human communication [1] [14].

The factors that have allowed the development of NLP in the last years twenty years, according to [14], are: (i) increase in computing power, (ii) the availability of large amounts of linguistic data, (iii) the development of successful machine learning methods, and (iv) a richer understanding of the structure of human language and its deployment in social context.

The types of linguistic knowledge in NLP can be divided in the following categories [15] [1]:

1. Phonetic and Phonological Knowledge;
2. Morphological Knowledge;
3. Syntactic Knowledge;
4. Semantic Knowledge;
5. Pragmatic Knowledge;
6. World Knowledge;
7. Discourse Knowledge.

An overview of each type of knowledge and its analysis in NLP is presented next.

Phonetic and Phonological Knowledge. Phonetic and phonological knowledge concerns how words are realized as sounds [15].

Morphological and Lexical Analysis. Morphology is the study of the way words are built up from smaller meaning-bearing units, morphemes [1]. Morphological parsing is the process of finding the constituent morphemes in a word. For instance, the morphological parsing of the word “cats” should reflect this word is the *plural* of the *noun* stem “*cat*” (*CAT* +*N* +*PL* for “cats”) [1].

This phase starts by the process of tokenization, the task of separating words from running text. Part-of-speech tagging (POS) is a goal of this phase. POS is the process of automatically assigning a morpho-syntactic tag (such as noun or verb) to each word. To perform POS tagging, the issue that the same word can have multiple tags (ambiguity) must be resolved. Methods for POS tagging include [1]: *Rule-based taggers*, that use hand-written rules to disambiguate between tags; *HMM taggers*, that maximize the product of word likelihood and tag sequence probability, trained with hand labelled data. The Viterbi algorithm is an example of an algorithm used that given an HMM and an input string returns the optimal tag sequence. Other machine learning models used include: maximum entropy and other log-linear models; decision trees; memory-based learning; and transformation-based learning [1].

Syntactic Analysis. Syntactic analysis concerns how words can be putted together to form correct sentences [15]. It defines how words relate to one another [15]. Syntactic parsing is the task of recognizing a sentence and assign a syntactic structure to it [1]. The syntactic structure of a sentence is typically represented in the form of a tree. In order to perform syntactic parsing, it is needed a grammar and a parsing algorithm. The grammar specifies the allowed structures of the language and the parsing algorithm defines how to generate the structure of a sentence given the grammar. Parsing algorithms can be divided in two groups: bottom-up and top-down. Top-down parsers try to build a parse tree from the root node down to the leaves. On the other hand, bottom-down parsers start with the words of the input and tries to build the tree from the words up.

Syntactic parsing must also deal with ambiguity, given that for the same sentence it can be possible to have multiple parsing trees that agree with the grammar. An approach to solve this issue is the use of probabilistic context-free grammars (PCFG). In PCFGs a probability is added to each rule, and the probability of a sentence can be obtained by multiplying the probability of each rule used in the parsing of the sentence [1]. The parsing algorithms can also be adapted to use the probability information.

Semantic Analysis. Semantic analysis concerns the meaning of individual words and how the words combined create the meaning of a sentence [15]. Several approaches are possible to represent this meaning in NLP. [1] identifies the following meaning representation languages: (i) First Order Logic; (ii) Semantic Networks; (iii) Conceptual dependency; and (iv) Frame-based.

Ambiguity is also present at the semantic level. The methods to solve this ambiguity can be divided in two major groups: (i) machine learning methods, that use features that are predictive of word senses (such as bag of words or collocational features), and (ii) methods that use lexical resources, such as dictionaries or syntax information.

Pragmatic Knowledge. Pragmatic knowledge concerns how sentences are used in different contexts and how it affects the interpretation of the sentence [15].

World Knowledge. World knowledge includes general knowledge about the world.

World knowledge is used in: (i) disambiguating the meaning of a sentence; (ii) connecting parts of discourse (bridging); (iii) finding how two segments of discourse are logically connected to one another (discourse relations); and (iv) resolving rhetorical figures such as metaphors and metonyms [16].

Discourse Knowledge. Discourse knowledge widens the analysis to more than single independent sentences. It takes in account the general discourse when interpreting a sentence, using information from preceding sentences. It is used in solving co-references.

As seen above, ambiguity is present on various levels of analysis in NLP. It presents one of the main challenges in NLP.

2.6 Natural Language Understanding in Dialog Agents

There are various possible structures to represent the meaning of linguistic expressions. Modern task-based dialog systems are based on a domain ontology, a knowledge structure representing the kinds of intentions the system can extract from user sentences [17]. The ontology defines a frame-based representation, with one or more frames, each a collection of slots, and defines the values that each slot can take.

Dialog agents typically have a natural language understanding module. NLU is responsible for the semantic parsing of user utterance, i.e., it gives semantic meaning to user utterances. This module is responsible for selecting the appropriate frames and filling the slots of the before mentioned domain ontology structure. This module objective is therefore to extract three things from the user's utterance [17]:

- **domain classification:** if the systems is not single-domain, there is the need to determine what domain is the user referring to.
- **intent determination:** what general task or goal is the user trying to accomplish. For example, the task could be to Find a Movie, or Show a Flight, or Remove a Calendar Appointment.
- **slot filling:** extract the particular slots and fillers that the user intends the system to understand from their utterance with respect to their intent.

Consider the sentence *"Book me a table for two for Friday night at Sushi Place"*. The NLU module would recognize the domain as "restaurant"; the intent as "book table" and would fill the time slots with "night" and "Friday"; the restaurant name slot as "Sushi Place"; and finally the slot for the number of seats as "two".

The domain and intent determination are usually treated as a semantic utterance classification (SUC) problem and the slot filling as a sequence labelling problem [18].

Possible methods used by for domain/intent recognition and slot filling include: (i) hand-written rules; (ii) semantic grammars, that are context-free semantic grammar in which the left-hand side of each rule corresponds to the slot names; and (iii) supervised machine learning, using a training set that associates each sentence with the semantics, we can train a classifier to map from sentences to intents and domains, and for slot filling a sequence model can be used [17].

2.7 Existing Chatbot Services

As seen in section 2.6, it is possible to create NLU from scratch by either implementing rules or applying machine learning algorithms. Training machine learning models requires having access to rare expertise, large datasets, and complex tools, which presents a barrier to smaller companies [19]. The availability of NLP services in the cloud has powered the widespread use of chatbots. From the rise of open source tools to the arrival of cloud APIs, NLU capabilities once limited to the academic and research community are now accessible to a wider audience across industries [19].

Today there are various solutions for businesses looking to build a chatbot without having to deal with the hard task of creating a NLU system from scratch. The major tech companies are all taking efforts in developing cloud platforms that can be used by other companies to create their own chatbots.

Nowadays, there is a diversified offer of services with various degrees of complexity, from services that require no coding to programming frameworks. In this section we will analyse in more detail solutions offered by some of the world leading technologic companies, namely: Google, IBM and Microsoft. For each of the solutions we will focus on following aspects:

- **Concepts present:** define what are the concepts that are used and configured in order to create the conversational agent.
- **Channels Integration:** explore what are the channels where the agent can be integrated. Some platforms offer integrations to existing conversational interfaces, making the process of creating a UI for the agent much simpler, because the fronted for displaying text and other elements, such as lists of buttons, is already available.
- **Integration with external services:** How does the solution allows the integration with external services is especially important to be considered when businesses are adapting existing business processes to a conversational UI, because typically there will be the need to communicate with the company's backend to either retrieve information from it or trigger actions. Today many companies develop their systems following some sort of SOA architecture, it is then important to realize how can the bot platform communicate with it and integrate it with the flux of the conversation.

Information about particular methods used by these NLU services for intent determination and slot filling could not be found publicly.

Common Concepts. There are some concepts that are present in most chatbot platforms, namely **Intents** and **Entities**. These two concepts are fundamental pieces to understand the intention of the user (Intents) and respond accordingly, and after that retrieve important pieces (Entities) of information from the user input. These two concepts correspond to the concepts explored in section 2.6.

By IBM's definition for Watson Assistant "**intents** are purposes or goals expressed in a customer's input, such as answering a question or processing a bill payment. By recognizing the intent expressed in a customer's input, the Watson Assistant service can choose the correct dialog flow for responding to it." [20], while **entities** "represent a class of object or a data type that is relevant to a user's purpose. By recognizing the entities that are mentioned in the user's input, the Watson Assistant service can choose the specific actions to take to fulfil an intent." [21]

The definitions above are transversal to all platforms described in this section. In each platform section we will focus only on the unique concepts but is important to reinforce that intents and entities are not only present as they are the fundamental concepts of each one of the solutions presented next.

2.7.1 Dialogflow

Dialogflow is the solution provided by Google for developing conversational agents running on the cloud. Dialogflow offers the possibility to create the conversational agent entirely using its website user interface. Additionally, it's also provided an SDK that allows to send some user message to Dialogflow and receive Dialogflow interpretation (such as detected intent and entities.). This SDK makes also possible to make changes in the agent itself, by changing its parameters, such as adding new intents.

The concepts present in the Dialogflow service are:

Contexts: Contexts let you control conversation flows by letting you define specific states that a conversation must be in for an intent to match [22]. There are two types of contexts in Dialogflow: the input context and output context. These contexts can be associated with intents. We can associate an input context to an intent so that the intent is matched only if the associated context is activated. Associating an output context to an intent, we can activate the context when the intent is matched.

Follow-up intents: Follow-up intents are provided in order to facilitate the flux of the conversation without having to use contexts manually [23]. It is possible using follow-up intents to shape the next interaction based on the information given in the previous one. It is useful to create intents that are only matched if the user gave a specific answer to the parent intent.

Events: Events are another way to activate intents but not by interpreting something that the user writes, but rather when something happens [24]. These events can be triggered from some of the platforms that Dialogflow integrates with, for instance when a user starts a conversation with the Dialogflow agent via Facebook platform, there is an event triggered to Dialogflow.

Integration with external Systems. In order to integrate the bot with external services, Dialogflow provides the Fulfillment concept. Fulfillment is code that's deployed as a webhook that lets the Dialogflow agent call business logic on an intent-by-intent basis [25]. Fulfillment allows the use of information extracted by Dialogflow's natural language processing to communicate with some external system via webhook and generate dynamic responses (using the external system as source), or trigger actions in the external system.

2.7.2 Microsoft Bot Framework

Microsoft bot Framework, unlike what we have seen in Google Dialogflow, is the result of various independent components. The three most essential components to build a chatbot are: BotBuilder SDK, Language Understanding (LUIS) and Azure Bot Service. Each component is explained next.

BotBuilder SDK. BotBuilder is the SDK that Microsoft provides in order to programmatically create our bot. This is the central piece of the bot and it's where the conversational flux is defined. To extend the conversational capabilities, it can be connected to other Microsoft services, such as LUIS for intent and entity recognition or Bing Spell Check to perform user's spelling correction.

Any call to an external service can also be done in the BotBuilder, by simply using the methods available in the SDK language.

Luis. Luis is one of the cognitive services provided by Microsoft. It is a natural language understanding service. It is an essential piece in the Microsoft Bot Framework, enabling the bot to interpret the user utterance. This is where the previously common concepts of "intent" and "entity" are defined.

The model is trained in LUIS website, where the bot creator gives training data to the model. It is possible to define all the intents and entities that our bot should recognize. Then we should add multiple utterances examples for each intent, marking the previously defined entities in that utterances.

The LUIS service is exposed via an endpoint URL and key. When LUIS receives query, it returns a json file with the query received, the top scoring intents detected, and the entities detected. This endpoint can then be called by the botbuilder application but can also be used by any other application to add NLU capabilities.

Azure Bot service. Azure Bot service offers an integrated environment to develop bots. It integrates the botbuilder sdk, making possible to create the bot directly from the azure website and host it in the azure servers.

Furthermore, the Azure bot service can be used to just register the bot, by providing the endpoint where the bot is hosted (hosting in azure is not required, any SSL endpoint is accepted). When the bot is registered, it is created a Microsoft app id and secret key that is used in the botbuilder sdk.

It is also in Azure Bot Service that the integrations with channels such as Facebook messenger or Skype is done

.

2.7.3 Watson Assistant

Watson Assistant is used in a similar manner to DialogFlow in the sense that both the intents/entities training, and the conversation flux is defined in the Watson website UI, as opposed to Microsoft Bot Framework which defines the conversation flux on a programming SDK.

The conversation concepts present in Watson are [26]:

- **Dialogs:** The dialog matches intents to responses. The dialog is represented graphically in the Watson Assistant tool as a tree. A branch should be created to process each intent that conversation should handle. A branch is composed of multiple Dialog nodes.
- **Dialog nodes:** Each dialog node contains, at a minimum, a *condition* and a *response*. A condition specifies the information that must be present in the user input for this node in the dialog to be triggered. The information is typically a specific intent while a response is the utterance that the service uses to respond to the user.

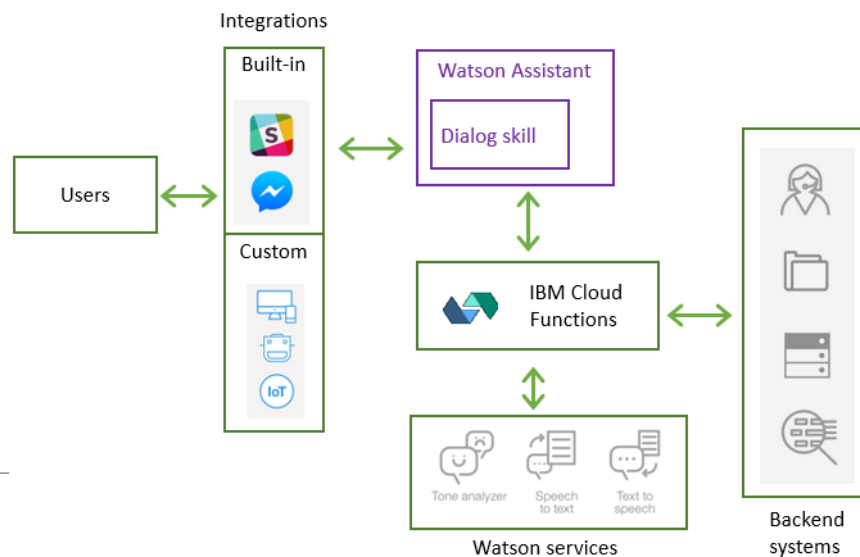


Figure 2 - Watson Assistant Overall Architecture – source [27]

Integration With external Systems. Watson assistance allows to associate programmatic calls to dialog nodes. This can be achieved using IBM Cloud functions, where the script that interacts with the external system is deployed [28]. In the dialog node it is defined the cloud function that should be run and can also be defined entities values to be passed to that function. The cloud function then returns an answer that can be accessed in a variable inside dialog node, in order to respond to the user.

2.7.4 Performance Comparison

A study [29] compared the performance of several NLU services, with the goal to enable both researchers and companies to make more educated decisions about which service to use. In order to test the performance on multiple domains, there were created multiple corpus, namely: *chatbot*, *web apps* and *ask ubuntu*. The “chatbot” corpus was obtained from questions gathered by a Telegram chatbot in production use, answering questions about public transport connections. The “ask ubuntu” and “Web Applications” corpus are based on data from the *StackExchange* web platforms with the same name.

Chatbot corpus consisted of the intents “Departure Time” and “Find Connection”, with entities such as “Station Start” and “Station Destination”. The web apps corpus with intents such as “Change Password” or “Delete Account”. Finally, ask ubuntu consisted of intents such as “Make Update” or “Setup Printer”.

The results of the study, regarding intent and entity recognition for the platforms analysed in this paper, can be seen in Figure 3.

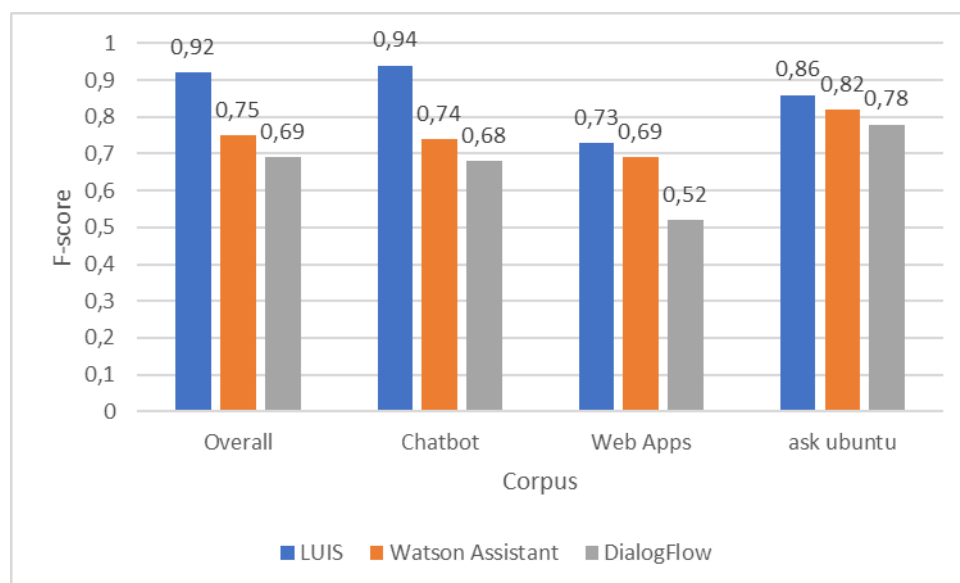


Figure 3 - F-scores for the different NLU services, grouped by corpus – source [29]

It is possible to conclude that Microsoft’s LUIS obtained the best results in all the corpus tested, having a significant overall score difference from the rest. Additionally, the authors of the study note that the results of Dialogflow were heavily influenced by its shortcomings regarding entity detection and its results would improve if only intent detection was considered.

2.7.5 Comparison Summary

In Table 2, it is presented a summary of the differences between the three studied platforms. The factors summarized are: the chat channels with supported integration; the number of languages supported; how the integration with external services is enabled; the summary of the performance results; if the dialog manager provided by the platform can be extended to be used with another NLU service; and finally if the service offers sentiment analysis.

Table 2 - Comparison summary of studied platforms

	DialogFlow	Watson Assistant	Microsoft Bot Framework
Channel Integrations	15	2	12
Languages Supported	27	13	12
Requires Programming Knowledge	No	No	Yes
Integration with ext. systems	Inside Dialogflow platform website via fulfilment and webhooks it is possible to call external services	Inside Watson Assistant website platform, calling IBM Cloud functions	Using botbuilder SDK
Performance (Overall F-score result)	0.69	0.75	0.92
Dialog Manager can be integrated with other NLU services	No	No	Yes
Sentiment Analysis	Yes	Yes, if integrated with Watson tone analyser	Yes
External Login Support	No	No	Yes, OAuth2

In summary, Dialogflow offers more chat channels integrations and is the solution that supports more idioms. Both DialogFlow and Watson assistant offer graphical interfaces to create the chatbot, not requiring programming knowledge. This can accelerate development time, while Microsoft Bot Framework requires programming knowledge that makes it more extensible. Concerning the natural language understanding performance, i.e. intent and entity recognition, Microsoft's solution has substantially better results with LUIS than both other solutions. Finally, Microsoft Bot Framework is the only solution that offers support for authentication with external services, by providing support for OAuth 2 protocol.

3. Chatbot Use Case Evaluator

This section identifies use case factors that should be considered when assessing the suitability of a use case for a chatbot implementation. A *use case selection process* is then proposed. This process allows the evaluation and ordering of a set of use cases, enabling the identification of the more suitable ones for chatbot implementation. After the definition of the *use case selection process*, it is applied to a set of use cases in the healthcare domain.

3.1 Use Case Evaluation Factors

In order to enable the evaluation of use cases, several factors are considered, reflecting the characteristics a use case should have in order to be appropriate to be implemented in a chatbot.

Chatbots lie between human operators and traditional graphical user interfaces (GUI) applications. In one hand, they can be used in the place of a human, offering a similar way of interaction, by using natural language. On the other hand, they can also be used instead of a traditional application, replacing a traditional graphical user interface with natural language.

The fact that chatbots share characteristics of both humans and GUI applications, fosters the division of the factors in three major groups:

1. **General Factors:** general factors that are essential to be considered to assure the viability of the use case to be implemented in a conversational UI.
2. **Factors over GUI application:** this group of factors reflect characteristics of a use case that can indicate that a chatbot is more adequate to expose it, instead of a traditional GUI application.
3. **Factors over Human:** factors that reflect characteristics of a use case that can indicate that the use case would benefit from being implemented by a chatbot instead of a human operator.

An analysis of such factors for each category yielded the following result.

General Factors

- **Business Rules well defined:** Chatbots perform better solving specific requests were the process to solve it is standard [30]. This facilitates the creation of the flow of the conversation based on that business rules.
- **Integration with Existing Systems:** concerns if it's possible to integrate the bot with the organization systems, via existing APIs. This factor guarantees that the chatbot can access the business logic and data required to the use case in question.

Factors over GUI Applications

- **Multiple steps or multiple input parameters** [12]: A simple traditional UI might be more practical to use cases that are simple and require only one step, but for tasks that require several user data, using NLU we can sometimes get all the information that the user would input in a form, for instance, in a simple sentence. Consider the sentence “Can you rebook my flight to Madrid to the following Monday after 3pm and get me a window seat”. A traditional GUI would require the user to insert the different pieces of information in different steps of the process, while a chatbot would recognize all the information parameters directly from the user sentence.
- **Notifications required**: Messaging applications already include an efficient and functional push notification system, which is available by default without any additional implementation effort [31].
- **Authentication required** [31]: Usually, for each new application, users must create a new account to be uniquely identified. With bots, user authentication is not necessarily needed. The messaging platform used already provides a reliable identification of the user. Users are uniquely identified by default. This reduces the effort asked to the user to start using the service, not requiring to create an additional account.

Factors over Humans

- **High Volume, Simple tasks, performed by humans**: For simple, well defined, repetitive tasks, a chatbot can be more suitable than a human, in the sense that is more economical and frees the human resources for another tasks [12].
- **Consistency required**: For use cases that is important consistency in the performance, i.e., the use case must be performed the same way in every occurrence, chatbots can be more suitable than a human operator [11]. Bots are intrinsically more consistent than human operators.
- **Scalability required** [11]: some use cases have unstable loads of requests from users. Bots can scale-up to fulfil the requests. Using human operators, is hard to handle sudden increases of requests.

These factors can be used not only to find out if a use case is suitable to a chatbot implementation, but can also indicate if a chatbot is more appropriate when compared with a human operator, or a traditional GUI application. The use case evaluation factors are summarized in Figure 4.

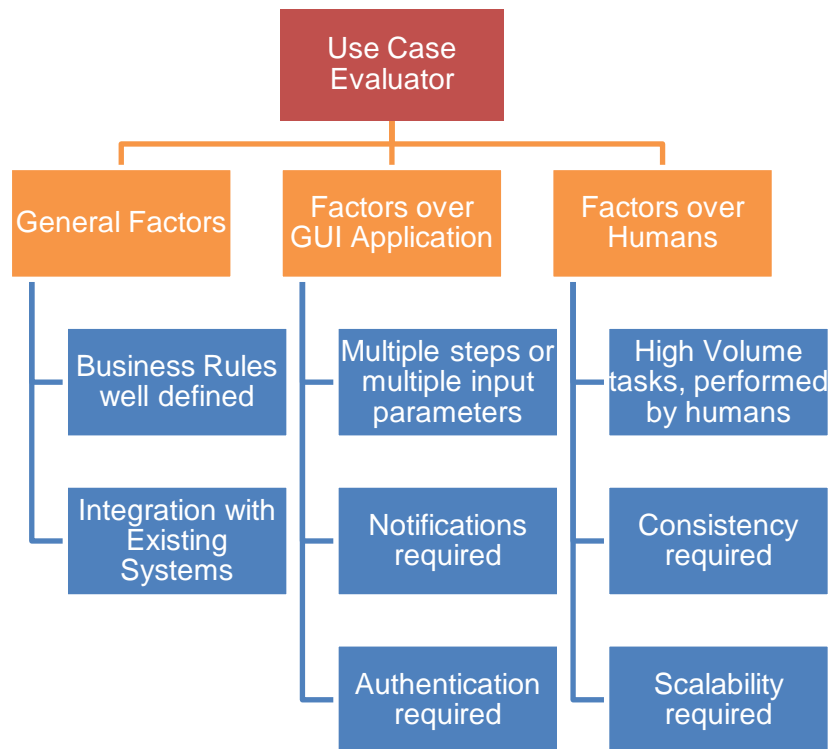


Figure 4 - Use Case Evaluator Factors

3.2 Use Case Selection Process

This section describes how the use case factors, defined in the previous section, can be used in order to select use cases suitable to implement in a chatbot. In order to accomplish the selection of use cases, a *use case selection process* is defined.

The first question the process aims to answer, for each use case, is:

- Is the use case viable to successfully implement in a chatbot?

If the answer found to this question is negative, the use case is evaluated as not suitable to implement in a chatbot and the use case evaluation stops. On the other hand, if the UC is viable to implement in a chatbot, the following two questions arise:

- Is the use case more adequate to implement in a chatbot when comparing with a GUI application?
- Is the use case more adequate to implement in a chatbot when comparing with a Human Operator?

Finally, after answering these questions for each use case, the process also aims to answer:

- What use cases should be prioritized for chatbot implementation?

The process proposed consists of three major steps, summarized in Figure 5.

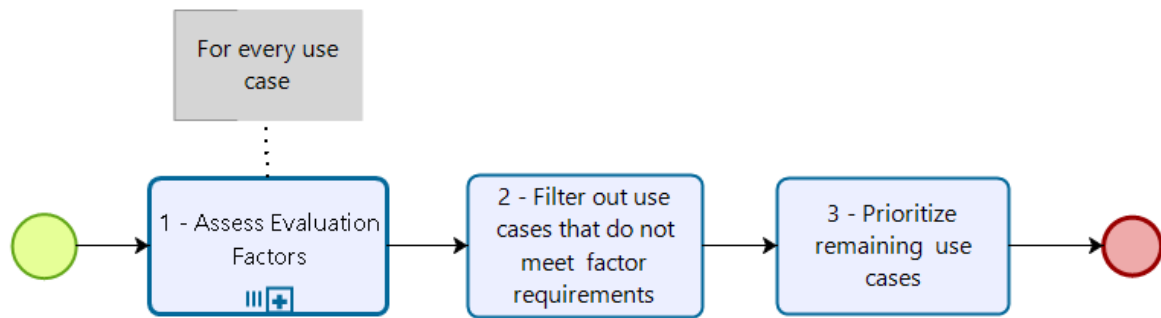


Figure 5 - Use Case Selection Process: overview

The following *use case selection process* should be applied in order to identify and prioritize use cases to be implemented in a chatbot:

1. Assessment for each use case:
 - 1.1. Assess if the UC is compliant with all the *General factors*.
 - a. If the use case is not compliant with any of the factors, it should not be implemented in a chatbot altogether and the process stops. The *general factors* function as minimum requirements to a UC to be considered for a chatbot implementation;
 - b. If the use case is compliant with all general factors, continue to step 1.2.
 - 1.2. Assess *Factors over GUI application*
 - a. If some UC factors are compliant, a chatbot implementation is considered advantageous over a GUI application;
 - b. If none of the factors are compliant, a chatbot is not considered as more suitable than a GUI application.
 - 1.3. Assess *Factors over Human operator*
 - a. If some UC factors are compliant, a chatbot implementation is considered advantageous over a Human operator;
 - b. If none of the factors are compliant, a chatbot is not considered as more suitable than a Human operator.
2. Filtering use cases: the use cases considered for chatbot implementation are only the ones that fulfil the following requirements:
 - Meet the general factors (step 1.1.b);
 - Advantageous to implement over both a GUI application (step 1.2.a);
 - Advantageous to implement over a Human operator (step 1.3.a).
3. After applying this evaluation process, and filtering out use cases using the before mentioned requirements, there might still be several use cases left as candidates to implement in a chatbot. The method purposed to prioritize the use cases is to prioritize the ones that meet a greater number of factors, or that meet factors that are of greater importance to the organization implementing the

use case. Organizations can define weights for each factor and prioritize use cases that have a greater sum of weighted factors.

While Figure 5 summarizes the use case selection process, Figure 6 presents in detail the first step of the process, that is applied to each use case.

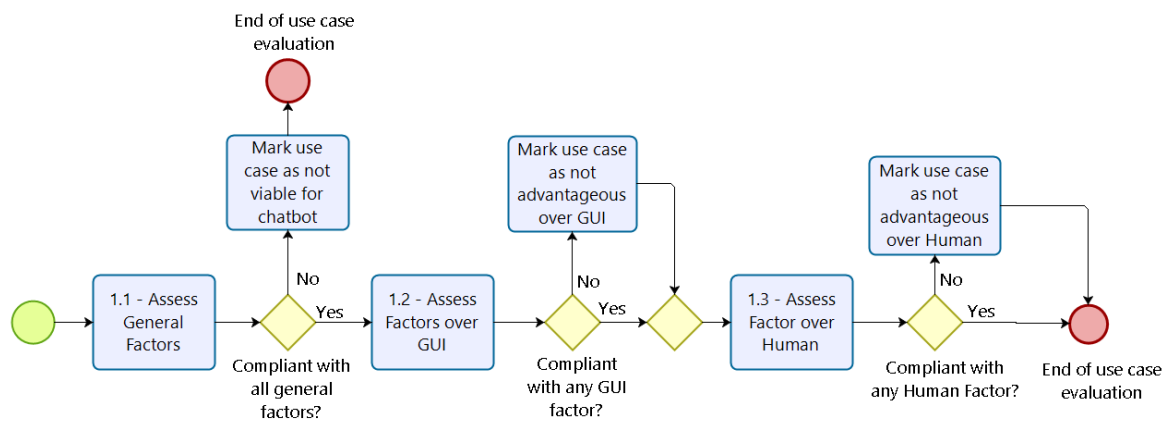


Figure 6 - Assess Evaluation Factors subprocess

3.3 Use Case Evaluator Application

In this section, the use case evaluation factors defined in section 3.1 are applied using the *use case selection process*, described in section 3.2, in order to select suitable chatbot use cases in the healthcare domain.

The healthcare use cases considered are:

- Scheduling an appointment;
- Paying for an appointment;
- Performing a medical diagnosis.

Each use case is evaluated next. In the end of the section, the conclusions about the most suitable use cases for implementation are presented.

3.3.1 Scheduling an Appointment

3.3.1.1 Use Case Definition

In order to schedule an appointment, the user must: select the specialty needed; choose one of the available doctors; and finally choose one of the available time slots (see Figure 7). This use case is typically performed with clerks, either by phone or in person, or in the case of some clinics, an application is available to the user schedule the appointment independently. Users are notified of the appointment close to the scheduled date, in order to confirm the presence or optionally reschedule (see Figure 7).

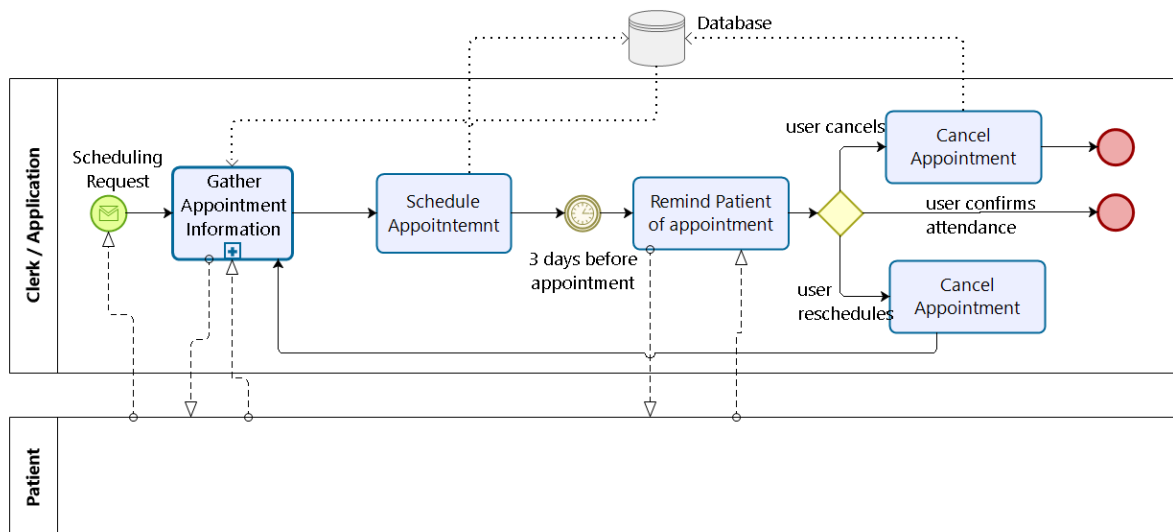


Figure 7 - Scheduling appointment business process

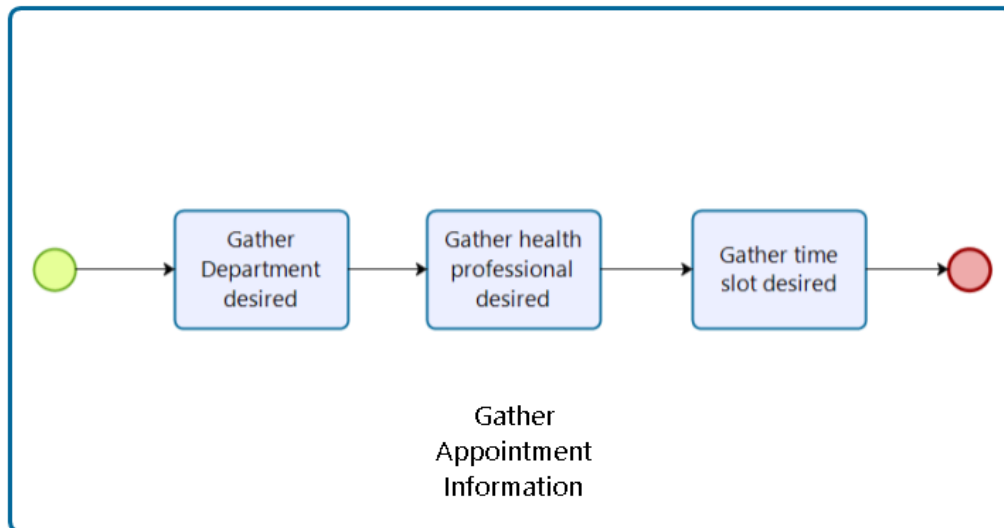


Figure 8 - Gather Appointment Information subprocess

3.3.1.2 Use Case Assessment

Step 1.1: General Factors. Scheduling an appointment is a common use case that has business rules well defined. In the case of MedClick, an API is available in order to request all the information needed to perform this use cases, including list of doctors, available time slots, and scheduling the appointment.

Table 3 - General factors assessment for scheduling appointment UC.

Factor	Assessment
Business Rules Well Defined	Yes
Integration with existing Systems	Yes

Step 1.2: Factors Over GUI Application. Scheduling an appointment has multiple input parameters, namely the desired medical specialty, the name of the doctor and the time slot. The identification of the user must also be known. The user must be notified close to the date of the appointment, in order to confirm its presence or, optionally, reschedule the appointment.

Table 4 - Factors over GUI application assessment for scheduling appointment UC.

Factor	Assessment
Multiple Steps/ Multiple Input parameters	Yes
Notifications Required	Yes
Authentication Required	Yes

Step 1.3: Factors Over Humans. Hospitals and clinics usually address high volume requests for appointment scheduling. The volume of requests may vary in an unpredictable way, requiring scalability. This use case is still commonly performed by human operators.

Table 5 - Factors over Humans assessment for scheduling appointment UC.

Factor	Assessment
High Volume, Simple Tasks, performed by humans	Yes
Consistency required	No
Scalability required	Yes

Evaluation Conclusions. The fact that this use case meets the two general factors indicates the viability to implement it in a chatbot. Furthermore, it is possible to conclude that the use case might benefit from the implementation in a chatbot over a traditional GUI application, meeting the three factors. Being a use case with multiple input parameters, it might benefit particularly from the extraction of the parameters directly from a user sentence, by using NLU. It might also be adequate to implement it in a chatbot over a human, in the sense that can free-up human resources and can easily scale, which is important to a use case with unpredictable loads of requests as this.

3.3.2 Paying for an Appointment

3.3.2.1 Use Case Definition

In the end of each medical appointment, the patient performs its payment. It is typical performed with clerks at the clinic/hospital. The clerk informs the patient of the value of the appointment and receives the payment. The price may depend on the patient's assurance.

3.3.2.2 Use case Assessment

Step 1.1: General Factors. Paying for an appointment has well defined business rules. In Medclick context the integration with existing systems could also be made by communicating with Medclick's API, and the information about healthcare prices and assurances can also be obtained by using the API.

Table 6 - General factors assessment for paying appointment UC

Factor	Assessment
Business Rules Well Defined	Yes
Integration with existing Systems	Yes

Step 1.2: Factors Over GUI Application. Paying for a medical appointment is a straightforward process, without multiple steps or input parameters. It does not require notifications, because the payment is typically immediately performed and processed, after the appointment. The patient paying should be identified.

Table 7 - Factors over GUI application assessment for paying appointment UC

Factor	Assessment
Multiple Steps/ Multiple Input parameters	No
Notifications Required	No
Authentication Required	Yes

Step 1.3: Factors Over Humans. Hospitals and clinics usually address high volume requests for payment of appointments at the end of each appointment performed. Because the request for appointments may vary in unpredictable way, the number of payments also vary, requiring scalability. This use case is still commonly performed by human operators.

Table 8 - Factors over Humans assessment for paying appointment UC

Factor	Assessment
High Volume, Simple Tasks, performed by humans	Yes
Consistency required	No
Scalability required	Yes

Evaluation Conclusions. The fact that this use case meets the two general factors indicates it as viable to implement in a chatbot. Comparing with a GUI application, it only meets the *authentication required* factor, so it would not take advantage of NLU capabilities of a chatbot associated with the factor *multiple steps/ multiple input parameters* and would not take advantage of chat notifications. Therefore, the benefits over a GUI app exist but are not major. When comparing with a human operator, a chatbot would be more beneficial, in the sense that it is a high-volume use case that would free human resources and offer better scalability.

3.3.3 Performing a Medical Diagnosis

3.3.3.1 Use Case definition

The diagnostic process is a complex, patient-centred, collaborative activity that involves information gathering and clinical reasoning with the goal of determining a patient's health problem [32].

The diagnostic process proceeds as follows: First, a patient experiences a health problem. Once a patient seeks health care, there is an iterative process of information gathering, information integration and interpretation, and determining a working diagnosis. In order to accumulate information that may be relevant to understanding a patient's health problem, health professionals may use the following approaches: (i) perform a clinical history and interview, (ii) conduct a physical exam, (iii) performing diagnostic testing, and (iv) referring or consulting with other clinicians. The information-gathering

approaches can be employed at different times, and diagnostic information can be obtained in different orders [32].

3.3.3.2 Use case Assessment

Step 1.1: General Factors. As seen in the use case definition, the diagnostic process is a complex iterative process, and is not performed always in the same manner. Consequently, the business rules are not considered to be well defined.

Because the *business rules well defined* general factor is not met, and following the *use case selection process* (see section 3.2), this use case is evaluated as not suitable for a chatbot implementation and its individual assessment stops.

3.3.4 Use Cases Selection

The assessment results of the three use cases considered are summarized in Table 9. The table is the result of applying the step 1 of the *use case selection process* (see section 3.2).

Table 9 - Use cases individual assessment summary (step 1)

Use Cases Factors		UC1: Scheduling Appointment	UC2: Paying Appointment	UC3: Medical Diagnosis
General Factors	Business Rules Well defined	✓	✓	✗
	Integration with existing Systems	✓	✓	-
Factors Over GUI	Multiple Steps/Input parameters	✓	✗	-
	Notifications Required	✓	✗	-
	Authentication Required	✓	✓	-
Factors Over Human	High Volume tasks, performed by humans	✓	✓	-
	Consistency Required	✗	✗	-
	Scalability Required	✓	✓	-

With the results of the assessment of each use case, it is possible to perform the step 2 of the process, that filters out use cases that do not meet the defined requirements. Because the use case of performing a medical diagnosis (UC3) does not meet the *business rules well defined* (general factor), it is excluded from the candidates to implement in chatbot (step 1.1.a of the process). Both UC1 and UC2 meet all the general factors and are considered advantageous over a GUI and over a Human, because meet at least one factor of both *factor over GUI* and *factor over human* categories. Therefore, UC1 and UC2 are not filtered out.

Finally, step 3 is applied, and the use cases that were not filtered out are sorted by order of priority. *UC 1* and *UC 2* both meet the same *factors over human*. The differences are present in the *Factors over GUI*. *UC 1* meets all the three factors over a GUI, while *UC 2* only meets one factor (notifications required). Because the *scheduling appointment (UC 1)* meets more factors than *paying appointment* it is considered as more adequate and having more priority for a chatbot implementation (see Table 10).

Note that the all the factors that *UC 2* meets, are also met by *UC 1*, making the decision of priority clearer and a possible attribution of weights for each factor not needed entirely.

Table 10 - Use cases priority rank and conclusions

Assessment Questions Use Cases	Viable for chatbot implementation	Chatbot advantageous over GUI	Chatbot advantageous over Human	Priority rank
UC1: Scheduling Appointment	✓	✓ (3/3)	✓ (2/3)	#1
UC2: Paying Appointment	✓	✓ (1/3)	✓ (2/3)	#2
UC3: Medical Diagnosis	✗	-	-	-

4. Chatbot Implementation

This section explains how the chatbot that supports the use case of *scheduling a medical appointment* is implemented. In section 4.1, an overview of all the components and the high-level architecture of the bot is defined. In section 4.2, it is explained how the dialog management is performed by the bot. Later, in section 4.3 it is explained in detail how the natural language understanding is realized. In section 4.4 it is analysed how the bot can actively learn from user interactions. Section 4.5 specifies how the login is achieved in the chatbot. Finally, section 4.6 explains how the user appointment notifications are sent to the users via chatbot.

4.1 Components and Integrations Overview

The main goal of the chatbot implementation is to support users in the *Schedule Appointment* business process. To enable this use case, a bot application is created, that is served by other external components. The bot application is exposed via chat platforms, such as the Facebook Messenger or Skype, where users can interact with the bot. The high-level architecture of the solution for the implementation of the chatbot is defined in Figure 9, where the several applications and services that interact with the bot application are represented. The end of each component is explained next.

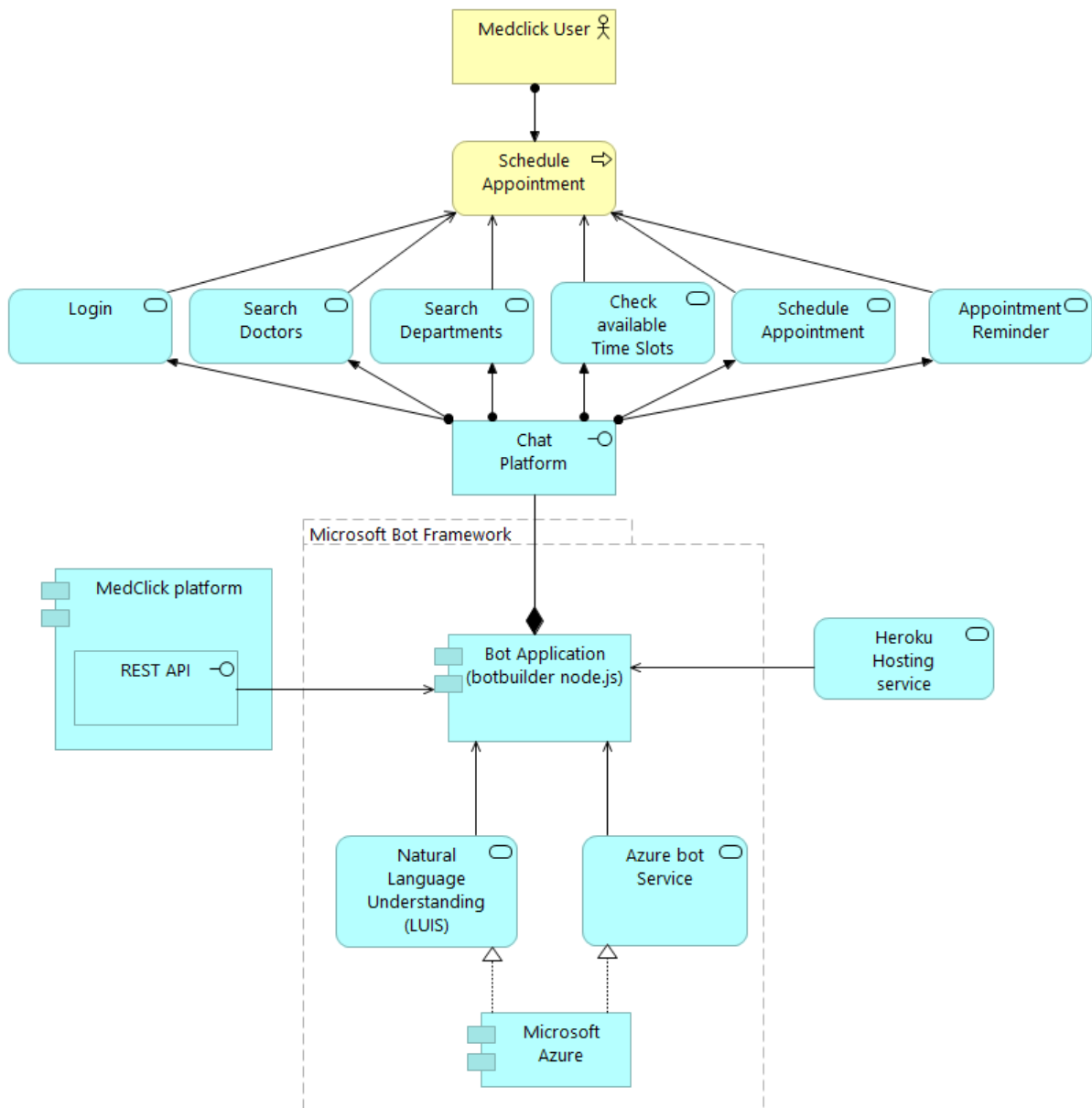


Figure 9 - Chatbot Architecture

Medclick Platform. The Medclick platform is the system that provides the business data needed to schedule an appointment (domain object *HealthCare*). The domain model containing all the business objects and its relations relevant to bot's scope is detailed in Figure 10. Medclick exposes the business objects via a REST API, where objects such as the health professionals or the list of medical departments (*specialty* domain object) can be fetched. It also exposes the relations between the several business objects. It is possible to get, for instance, the list of all the health professionals that practice a given specialty.

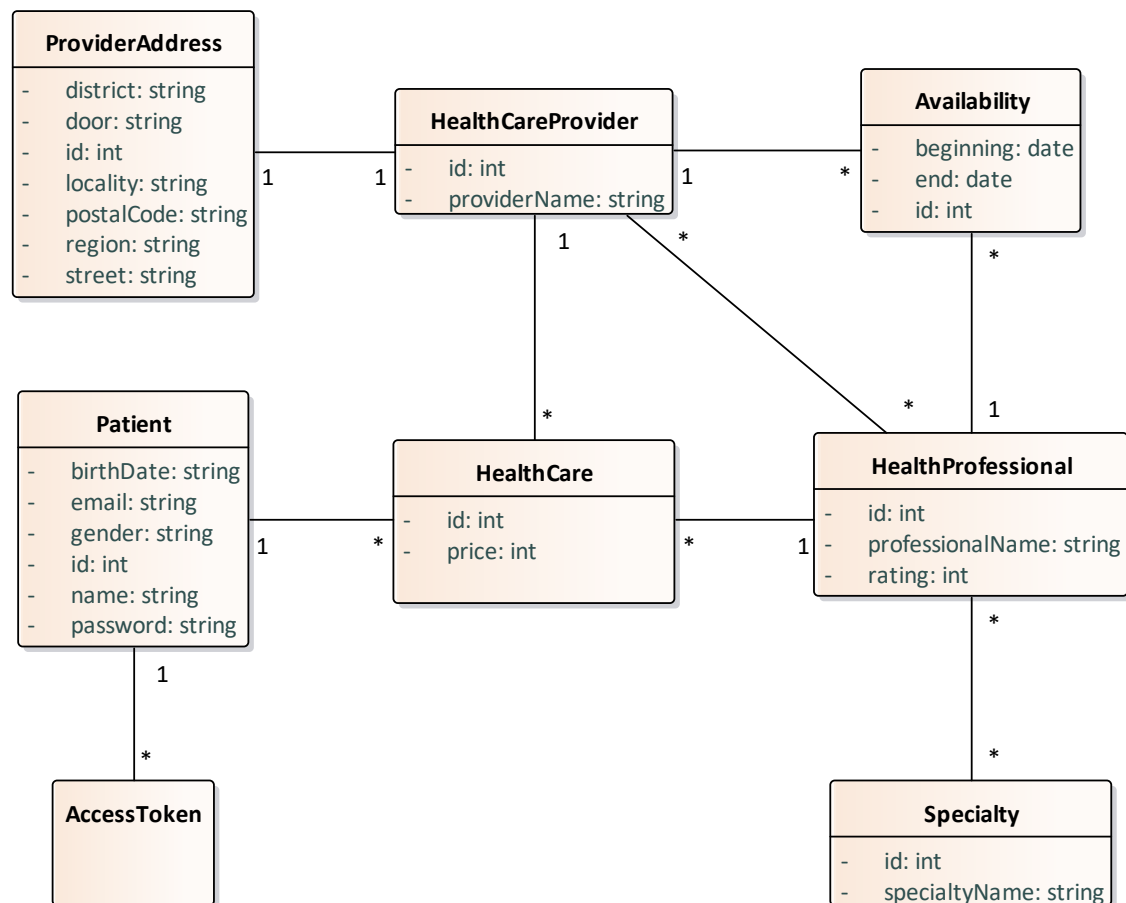


Figure 10 - Medclick domain objects

Microsoft LUIS. Microsoft LUIS is the natural language understanding component that is responsible to give semantic meaning to the users' utterances. It is exposed via an internet endpoint, that receives an utterance, and returns the intent and entities identified. Luis was chosen as the NLU service due to the fact that it is the service with better intent and entity detection performance from the services explored in section 2.7.

Bot Application. The Bot Application is a node.js web application. It is where the dialog management is defined, using the *botbuilder sdk* from the Microsoft Bot Framework. It is exposed in the chat channels such as Facebook Messenger. It receives the user utterances and defines the answers. This central component interacts with all the other, namely: (i) The Medclick platform, to obtain the business data needed for scheduling the appointment with the user; (ii) Microsoft LUIS application, to understand the meaning of the users' utterances; (iii) The azure bot service, in order to connect to chat platforms such as the Facebook messenger or Skype.

The *botbuilder sdk*, from Microsoft Bot Framework, has the unique benefit of supporting login using the OAuth2 protocol. None of the other solutions analysed in section 2.7 support external application login,

that is a requirement to enable login in Medclick system from the bot application. This reason makes it the only suitable solution from the ones analysed in section 2.7.

Heroku Hosting Service. Heroku is the cloud platform as a service (PAS) where the node.js Bot Application is deployed.

4.2 Dialog Management

This section discusses how the dialog management is performed in the bot created. Dialog management concerns how the flow of the conversation is defined, and how the bot defines the output to the input use utterances. It explains how the bot controls the flux of the conversation using the concept of state-machine, and the evolution from the first version of the state machine to the final one.

4.2.1 State Machine: first iteration

The first iteration of the dialog manager of the dialog system is based on a finite-state automata (see Figure 11). The user enters the *schedule* state when the intention of scheduling an appointment is detected in the user utterance. The next states of the automata have the goal of filling the different slots needed to complete the scheduling in this frame-based conversation, i.e, the *department*, *doctor* and the *timeslot* frames.

After detecting that the user wants to schedule an appointment the dialog system enters the *askLocation* state, where the user selects the provider (clinic or hospital). The providers are filtered by asking the district and locality first, in sub-states of this state. The user system then enters the *askDepartment* state where the user is prompt to insert the department desired. After the department is known the system enters the *askDoctor* state, where the system prompts the user to choose between the doctors belonging to the department and the provider selected in the previous states. After the doctor is known, the system enters the *askTimeSlot* state where the user chooses between the doctor's availabilities. At this state the three frames are filled. Additionally, the system enters the *login* state in order to identify the user and the *askConfirmation* to allow the user to review the slots inserted and either: confirm the schedule of the appointment; cancel it; or restart the scheduling. The user might exit the *schedule* state at any sub-state, by using pre-defined exit/cancel keywords.

Figure 12 shows the implementation of this state machine, interacting with a user on Skype chat platform.

This dialog management is system-initiative in the sense that the system has the initiative of asking the questions, and the user answers to a specific given question. This makes the language understanding easier, because the system knows the context of the answer given by the user. However, it does not provide the flexibility of allowing user to freely mention another entities in an utterance.

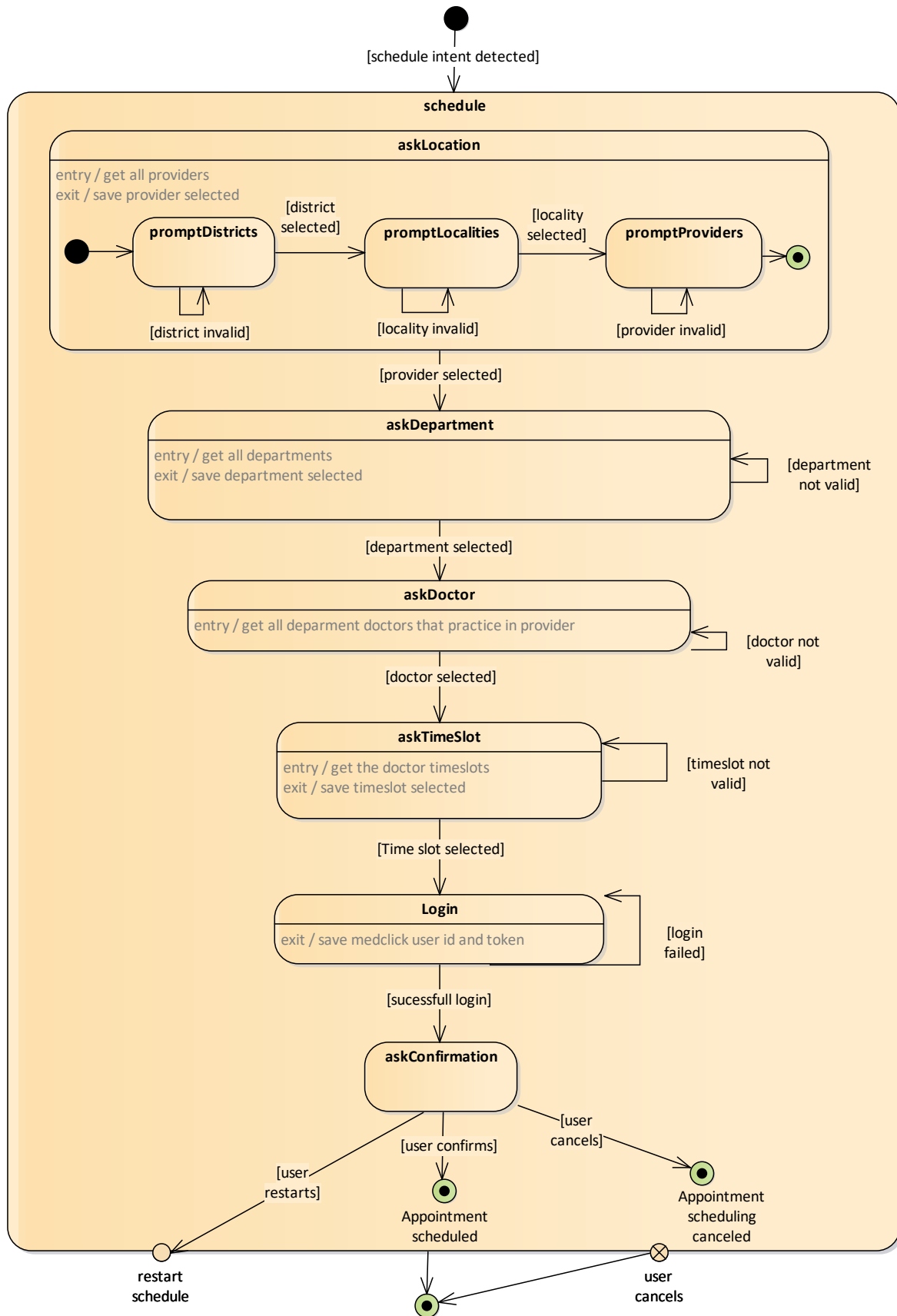


Figure 11 - Schedule dialog first state machine

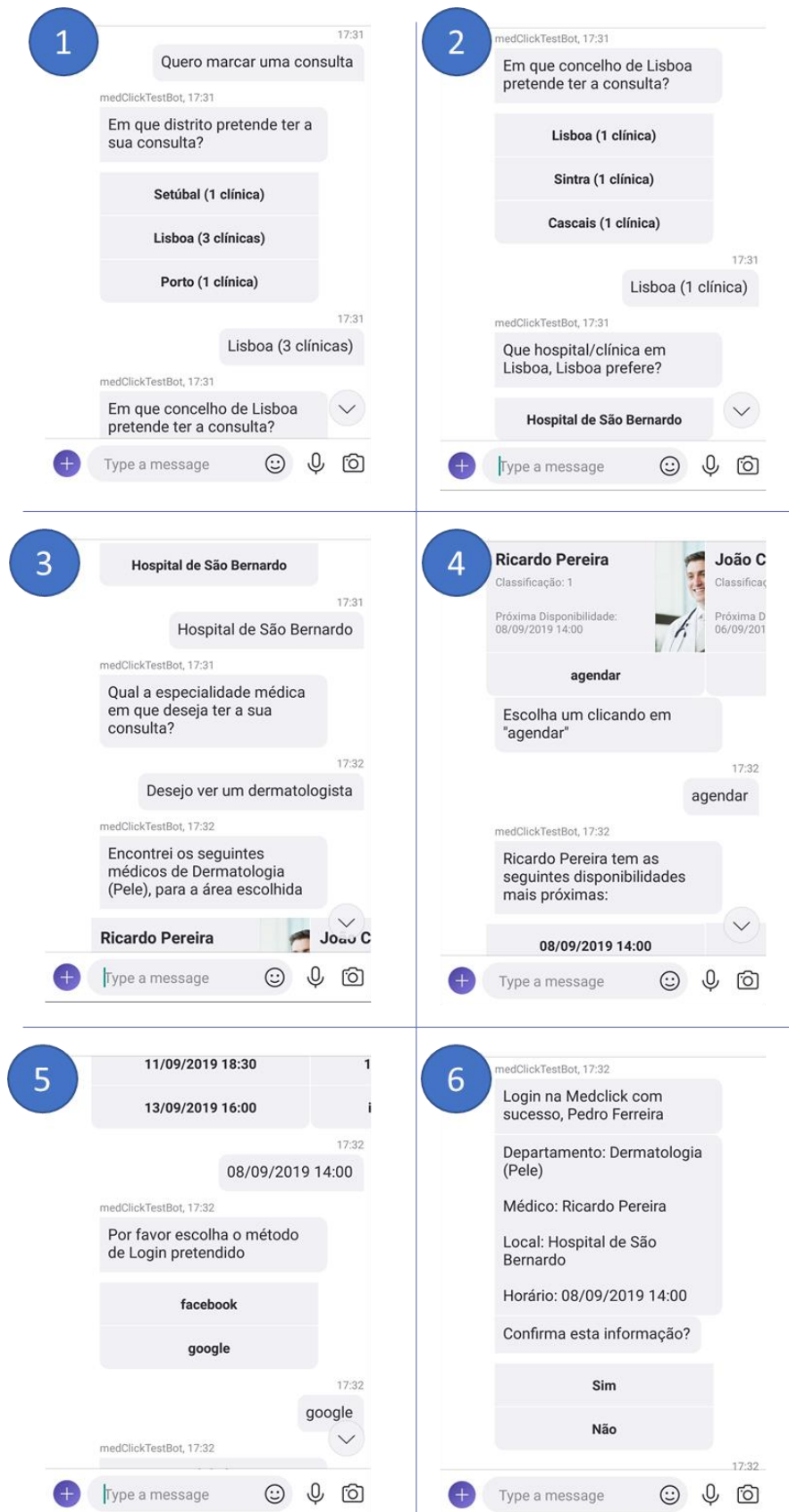


Figure 12 - Chatbot first iteration in skype

4.2.2 State Machine: final version

The system-initiative dialog management strategy of the first iteration (see section 4.2.1), does not adapt to the case where the user gives an utterance with several slots information. The user might say “*schedule appointment for tomorrow with Dr. James*” and the system would behave the same way if the user simply said “*schedule an appointment*”. Therefore, it would not benefit from the fact that this use case has multiple steps and multiple input parameters. This factor is one of the advantages of implementing such a use case in a chatbot instead of in a traditional GUI application, as discussed in section 3. The second iteration of the dialog management of the bot here presented, has the goal of taking advantage of NLU to extract multiple inputs and reduce the number of steps when the user utterance contains multiple input information (such as the name of the department or the name of the doctor desired). In order to achieve this, a new state machine was created (see Figure 13), extending the state machine of the first iteration.

In this iteration of the state machine, the system starts by entering the *processEntities* state. This state responsibility is to detect if the user mentioned the doctor, department or time entity. The next states of the automata are dependent of the entities detected:

- If the doctor entity is detected, the system skips the *askLocation* and *askDepartment*, then, instead of entering the *askDoctor* state, it enters the *confirmDoctor* state, that prompt the user to choose between the doctors with the most similar name to the one detected in the *processEntities* state. It then enters the *asDoctorLocation* state, that prompts the user to choose only between the providers where the selected doctor practices;
- If the user mentions the department, but not the doctor, the system skips the *askDepartment* state.
- The *askTimeSlot* was adapted to when *processEntities* detects date entity, prompt the user with time slots only in the period detected, when free time slots for the period requested are found.
- If there are no entities detected, the user behaves the same way as in the first iteration of the state machine.

This way, the dialog system adapts to the information that the user provides in the first utterance, allowing to skip steps all together, or prompt a restrict subset of domain objects based on the NLU entities found in the user sentence.

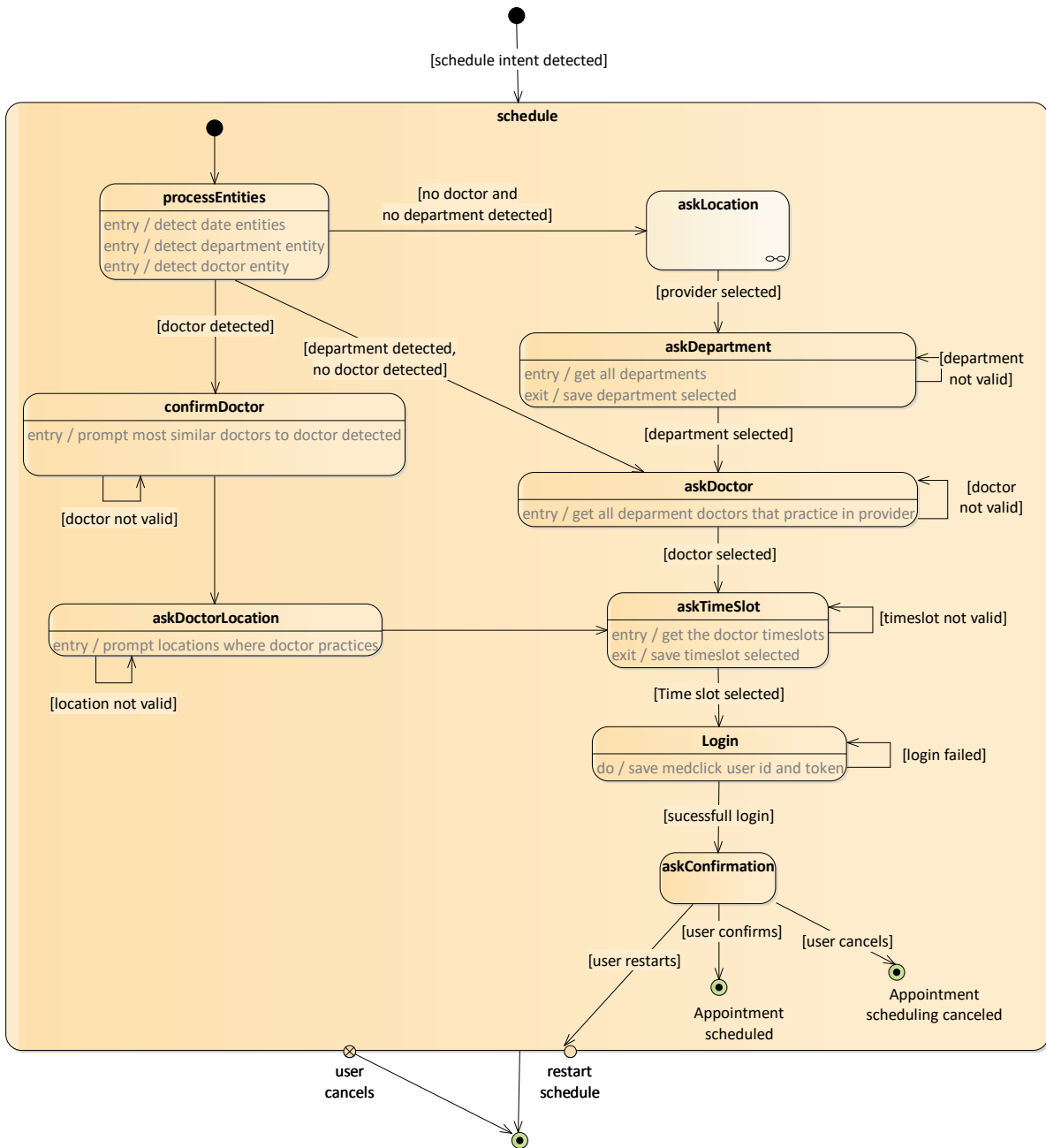


Figure 13 - Schedule dialog final state machine

In Figure 14, it is possible to observe an interaction with a user that mentions the name of the doctor and the date wanted in the beginning of the conversation. The system adapts to this information, by directly showing the most similar doctors found in medclick database, and by searching the doctor availabilities in the time frame requested by the user.

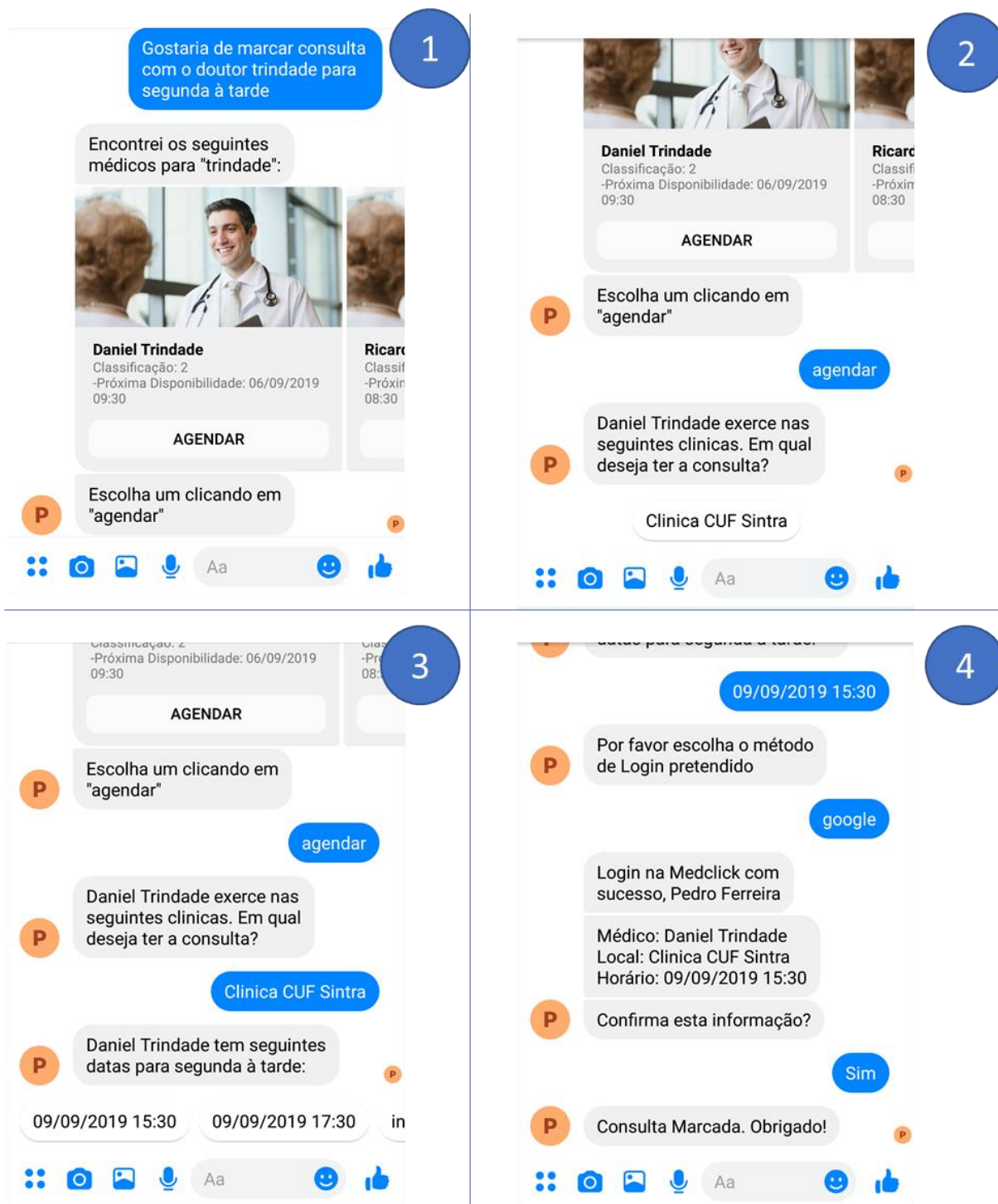


Figure 14 - Chatbot final version on Facebook Messenger

4.3 Natural Language Understanding

In order to extract meaning from the user utterances, the Microsoft LUIS service needs to be trained with labelled utterances. LUIS supports a frame-based approach, applying supervised machine learning to detect intent and its entities from utterances. The first step is therefore to model what are the user intents that we want LUIS to detect, and what entities should be extracted.

The main goal of the dialog agent is to allow scheduling of appointments. Therefore, should exist a *schedule* intent. The pieces of information we want to extract from the user that wants to schedule a medical appointment are:

- **Department:** The name of the medical department/specialty the user is seeking;
- **Doctor:** The name of the doctor the user wants to schedule;
- **Date:** When does the user want to schedule the appointment for.

The *department*, *doctor* and *date* translate as entities of the *schedule* intent, in the frame-based modelling of the NLU.

Additionally to the *schedule* intent, there should also be a *none* intent, to detect the cases that do not fall in any other intent, in this case utterances that do not represent the *schedule* intent. The summary of the intent and entities modelling can be seen in Figure 15.

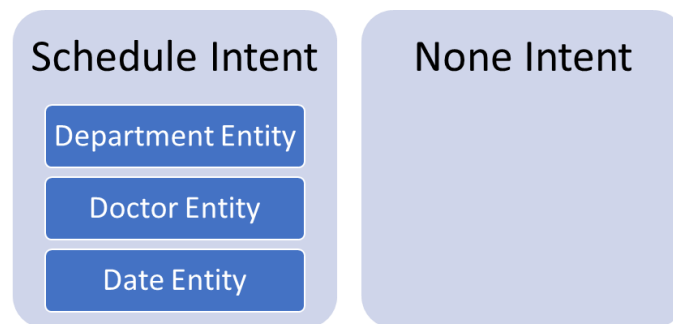


Figure 15 - Intent and entities modelling

After modelling the intent and entities needed, the LUIS model is trained with example utterances. The goal is to provide the LUIS model with a good enough variability of sentences that map to the intent, using common combinations of entities, in order to create a model that can generalize to recognize and label new sentences. In Figure 16 are presented some of the utterances used to train the *schedule* intent. Each sentence is inserted as an utterance the user might use, and then the entities are labelled. For instance, the first sentence of Figure 16 was inserted as “*marcar consulta com o doutor bernardo amanhã*”, and after labelling the entities it seen as “*marcar consulta com o doutor [doctor] [datetimeV2]*”, i.e., the entities instances are replaced by the general label identifier.

schedule

Labelled entities: [doctor](#), [department](#)








 Edit		 Reassign intent	 Add as pattern	 Delete	 Search		 Filter
<input type="checkbox"/>	Example utterance				Score		
Enter an example of what a user might say and hit Enter.							
	marcar consulta com o doutor	doctor	datetimeV2		1.00		
	quero marcar consulta com dr	doctor	para	datetimeV2	1.00		
	agendar consulta com	doctor			0.99		
	marcar com dr	doctor			1.00		
	agendar consulta com a doutora	doctor			0.99		
	marcar consulta com o doutor	doctor			1.00		
<input type="checkbox"/>	quero marcar um	departmentsList	para	segunda datetimeV2	: 0.99		
	marcar consulta	datetimeV2	sff		0.99		

Figure 16 - Training LUIS app with labelled utterances

LUIS has different types of entities. When creating the entities, it must be considered what is the most appropriate type for the entity in question. Entities can be extracted with machine-learning, which allows LUIS to continue learning about how the entity appears in the utterance or can be extracted without machine-learning, matching either exact text or a regular expression. Only Machine-learned entities need to be marked in the example utterances.

The different relevant types of entities LUIS offers are summarized in Table 11.

Table 11 - LUIS entities types

Entity Type	Machine Learned	Purpose
List	No	List of items and their synonyms extracted with exact text match.
Prebuilt	No	Already trained to extract various kinds of data.
Simple	Yes	Contains a single concept in word or phrase.
Regular Expression	No	Uses regular expression to match text.

Additionally, LUIS also offers other resources to improve accuracy. One of the resources is the *phrase list*. A *phrase list* includes a group of values (word or phrases) that belong to the same class and must be treated similarly. What LUIS learns about one of them is automatically applied to the others as well. Unlike the *List Entity* type (see Table 11), a *phrase list* is not used as an exact match and does not need to be every possible value expected.

4.3.1 Entity Recognition and Mapping

After training the LUIS app with labelled utterance, it is able to identify the intent of the user and the parts of the utterance that map to the modelled entities. But we still need to map the NLU entity identified to the specific domain business object the user is referring. Consider the query result in Figure 17. When the bot application receives this query result, it is possible to determine that the desired doctor name is “bernardo”, and that the user wants to schedule the appointment for tomorrow (“amanhã”). The question is who is the “bernardo” doctor in medclick’s database, and to which available time slots does tomorrow map to?

```
{
  "query": "marcar consulta com o doutor bernardo amanhã",
  "topScoringIntent": {
    "intent": "schedule",
    "score": 0.9965413
  },
  "intents": [
    {
      "intent": "schedule",
      "score": 0.9965413
    },
    {
      "intent": "None",
      "score": 0.0029450343
    }
  ],
  "entities": [
    {
      "entity": "bernardo",
      "type": "doctor",
      "startIndex": 29,
      "endIndex": 36,
      "score": 0.970532835
    },
    {
      "entity": "amanhã",
      "type": "builtin.datetimeV2.date",
      "startIndex": 38,
      "endIndex": 43,
      "resolution": {
        "values": [
          {
            "timex": "2019-08-16",
            "type": "date",
            "value": "2019-08-16"
          }
        ]
      }
    }
  ]
}
```

Figure 17 - LUIS response to query "marcar consulta com doutor bernardo amanhã"

The pipeline that resolves entities needs to be able to receive a user utterance, extract the NLU entities and map them to business objects. Figure 18 demonstrates the sequence of interactions required to perform the mapping.

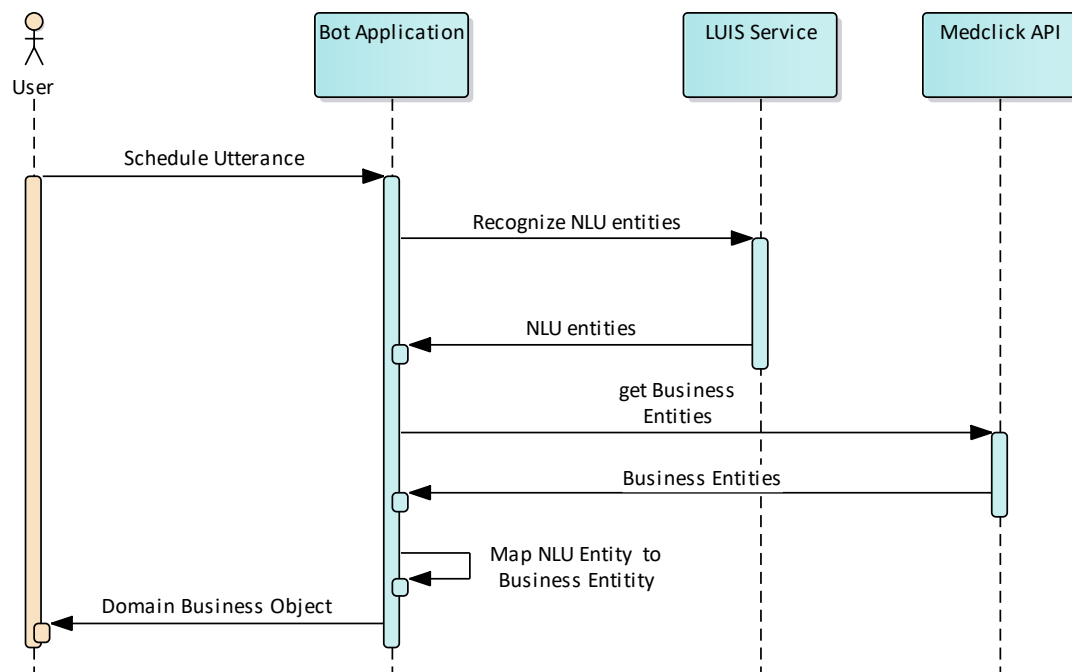


Figure 18 - Mapping user utterances to domain objects

The mapping is not realized exactly the same way for each entity, due to intrinsic differences between them. It is explored next the differences in the NLU entity recognition level and at the domain business object mapping level, explaining and how this pipeline is implemented for each entity in particular.

4.3.1.1 Department Entity

The department entity has a static nature, given that there is a close set of medical departments. It is not expected great variability on the way the departments are mentioned by the users, when compared to the other entities. Typically, the departments are mentioned by the name of the department or by the name of the profession (e.g. “ophthalmology” or “ophthalmologist”).

Given the static nature of the entity, the LUIS entity type chosen was the *List*. This entity type allows us to create a canonical form for the name of the department and add synonyms to it. LUIS then detects this entity by applying exact text match by searching for the canonical form or any synonym in the user utterance.

The Medclick API provides a list of all departments. It was extended to be able to associate synonyms to each department name. This way, additional synonyms can be added to the departments, as they are discovered in interactions with users. A script was created that loads every department from Medclick API and its synonyms and adds to the LUIS’ list entity the name of the department as the LUIS canonical form and the synonyms associated. This way LUIS will return the name of the department if there is a string match with the department canonical form or any of the synonyms loaded. By running this script there is a guarantee that the LUIS model is aware of all the nomenclatures for each department present in Medclick’s database.

Given that LUIS returns the canonical form of the department entity detected, the mapping to the business object is direct, because this canonical form is the name of department in medclick database. This way we simply query Medclick API for the department with the name equal to the canonical form returned by LUIS.

In Figure 19 it’s shown the LUIS result for the query “*marcar oftalmologista*”. LUIS detects “*oftalmologista*” as a department entity (departmentsList), and resolves it to the canonical form “Oftalmologia (Olhos)”. This canonical form is equal to the name of the business entity from the API, resulting in a direct mapping by these two strings.

LUIS Query Result

```
{
  "query": "marcar oftalmologista",
  "topScoringIntent": {
    "intent": "schedule",
    "score": 0.9563542
  },
  "intents": [
    {
      "intent": "schedule",
      "score": 0.9563542
    },
    {
      "intent": "None",
      "score": 0.02482305
    }
  ],
  "entities": [
    {
      "entity": "oftalmologista",
      "type": "departmentsList",
      "startIndex": 7,
      "endIndex": 20,
      "resolution": {
        "values": [
          "Ofthalmologia (Olhos)"
        ]
      }
    }
  ]
}
```

Business Entity (from Medclick API)

```
{
  "specialtyName": "Ofthalmologia (Olhos)",
  "price": 50,
  "code": "E07",
  "order": 28,
  "id": 28,
  "synonyms": [
    {
      "name": "oftalmologia",
      "id": 38,
      "specialtyId": 28
    },
    {
      "name": "oftalmologista",
      "id": 39,
      "specialtyId": 28
    }
  ]
}
```

Figure 19 - Mapping department list entity to business entity

4.3.1.2 Doctor Entity

The doctor name entity, as opposed to the department entity, is more dynamic and presents more variability. New doctors may be added and removed from the database at any time and may be referenced by the user in multiple forms. Given the fact that LUIS does not provide a *person name* built-in entity for the Portuguese culture, it was chosen to use the *simple entity* type.

The LUIS app was trained with some sentences containing doctor names. This training data with some labelled doctor names does not suffice for LUIS to be able to predict new doctor names effectively. Because a name can be anything, LUIS predicts entities more accurately if it has a *phrase list* of words to boost the signal of the *simple entity*. It was taking advantage of the fact that the possible doctor names we want LUIS to detect at a given time can be known by querying Medclick API for all doctors. A script was created to automatically create a *phrase-list* in LUIS with the names of all the doctors in Medclick at the moment the script is run. This way everything the model learns for the names inserted in the training phrases is also learned for the other names in the *phrase list*.

One possible way to address the mapping of the string returned by LUIS to the doctor domain object would be to apply exact string matching. However, this would be problematic, given that the name of the doctor in the database might not be exactly equal to how the user is referencing he or she. Personal names are referenced with several variations and errors, which makes an exact string matching approach problematic, creating the need to use approximate matching techniques [33]. Some variations that can be easily identified of how the user might reference the doctor are: the first name only; first and last name; full name.

The choice of the matching technique to be applied to the doctor name entity was based on a study [33] that compared several techniques performance for name matching applied to personal names. The experimental results on the four datasets tested are presented in Table 12. Even though the study concludes that there is no clear best matching technique for every case, it indicates Jaro and Winkler techniques as suitable for personal name data, given their good performance on all four data sets. Considering the results of the study, the algorithm chosen was the Winckler (also known as Jaro-Winkler).

Table 12 - F-measure scores for different Personal Name Matching Techniques

	Midwives			COMPLETE surnames
	given names	sur- names	full names	
Soundex	.342	.341	.376	.485
Phonex	.423	.369	.499	.579
Phonix	.339	.330	.368	.617
NYSIIS	<u>.275</u>	<u>.296</u>	<u>.299</u>	<u>.351</u>
DMetaphone	.304	.306	.330	.410
FuzSoundex	.327	.311	.359	.396
Leven dist	.658	.513	.737	.624
Dam-L dist	.659	.517	.739	.625
Bag dist	.597	.522	.670	.616
SWater dist	.889	.579	.802	.617
LCS-2	.915	.564	.877	.514
LCS-3	.909	.529	.866	.500
1-grams	.839	.588	.787	.627
2-grams	.885	.498	.867	.519
3-grams	.783	.442	.833	<u>.416</u>
Pos 1-grams	.890	.574	.724	.653
Pos 2-grams	.880	.473	.697	.508
Pos 3-grams	.768	<u>.416</u>	.659	<u>.416</u>
Skip grams	.844	.496	.825	.521
Compr BZ2	<u>.458</u>	.547	<u>.568</u>	.633
Compr Zip	.532	.456	.684	.481
Jaro	.853	.601	.829	.712
Winkler	.891	.588	.868	.707
Editex	.631	.561	.706	.646
SyllAlign dist	.656	.426	.710	.532

The algorithm that calculates the distance of the doctor name mentioned by the user, to a doctor in the database is presented in Figure 20. The algorithm receives two strings as arguments: the *doctor mentioned* recognized by LUIS, and the full name of some *database doctor*, and returns the distance between the two names. The algorithm starts by creating an array with all the names of the *doctor mentioned*, by separating the name string by words. The same pre-process is applied to the *database doctor* name that will be compared. Then for each name of the *mentioned doctor*, it's found the most similar name of the *database doctor* that is being compared, by finding the least jaro-wrinkler distance. The total distance between the *mentioned doctor* name and the *database doctor* is the sum of the minimum distances of each *mentioned doctor* name to each *database doctor* name.

```

function calculateNameDistance(doctorMentioned, doctor){
  doctorMentioned = doctorMentioned.match(/\S+/g);
  doctor = doctor.match(/\S+/g);
  let fullNameDistance = 0;
  for (let mentionedName of doctorMentioned){
    let minDistance = Number.MAX_SAFE_INTEGER;
    for (let doctorName of doctor){
      let distance = winkler(doctorName, mentionedName).distance;
      if (distance < minDistance){
        minDistance = distance;
      }
    }
    fullNameDistance += minDistance;
  }
  return fullNameDistance
}

```

Figure 20 - Algorithm used to calculate distance of the doctor mentioned name to a database doctor name

The algorithm in Figure 20 is applied to all the doctors in the database, in order to compare the mentioned doctor name with all the database doctors. This comparison allows to find the database doctor with the smallest edit distance to the doctor mentioned, i.e., the doctor with the most similar name.

4.3.1.3 Time Entity

The date when the user wants to schedule the appointment can be referenced by the user with great variability of format. In natural language it cannot be expected that the user mentions the date in a standard format. Consider the user wants to schedule an appointment for tomorrow. The user might reference tomorrow in a more standard format such as *dd/mm/yyyy* or might simply use the word “tomorrow” or even reference this date by the weekday. The solution to extract the time entity and to resolve it to a standard format is not trivial.

Microsoft LUIS service provides a built-in entity type to recognize date and time values from the user utterance. This entity type is named *datetimeV2*. The service not only identifies the part of the utterance containing the *datetimeV2* entity, but also resolves the date from natural language to a standard format. The use of this entity simplifies substantially the extraction and resolutions of dates, given that it is already trained for the Portuguese idiom, supporting known temporal expressions of the language. The entity chosen to represent the date that the user wants to schedule the appointment was therefore the built-in pre-trained *datetimeV2*.

```
{
  "entity": "amanhã à tarde",
  "type": "builtin.datetimeV2.datetimerange",
  "startIndex": 16,
  "endIndex": 29,
  "resolution": {
    "values": [
      {
        "timex": "2019-08-20TEV",
        "type": "datetimerange",
        "start": "2019-08-20 16:00:00",
        "end": "2019-08-20 20:00:00"
      }
    ]
  }
}
```

Figure 21 - LUIS resolution for "amanhã à tarde" datetimeV2 entity detected

Figure 21 presents a LUIS datetimeV2 entity detected. It resolves the “*amanhã à tarde*” entity to a standard date format with a start and end date. Using this standard format, the bot application can query the Medclick API for availabilities only in this time frame. This query to the Medclick is done by applying a filter to the http request that queries for the availabilities that queries availabilities only in the time frame resolved.

4.3.1.4 Entity Recognition and Mapping Summary

Table 13 summarizes for each entity the LUIS entity type chosen and the strategy implemented to map this detected entity to the domain object.

Table 13 - Entity Recognition and Mapping Summary

Entity	LUIS Entity Type	NLU entity to domain Object mapping
Department	List Entity	Direct match between canonical form returned by LUIS and Department Name in database
Doctor	Simple Entity	Calculating most similar name (least edit distance) in database using jaro-winckler algorithm
Date	Built-in (dateTimeV2)	Querying API for availabilities in the date or time frame resolved by LUIS

4.4 Active Learning

Dialog systems should be capable of learning from previous conversations with users, in order to enhance the quality of future conversations. Linguistic variability is one of the factors that makes active learning important. Because a user can express the same idea in several different ways, we aim to capture these variations actively in each interaction. When a dialog system is first deployed, it is useful to have a dataset of real-world conversations, that can be used to initially train the bot. This training can be done by labelling the intents/entities of each utterance in the dataset, in order to train a machine learning model. In the context of the bot created, it was not possible to gather real conversations of medical appointment scheduling. The fact that there is no real initial training data, foster the search for ways to mitigate this issue, and makes active learning a matter of increased importance.

Menu options and buttons. The chatbot may not be able to recognize an entity referenced by the user. The strategy used as a fallback is to present a list of options to the user. Consider the department entity. If a department is not detected in the first utterance, the bot will prompt the user to insert the name of the department. The text answer to the department question is analysed (using NLU by calling LUIS service), but the system might not be able to identify a department in the user response. In the case of failure of NLU, the system uses as fallback a less conversational approach, by showing the user a list of departments and asking the user to choose one (see Figure 22). This approach allows to let the user insert the department using natural language, and only in case of failure presents the list of departments for the user to choose, and looks for the most similar to the text inserted by the user. This strategy allows to learn how the user mentions the departments, as seen next in the next topic *synonyms*.

Review endpoint utterances ?

Filter: None	<div><div></div></div> Tokens View	
Utterance	Aligned intent ?	Add/Delete
marcar	schedule (0... ▾	<div><div>✓</div><div>✗</div></div>
marcar consulta amanhã	schedule (0... ▾	<div><div>✓</div><div>✗</div></div>
marcar consulta com doctor ricardo pereira	schedule (0... ▾	<div><div>✓</div><div>✗</div></div>

Figure 23 - Reviewing LUIS NLU predictions

In summary, by using active learning, the bot will be able to improve its NLU performance with user interactions. The increase in NLU performance, will better the conversational experience. Moreover, by using less conversational approaches, such as menu options and buttons when NLU fails, it is possible to assure that user will still be able to complete the appointment scheduling, even if using a more traditional GUI approach. This approach is a compromise that aims to solve the paradox that a conversational system needs good training data to offer a good experience, but to have good training data, the system needs user interactions.

4.5 Login

Medclick platform requires that a user is authenticated to allow the scheduling of appointments. The authentication is implemented using access tokens. Medclick's API has an endpoint for user login that receives the user credentials (email and password) and returns the access token. This access token allows the user to make authenticated requests to protected API endpoints, and functions as identifier of the user in Medclick backend.

The login could be trivially implemented in the chatbot, by just asking the user to insert the email and password in the chat window, and the bot would be able to request the user's access token to Medclick API. The problem with this approach is that when the user is interacting with the bot in channels such as Facebook Messenger or Skype, there is the additional issue that login credentials (email and password) must not be exposed in the channel. By simply asking the username and password in the chat window, not only would this information be exposed to the channel provider, but there would be no way to hide the password from the chat window when the user is typing it and after from the conversation history. This approach, even though simple to implement, would raise security concerns.

Microsoft bot framework solves the issue mentioned by using a card that is presented to the user, in the chat channel, called *OAuth card*. This card initiates an oauth2 flux and is configured in the azure portal, and can be later used in the bot. After the user clicks the OAuth card in the chat window, he or she is redirected to the OAuth2 provider login page. After a successful login, the bot receives the access token. This way, the credentials are inserted in the scope of the OAuth2 provider in question, instead of being inserted in the chat window. This authentication can be more easily implemented when the login is made with a known OAuth2 service provider. The bot service is already configured to deal with known providers, such as uber, facebook or google. Although it also allows to configure authentication to any "generic oauth 2", where fields such as "authorization URL" or "token URL" should be manually inserted.

The Medclick login system does not support OAuth2. In order to take advantage of the OAuth card, Medclick login was extended to support social login with facebook and google. It was created a *register* and *login* social endpoints, both for facebook and google:

- The *register* endpoint receives the social user token and uses it to retrieve user data from facebook/google, such as the name, email and facebook/google id. With this data, it registers the user as medclick user. The medclick user is linked with the facebook/google ID. This way, each user has a medclick ID and a facebook/google ID.
- The *login* endpoint receives a social token and uses this token to get the facebook/google id from the oauth2 provider. After having the facebook/google ID, medclick server uses it to identify the medclick user with that id associated. Finally, the medclick user token is returned.

After these extensions to the medclick API were completed, the azure bot service was configured to support OAuth login with facebook and google. The bot application presents an *OAuth card* to the user that redirects the user to facebook/google login, and as a result the bot application receives the facebook/google token. This social token is used to login the user, and the result is the medclick's user

token of the user with the facebook/google id associated. If no user with this social ID is found, the *register* endpoint is called first. The azure bot service stores the social tokens, so after the first time the users logs in, if the social token is still valid, azure bot service will return it to the bot, and there is no need to present an *OAuth card* to the user.

Figure 24 summarizes how the bot application retrieves the social token and uses it to retrieve the user token, calling the medclick API.

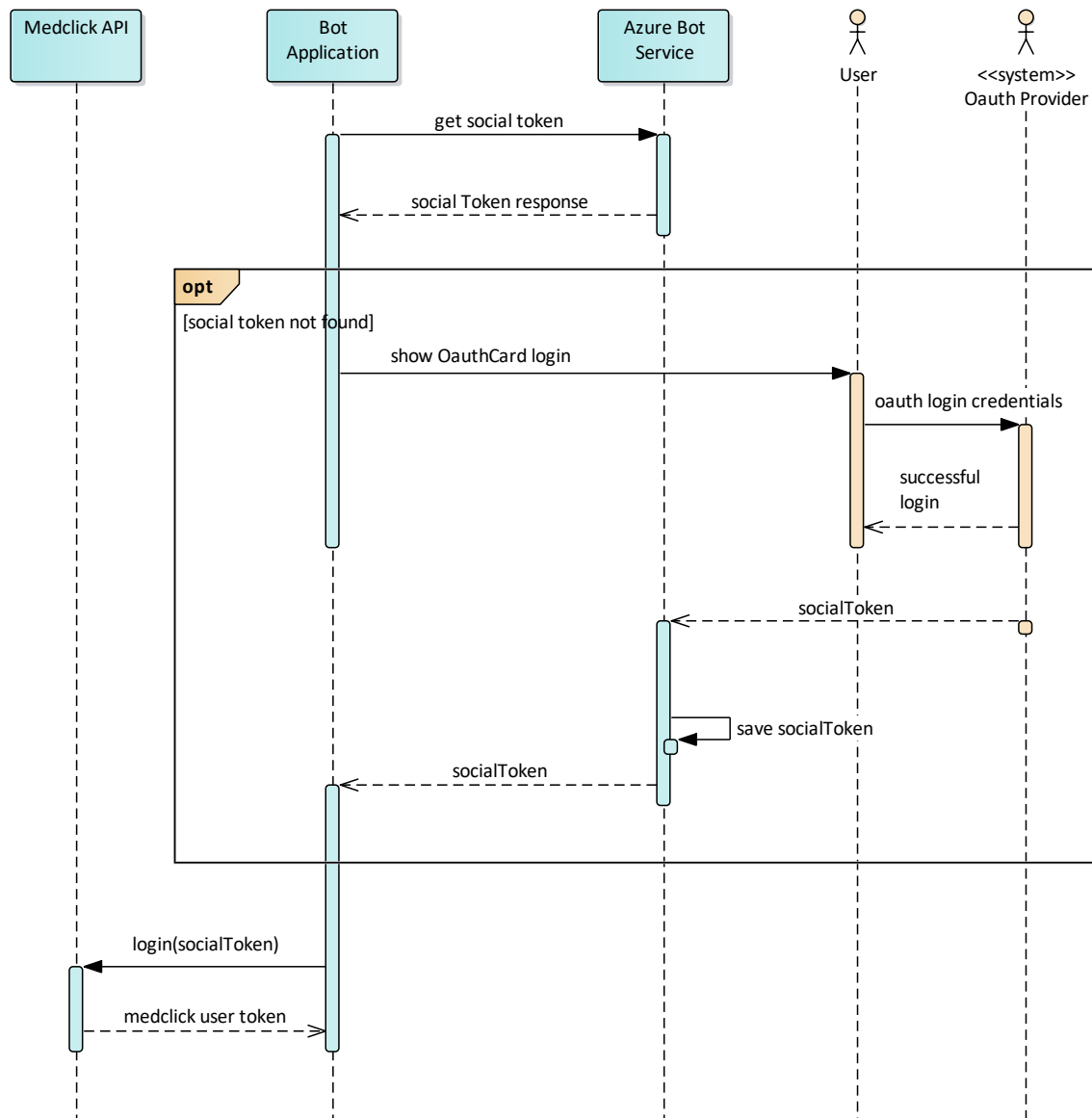


Figure 24 - Login with social token and Medclick token flux

In summary, there are two tokens: (i) the *social token*, from facebook or google; and (ii) the *medclick user token* from medclick backend. Each medclick user have three ids associated: (i) The medclick ID; (ii) The facebook ID; and (iii) The google ID. When the oauth card is presented to the user, the user logs

in facebook/google, and the bot receives the social token. Then the bot uses the *social token* to make a login request to medclick backend. The medclick backend uses the *social token* to request the oauth provider the google/facebook ID associated with the *social token*. Finally, medclick finds the medclick user with the google/facebook ID associated and returns to the bot application the *medclick user token*.

4.6 Appointment Reminder

This section describes how the bot appointment reminders to the users are implemented. When the scheduled date of an appointment is approaching, the bot sends a notification to the user, reminding the user of the details of the appointment, and asking the user for confirmation if he or she will be attending the appointment (see Figure 25). The usage of notification is important in order to help reducing the number of healthcare that patients miss without previous notice. When a patient misses an appointment without noticing, the slot attributed to the healthcare becomes vacant and cannot be used by other patients. The notifications are typical either done by phone call, or by sending a text message. A systematic review [34] of several studies on the effect of text-based electronic notifications found that patients who received notifications were 23% more likely to attend clinic than those who received no notification.

The usage of chatbot notifications, instead of SMS, has the advantage that the user receives the reminder in the same platform used to schedule the appointment, so there is no context-switch. Furthermore, the patient can directly answer if he or she will be attending the healthcare appointment in the chatbot.

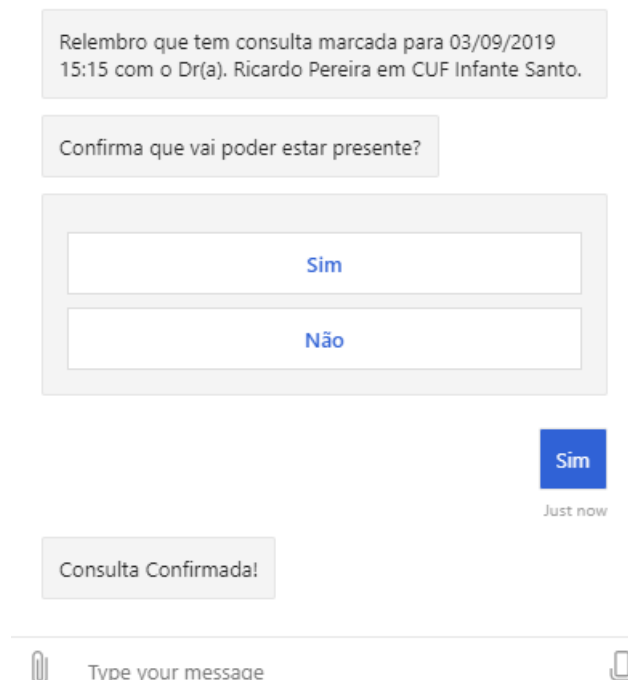


Figure 25 - Appointment reminder sent to user

The notifications were implemented in a way that the server has the initiative to request the bot application to send a notification to the user. This request is triggered when the appointment is three days away. In order to enable the bot to send reminder notifications to the bot users, the Medclick backend was extended. When the Medclick backend sends this request to the bot, it must contain the appointment details (*healthcare* business object) and the information needed for the bot to send the reminder to the associated user. Additionally, the server must know which appointments were scheduled via chatbot. The user attendance answer to the reminder should also be recorded in the Medclick database. These requirements were fulfilled by extending the Medclick backend as seen in Figure 26. Note that each healthcare object was associated with a channel object, in order to know if the appointment was scheduled via chatbot. The healthcare object properties were also extended to record the user answer to the reminder notification. When the user first schedules the appointment, the *healthcare* object is created via Medclick API, and the *bot conversation data* is associated with it. This *bot conversation data* is the data the bot uses to route messages to the correct chat users.

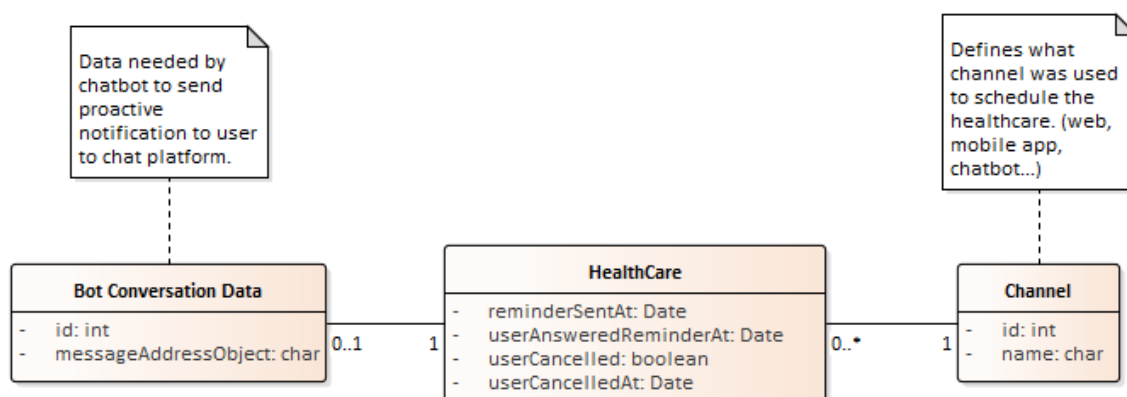


Figure 26 - Medclick backend extensions for notifications

The bot was also extended in order to create a POST endpoint that receives reminder requests from the Medclick server. The full flux of the notification implementation is described in Figure 27.

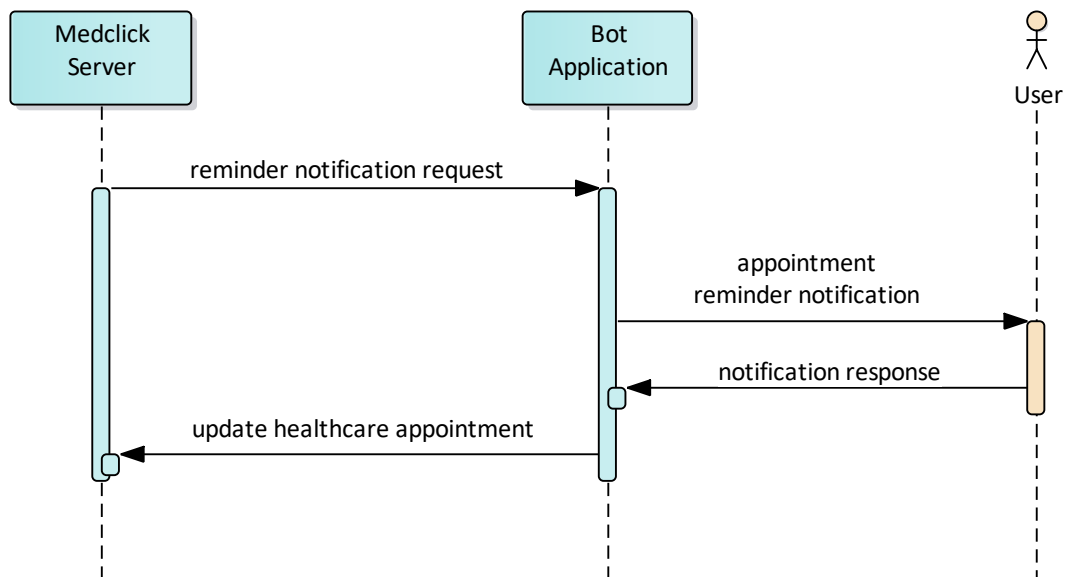


Figure 27 - Appointment notification communication sequence

5. Evaluation

This section discusses the evaluation of the *chatbot use case evaluator* and of the chatbot implementation. The use case of *scheduling a medical appointment* was selected as the more suitable (see section 3) for a chatbot, between the evaluated use cases. It was then implemented as described in section 4. Using the *chatbot use case evaluator*, it was defined that the implementation in a chatbot over a GUI is advantageous. In order to confirm if the UC is indeed advantageous in a chatbot, user tests are performed both in Medclick website (traditional GUI) and using the chatbot in Facebook messenger. It is also important to test the quality of the chatbot itself. The use case might be appropriate for a chatbot, but the bot implementation might not be performant enough.

5.1 Evaluation Process

In this section, it is described the process that enables the evaluation of both the chatbot and the *use case selection process*.

5.1.1 Website and Chatbot Scheduling

Users are asked to schedule an appointment using both the chatbot and the website. Metrics are extracted from logs of both interactions.

The metrics extracted from the website interaction are:

- **Task completion success:** was the user able to perform the desired task?
- **Efficiency:** how long did the interaction take.

The metrics extracted from the bot interaction are:

- **Task completion success:** was the user able to perform the desired task?
- **Efficiency:** how long did the interaction take.
- **Intent detection error rate:** how often did the system misclassified the user intention.
- **Entity detection error rate:** how often did the system filled an entity slot with the wrong value or did not detect that an entity was given.

With the beforementioned metrics, it is possible to directly compare the task completion success and efficiency of the GUI and chatbot interaction. The intent and entity detection error rate are specific to the chatbot, and indicate the quality of the natural language understanding of the bot. Both website and chatbot were adapted to produce logs of the interactions, to enable later extraction of these metrics.

The information given to the user for scheduling both the appointments is:

- Doctor Name: The name of the doctor that the user wants to schedule
- Time Frame: The date when the user desires to visit the doctor.

5.1.2 Questionnaire

In the end of the website and chatbot interaction, users are also requested to answer a questionnaire. The questionnaire uses the questions of the short version of the “User experience questionnaire (UEQ)” [35], in order to evaluate the chatbot user experience (see Figure 28). Participants can rate each item on a 7-point Likert scale. The answers are scaled from -3 (fully agree with negative term) to +3 (fully agree with positive term). The questionnaire aims to evaluate the pragmatic and the hedonic quality of the bot. Pragmatic quality concerns task or goal related quality aspects, while hedonic quality relates to pleasure or fun while using the product. Half the questions concern pragmatic quality and the other half hedonic quality.

obstructive	o o o o o o o	supportive
complicated	o o o o o o o	easy
inefficient	o o o o o o o	efficient
confusing	o o o o o o o	clear
boring	o o o o o o o	exiting
not interesting	o o o o o o o	interesting
conventional	o o o o o o o	inventive
usual	o o o o o o o	leading edge

Figure 28 – Chatbot user experience questions asked in the questionnaire

The use of the User Experience Questionnaire (UEQ) allows to quantify the user experience of the chatbot. The UEQ contains a benchmark that helps to judge how good or bad a measured product is in comparison to other products. The benchmark contains data from 9905 persons that evaluated 246 different interactive products. [36] adapted this benchmark for the short version of the UEQ, based on the same dataset. The proposed scale is the following:

- **Excellent** (In the range of the 10% best results): *Pragmatic Q.* greater than 1.73, *Hedonic Q.* greater than 1.55, *Overall* greater than 1.58.
- **Good** (10% of the results in the benchmark data set are better and 75% of the results are worse): *Pragmatic Q.* between 1.55 and 1.73, *Hedonic Q.* between 1.25 and 1.55, *Overall* between 1.4 and 1.58.
- **Above average** (25% of the results in the benchmark are better than the result for the evaluated product, 50% of the results are worse): *Pragmatic Q.* between 1.15 and 1.54, *Hedonic Q.* between 0.88 and 1.24, *Overall* between 1.02 and 1.39.

- **Below average** (50% of the results in the benchmark are better than the result for the evaluated product, 25% of the results are worse): *Pragmatic Q.* between 0.73 and 1.14, *Hedonic Q.* between 0.57 and 0.87, *Overall* between 0.68 and 1.01.
- **Bad** (In the range of the 25% worst results): *Pragmatic Q.* less than 0.73, *Hedonic Q.* less than 0.57, *Overall* less than 0.68.

Additionally to the UEQ questions, the questionnaire created also contains a question inquiring the tester what the preferred method for scheduling appointments is, based on the two product interactions.

5.1.3 Evaluation Process Summary

In summary, the evaluation process consists on the following steps:

1. User schedules appointment on Medclick website (see Figure 29);
2. User schedules appointment using the chatbot on Facebook Messenger (see Figure 30);
3. User answers the questionnaire;



Figure 29 - Website interface



Figure 30 - Chatbot interface on Facebook Messenger

Figure 31 summarizes the steps of the process and indicates the metrics that can be extracted from each individual step.

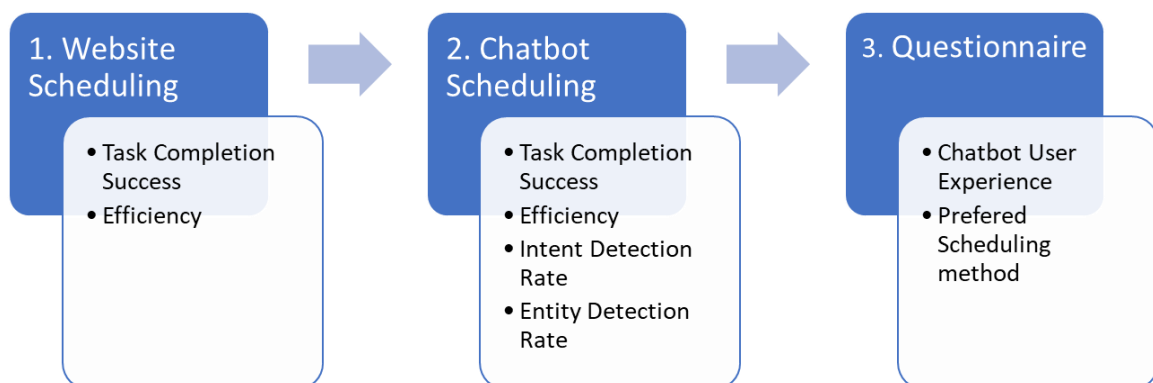


Figure 31 - Evaluation with users process

5.2 Evaluation Results

This section presents the results of applying the evaluation process described in section 5.1. The full process was performed by 10 test users. The demographic data collected indicates that test user age range from 18 to 64 years old (see Figure 32), and the distribution regarding sex can be seen in Figure 33. Furthermore, half the users indicated that were using a chatbot for the first time, while the remaining half already had interacted with a chatbot before.

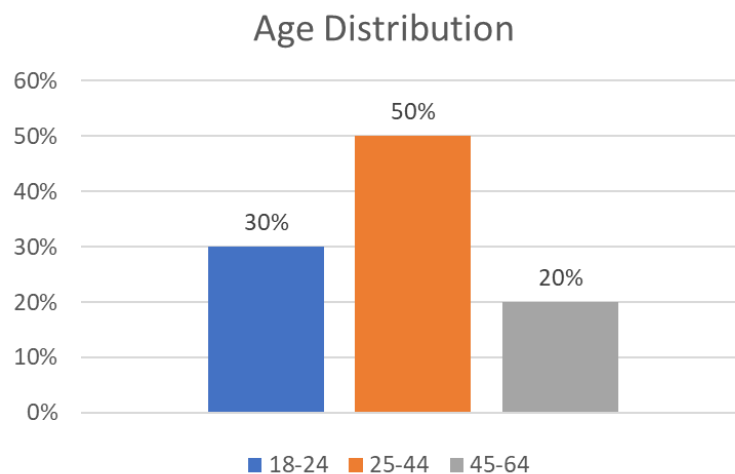


Figure 32 - Age distribution of test users

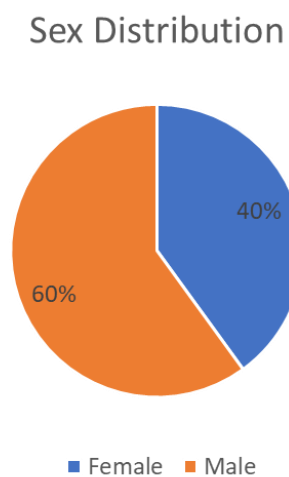


Figure 33 - Sex distribution of test users

The application of the evaluation process yielded the following results.

5.2.1 Chatbot Results

Users were able to complete the scheduling in all chatbot interactions. Furthermore, the bot was able to detect the intention of *scheduling* in the users' utterance and successfully detected and map the *doctor name* and *time* entities in every interaction (see Table 14). The doctor name is considered successfully detected when the first doctor suggested is the one that was asked the user to request. The time entity is considered successfully detected when the bot searches for timeslots in the time frame mentioned by the user and presents the ones that exist in the database in such time frame.

Table 14 - Chatbot user testing results

Task Completion Success		100%
Completion Time (in seconds)	Average	47,8
	Median	45
	Std. Dev.	13,12
Intent Detection Rate		100%
Entity Detection Rate		100%

5.2.2 Website (traditional GUI) and Chatbot Results Comparison

Comparing the average time users took to schedule the appointment in the website and in the chatbot, it is possible to conclude that the chatbot was more efficient, in the sense that users took less time to complete the task (see Table 15). All the users were able to complete the scheduling in both interactions, so the task completion success is the same.

Table 15 - Task completion and efficiency comparison (chatbot and website)

Scheduling Method		Website	Chatbot
Metrics			
Task Completion Success		100%	100%
Completion Time (in seconds)	Average	65,7	47,8
	Median	58	45
	Std. Dev.	20,27	13,12

Analyzing for each individual user test which ones were faster on the bot and in the website, it is concluded that most of the users were faster scheduling the appointment via chatbot than using the website (see Figure 34).

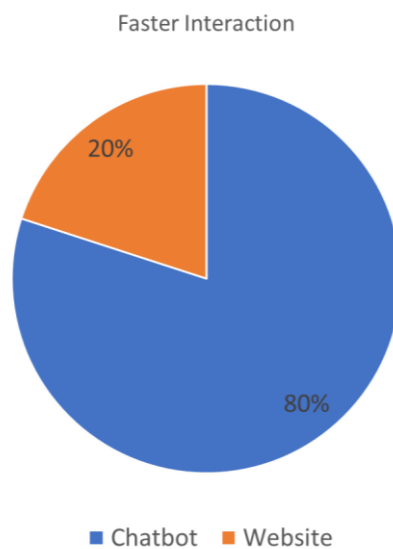


Figure 34 - Comparison of faster method per user test

5.2.3 Questionnaire Results

The results of the answers to the UEQ in the questionnaire can be found in Table 16. Note that the mean value is already in the -3 (most negative) to +3 (most positive) scale.

Table 16 - Short UEQ results

Item	Mean	Variance	Std. Dev.	No.	Negative	Positive	Scale
1	↑ 2,2	0,6	0,8	10	obstructive	supportive	Pragmatic Quality
2	↑ 2,5	0,9	1,0	10	complicated	easy	Pragmatic Quality
3	↑ 2,7	0,2	0,5	10	inefficient	efficient	Pragmatic Quality
4	↑ 2,5	0,5	0,7	10	confusing	clear	Pragmatic Quality
5	↑ 1,8	1,1	1,0	10	boring	exciting	Hedonic Quality
6	↑ 2,3	0,7	0,8	10	not interesting	interesting	Hedonic Quality
7	↑ 1,9	0,5	0,7	10	conventional	inventive	Hedonic Quality
8	↑ 1,8	0,4	0,6	10	usual	leading edge	Hedonic Quality

Considering the averages of the Pragmatic and the Hedonic Quality fields, it is possible to yield the following general results:

- **Pragmatic Quality:** 2,457;
- **Hedonic Quality:** 1,950;
- **Overall:** 2,213.

With these results the benchmark proposed by [36] is applied, as described in section 5.1.2. The chatbot is evaluated as *Excellent* (In the range of the 10% best results) in pragmatic, hedonic and overall quality, as the values obtained for each category are higher than the requirement of the benchmark level.

When asked what the preferred method for scheduling was, most users (70%) chose the chatbot as preferable (see Figure 35).

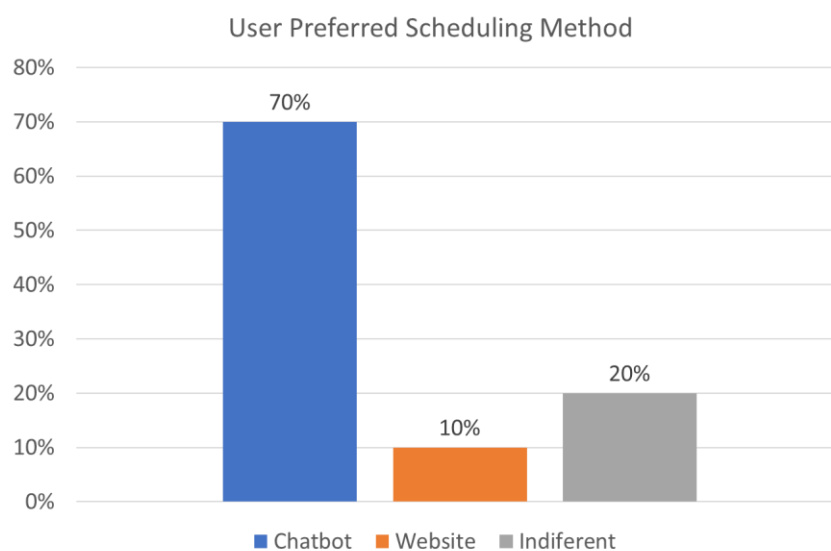


Figure 35 - User preferred scheduling method results

5.3 Evaluation Results Conclusions

The fact that all the users were able to complete the scheduling using the chatbot, and that the UEQ benchmark positions the user experience in the range of the 10% best results, indicates the use case of scheduling a medical appointment as suitable for a chatbot. Furthermore, considering that: (i) 70% of the users answered that prefer to schedule an appointment using the bot instead of a website (GUI), and that (ii) the chatbot interactions were more efficient in 80% of the tests, it is possible to consider the chatbot implementation as advantageous over a traditional graphical user interface.

The suitability of the use case for a chatbot combined with the observable advantages over a GUI, indicates that the *use case selection process* (see section 3.2) selected an appropriate use case for implementation.

6. Conclusion

The heavy usage of chat platforms allied with the developments in natural language understanding create an opportunity for organizations to engage users using dialog agents. Chatbots lie between a human operator and a graphical user interface (GUI) application. Chatbots share the advantage of the more natural interaction of human operators and the automation benefits of GUI applications. Although several advantages can be found in chatbots, it is not obvious how can organizations define *a priori* the use cases that are suitable for a conversational user interface implementation. Evaluating the factors that are required for a viable implementation (general factors), and the factors of the use case that make it beneficial over a GUI application or over a human, it is possible to further understand if a use case is not only viable for a chatbot, but if it is beneficial over a GUI or a human operator. This research proposes a process that using the evaluating factors gathered aims to aid organizations in this analysis.

The application of the use case selection process to the health care domain, indicates scheduling a medical appointment as beneficial to be implemented in a dialog agent. After conduction user tests using the bot implementation of this use case, it was possible to conclude that the chatbot was more efficient when comparing to a traditional GUI (website), and that most users preferred to schedule an appointment using the bot over the website.

6.1 Contributions

This research enables organizations to make more informed decisions regarding dialog agent implementations, offering a systematic process for use case suitability assessment. In the healthcare domain, it was possible to evaluate several use cases using this process.

The major contributions identified are:

- Gathering of evaluating factors that should be considered when implementing a use case in a dialog agent. This analysis resulted in three categories of factors: (i) *general factors*, that indicate the viability of a use case for a chatbot, (ii) *factors over GUI application*, that allow to determine if a conversational interface implementation is advantageous when comparing to a GUI and (iii) *factors over humans*, that allow the assessment of the benefits of implementing a use case in a chatbot over a human operator.
- A use case selection process, that can be used by any organization to evaluate and prioritize use cases for a chatbot implementation, in a systematic way. The process consists of three major steps. In the first step the evaluation factors are applied to each use. In step 2, use cases that do not meet a set of requirements that result of the assessment of the evaluation factors are filtered out. Finally, in the final step, the remaining use cases are ordered, allowing organizations to prioritize the more suitable use cases for implementation in a chatbot;

- Application of the use case selection process to use cases in the healthcare domain, resulting in the conclusion that scheduling medical appointments is a good candidate to be implemented by healthcare organizations in a chatbot;
- A reference chatbot architecture, applied to the healthcare domain, that allows the scheduling of medical appointments. The major components are: (i) the *bot application*, using the Microsoft bot framework, where the dialog management is defined and that interacts with the other components, (ii) The *natural language understanding module*, that gives semantic meaning to the user sentences, by recognizing the intent and entities present in it, and (iii) The medclick API, that provides the business data for scheduling the appointment. The architecture is validated via user testing and can be used as a reference for other implementations.

The secondary contributions identified are:

- Review of the state of the art of chatbots, and the current use cases that are being implemented using dialog agents;
- Review of chatbot and natural language understanding services, comparing aspects such as the performance in intent and entity recognition, number of languages and chat channels supported or the support for integration with existing systems.

6.2 Future work

Future work can be performed in order to further develop and evaluate the use case evaluation process. Regarding the evaluation of the process, it can be further validated by using it to assess other use cases and compare the success of the implementation with the result of the assessment using the process.

The evaluation of the chatbot implemented can also be improved, by adding more variability of entities given (doctor name, time, specialty) and by increasing the number of test users. It would be beneficial to use a corpus of real appointment schedule conversations, in order to train the chatbot natural language understanding module. The usage of such corpus would also enable to further evaluate and improve the intent and entity detection rate of the dialog agent, even though the proposed bot can learn as more interactions are performed.

7. References

- [1] D. Jurafsky and J. Martin, *Speech and Language Processing*, 2nd ed., New Jersey: Prentice Hall, 2008.
- [2] A. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, pp. 433-460, 1950.
- [3] Facebook, "Chatbots 101," [Online]. Available: <https://kore.ai/wp-content/uploads/bots-101-ebook-general.pdf>. [Accessed 20 September 2018].
- [4] A. Shevat, *Designing Bots*, Birmingham: O'Reilly Media, 2017.
- [5] "Safe In Breastfeeding," [Online]. Available: <https://www.safeinbreastfeeding.com/safedrugbot-chatbot-medical-assistant/>. [Accessed 24 October 2018].
- [6] H. Kazi, B. Chowdhry and Z. Memon, "MedChatBot: An UMLS based Chatbot for Medical Students," *International Journal of Computer Applications*, vol. 55, no. 17, 2012.
- [7] "A Chatbot for Psychiatric Counseling in Mental Healthcare Service Based on Emotional Dialogue Analysis and Sentence Generation," in *International Conference on Mobile Data Management*, 2017.
- [8] B. Heater, "Babylon Health to power NHS 111 with 'AI triage' bot," 5 January 2017. [Online]. Available: <https://www.digitalhealth.net/2017/01/babylon-health-to-power-nhs-111-with-ai-triage-bot/>. [Accessed 12 October 2018].
- [9] Florence, "Florence (Beta) - Your health assistant," PACT Care BV, [Online]. Available: <https://www.florence.chat/>. [Accessed 11 December 2018].
- [10] J. Hill, R. Ford and I. Farreras, "Real conversations with artificial intelligence: A comparison between human-human online conversations and human-chatbot conversations," *Computers in Human Behavior*, vol. 49, pp. 245-250, 2015.
- [11] S. Janarthnam, *Hands-On Chatbots and Conversational UI Development*, Packt Publishing, 2017.
- [12] Accenture, "Chatbots in Costumer Service (white paper)," 2016.

- [13] J. Ask, M. Facemire and A. Hogan, "The State Of Chatbots," 2016.
- [14] J. Hirschberg and C. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261-266, 2015.
- [15] J. Allen, Natural language understanding, Benjamin/Cummings Publishing, 1988.
- [16] E. Ovchinnikova, Integration of World Knowledge for Natural Language Understanding, Atlantis Press, 2012.
- [17] D. Jurafsky and J. H. Martin, Speech and Language Processing (Third Edition draft), 2018.
- [18] X. Zhang and H. Wang, "A Joint Model of Intent Determination and Slot Filling for Spoken Language Understanding," *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 2993-2999 , 2016.
- [19] A. Raman and W. H. Tok, A Developer's Guide to Building AI Applications, O'Reilly Media, 2018.
- [20] IBM, "Defining intents," 27 November 2018. [Online]. Available: <https://console.bluemix.net/docs/services/assistant/intents.html#defining-intents>. [Accessed 30 November 2018].
- [21] IBM, "Defining entities," 15 November 2018. [Online]. Available: <https://console.bluemix.net/docs/services/assistant/entities.html#defining-entities>. [Accessed 30 November 2018].
- [22] Dialogflow, "Input and output contexts," [Online]. Available: <https://dialogflow.com/docs/contexts/input-output-contexts>. [Accessed 3 October 2018].
- [23] Dialogflow, "Follow-up intents," [Online]. Available: <https://dialogflow.com/docs/contexts/follow-up-intents>. [Accessed 2 October 2018].
- [24] Dialogflow, "Events Overview," [Online]. Available: <https://dialogflow.com/docs/events>. [Accessed 2 October 2018].
- [25] Dialogflow, "Fulfillment Overview," [Online]. Available: <https://dialogflow.com/docs/fulfillment>. [Accessed 2 October 2018].

- [26] IBM, "Dialog overview," 30 November 2018. [Online]. Available: <https://console.bluemix.net/docs/services/assistant/dialog-overview.html#dialog-overview>. [Accessed 27 November 2018].
- [27] IBM, "About," 6 November 2018. [Online]. Available: <https://github.com/IBM-Bluemix-Docs/assistant/commits/master/index.md>. [Accessed 30 November 2018].
- [28] IBM, "Making programmatic calls from a dialog node," 27 November 2018. [Online]. Available: <https://console.bluemix.net/docs/services/assistant/dialog-actions.html#dialog-actions>. [Accessed 30 November 2018].
- [29] D. Braun, A. Mendez, F. Matthes and M. Langen, "Evaluating Natural Language Understanding Services for Conversational Question Answering Systems," in *Proceedings of the SIGDIAL*, 2017.
- [30] R. Sengupta and S. Lakshman, "Conversational Chatbots – Let's chat," June 2017. [Online]. Available: <https://www2.deloitte.com/content/dam/Deloitte/in/Documents/strategy/in-strategy-innovation-conversational-chatbots-lets-chat-final-report-noexp.pdf>. [Accessed 25 September 2018].
- [31] L. Klopfenstein, S. Delpriori, S. Malatini and A. Bogliolo, "The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms," in *Proceedings of the 2017 Conference on Designing Interactive Systems*, 2017.
- [32] E. Balogh, B. Miller and J. Ball, "The Diagnostic Process," in *Improving Diagnosis in Health Care*, The National Academies Press, Washington, DC, 2015, pp. 31-69.
- [33] P. Christen, "A Comparison of Personal Name Matching: Techniques and Practical Issues," *ICDMW*, pp. 290-294, 2006.
- [34] D. Robotham, S. Satkunanathan, J. Reynolds, D. Stahl and T. Wykes, "Using digital notifications to improve attendance in clinic: systematic review and meta-analysis," *BMJ Open*, 2016.
- [35] B. Laugwitz, T. Held and M. Schrepp, "Construction and Evaluation of a User Experience Questionnaire," *HCI and Usability for Education and*, vol. 5298, pp. 63-76, 2008.
- [36] A. Hinderks, M. Schrepp and J. Thomaschewski, "A Benchmark for the Short Version of the User Experience Questionnaire," *Proceedings of the 14th International Conference on Web Information Systems and Technologies*, pp. 373-377, 2018.