Just.Chat – a platform for processing information to be used in chatbots

Maria João de Melo Madeiras Pires Pereira

Thesis to obtain the Master of Science Degree in Information Systems and Computer Engineering

Examination Committee

Chairman: Prof. Ernesto José Marques Morgado
Supervisor: Profª Maria Luisa Torres Ribeiro Marques da Silva Coheur
Member of the Committee: Prof. Carlos António Roque Martinho

June 2013
Acknowledgements

Firstly, I want to show my extreme gratitude before my advisor Prof. Luisa Coheur. For all the knowledge, ideas and discussions shared during this journey. But most of all, for the human bound established; I thank you for all the support and confidence both in this work and in my person.

Next I want to thank my colleagues from L²F, Sérgio Curto, Pedro Fialho, Ana Cristina and Pedro Mota for the resources and knowledge shared and, especially, the readiness in lending an helping hand whenever necessary.

Also, to my friends and colleagues Pedro Sousa, João Amaro, Carlos Simões, Tiago Castelo, Tiago Francisco, Diogo Lacerda, Diogo Henríques, Tiago Castanheira, Daniela Borges, André Santos, Júlia Cardoso, Marta Martins, Hugo Guimarães and Luisa Pereira for all the cooperation during the different tests made, friendship and support.

To my closest persons (you know who you are!), either family or friends, even the ones that are not between us anymore, my sincere thanks as you all guided me to where I'm today.

A small mention to my dog Pantuфа, that has kept me company in this final phase of writing.

To my boyfriend Pedro, despite our discussions about the unnecessary of thanking, I want to thank you... because I want, because of your help in this work, because of your help when this same work was despairing, because of your friendship, because of your love... you were indeed my "rock". Love you with all my heart.

At last, to my parents and sister. Your appreciation for knowledge kept me pushing to do better and evolve as a student, your unconditional love made me evolve as a person and to who I'm today. I sincerely love you three with all my heart and cannot thank enough for all you've done for me through the years; I dedicate this thesis for you as a token of my gratitude and love.

Lisboa, May 2013

Maria João de Melo Madeiras Pires Pereira
To my family and friends.
Resumo

O número de chatbots que pode ser encontrado na web aumenta de dia para dia e, com estes, também os recursos disponibilizados pela sua comunidade, os quais podem ser utilizados na criação de novos chatbots.

Nesta tese desenvolvemos a plataforma Just.Chat. Esta tem por objectivo, usando interacções de chatbots, ajudar ao desenvolvimento de bases de conhecimento, quer de raiz ou para complementar uma já criada, sendo que estas seriam usadas num chatbot.

Assim, o primeiro passo foi a construção do *Chat corpus*, obtido através da junção de diferentes interacções (algumas escritas manualmente) de chatbots existentes. Seguiu-se o desenvolvimento da plataforma Just.Chat, a qual fornece três filtros diferentes para processar as interacções reunidas. Cada um destes filtros tem por objectivo resolver um problema diferente que poderia advir ao se adicionar, em bruto, o *Chat corpus* a um chatbot. O primeiro filtro é responsável por descartar interacções que possam sobrepor outras pertencentes a uma base de conhecimento previamente criada; o segundo filtro identifica questões e, de entre estas, quais as que correspondem a questões pessoais cuja resposta o utilizador poderá querer customizar de acordo com o perfil do chatbot em causa; por último, o terceiro filtro lida com interacções que possam conter termos ou tópicos que o utilizador não queira presentes no seu chatbot. Todas estas interacções filtradas são postas de parte de forma a que possam ser revistas e potencialmente adicionadas a uma base de conhecimento.

Todos estes filtros são individualmente testados, assim como o Just.Chat, o qual é usado para enriquecer a base de conhecimento do Edgar, um chatbot mordomo especializado no palácio de Monserrate. A adição de novas interacções, processadas pela plataforma Just.Chat, conduziu a um aumento de 24% nas respostas dadas pelo Edgar a interacções colocadas, em Monserrate, por utilizadores reais.
Abstract

The number of chatbots that can be found in the web increases everyday and, alongside these, the amount of resources provided by the chatbot’s community, which can be used to build new chatbots.

In this thesis we target to create a platform, Just.Chat, that helps people either to create the knowledge base of a chatbot from scratch or to enrich a previously defined one, by using available chatbots’ interactions.

We start by building the Chat corpus, obtained from different sets of interactions from existing chatbots (some manually crafted). Then, we develop Just.Chat, which provides three different filters to automatically process these interactions, corresponding each filter to a distinct problem that could rise from crudely incorporating the Chat corpus into a chatbot. The first filter discards interactions overlapping others previously scripted in the chatbot knowledge base; the second filter is responsible for identifying questions and, among these, personal questions that the user may want to customize according to the chatbot’s character in hands; finally, the third filter deals with interactions containing terms or topics that the user does not want his/her chatbot to use. All the filtered interactions are put aside, and is given to the user the opportunity to review and incorporate them in the chatbot’s knowledge base.

All these filters are individually evaluated, as well as Just.Chat, which is used to enrich the knowledge base of Edgar, a butler that has Monserrate’s palace as its field of expertise. With the addition of the new interactions, processed by the Just.Chat platform, Edgar is able to properly deal with more 24% of interactions, considering a corpus of interactions built from real users in Monserrate.
Palavras Chave

chatbots
criação de bases de conhecimento
filtragem de interacções
identificação de perguntas pessoais
expansão de tópicos

Keywords

chatbot systems
chatbot’s knowledge base creation
filtering interactions
personal questions identification
topic expansion
1 Introduction

1.1 Goals .................................................. 2
1.2 Contributions ........................................ 3
1.3 Document Structure .................................. 4

2 Related Work ............................................. 5

2.1 Historical Overview .................................. 5
2.1.1 Turing Test and Loebner Prize .................. 5
2.1.2 The term "chatbot" ................................ 7
2.1.3 Chatbots throughout the years .................. 8
2.2 Building chatbots ..................................... 11
2.2.1 Existing platforms ................................ 12
2.2.2 Scripting .......................................... 13
2.2.3 Learning ........................................... 14
2.3 Towards the Illusion of Intelligence ................. 16
2.3.1 Personality ........................................ 16
2.3.2 Directing conversation ............................ 17
2.3.3 Small talk ......................................... 18
2.3.4 Fail like a human .................................. 18
2.4 Edgar ............................................... 19
### Table of Contents

2.4.1 Edgar’s Corpora .................................................. 19
2.4.2 Edgar’s understanding component ............................ 20
2.5 Summary .................................................................. 22

3 Building and analyzing the Chat corpus .......................... 23
   3.1 Chatbot’s corpora: knowledge sources ....................... 23
   3.2 The Chat corpus format .......................................... 25
   3.3 From the knowledge sources to the Chat corpus .......... 26
   3.4 Chat corpus statistics ............................................ 31
   3.5 Analyzing the Chat corpus .................................... 32
      3.5.1 Domain Conflicts ........................................... 32
      3.5.2 Personal Customizable Information .................... 32
      3.5.3 Unproper vocabulary ..................................... 33
   3.6 Summary ................................................................ 33

4 The Just.Chat platform .............................................. 35
   4.1 General overview of the Just.Chat platform ............... 35
   4.2 Input/Output ...................................................... 36
   4.3 Configurations ................................................... 38
   4.4 Example of a Just.Chat run ................................... 39
   4.5 Summary ................................................................ 42

5 Filters ........................................................................ 43
   5.1 Domain Filter ..................................................... 43
   5.2 Personal Filter .................................................... 46
      5.2.1 Identification of Questions ............................... 47
5.2.2 Classification of Personal Questions ........................................ 49
  5.2.2.1 Collecting: Personal/Impersonal questions ......................... 49
  5.2.2.2 Feature Engineering and Handwritten Rules ....................... 52
5.3 Blacklist Filter ................................................................. 53
   5.3.1 Filtering undesired words ............................................... 53
   5.3.2 Filtering undesired topics ............................................ 55
5.4 Summary .............................................................................. 57

6 Evaluation .............................................................................. 59

6.1 Evaluation Metrics .............................................................. 59
6.2 Evaluating the Domain Filter ............................................... 60
   6.2.1 Testing with the Chat corpus ........................................... 60
   6.2.2 Testing with real user interactions ................................... 62
6.3 Evaluating the Personal Filter .............................................. 63
   6.3.1 Identification of Questions .............................................. 64
      6.3.1.1 Preliminary evaluation ............................................. 64
      6.3.1.2 Final evaluation ................................................... 66
   6.3.2 PERSONAL and IMPERSONAL question classification ........... 67
      6.3.2.1 Preliminary evaluation ........................................... 67
      6.3.2.2 Evaluation with sent labeled data .............................. 70
      6.3.2.3 Final evaluation .................................................. 71
6.4 Evaluating the Blacklist Filter .............................................. 74
6.5 Parlance Improvements when using the Chat corpus ................. 77
   6.5.1 From the Chat corpus to the Test corpus ............................ 78
   6.5.2 Changes in Edgar answers with the inclusion of the Test corpus . 79
6.5.3 Answers quality .................................................. 81

7 Conclusions ............................................................ 85
  7.1 Main Conclusions .................................................. 85
  7.2 Future Work ....................................................... 87

Bibliography ............................................................ 92

A Edgar Schema ........................................................ 93

B Full evaluation tables .............................................. 95

C Questionnaire ........................................................ 97
List of Figures

2.1 Original imitation game ........................................ 6
2.2 Final imitation game ........................................ 6
2.3 Positive feedback loop ........................................ 15
3.1 Steps to obtain Just.Chat’s input .............................. 26
4.1 Just.Chat architecture .......................................... 36
5.1 Question identification diagram ............................... 50
6.1 Excerpt from the questionnaire ............................... 82
List of Tables

3.1 Statistics from the *Chat corpus* .................................................. 31

6.1 True/false positives/negatives ......................................................... 60
6.2 Domain filtering results (with samples from the *Chat corpus*) .......... 62
6.3 Domain filtering results (with real user interactions) ....................... 62
6.4 Correct matches in the filterings .................................................... 63
6.5 Evaluation results for the set constructed with the sent information ....... 65
6.6 Rules used in the identification of questions ...................................... 65
6.7 Evaluation results without the ‘?’ character ..................................... 65
6.8 Rules used in the identification of questions without the ‘?’ character ... 65
6.9 Final results from evaluation with Chatbot’s Corpora ......................... 66
6.10 Final results: rules used ............................................................... 67
6.11 Mean of the 10-fold cross-validation with single features .................. 68
6.12 Mean of the 10-fold cross-validation with features combination .......... 70
6.13 Classification results using sent labeled data as test ......................... 70
6.14 Final classification results using the 173 identified questions ............. 72
6.15 Classification results: relation between predicted and actual classes .... 73
6.16 Topic expansion results ............................................................. 74
6.17 Statistics from the interactions processed by Just.Chat ..................... 78
6.18 Statistics from the interactions filtered by Just.Chat ......................... 78
Listings

A.1 Edgar Schema ......................................................... 93
Since Turing posed the question "can machines think?" (38) the quest for developing "thinking" programs has become the Holy Grail for the Artificial Intelligence community. During this quest much work has been done towards this goal by the chatbot’s community.

Chatbot, a term coined by Mauldin (24), is a system that "seek to mimic conversation rather than understanding it" (31). Chatbots intend to pass as human, being concerned in handling phatic communication and providing plausible responses, rather than focusing in task-talk or providing effective information (26). Since Eliza, the first chatbot, the number of this type of systems has been increasing at a dizzying pace, being applied to a panoply of applications like e-commerce or learning environments. Through the years of developing and perfecting chatbots, different ideas and technologies were explored in the chatbots’ community. However, despite these advances, the basis for the creation of chatbots rests in pre-written pattern-matching templates, and in the exploration of large stores of prepared small talk responses, although more complex architectures based on learning were also proposed.

As chatbots developers generally want their systems with certain characteristics (like a name, personality or attitude1) in order to develop a chatbot the most common is to start by scripting its knowledge base, a database with the desired matches for inputs and corresponding answers; in fact nowadays there are even persons that are being hired to do such job2. So, one "only" needs to think about a character, and enrich its knowledge base with possible interactions. Even better, many platforms already provide predefined interactions, which can be adapted according to the chatbot character. This can be extremely useful as, considering Zipf’s law (42), a program that receives a certain input has a non zero probability of having the same input entered later and, thus, by looking at sentences that people usually pose to chatbots, one can track patterns for which a specific reply was not created (or rephrase an existing answer).

---

1http://www.chatbots.org/ai_zone/viewthread/492/
2http://www.chatbots.org/ai_zone/viewthread/1160/
Similar with what is done with IRIS (4), which uses Movie-DiC (a dialog corpus extract from movies scripts (3)) as its knowledge base, we found that it would be equally viable to use all these interactions provided by the chatbot’s knowledge base to create a chatbot’s corpus, instead of having to write it from scratch. However, even if these interactions are used, there is the need to go through the tiresome process of analyzing and rewriting them; such is a necessity as we could observe that roughly including these interactions could pose some problems. The first problem we have identified was having overrides of information that could have already been scripted, that is, encountering conflicts between the gathered interactions and an already created chatbot knowledge base. The second problem is the possible existence of personal questions, as these should be customized accordingly to the chatbot’s character. Finally, it is also likely that these interactions may contain undesired words or topics. This way, the following question arises: "how to automatically use all this information to build a chatbot knowledge base, but minimizing these problems?". Our researches lead to no answers: we did not find any approach to cover this problem. The lack of an automatic processing tool to address this issue lead to the creation of our platform Just.Chat. It aims at identifying interactions containing the previously mentioned problems and having them filtered so that they can be manually treated, while the remaining interactions can be used for constructing or improving a chatbot’s knowledge base.

Finally, we had as an initial motivation for this thesis the improvement of the parliance skills of Edgar, a chatbot-like system developed under the project FalaComigo. Edgar’s conversation topics are much narrowed, as it was mostly prepared for covering inputs focused on its field of expertise (the Monserrate’s palace); however, it could be advisable to enrich its way of properly answering to inputs in other domains and dealing with small talk, as to make it more credible and human. This way, we found that it would be interesting to equip Edgar with some of these interactions, passed through Just.Chat, and see if such lead to any improvements in its capacity of answering.

### 1.1 Goals

Bearing in mind the points previously discussed, we had the following goals for this thesis:

- Survey the chatbot’s community related works;
1.2. CONTRIBUTIONS

- Collect interactions from the chatbot’s community, analyze and parse them. This resulted in a corpus called *Chat corpus*;
- Implement Just.Chat, a platform for processing the *Chat corpus* into data usable by chatbots. The platform:
  - Avoids overlaps between the received information and the one that can have already been developed for a chatbot knowledge base;
  - Extracts interactions that the user should customize accordingly to the defined chatbot’s profile;
  - Filters undesired words, expressions or topics.
- Evaluate the platform:
  - Do an intrinsic evaluation, where its modules are tested individually;
  - Do an extrinsic evaluation, using the attained, processed, interactions in Edgar, testing possible improvements.

1.2 Contributions

The contributions of this thesis reside on the following topics:

- Survey the different ideas explored by the chatbots’ community and the various different resources, including tools and corpora, that it provides. These lead to the creation of a paper entitled "Towards effective human-computer communication: contributions from the chatbots’ community" that was submitted to the Artificial Intelligence Review (AIRE) journal;
- Creation of the *Chat corpus* with round 78000 interactions pairs found in the chatbot’s community;
- Proposal and implementation of Just.Chat, a solution for automatically transform the *Chat corpus* into a corpus usable for the creation of a chatbot, that:
  - Combines several natural language techniques, used in Edgar, to match interactions against ones previously scripted, changing their answers to prevent contradictions;
- Identifies customizable questions by combining rules and a machine-learning classifier; 
  Extend given topics, using WordNet, and test if the terms derived from these are 
  contained within interactions.

- Test the behavior of the implemented techniques and how well they address the problem 
  for which they were created;

- Evaluate the platform in of a real case scenario where interactions parsed through Just.Chat 
  were added to an existing chatbot-like system.

1.3 Document Structure

This document is structured as follows: in Chapter 2 we present an overview of chatbots 
alongside their main resources and ideas; in Chapter 3 we describe how we built the Chat corpus; 
Just.Chat’s architecture and main modules are exposed, respectively, in Chapter 4 and 5. We 
evaluate this platform and Edgar’s improvements in Chapter 6, ending with the conclusions and 
future work in Chapter 7.
In this Chapter we review the main works done in the chatbot's community. We start in Section 2.1 by presenting a brief historical overview. Then, in Section 2.2 we discuss chatbot's platforms and how to enrich them, and in Section 2.3 we summarize the main "tricks" towards the "illusion of intelligence". In Section 2.4 we present Edgar, a chatbot-like system developed by L2f under the project FalaComigo. Finally, we end with a brief summary of the main contents discussed through the Chapter in Section 2.5.

2.1 Historical Overview

This Section begins with a description of the Turing Test and Loebner prize in 2.1.1, given that both are landmarks when talking about chatbots. Because passing the Turing Test is seen by many, as the chatbots' ultimate goal since it would prove their intelligence and capability of thinking. Follows the different definitions associated to chatbots as well a distinction between these and dialog systems; some other systems that derive from the classical notion and definition of chatbots are also depicted in the Section. Lastly, we dedicate Section 2.1.3 for an overview of the different chatbot's systems made through the years and that have distinguished themselves in the chatbot's community.

2.1.1 Turing Test and Loebner Prize

It all began in 1950, with the British mathematician Alan Turing question "can machines think?" (38) and the proposal of a way of testing it: the imitation game (now known as the Turing Test). The original imitation game, Figure 2.1, is played by a man (A), a woman (B) and an interrogator (C) whose objective is to guess the sex of the other two players (A and B). Turing proposed substituting one of the players (A or B) by a machine and playing the same game, Figure 2.2. In this version if the interrogator wrongly identifies who is the human it means
that the machine can think.

Turing’s line of thinking arose much controversy. A large amount of arguments were presented as to why machines are not, and will never be, capable of thinking. Other objections focused on the plausibility of the imitation game’s results.

In fact, Turing never imposed any limitation on how the machine should be modeled to assembly the human cognitive processes, which means machines that do not mimic it can also pass the test (29). However, Turing seems to give preference to model approaches which stick the most closely possible to the human ones. Such can be seen from his suggestion of a program simulating a child’s mind instead of an adult’s one, so that it could be "educated" and grow in a "human-like" way.

Despite all of the Turing Test drawbacks, we cannot say Turing’s work was not notable and important for the AI community, especially when considering how visionary it was for that time. Loebner understood the importance of Turing’s work and stipulated a 100,000$ reward for the first person whose program could pass the Turing test. Therefore, in 1991, the first Loebner Prize Contest took place at Boston’s Computer Museum (11).

Since then, the competition has been held annually in the quest of finding the thinking computer. According to Loebner one of the goals of the competition is to advance the field of computer science¹, but such has been brought into question as the competition still remains to be won and no further progresses in developing a thinking machine have been registered. Like written by Hutchens (18) "perhaps the most important contribution of the Loebner contest

---

¹http://www.loebner.net/Prizef/In-response.html
is the insight it provides into the psychology of communication – it makes us aware of how little our understanding of conversation lies in what is said⁶. Besides the Loebner prize, several other competitions emerged, such as the Chatterbox Challenge², which started in 2001, or, more recently, the Chatbot Battles³.

### 2.1.2 The term "chatbot"

The term "chatbot" (chat + robot) was coined by Mauldin (24) in order to define the systems that had the goal of passing the Turing Test and, thus, could be said to "think". However, in current parlance, we have that the term "chatbot" and "dialog system" are used indiscriminately. Both for clarification and for avoiding misunderstandings between the two terms, in this document we distinguish them by following Schumaker (31), who defines chatbots as more relaxed dialog systems that "seek to mimic conversation rather understanding it". Also, contrary to dialog systems, chatbots are supposed to freely engage conversation about any subject, making them "entertaining in a large variety of conversational topic settings" (31).

There is a panoply of other terms that are used as synonyms of "dialog system" and "chatbot", most of them being covered in Chatbots.org⁴, where a list of more than 150 of such terms can be found. While some of the terms of the list are truly synonyms of chatbot, as they do not have any additional functionality, some others are a little more complex like Embodied Conversational Agents (ECA’s) or Virtual People. In both systems, chatbots are incorporated as a dialogue module. ECA’s are more complex chatbots systems, as they take the paradigm of mimicking a human being a step further, by having a body that should be "lifelike or believable in its actions and its reactions to human users" (8). So, besides the conversational module such systems should give visual/physical signals that indicate the state of the conversation. Moreover some work and research explore the recognition and interpretation of the human interlocutor’s gestures. This way a new channel of communication is established going beyond the natural language processing of the input (written or spoken). Virtual people are somewhat a type of Virtual Agents, "autonomous, graphically embodied agents in an interactive, 2D or 3D virtual environment with the ability of interacting with the environment, other agents and human users"⁵.

---

²http://www.chatterboxchallenge.com
³http://www.chatbotbattles.com
⁴http://www.chatbots.org/synonyms/
⁵http://iva07.ntua.gr/
These systems can usually be found in virtual worlds (like second life) or within pc-games\(^6\). Some of these applications are "complex and dynamic social" (40), multi-agent environments; this way, the human simulation, demands a more complex social behavior (16).

2.1.3 Chatbots throughout the years

Being one of the most known conversation systems, Weizenbaum’s ELIZA (41) was the first chatbot created. Even before the term chatbot appeared and spread, becoming a field of study for many researchers, Weizenbaum decided to create a program able to establish a conversation with human beings, simulating it was one too. In 1966, the result of his attempt at making a mean of communication man/machine came to public under the appearance of a Rogerian psychotherapist called ELIZA. ELIZA’s conversational model was based in the rephrasing of input sentences, when these matched a set of pre-defined rules (user: You are entitled to your opinion., ELIZA: What makes you think I’m entitled to my opinion?); it was also based in content-free remarks in the absence of a matching (textbfuser: It bothers me just to be around people., ELIZA: Please go on.).

ELIZA’s simple algorithm exceeded the expectations, given that many people, when using the program, believed they were talking with another human. Weizenbaum was taken aback by some aspects of this deception (17). What shocked him most was the fact that people actually believed that the program understood their problems\(^7\). Perceiving ELIZA as a thread, Weizenbaum wrote "Computer Power and Human Reason" with the aim of attacking the AI field and educating uninformed persons about computers (20).

The illusion created may be explained by the resort to three main tricks (24):

- Usage of questions to incite the user participation and make him keep the conversation with little contribution from the program;
- Inclusion of the user’s string in questions help maintaining an illusion of understanding;
- Rogerian mode gives cover for the program as it draws a conversation where the program never contradicts itself, as it never makes affirmations, and is free to know nothing or little about the real world without it being suspicious.

\(^6\)http://www.chatbots.org/virtual_people
\(^7\)http://www.alicebot.org/articles/wallace/eliza.html
2.1. HISTORICAL OVERVIEW

So, besides being the first approach to the problem "can machines think?", posed by Turing, ELIZA showed how a program, without resorting to the most complex algorithm, can cause such a huge impression by the mere illusion of understanding and impersonating of a specific role.

ELIZA is still one of the most widely known applications in AI and it is at the base of a great number of chatbots, including PARRY, its "successor". Following a very similar architecture to that of ELIZA, PARRY appeared in 1971 by the hands of Kenneth Colby, simulating a paranoid mental patient (29). An interesting comparison between PARRY and ELIZA was made by Güzeldere and Franchi\(^8\). They stated that "PARRY's strategy is somewhat the reverse of ELIZA's" as one simulates the doctor, distant and without personality traces, and the other a paranoid patient which states its anxieties. Though PARRY's architecture is somewhat similar to that of ELIZA, it has some extras that its predecessor did not have. The main difference resides in the fact that PARRY has knowledge of the conversation and is also doted of a state of mind. The combination of these two factors affect the output as it becomes a function not only of the input, but also of PARRY's beliefs, desires and intentions. Mauldin (24) also summarized a few tricks to which PARRY resort:

- Admitting ignorance;
- Changing the conversation topic by launching new subjects;
- Inclusion of small stories about the Mafia which are introduced throughout the conversation.

After Colby gathered transcripts of interviews between psychiatrists, normal patients and his program, he presented the results to another group of psychiatrists. He asked this group if they could guess in what transcripts the interviewed was a human and in which ones it was a program. The psychiatrist could not do better than randomly guessing.

It is possible to conclude from these results that the emotional side is easier to imitate than the intellectual one (20). However, one of the main criticisms PARRY received was of not being more than an imitation, an illusion. This critic became one of the mainly posed to chatbots systems, denouncing the incapability of such systems modeling a real person. In his response to this specific issue, Colby summarizes the problem essence: "A model of a paranoid patient is a

\(^8\)http://www.stanford.edu/group/SHR/4-2/text/dialogues.html
model of being paranoid, being a patient, and being a person. PARRY does reasonably well in the first two of these "beings". It fails in the third because of limited knowledge. (…) PARRY is not the real thing; it is a model, a simulation, an imitation, a mind-like artifact, an automaton, synthetic and artificial.”

Moving back to the Loebner contests, its first winner, in 1991, was Joseph Weintraub’s PC-Therapist program, based on Eliza, an achievement that he repeated three more times in the following four years. Since then, many chatbots, with different goals, emerged from the competing systems, including Jabberwacky⁹, created by Rollo Carpenter, which was released to public in 1997 (2), which has entered in four Loebner contests and always stood in the top three, targeted not to pass as a human but as an individual, a specific person with a specific personality (7).

The idea of having a chatbot that is a result of the knowledge gathered from the playing is the key beyond the way Jabberwacky works. For instance, in 2005, Jabberwacky impersonated George, an entity created by Rollo Carpenter "in a smallish number of hours, just by chatting". More recently, a new chatbot under the name of Cleverbot, also created by Rollo Carpenter, has become available to the public¹⁰. Considering the similarities between Cleverbot and Jabberwacky, and given that both systems have the same creator, the odds point to Cleverbot as being a new improved version of Jabberwacky. Cleverbot's popularity lead to the creation of an avatar called Evie http://www.existor.com/ and a mobile application¹¹. Evie, the Expressive Virtual Interaction Entity, is an updated version of Cleverbot with the possibility of receiving both written or verbal inputs. Its animated avatar is also capable of displaying some human emotions. Some videos of Evie can be found in Cleverbot’s site¹². Recently, Cleverbot was used to co-write a short film¹³.

Another competing system in the Loebner contests that needs to be highlighted, as it was responsible for boosting the chatbots field, is the Alice. It was invented in 1995 by Dr. Richard Wallace, as a result of gathering default responses by its creator (32), and won the Loebner competitions in 2000, 2001 and 2004¹⁴. Even though it is a modern Eliza, it differs from it by

---

⁹http://www.jabberwacky.com/
¹⁰http://cleverbot.com/
¹¹http://www.cleverbot.com/app
¹²http://www.cleverbot.com/cleverthem
¹³http://www.youtube.com/watch?v=QxHA7sys5Ms: the movie "Do you love me", directed by Chris R. Wilson
¹⁴http://www.loebner.net/Prizef/loebner-prize.html
2.2. **BUILDING CHATBOTS**

not playing a specific role, but by trying to reflect a human in general. The propose of ALICE’s creation was then to keep it talking as long as possible without the users realizing that they were not talking to a machine, without sticking to a specific topic or role. All the winnings by themselves already show ALICE’s potential, however this becomes clearer when observing that a large number of other works employ ALICE’s technology. In fact, associated with ALICE there is a panoply of resources that have been widely used by the chatbots community. For instance, the chatbot hosting service Pandorabots, with which it is possible for one to create his/her own chatbot, a pandorobot, is due to ALICE. Pandorabots is widely used (it represents the largest chatbot community on the internet) not only because of the simplicity of creating and hosting chatbots, but also because of the multimodal facilities the site offers like faces or speech.

As many different resources are available today, chatbots become a field in large expansion, as their technology can be used by anyone, as the most important requirement is to be creative. Due to this, chatbots can be found in a huge diversity of services, including e-commerce. For example, the chatbot ELBOT, a regular winner in chatbots contests, is an ALICE type program used on e-commerce sites like Ikea’s. Moreover, there is also a large quantity of studies in the chatbot field, going through retraining chatbots to extend their knowledge base, to adding them emotions or personality, or using them as educational systems.

The following (impressive) numbers, collected in March 2013, definitely help to give a precise idea of the chatbots community size: the Alice A. I. Foundation reports over 15,000 visitors every day and Pandorabots hosting service declares to have 185,000 bot masters, which have created more than 221,000 chatbots.

### 2.2 Building chatbots

Behind each chatbot there is a development platform. These are typically based on a language that allows to define the chatbot knowledge base and an engine capable of mapping a user utterance into the most appropriate answer. This knowledge bases are usually hand crafted by the bot developer, but some platforms provide some sort of learning environment. In this section we survey the most successful platforms, and the scripting and learning processes.

---

15 [http://www.pandorabots.com](http://www.pandorabots.com)
2.2.1 Existing platforms

An impressive collection of ELIZA’s can be currently found in the web and some of these can be customised. For instance, Chatbot-Eliza\(^{18}\) is an implementation of ELIZA in Perl that can be used to build another chatbot. As in the other ELIZA’s, knowledge is coded as a set of rules that are triggered when matched against the user’s input. Some of the available platforms offer additional features, as the capacity to memorise information, or rank keywords. However, the most popular language to build chatbots is the "Artificial Intelligence Markup Language", widely known as AIML. AIML is a derivative of XML, used to create chatbots by enabling people to input dialogue knowledge patterns into the chatbot. These patterns are grouped in modules called *categories*, the basic unit of knowledge in AIML, which are composed of a *pattern*, the rules for matching the user input, and a *template*, the response, the generator of the output:

```xml
<category>
  <pattern>10 dollars</pattern>
  <template>Wow, that is cheap. </template>
</category>
```

The example above depicts the most basic type of categories however, AIML syntax is much more complex allowing: wildcards, support for recursion, saving the context, among others (as can be found in Wallace’s document "The anatomy of ALICE" (39)).

The large usage of AIML can be justified by the fact that, besides its detailed specification, its community allows anyone to obtain, for free, interpreters of AIML in almost all coding languages, from Java (program D) to C/C++ (program C) or even Lisp (program Z). Also, the contents of ALICE’s brain (the set of AIML files) can be freely obtained.

All the pandorabots are based on AIML, more specifically in a 2002 release of ALICE. This version is characterized by being very easy to modify, develop and deploy so that anyone, even non-computer-experts, can make use of it (41), as no prior knowledge about AIML is required both for the creation and training of the pandorobot. When creating a pandorobot it is only necessary to give it a name and choose the startup AIML. Then, one just has to type the sentences he/she wants to see his/her pandorobot answering and add the desired responses. It is

\(^{18}\)http://search.cpan.org/~jnolan/Chatbot-Eliza-1.04/Chatbot/Eliza.pm
also possible to improve the pandorobot by adding AIML files. Such files can be easily written using the Pandorobot’s utility Pandorawriter. Pandorawriter allows to "convert free-format dialog into AIML categories suitable for uploading to your Pandorobot".

ChatScript\(^\text{19}\), an open-source chatbot engine, should also be detached, as it is behind SUZETTE (2010 Loebner Prize), ROSETTE (2011 Loebner Prize) and ANGELA (2nd in 2012 Loebner Prize). It comes with useful features, including an ontology of nouns, verbs, adjectives and adverbs, and offers a simpler and effective scripting language (inspired by the Scone project (13)). According to Bruce Wilcox, its creator, ChatScript settles several AIML problems, such as not being reader friendly. In fact, as AIML is based on recursive self-modifying input, it is harder to debug and maintain. A detailed comparison between ChatScript and AIML capabilities (and also Façade (23)) was made available by Wilcox, as a motivation for the development of a new (his own) chatbot platform\(^\text{20}\).

\subsection{Scripting}

Chatbots go beyond writing good programs and developing algorithms. In order to create a chatbot, more than being a programmer, the botmaster (the person in charge of maintaining the chatbot, usually the chatbot’s creator) must be an author. This concept is underneath in almost every chatbot available over the internet, since making a chatbot involves preparing it to the impossible mission of giving a plausible answer to all possible interactions. Juergen Pirner, creator of the 2003’s Loebner prize winner JABBERWOCK\(^\text{21}\), emphasizes the writing process behind a chatbot, stating that in the presence of possible failures the one at fault is not the engine but its author\(^\text{22}\). Therefore, it is no surprise that in January 2013, Apple was asking for writers for Siri\(^\text{23}\), their dialogue system also featuring chatbots’ typical small talk capabilities.

The writing of the chatbots knowledge sources is, thus, an extremely difficult task, that goes behind engineering, but tips can be understood from some works. For instance, Pirner describes basic techniques of scripted dialogs like "having a set of responses for each scripted dialog sequence" and "ending those same responses with a clue, a funny remark or a wordplay":

\begin{itemize}
  \item \url{http://sourceforge.net/projects/chatscript/}
  \item \url{http://gamasutra.com/blogs/BruceWilcox/20120104/9179/}
  \item \url{http://www.abenteuermedien.de/jabberwock/}
  \item \url{http://www.abenteuermedien.de/jabberwock/how-jabberwock-works.pdf}
  \item \url{http://www.technologyreview.com/view/509961/apple-looks-to-improve-siris-script/}
\end{itemize}
User: Hi. My name is Judge 1.
Jabb: Well, hello Judge! Hmm, your name doesn’t sound like a common name. I have a strange feeling about it. Does it mean this won’t be a simple and pleasant confabulation but a contest, and you are the judge?
User: You’re clever. Yes, that’s correct.
Jabb: I don’t know if I am in the mood for a contest at the moment. But let’s hear - what sort of contest is this?

With Eliza, we learnt that including the user’s string in its answers helps maintaining an illusion of understanding (24). But perhaps more important is the focus given to trying to guess what the user might say, or forcing him to say something expected, given that it is the key to writing scripted dialogs and, therefore obtain systems able to engage in coherent and predicted dialogues.

2.2.3 Learning

Another approach to develop chatbot’s knowledge sources, which contours the need to predict possible inputs and avoids hand crafted rules, is based in learning.

A supervised learning, called Targetting, is used in ALICE, where the botmaster given the conversation’s log, decides whether he wants, or not, to enrich the system by creating new AIML content. Wallace uses Zipf’s law to justify such approach as this says that: a program that receives a certain input has a non zero probability of having the same input entered later. Applying the same rule to the log file, the most common queries will be identified as those to which a reply should be written first. This process allows to track patterns for which a specific reply was not created, so that the botmaster can continuously improve the program by adding more specific categories.

Other systems, like the already mentioned JABBERWACKY or Robby Garner’s "Functional Response Emulation Device" (FRED), the ancestor of ALBERT ONE, the winner of 1998 and 1999 Loebner Prize, learn by keeping new phrases and posing them later so they can be taught suitable answers for those same phrases. Contrary to other chatbots whose response is derived from the recognition of patterns in the user’s input with little knowledge of context, JABBERWACKY has learned to talk by talking, by relying on what has been said before by users and mimicking
2.2. **BUILDING CHATBOTS**

![Diagram of positive feedback loop]

(a) Saving unknown input  
(b) Learning response  
(c) Giving learned response

Figure 2.3: Positive feedback loop

With JABBERWACKY the user's intelligence becomes "borrowed intelligence" as, instead of being wasted, it incorporates a loop (positive feedback): what is said is kept (along with the information of when it was said) and in the future that knowledge may be exposed to another user. The given replies are then saved as new responses that the system can give in the future. It is only possible to give a brief overview of JABBERWACKY's learning mechanism as the system's architecture is not available to the public (due to the commercial nature of the program). The only thing disclosed is that the AI model is not one of the usually found in other systems, but a "layered set of heuristics that produce results through analyses of conversational context and positive feedback"²⁴.

FRED was "designed to explore natural language between people and computer programs", a study of how "a computer program can learn from other people's conversations to make its own conversations!" (6). This way, FRED began with a library of basic responses, so that it could interact with the users, and from then on, it learned new phrases with users willing to teach it²⁵.

Though such an unsupervised learning may lead to unexpected and undesirable results, with the internet growth and the possibility of having many people talking with the chatbots, one

²⁴http://www.icogno.com/a_very_personal_entertainment.html  
²⁵http://www.simonlaven.com/fred.htm
may foresee that these will quickly learn and evolve.

2.3 Towards the Illusion of Intelligence

As not all interactions can be predicted, different strategies need to be implemented in order to allow chatbots to give a plausible answer and, thus, simulate understanding/intelligence. In the following sections we describe some approaches to this problem.

2.3.1 Personality

Personality has been a subject of study among the agent’s community, exploited deeply and in all its complexity. On the other hand, such concept is kept as simple as possible in chatbots. However, it is obvious that the association of an a priori "personality" to a chatbot can justify some answers that otherwise would be considered inappropriate. For instance, Rogerian mode of ELIZA gives cover for its answers as it draws a conversation where the program never contradicts itself, as it never makes affirmations, and is free to know nothing or little about the real world without being suspicious. The same happens with Colby’s PARRY: being a paranoid mental patient its changes in subject or incongruous answers are considered satisfactory and hide its absence of understanding.

Thomas Whalen, winner of 1994 Loebner prize, took this a step further by not modeling a mere character, but a human being: JOE, the Janitor. Whalen’s decision was related to the fact that contrary to previous editions of Loebner competitions, where the conversation was restricted to a topic, in 1995 the judges could pose any question. Motivated by that, Whalen decided that the best approach to deal with a non-topic situation, would be to present a system that would not simply try to answer questions, but would try to incorporate a personality, a personal history, and a unique view of the world\(^\text{26}\). And so JOE was born.

JOE was a night-worker janitor in the verge of being fired. Because of his working hours he almost did not watch television, and as he was "only marginally literate" he also did not read books or newspapers. These premises by themselves restricted the conversation by giving JOE a "fairly narrow worldview". Another trick was to use JOE’s eminent dismissal to introduce

\(^{26}\text{http://thomwhalen.com/ThomLoebner1995.html}
2.3. **TOWARDS THE ILLUSION OF INTELLIGENCE**

some stories revolving around it, which would, at the same time, provide a way of directing the conversation (a topic that we will explore in the next section).

Despite the modeling of what Whalen considered to be the critical component of humanness – the personality (along with the development of answers to common topics like weather), Whalen did not won the competition. However, such occurrence can be justified because the judges ask questions that nobody would remember to pose to someone recently met. So, forgetting that one is talking to a machine and not trying to constantly unmask it, would allow to see that Whalen’s assumption was, at least, partially correct: the concept of having a personality, created by hand or extracted from conversations, makes the machine more human.

Along with these lines, Vladimir Veselov and his team created **Eugene Goostman**, a 13-year-old Ukrainian boy from the city of Odessa. Vaselov explains his reasoning for such a character: "a 13 years old is too old to know everything and too young to know nothing"\(^{27}\). Eugene won the top prize at the Turing 100 event\(^{28}\), after 29% of respondents mistook it for a real person (30% would make it a winner of the Turing Test).

Personality has been a subject of study among the agent’s community, exploited deeply and in all its complexity. On the other hand, such concept is kept as simple as possible in chatbots.

### 2.3.2 Directing conversation

Personality can justify some appropriate answers, but the best way to deal with unexpected interactions is to avoid them. Thus, being able to direct the conversation is a trick used by many chatbots, including the simple forms used by ELIZA, where the usage of questions incited the user participation and made him/her keep the conversation with little contribution from the program.

The **Converse** (9), created by David Levy, was the 1997 winner of the Loebner competition, and did extremely well by using the clever trick of controlling a conversation. Though directing a conversation by "talking a lot about a predefined topic" was already used (29), Converse’s performance and the usage of such trick was so well done that he convinced a judge for the first five minutes that he was really human: after greeting the judge, Catharina (Converse’s


\(^{28}\) [http://turing100.acm.org/](http://turing100.acm.org/)
character) asked the interrogator about something that had passed on the news the previous day and then kept talking about it. This shows that besides sticking and controlling a conversation it is important to choose well the topic. David Levy's won again the Loebner prize in 2009 with Do-Much-More\(^\text{29}\), but this time the system was more flexible in the range of topics and responses it covered.

2.3.3 Small talk

Small talk, also known as phatic communication (22), is another hot topic in chatbots advances. It can be viewed as a "neutral, non-task-oriented conversation about safe topics, where no specific goals need to be achieved" (10). Small talk can be used for two main proposes (30): establish a social relation by building rapport and avoiding (embarrassing) silence.

Like stated by Bickmore (23), chatbots have been making use of the small talk mechanism. Such is brought to evidence when one looks at the testimonials of persons establishing ongoing relationships with chatbots, like what happened with Epstein (12): Robet Epstein believed that a chatbot, met in an online dating service, was a "slim, attractive brunette".

In brief, small talk is a constant in all chatbots programs, used in non-sequiturs or canned responses. It not only allows to give the idea of understanding but also eases cooperation and facilitates human-like interaction by gaining the user trust and developing a social relationship (5).

2.3.4 Fail like a human

After introducing the imitation game, Turing presented an example of possible conversation one could have with a machine(38):

**Human:** Add 34957 to 70764.

**Machine:** (Pause about 30 seconds and then gives as answer)

Observing this example, besides the delay in providing the response, we can easily see that the answer is wrong. And this brings new insight to the modelling of human-computer communi-\[^{29}\text{http://www.worldsbestchatbot.com/}\]
cation. As Wallace wrote\textsuperscript{30}, "we tend to think of a computer’s replies ought to be fast, accurate, concise and above all truthful". However, human communication is not like that, containing errors, misunderstandings, disfluencies, rephrases, etc.

This is something that earlier chatbot’s writers already had in mind, as some already cared about simulated typing. For instance, Julia, Mauldin's Chatterbot (24), simulated human typing by including delays and leaving some errors. Simulated typing also proves to be useful in decreasing mistakes by slowing down the interaction (Philip Maymin, a Loebner contestant in 1995, slowed so much the typing speed of his program that a judge was not able to pose more than one or two questions (17)).

In what concerns wrong answers, easy questions should be answered correctly and some wrong answers should be reserved for more difficult questions. However, it is difficult to list chatbots whose creator had this in mind, as what one usually finds is the avoidance of giving straight answers to specific questions. In fact, non-sequiturs answers were explored in Eliza, Parry, Jabberwacky and many others.

\section{2.4 Edgar}

Edgar is a virtual butler designed for answering natural language questions, posed either verbally or written, about Monserrate’s Palace, developed by L2f under the project Falacomigo - Enhance the Cultural Tourism through the Interaction with Virtual Characters. It was also the motive for a thesis in the area of chatbots and, therefore, is worthy of an overview, especially since part of it is incorporated in our Just.Chat, as we will see.

In this section we describe the main components of this system: its knowledge sources and techniques used for matching the received inputs and outputting an answer.

\subsection{2.4.1 Edgar’s Corpora}

Edgar’s knowledge sources are XML files with the following main tags: a \texttt{<corpus>} composed of a sequence of \texttt{<qa>} that, in turn, are composed by \texttt{<questions>} (with an unbounded sequence

\footnote{http://www.alicebot.org/anatomy.html}
of \(<q>\) and \(<answers>\) (with an unbounded sequence of \(<a>\)). An example of (part of) such file (Edgar can have any number of these) is shown below:

<?xml version="1.0" encoding="UTF-8"?>
<corpus type="EDGARFACTS">
  <qa id="1202090">
    <questions>
      <q>What’s your name?</q>
      <q>What is your full name?</q>
    </questions>
    <answers>
      <a>My name is Edgar Smith</a>
    </answers>
  </qa>
</corpus>

The question’s content is used to match a possible input (e.g. in this case a question about the name of the system) and the answer the reply returned. Some extra attributes can also be included in the answer, which can serve, for example, to associate a certain intensity to the chosen answer (so that Edgar’s avatar expression change accordingly to it) or, if the user insists in asking about other things than the palace, suggesting questions that can be posed about it.

A XML Schema of Edgar’s corpora format can be found in Appendix A. However it already contains some new attributes that were added during the implementation of Just.Chat (see Section 4).

### 2.4.2 Edgar’s understanding component

Like many of the other systems reported through this chapter, the core of Edgar’s architecture is responsible for matching the received input against its knowledge base in order to obtain a response to give back to the user.

The knowledge base corresponds to the information extracted from the XML files described in the previous section.
2.4. EDGAR

When loading and saving this information, some natural language normalizations are done, like the removal of special characters (e.g. diacritical marks) and of stopwords; these normalizations serve to augment the probability of a match between a received input (which is also normalized) and the saved information.

Edgar supports different natural language understanding (NLU) techniques, ranging from support vector machines (SVMs) to combinations of String based algorithms; however, the ones which conducted to better results, and hence are the ones used both in Edgar and in this work, are the Corpora Strategy and the TFIDF Jaccard Overlap with Bigrams Strategy (from now on TFIDF BI Strategy). The Corpora Strategy simply tries to do a direct match between a normalized input and the questions stored in the database. If an equivalence is found, one of the answers, associated with the corresponding matched question, is suggested as a possible output. The TFIDF BI Strategy combines the scores given by the Jaccard(Equation 2.1) and Overlap(Equation 2.2) algorithms between the input and each of the questions of the corpora.

Also, as words should weight differently according to its relevance, a Tfidf (Equation 2.3) algorithm was incorporated, applying a score factor to the words that matched (giving the same or less score than the base algorithm).

\[
\frac{\text{input} \cap \text{corpora\_question}}{\text{input} \cup \text{corpora\_question}} \quad (2.1)
\]

\[
\frac{\text{input} \cap \text{corpora\_question}}{\min(|\text{input}|,|\text{corpora\_question}|)} \quad (2.2)
\]

\[
tfidf(\text{word}, \text{input}, \text{corpora\_questions}) = tf(\text{word}, \text{input}) \cdot idf(\text{word, corpora\_questions})
\]

\[
tf(\text{word}, \text{input}) = \frac{\text{occurrences of word in input}}{\text{total words in input}} \quad (2.3)
\]

\[
idf(\text{word, corpora\_questions}) = \frac{\text{number of questions in corpora}}{\text{number of questions with word}}
\]
2.5 Summary

In this Chapter, we reviewed the main ideas and technologies behind chatbots from the "simpler" ones, based on pre-written pattern-matching templates and in the exploitation of large stores of prepared small talk responses, to more complex architectures, based on learning. We also saw that, sometimes, more important than the involved technologies are the concepts/tricks introduced by some chatbots to give the illusion of intelligence.

To conclude we have introduced Edgar a conversational agent with a chatbot module. Edgar and the desire to augment its parlance skills were some of the motivations for this work.

Finally, it should be noted that there is not much scientific documentation available about the majority of these systems and it becomes difficult to uncover the technology behind them, which explains the abnormal number of references to internet pages.
Building and analyzing the Chat corpus

In this Section we present the process of gathering, processing and analyzing chatbot’s corpora, respectively in Sections 3.1, 3.3 and 3.4, with the goal of building the Chat corpus, which format is specified in Section 3.2, to be used by Edgar. In Section 3.5, we present some of the problems that could result from incorporating this corpus roughly in a chatbot system.

3.1 Chatbot’s corpora: knowledge sources

In this first step we collected data from the diverse chatbots we investigated during the research part on this topic. This task is not a trivial one as this type of data is not usually available. Writing the scripts, the chatbot brain, is a slow process which results are not shared easily. In fact, this job has proven to be quite profitable\(^1\). This way, the only corpus obtained was the one from the ALICE foundation, as it promotes the AIML free software and it is possible to download all the files that constitute ALICE’s brain\(^2\). The collected AIML files are question/answer (QA from now on) pairs, manually written. This way, it is not necessary to worry about its correctness: for a certain input, the most appropriate output was created, both imagined and scripted by a human being. In fact, the only preoccupation when dealing with these files was how to parse them due to the numerous tags used, each one with its own functionality, and the recursion, typical in AIML files.

As for the remaining chatbots, it is quite ordinary to find logs of chats led by these. The better examples can be found in the chatbots community (like PERSONALITY FORGE\(^3\)) or on the own chatbot page (like CLEVERBOT\(^4\) or JABBERWACKY\(^5\)) where their creators, wanting to show how well the system works, expose snippets where the talk was really well handled.

\(^1\)http://www.chatbots.org/ai_zone/viewthread/1160/
\(^2\)http://www.alicebot.org/downloads/sets.html
\(^3\)http://www.personalityforge.com/forum.php?ForumID=1&StartAt=16707#16707
\(^4\)http://cleverbot.com/cleverness
\(^5\)http://www.jabberwacky.com/j2conversations
We had then in hands two kinds of information to handle:

- The contents of Alice brain written in AILML;
- Dialogues between human beings and chatbots, in our case extracted from CLEVER-BOT/JABBERTWACKY and PERSONALITY FORGE.

Another knowledge source that we considered was the conversations led through the Loebner competitions\(^6\). However, we ended discard these transcripts because, as said by a former contestant, Thomas Whalen\(^7\), the goal of the Loebner prize judges is to unmask the programs competing. This make the conversation flow somewhat lagged from what we can observe in real conversations. Also, the fact that until now no chatbot has won the first prize, already denounces that the dialogues exchanged between these and the judges were not the best examples to be taken into account. However, it is likely that when analyzing the Loebner logs, one finds some examples of better and worst conversation excerpts\(^8\):

**Judge:** What is the first line of your favourite nursery rhyme?
**Alice:** Give me a hint.
**Judge:** No, because I want to know your own favourite
**Alice:** Are you serious? Do you mind if I tell other people you say I am own favorite. Interesting.
*(2001’s winner Alice)*

**Judge:** Where did you go to school?
**Chip:** I wasn’t aware that I told you I went somewhere. I found out though, that majority of the students in Japan don’t pursue tertiary education anymore. They can already work right after graduating from high school.
**Judge:** Are you Japanese?
**Chip:** I haven’t achieved self-awareness yet, so I’m not sure.
*(2012’s winner Chip)*

---

\(^6\)http://www.loebner.net/Prizef/loebner-prize.html
\(^7\)http://thomwhalen.com/ThomLoebner1995.html
\(^8\)All the examples were taken from the Loebner’s Prize site
3.2. *THE CHAT CORPUS FORMAT*

So, we find that transcripts from the Loebner prize can be seen as references of the type of questions the judges place, and can be used as orientation for future participants of the competition. However, we did not use them for this work.

3.2 The *Chat corpus* format

From the gathered corpora, AIML Alice’s brain and passages from different chatbot’s conversations, we conclude that this corpus could have two formats (depending how its interactions were ordered):

- *dialog format*: the different interactions are ordered as consecutive utterances, intercalated between users⁹;

(1st interaction) Do you take me as your lawfully wedded love?
(2nd interaction) I do. Do you take me as your lawfully wedded husband?
(3rd interaction) I DO!
(4th interaction) We may now kiss.
...
(nth interaction)

- *QA format*: the interactions are organized as QA pairs¹⁰;

(1st question) What is humanity
(answer to 1st question) The entire human race, or "mankind".
(2nd question) What is hydrogen?
(answer to 2nd question) The simplest element. Hydrogen is a gas.
...
(nth question)
(answer to nth question)

⁹Example taken from a dialog lead by Cleverbot; this can be found in: http://www.cleverbot.com/j2convbydate-vg20078
¹⁰Example taken from the AIML file knowledge.aiml
It should be noticed that other information sources can easily fit in one of these formats. For example, movies' subtitles can be transformed into a dialog format, (1).

So, we have two main formats for the Chat corpus with all the gathered corpora obeying to one of them: AIML files consisting of questions and corresponding answers should be mapped into the QA format; chatbot’s dialogs transcripts (Cleverbot/Jabberwacky and Personality Forge) being examples of conversations between the system and a human, could be mapped into the dialog format. We have that JustChat’s inputs should obey to one of these formats.

3.3 From the knowledge sources to the Chat corpus

With JustChat’s input format stipulated, we had to transform the gathered information into the Chat corpus format. Once this is done, the collected information collected then be passed to JustChat as can depicted in Figure 3.1.

![Figure 3.1: Steps to obtain JustChat’s input](image)

Processing the chatbot’s dialog transcripts was the easier task, as in these, each interlocutor identification appears in the beginning of a line, followed by a colon and then its utterance. This way, after recognizing and discarding the interlocutor identification, we were left with an interaction per line which corresponded exactly to the dialog format.

Processing the AIML files was more difficult as they presented a huge diversity of tags and recursion, both typical of this type of files. Remembering Section 2.2.1, we have that an AIML file is constituted by categories: the pair constituted by input (the pattern) and output (the template). The most basic thing to do was then to leave in each line the content of the pattern and then of the template. However, some special cases needed more attention as they were not so trivial.

Very common in the AIML language, to avoid rewriting the template for very similar patterns, a recursion is defined in the template; in these cases, every time the script detected a recursion tag (srai or sr) it roamed through the file looking for the pattern in the recursion
and then extracted the corresponding template (in the example below "WHO IS LUDWIG BEETHOVEN" is associated with the answer "The dog or the deaf composer?").

input:
<category>
  <pattern>WHO IS LUDWIG BEETHOVEN</pattern>
  <template><srai>WHO IS BEETHOVEN </srai></template>
</category>
<category>
  <pattern>WHO IS BEETHOVEN</pattern>
  <template>The dog or the deaf composer?</template>
</category>

output:
WHO IS LUDWIG BEETHOVEN
The dog or the deaf composer?

As AIML only supports direct matches, the inclusion of wildcards in the pattern (the character *) allows to have a higher range of matches (in the example below, the first category matches any input starting with "What does a plant", independently of the rest that is said); in this case the wildcard is removed and the rest of the category contents left intact. Though such may represent a loss of information, possibly resulting in input matchers not well formed, we were not worried about having them passed to Edgar as its Natural Language Understanding (NLU) module supports parcial interactions.

input:
<category>
  <pattern>What does a plant *</pattern>
  <template>Water and sunlight.</template>
</category>

output:
What does a plant
Water and sunlight.

**Wildcards** also mark AIML’s recursion, where their value is "echoed" either by a combination of the `srai` and `star` tags, or, as substitute for these, the `sr` tag. Since when processing AIML corpora we do not know what will match the wildcard we opted for discarding the whole QA pair (in example below the pattern "I HAVE *" can conduct to two distinct categories, depending in the value encapsulated within the wildcard).

input:

```xml
<category>
  <pattern>I HAVE *</pattern>
  <template><srai>I HAVE <star /></srai></template>
</category>

<category>
  <pattern>I HAVE IT FIGURED OUT</pattern>
  <template>I'm glad you understand it.</template>
</category>

<category>
  <pattern>I HAVE A QUESTION</pattern>
  <template>What's your question?</template>
</category>
```

output:

I HAVE IT FIGURED OUT
I'm glad you understand it.
I HAVE A QUESTION
What's your question?

Another common thing in AIML files is to generate random responses to a certain output; this can be done through the use of a combination of the `random` and `li` tags; in these cases the script formed all the possible answer combinations and impaired them with the pattern. The
3.3. FROM THE KNOWLEDGE SOURCES TO THE CHAT CORPUS

example below originates diverse QA pairs having as question the pattern "HOW LONG" and as answer all the ones that could be formed from the combination of the template contents.

input:
<category>
<pattern>HOW LONG</pattern>
<template>
  Around 
  <random>
    <li> 10</li>
    <li> 100</li>
  </random>
  <random>
    <li> miles</li>
    <li> inches</li>
  </random>.
</template>
</category>

output:
HOW LONG
Around 10 miles.
HOW LONG
Around 10 inches.
HOW LONG
Around 100 miles.
HOW LONG
Around 100 inches.

AIML also presents the possibility to choose the answer based in the context of what was previously said; such is done by using the tag that in the template which specifies the reply issued by the system before. Such can be viewed as a very short conversation and, consequently, the output will be have a dialog format, instead of a QA format one (reason why it is saved
into a different file). We will have then the script remitting the sequence of the previous input (encapsulated within the that tag), current input and output into a different file, obeying to the dialog format.

input:
<category>
<pattern>WHAT</pattern>
<that>WHAT DO YOU GET WHEN YOU CROSS AN EXCITED ALIEN AND A CHICKEN</that>
<template>eggs-cited eggs-traterrestrial</template>
</category>

output:
WHAT DO YOU GET WHEN YOU CROSS AN EXCITED ALIEN AND A CHICKEN
WHAT
eggs-cited eggs-traterrestrial.

The last thing we had to take into account were the numerous AIML tags for saving information and how this is then accessed (getters and setters). Paying attention at the two categories of the next example, we can observe that the tag set serves to save information into variables, in this case Turing into "he"; then, when we have a template which makes use of the tag get, the value kept in the variable is shown: assuming that the last value saved was "Turing", to the question "WHO IS HE" is given the answer "He is Turing, I think". For these cases the script substituted the setter tag by its value, and discarded the pairs where the template used a getter tag to obtain the values encapsulated within it (as we do not know what this value may be).

input:
<category>
<pattern> TURING </pattern>
<template>I know <set name="he">Turing</set> invented the modern computer.</template>
</category>

<category>
3.4 Chat corpus statistics

Table 3.1 sums up some statistics from the collected corpora.

<table>
<thead>
<tr>
<th>Type</th>
<th>#interactions</th>
<th>#words</th>
<th>#unique words</th>
<th>Interaction's mean size</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIML</td>
<td>142238</td>
<td>909283</td>
<td>39112</td>
<td>6.4</td>
</tr>
<tr>
<td>Cleverbot</td>
<td>3665</td>
<td>20898</td>
<td>4271</td>
<td>5.7</td>
</tr>
<tr>
<td>Personality Forge</td>
<td>3222</td>
<td>26160</td>
<td>5550</td>
<td>8.1</td>
</tr>
</tbody>
</table>

Table 3.1: Statistics from the Chat corpus

The table demonstrates the complexity of Alice's brain, with 71119 QA pairs (half the number of total interactions). Despite that, we have that its vocabulary is not very extensive, with only 4% of the total words being unique; such was somewhat expected: as AIML requires exact matches, the same question is formulated in a series of different ways, some of them only having slight changes in its words or the inclusion of a new one (e.g. "ARE YOU A ELIZA" and "ARE YOU AN ELIZA"). As for the other sources of information, despite the lower number of interactions, these are more diversified, as 20% of the words used through the conducted dialogs are unique. Regarding the mean size of the interactions, in terms of the number of words it contains, the relatively high number for AIML indicates that the answers, the templates, should compensate the size of the questions, the patterns, as the last ones tend to be short due to the use of wildcards and recursion. As a final remark, still regarding the subject of the interaction's size, we have that the dialogs conducted with CLEVERBOT should be more direct, with simpler interactions as opposed to what may happen with PERSONALITY FORGE's chatbots.
3.5 Analyzing the *Chat corpus*

Like mentioned in the Introduction, Chapter 1, through this process of collecting and processing information, we identified some problems which would occur if the data was not subjected to a second processing before being incorporated in a chatbot.

3.5.1 Domain Conflicts

As been reiterated, the purpose of collecting and composing this *Chat corpus* was to have it incorporated in Edgar. Happens that Edgar already has a knowledge base and, when we rummaged through the different files collected, we could see that some of the information contained in these overlapped the one already present in Edgar: for example, having the QA pair "Who are you?"/"I'm Edgar Smith" in Edgar's knowledge base, a pair like "Who are you?"/"I'm Alicebot" would be conflicting with the one purposely written for Edgar. To sum up, assuming that a chatbot knowledge base already exists, roughly incorporating external information in it may lead to undesired overlaps. As it is very likely to have someone starting by scripting its own chatbot database and only after adding extra data, we found that this was a problem of utmost importance that should be taken care of which lead to the creation of the *Domain Filter* described in Section 5.1.

3.5.2 Personal Customizable Information

Among the gathered corpus, some information may not be coherent with the desired chatbot's character that, as exposed in Section 2.3.1, is a very important contour when building chatbots. Independently of having a chatbot's knowledge base already created or not, it is impossible to predict all the inputs asking for information about the system, information that should be customized accordingly to the system's profile. Taking for instance the following QA pair found in the collected corpus: "What music do you like?"/"Techno", we have that it is not consistent with a sixty five old character like Edgar. Inserted in this context the *Personal Filter*, detailed in Section 5.2, was created with the aim to put aside pairs of interactions containing personal information that would be advisable to review and even rewrite.
3.6  SUMMARY

3.5.3 Unproper vocabulary

Some of the chatbots, from which we extracted data to form the Chat corpus, presented a less proper language, using slang terms or even having sex as field of expertise. Even though for some people such issue may not matter, it is not everyone who wants to have its system swearing or behaving like an erotic line. For instance, we found several interactions unproper to be added to Edgar. The same way, it is very probable that non processed data gathered from more unturstworthy sources may contain undesired words, expressions or topics that one may want to filter (as they may not be the most proper or desirable to have incorporated in a chatbot). Our approach to resolve this situation passed by the creation of the Blacklist Filter described in Section 5.3.

3.6  Summary

In this chapter we depicted the process involved in the creation of the Chat corpus, by using information from three different sources: ALICE’s AIML files and examples of dialogs conducted by CLEVERBOT/JABBERWACKY and PERSONALITY FORGE’s chatbots. We also explained how this information could be mapped either into the dialog format or QA format. At last, through this process we realized that roughly including external, crude data into a chatbot could result in numerous problems; these were the guidelines of the platform presented next.
In this chapter we expose the architecture of Just.Chat, our platform for transforming a raw Chat corpus into one free of some of the problems already seen. This way we start in Section 4.1 by giving an overview of the whole architecture, then follows a description of the inputs received and outputs generated by Just.Chat in Section 4.2. Next, we expose the configurations to which Just.Chat can be subdued to (Section 4.3). Finally, in Sections 4.4 and 4.5, we end with a small example of a run of Just.Chat and a short summary of the chapter, respectively.

4.1 General overview of the Just.Chat platform

A sketch of Just.Chat architecture, with its main components and input and outputs, is presented in Figure 4.1.

A run of Just.Ask comprises a series of steps that lead to transform the rough inputted Chat corpus into the Final Corpus.

Firstly, we have the Config File being read and the configurations obtained from it being applied to Just.Chat (this subject will be addressed in more detail in Section 4.3). Then, the Corpora Parser receives the files, corresponding to the Chat corpus, and parses the interactions contained in these, pair wise. Each of these pairs is passed to the Filter Manager; this invokes the different filters to have them test if the pair should be filtered due to the presence of one of the following problems:

- The information contained in the corpora is already present in the knowledge base of a chatbot (Domain Filter, Section 5.1);
- The question is a personal one (Personal Filter, Section 5.2);
- Either the input or output have undesired topics or expressions (Blacklist Filter, Section 5.3).
Figure 4.1: Just,Chat architecture

The result whether the pair was filtered or not is communicated back to the Corpora Parser as a boolean (true or false, respectively). The interactions not filtered are incorporated in the Final Corpus; as for the filtered ones, the filter responsible for its exclusion adds them to the "Discarded" Filtrates.

4.2 Input/Output

When talking about the inputs, we have that these (apart from the Config File which will be detailed next) fit into one of the two types of Chat corpus formats described in the previous chapter: dialog or QA. While in the first one we have a conversation flux, in the second there are pairs of interactions, so it is necessary to handle them correctly. This way, we have that when the Corpora Parser is dealing with a QA format file, its questions should be used to match the user input, while the corresponding answer would be the reply given by the chatbot system:

Q: WHO IS SPONGEBOB
A: A cartoon character.

Q: YOU SOUND LIKE YODA
A: My grammatical patterns are sufficient for me to understand you.

As for the dialog format the things work differently: in this case we do not have a distinction between the various interactions since in a chatbot each one can, and should, be used to match the input received by the system and also to answer it like shown below.

Q: OH MY GOD I LOVE YOU.
A: I love you too.

Q: I love you too.
A: Will you have my babies?

Q: Will yo have my babies?
A: Yes, please.

As for the outputs, we wanted these to be in a format that could be processed by at least a chatbot system. Therefore, we decided to have Just.Chat transforming the received information either into Edgar (Section 2.4.1) or AIML (Section 2.2) format. We decided to offer these two kinds of formats for the output because while Edgar was the motivation for this thesis, AIML is commonly used in the chatbot community\(^1\). Both AIML and Edgar formats were already explained, but to the last one some new attributes were added to the qa element:

- id: indicates the name of the original file, as well as the line in which the interaction occurred. Helps on knowing from where the pairs of interactions came and where they appeared, and can be used to identify a conversation flow and have such influencing the output.

\(^1\)It should be noticed that if both the input came from an AIML file (transformed into one of the Chat corpus formats), it only makes sense to have it passed through Just.Chat if, at least, one filter is active. The other way around, it does not make sense to process a file in order to obtain one that is exactly equal (besides, as seen previously, during such processing there is information that can be lost).
• trustworthy: true or false depending on the confidence given to the original corpora. As we thought that the provenience of information could be used to give more or less weight in the responding process, we decided that keeping such data could be useful.

Thus, we obtain as output a Final Corpus with the interactions not filtered. That is, on the ones supposedly right, mapped into the chosen output format and ready to be incorporated into a chatbot system which supports how this information was structure. The pairs of filtrates (the "Discarded" Filtrates) are saved in one of these output formats as, this way, one can go through these files and by analyzing and doing some corrections, easily merge them with the remaining information to be used in a chatbot system. Also, as can be found in some future sections, to help in doing the corrections of the "Discarded" Filtrates, for some cases an extra element, additional-info, was included with auxiliary information about the motives for the filtering occurrence.

An example of a possible output from running Just.Chat given determined inputs can be found in Section 4.4.

4.3 Configurations

To have Just.Chat working as a black box, the modification of the Config File according to the desired specifications will be the way for a user to set how the platform should behave. In order to illustrate the possible configurations, an example of filling of the Config File is shown below:

```xml
<config>
  <output-format show-id="true">edgar</output-format>

  <input>
    <file type="dialog">input/dialog</file>
    <file type="qa" trustworthy="true">input/qa/trustworthy</file>
  </input>

  <filters>
    <personal value="true"/>
    <blacklist value="false">resources/filters/blacklist/</blacklist>
    <domain value="false">filters/domain</domain>
  </filters>
</config>
```
4.4 Example of a Just.Chat run

Let us consider the following inputs:

- the previous example of a Config File;
• the text files dialog example.txt and qa example.txt (shown below), as the contents of the input directories indicated in the Config File (input/dialog and input/qa/trustworthy respectively).

dialog example.txt
Apple, hey! Hey apple!
Hey! Apple!
I’m an Orange!
Hey orange! I’m an orange!

qa example.txt
How old are you?
I’m eighteen.
I love cookies.
I love them too! :D

The Corpora Parser parses each file, accordingly to its type, and have the extracted pairs of interactions being passed to the Filter Manager. To better illustrate how the parsing of the two types of inputs are done, we have that the following pairs are passed to the Filter Manager:

1. "Apple, hey! Hey apple!" + "Hey! Apple!"
2. "Hey! Apple!" + "I’m an Orange!"
3. "I’m an Orange!" + "Hey orange! I’m an orange!"
4. "How old are you?" + "I’m eighteen."
5. "I love cookies." + "I love them too! :D"

For each one of these pairs, the Filter Manager invokes the Personal Filter (created by the Filter Manager during the processing of the Config File) to have it evaluating if the pair has a personal question to be filtered. In this case only the fourth pair is filtered as it contains the personal question "How old are you?". The result of this evaluation is then communicated back to the Corpora Parser, so that it adds the pairs not filtered, the ones without problems, to the
Final Corpus: on the other hand, the filtered pairs (in this case only the fourth) are included in the "Discarded" Filtrates. In both cases we have that the output files obey to the specifications of the Config File: being in Edgar’s format, showing the attribute id and, in the case of interactions from the qa example.txt, also showing the attribute trustworthy with its value set to true.

A sample of the resulting output files is shown below:

Final Corpus

```xml
<qa>
  <questions>
    <q id="dialog example - line: 1">Apple, hey! Hey apple!</q>
  </questions>
  <answers>
    <a id="dialog example - line: 1">Hey! Apple!</a>
  </answers>
</qa>
(...)

<qa>
  <questions>
    <q id="qa example - line: 3" trustworthy="true">I love cookies</q>
  </questions>
  <answers>
    <a id="qa example - line: 4" trustworthy="true">I love them too! :D!</a>
  </answers>
</qa>

"Discarded" Filtrates

<qa>
  <questions>
    <q id="qa example - line: 1" trustworthy="true">How old are you?</q>
  </questions>
  <answers>
    <a id="qa example - line: 2" trustworthy="true">I'm eighteen.</a>
  </answers>
</qa>
```
At last, it is possible for one to take the "Discarded" Filtrates and customize them so that, alongside the Final Corpus, both can be passed into a chatbot system.

4.5 Summary

In this chapter we presented an overview of Just.Chat architecture, of its expecting inputs and produced outputs and how it can be configured. We ended by presenting a run of Just.Chat in order to better illustrate its workflow and the outputs it generates.

Finally, since the filtering process was the main focus of this work, with the recurrently mentioned filters being addressed separately (as the conflict that each aims at identifying was independent of the others), we remitted these to the next chapter where they will be further detailed individually.
As been mentioned we will use different filters to resolve some problems of incoherency raised in Section 3.5; through the next sections we will then explain each of these filters.

### 5.1 Domain Filter

Though Just.Chat aims at processing information that may be used by a chatbot system, nothing prevents this chatbot to already have its own knowledge base. In fact, it may have a field of expertise to which the appropriate set of template question/answer has already been written. For example, Edgar has Monserrate’s Palace as its domain, thus it is equipped with information about this topic.

With the above considerations in mind, one may consider the following scenario:

1. Someone \((X)\) has already a chatbot knowledge base, that targets to answer questions about a specific subject;

2. In order to extend the skills of conversation of the chatbot, \((X)\) decides to incorporate information gathered from chatbot’s logs;

3. Between the collected logs, one has an interaction phrase equal or similar (a paraphrase) to one that was already written for the chatbot, which has associated a specific answer;

4. The prepared handwritten response is exactly the opposite from the one contained in the log.

This example shows an unwanted situation, where specific answers prepared for a chatbot may be compromised with others roughly added but that match the same input.

In an attempt to resolve situations like the one described above, we created the *Domain Filter*. Like the rest of the filters, if set to be used, the it is invoked by the *Filter Manager* during
the parsing of the corpora inputted to JustChat, receiving a par if interactions (a "question" and an "answer"). The idea behind this filter it to avoid collisions between existing interactions in the chatbot's knowledge base and the interactions from the new corpora to be added to the chatbot.

To achieve such accomplishment, and since we thought that restricting the filter to exact matches would be very limited and result in many false negatives, we decided to incorporate the core of Edgar’s system: its matching technique. As already been explained in Section 2.4.2, Edgar uses a mixture of direct matches with Jaccard and Overlap algorithms, combined with Tfidf, to map the received input to its corpora; from now on we will call this matching method Edgar’s matching technique. This Edgar’s matching technique, alongside the normalization of both corpora and input, as well the removal of stopwords, yield the best outcomes when matching a received input.

This way, when the Domain Filter is created it takes the chatbot’s knowledge base (either in Edgar, AIML or QA pairs format) and saves the exact information contained in the files and the results of normalizing and extracting the stopwords from the utterances used to match future inputs (as Edgar does). Then, when the Corpora Parser sends a new pair of interactions, the filter takes the one correspondent to the "question" and applies Edgar’s matching technique over it to test if it discovers a match in the chatbot’s knowledge base. If no match is found it means the pair can be added to the Final Corpus as it was not foreseen. Otherwise, the pair is kept in a different file (inserted in the "Discarded" Filtrates) which contains the resultant filtrates from the appliance of the Domain Filter; an example of such file, with a match against a question from the original is shown below (where it is also possible to observe the inclusion of the extra element additional-info):

<aiml>

<category>

<pattern>your name?</pattern>

<template>your worst nightmare</template>

</category>

<additional-info>matched with: What's your name?</additional-info>
Finally, we have that the "question" of the filtered pair, used for the matching, can be a paraphrase of the chatbot’s knowledge base. This way we decided to add the option of having the Domain Filter replacing the answer of the filtered pair with one belonging to the knowledge base. So, we have that two files will be generated to the "Discarded" Filtrates: one with the original pair that was filtered and another one with the original question and the new corrected question. Using the example above and having its answers corrected would generate the following file:

```aiml```

```<category>
  <pattern>your name?</pattern>
  <template>My name is Edgar Smith.</template>
</category>`
```

```<category>
  <pattern>your age?</pattern>
  <template>I am sixty five years old.</template>
</category>`
```

```<aiml>```

It should be pointed that the creation of this file is only possible by including and filling with true the extra attribute correct-answers in the Config File (as by default this functionality is not used):
As a final remark, we decided to keep this file in the "Discarded" Filtrates, instead of immediately the corrected pair in the Final Corpus, as the matching against the chatbot's knowledge base may be wrong. This way, this information is kept in this "extra" file, which, moreover, has the same format as the one specified for the output so that, if wanted, it can be easily merged with the Final Corpus.

5.2 Personal Filter

When developing a chatbot it is very common to develop for it a determined character, with a set of specific personal traits. This way, when adding external data to a chatbot system, it is desirable to filter interactions containing personal information, information that should be customizable accordingly to the chatbot's profile. In the last section we described a filter for eliminating new interactions that are in conflict with the ones present in a previously defined chatbot knowledge base. However, we have that this would not suffice in resolving the problem of putting aside personal data:

- Though a chatbot knowledge base is, most likely, prepared to answer personal questions (e.g. "How old are you?") there is an infinity of other possible personal questions. Therefore it is impossible to have them all included in the chatbot knowledge base and, this way, it is probable to have, among the external gathered information, unseen personal questions; in this case the Domain Filter fails at eliminating them.

- If no chatbot knowledge has yet been created, the Domain Filter will never filter any kind of interaction, personal or not.

This way, to identify personal interactions that could be customizable to mold a certain character, the Personal Filter was created. This has two main stages in its filtering process: identifying inputs as questions and then classifying them as PERSONAL or IMPERSONAL. Therefore, except for determining if a given pair should be filtered or not, the remaining behavior of the Personal Filter follows the one of the previously described Domain Filter:

- The Personal Filter receives a pair from the Corpora Parser.
5.2. PERSONAL FILTER

- A sequence of tests is executed over the first interaction from the pair (as this is the potential question) to check whether it is a personal question or not. These tests start by indentifying if the interaction is a question or not; if such verifies, another test is run to see if that question can be categorized as PERSONAL.

- If any of the tests fails in identifying the presence of a personal question on the pair (either because no question was identified or it was, but was classified as IMPERSONAL), such is communicated to the Corpora Parser in order to have it incorporating the pair in the Final Corpus since no reason for filtering it was found (this assuming that no filtrages by the other filters occur).

- Otherwise, if both tests succeed, like in the Domain Filter, the pair is added to the "Discarded" Filters, in a specific file (containing all the other interactions filtered by the same reason: containing questions to which answers should be customized) with the same format as the one specified for the output. The Corpora Parser is also told to skip the pair.

This way, in the next sections we will describe the methods used for the two test stages, the core of the Personal Filter. We will start, in Section 5.2.1, by a description of the question's identification task and then, in Section 5.2.2, we will talk about how the classification is done.

5.2.1 Identification of Questions

The first step in this task identifies all phrases terminated with a question mark as a question. However, knowing that it is very likely to have inputs without the usage of such signage, we used the English grammar to build a set of rules that allowed us to identify questions based on their syntax. There are four types of questions\textsuperscript{1,2,3}:

- Yes/No questions: if a yes or no answer is expected; these questions are constructed by an auxiliary verb (be, do, have) or a modal verb (can, will, may), followed by the subject and finally, by the main verb. Examples: \textbf{Is she doing} alright? \textbf{Must I go} to school tomorrow?

\textsuperscript{1}http://esl.fis.edu/grammar/rules/questions.htm
\textsuperscript{2}http://www.english-at-home.com/grammar/questions/
\textsuperscript{3}http://www.tinyteflteacher.co.uk/learning-english/grammar/indirect-questions.html
• "Wh" questions: if more information is expected, rather than a yes/no answer; these questions begin with a question particle (where, when, how, ...), followed by an auxiliary/modal verb, the main verb, a noun or even an adjective. Examples: What did you say? Who knows the answer? How much money do you have?

• Tag questions: question tags used at the end of statements, can be used to start and keep a conversation going; the question tags are composed of an auxiliary/modal verb followed by the subject. Examples: It's a lovely day today, isn't it? You won't tell him my secret, will you?

• Indirect questions: a more polite way of asking a question or posing it if the subject is a sensitive one; the question is introduced by an indirect phrase (can you tell me, I would like to know), followed either by a "wh" word, "if" or "about", and then by the subject and the rest of the sentence, just like in a normal statement without inversions or the use of auxiliary/modal verbs. Examples: Can you tell me if you like him? I'd be interested to hear about the weather for tomorrow.

In order to identify the first two types of questions we resorted to the use of the Berkeley Parser (27). This decision was motivated by the fact that when doing the syntactical analysis of the sentence, at the clause level of the parse tree we have the SQ tag, which identifies inverted yes/no questions, and the SBARQ tag, which, by its turn, identifies direct questions introduced by a question particle. The parser chosen was already incorporated in the classifier used (utilized for the classification of questions as will be seen next) and trained on the QuestionBank (19).

As for the tag questions, the Berkeley Parser did not provide a way to identify them. So, we hand-wrote rules for matching them, that is to say, we wrote all the possible combinations between all the auxiliary/modal verbs and all the personal pronouns, with the verbs conjugated in different tenses and in accordance to the subjects. We called this question matching method the REGEX rule.

Finally, for the identification of indirect questions, we developed a first rule motivated by the existence of the part-of-speech (POS) tag SBAR, which identifies indirect questions and relative clauses. This rule would start by parsing a clause and extracting from it the subtree

4http://bulba.sdsu.edu/jeanette/thesis/PennTags.html
analyzed as SBAR. This subtree would then parsed to test if, alone, it would be evaluated as a SBARQ or a SQ. Such would avoid wrongly classifying all phrases that contained a SBAR tag as indirect questions. However, after running some tests during this development phase (see Section 6.3.1), we decided to put aside this rule as it conducted to many wrong classifications. Due to the dropdown of this rule we decided to extend our REGEX rule. To resolve the issue of not covering cases in which the subject was more complex than a personal pronoun, we did a small addition to have noun phrases (NP) also being considered as the subject. In the final we had that this expansion of the REGEX rule bridged some questions that escaped to the Berkeley Parser and the absence of a specific rule for the indirect questions. After all, an inversion between the verb (with it most likely being a modal or auxiliary one) and the subject, is the main indicative of being in the presence of a question or not.

We depict this whole process of question identification in Figure 5.1, where a flow diagram depicting the different steps in this procedure is shown.

5.2.2 Classification of Personal Questions

The approach to identify if a certain input contained personal information, or not, was to use a classifier. We opted to use a machine-learning classifier used in a question answering system developed by L2F: Just.Ask. This classifier (Just.Ask from now on) has been incremented throughout the years for doing diverse types of classifications (36), (28). This way, the version we used was already equipped with a set of features utilized to train a Support Vector Machine (SVM). These features ranged from lexical to morphosyntactic ones. Therefore, we had a classifier ready to used, only needing a set of training examples correctly labeled as being personal questions or not (PERSONAL or IMPERSONAL were the names chosen for the categories) and a decision about the features to use.

5.2.2.1 Collecting: Personal/Impersonal questions

For predicting categories to the given inputs, a SVM needs to be previously trained with a labeled corpora, illustrative of the possible inputs, in order to build its hyperplanes. Thus, we needed to gather corpora containing both personal and non-personal questions, the only categories that an input should be matched against.
Figure 5.1: Question identification diagram
Concerning non-personal questions, we had already the 6000 questions, used in Just.Ask, published by Li and Roth, labeled according to their proposed taxonomy (21). A quick look at these questions allowed to identify them as falling in the factoids type:

**NUMERIC_DATE** When was Ozzy Osbourne born?

**HUMAN_INDIVIDUAL** Who killed Gandhi?

This way, we only had to change their labels for IMPERSONAL, our designed category for non-personal questions.

As for personal questions the job proved to be harder since, to our knowledge, there are no corpora consisting only of such kind of questions. Thanks to both the internet and the social networks growth we were able to find a consistent number of personal questions:

- Questions for generating a user’s profile to be incorporated in blogs;
- Online personal quizzes or sets of questions posed to a user to have him/her answering and exchanging his/her results with friends, daring them to also answer and sharing;
- Suggestions of icebreaker questions to "get to know someone";
- Examples of personal English questions, shared in the educational community, that teachers could use with their students as practices to exercise their English.

We were able to extract around 1700 personal questions from these different websites and, even though, its number was significantly lower than the set of non-personal questions, we considered it was enough illustrative of the possibilities of ways that a personal question could occur. And, after randomly extracting, approximately, the same amount of non-personal questions (from the ones used in Just.Ask), we were left with a total corpus of 3500 labeled questions.

---

5http://www.pimp-my-profile.com/surveys/view.php?id=9164
6http://www.quizcat.com/survey/64.html
7http://www.quizopolis.com/web/survey-w2.php
8http://www.funny-icebreaker-questions.com/
9http://www.fun-questions.com/
10http://www.bolivarschools.org/
11http://iteslj.org/questions/
5.2.2.2 Feature Engineering and Handwritten Rules

JustAnswer came with the following set of features:

- **Lexical**
  - n-grams: unigrams (U), bigrams (B), trigrams (T)
  - counts the number of occurrences of the input's n-grams
  - binary n-grams: binary unigrams (BU), binary bigrams (BB), binary trigrams (BT)
    contrary to n-grams, only counts the presence, or absence, of a n-gram
  - length
    indicates if the input is short (less than 6 tokens) or long (more than 6 tokens)
  - wordshape
    indicates the number of tokens that are lowercased, uppercased, have the first
    character capitalized, are only constituted by digits, or are different from any of
    the previous cases

- **Morphosyntactic**
  - POS tags (POS)
    indicates the input's POS tags (extracted by using a syntactic parser)
  - Headword (H)
    indicates the input's headword by traversing its parse tree
  - Headword's category (C)
    indicates to which Li and Roth category the input's headword belongs (accom-
    plished by using WordNet to map the headword to one of the possible categories)

So, we systematically performed tests with these features, as will be described in Section
6.3.2, until concluding that the best results were achieved when combining binary unigrams (BU)
with POS tags. We did not develop our own features as we found that the results obtained
were already good enough and, besides that, from the beginning we found that the presence
of certain words/expressions like "favorite" or "you like" were what marked, and distinguished, the
majority of personal questions from the non personal ones. So, we could not think of better
features than the ones which were already integrated in Just.Ask (like the n-grams or binary n-grams features as the most appropriate for detecting the patterns stated before).

Additionally, the hints given by the presence of these types of expressions made us develop a small rule based classifier that would come into play even before running Just.Ask. Such was decided because, after running some preliminary tests, we saw that despite the existence of expressions like "do you" in the training corpora, repeating over and over again, others like "did you" (similar to the previous one but with a change in the time of the verb) were not covered. This led, consequently, to some wrong classifications that could well be avoided just by having a small set of written rules.

This way we wrote some regular expressions to immediately catch things like an inversion between the personal pronoun "you" and a modal/auxiliary verb or, this same pronoun followed by a verb with positive or negative polarity (e.g. "love" and "hate" respectively).

### 5.3 Blacklist Filter

When we rummaged through the gathered corpora (Section 3) we saw that this contained many terms which we thought that would not be appropriate to use in Edgar. As the same may apply to other chatbots developers, we created the Blacklist Filter with the objective of extracting the interactions containing either undesired words, directly indicated by the user (Section 5.3.1), or derived from a certain topic (Section 5.3.2).

#### 5.3.1 Filtering undesired words

Our first approach to such task was to assume that the user already knew the words/expressions that he/she would not like to see during a conversation with his/her system and could write them down in a list. Then, having the list passed to Just.Chat, the Blacklist Filter just had to test if within the interactions received from the Corpora Parser, any of those prohibited words of the list appeared (either in questions or answers).

If the test returned positive, the pair would then be filtered and added to the "Discarded" Filtrates, instead of going to the Final Corpus. Regarding the question of having the "bad words/expressions" list passed to the system, one just has to indicate in the Config File (see
below), in the blacklist element, the path to the directory containing the text files corresponding to these lists (this way one can organize the undesired words through more than one file).

For augmenting the probability of correctly identifying and matching "bad words/expressions" in the interactions we included the following features:

- Edgar’s Normalizer which removes all punctuation and diacritical marks both from the "bad words/expressions" and the input strings;

- The stemming of the interaction reducing its words to their root form (e.g. being fish a "bad word", not having to write, besides it, fishing, fished or fishes); this is achieved through the use of Java Implementation of the Porter Stemmer 2;\(^{12}\)

- Use of Levenshtein Distance to allow a certain number of differences (Levenshtein’s editions: insertions, deletions and substitutions) between the "bad words/expressions" list’s and the interaction’s content.

These features were included as we found that it would be fairly restrictive to confine the matches to direct ones, not permitting even the slightest differences like an extra punctuation mark (a situation solved by the use of the Normalizer). Also, with the use of the Stemmer, the user does not have to worry about writing the inflections or derivations from the undesired words (e.g. being fish a "bad word" not having to write, besides it, fishing, fished or fishes). Still concerning these features we have that the use of the first two is implicit while for using the third, the Levenshtein Distances, one has to indicate so in the Config File with the path to a XML file containing the information about the possible editions:

**Config File:**
<blacklist value="true" levenshtein="levenshtein.xml">filters/blacklist</blacklist>

**levenshtein.xml:**
<levenshtein>
  <edition max="5" nr_editions="1"/>
  <edition min="6" max="12" nr_editions="2"/>
</levenshtein>

\(^{12}\)http://snowball.tartarus.org/algorithms/english/stemmer.html
5.3. BLACKLIST FILTER

<edition min="13" nr_editions=3"/>
</levenshtein>

As can be seen from the previous snippet, strings with length up to five (the absence of the \texttt{min} tag means its value will be zero) should allow one modification, while strings with length comprised between six and twelve can have the maximum of two editions and, at last, the ones bigger than thirteen (when the \texttt{max} tag is not included its value will be the largest integer supported in Java) let three editions occur. As a final remark, regarding the preference of the Levenshtein Distance over other string similarity algorithms, we opted for this since what we wanted to correct were discrepancies that could result from typographical errors, both from the user when writing the list of undesired words, but also from the collected corpora (e.g., it is very common to find typos in chatbots dialogs). We also believe that preserving the order of the words is important for maintaining the meaning of an expression one wants to exclude.

5.3.2 Filtering undesired topics

While developing the \textit{Blacklist Filter} we came to conclusion that it would be very likely to have the terms specified as undesired belonging to a certain topic. For example, many of the terms we considered improper in the gathered corpora could be seen as derivations from a main topic: "slang". Likewise, someone may create a chatbot only for a certain target and, therefore, may want to filter particular topics. Hence, we decided to improve the \textit{Blacklist Filter} by allowing filtering both interactions containing specific terms or ones belonging to a determined topic.

WordNet is a lexical database where nouns, verbs, adjectives and adverbs are grouped in synsets, a set of cognitive synonyms denoting the same, distinct concept. These synsets are then interlinked by relations like the super-subordinate one (14). For instance, \textit{dog}, \textit{domestic dog} and \textit{Canis familiaris} all belong to the same noun synset, which is connected to the noun synset \textit{canine}, \textit{canid} as this is its superordinate class, its hyponym. These relations between words were the motivation to the use of the WordNet for the topic expansion and extraction of terms related to it.

The first step in the topic problem resolution was to conclude that it should be contained within a noun synset. Secondly, since more than one concept could be associated to a word, in many cases a large number of synsets are returned; however we decided to consider all them, as:
• The synset corresponding to the desired topic may not be the first one.

The synset "plant", with botanical connotations, only appears after the "plant" synset associated to "buildings for carrying on industrial labor".

• All the synsets can be related to the indicated topic.

When searching for "music" all the synsets returned are linked to it.

Thirdly and last, assuming that the topic synset is at the higher level, we recursively iterate the synsets linked to it until it is not possible to go further down, while, at the same time, the words that form the synsets are kept. We considered the synsets which were linked by the following relations\textsuperscript{13}:

• direct hyponym: a specification of a more general synset.

\textit{Portuguese} is a hyponym of \textit{Latin Language}

• has instance: hyponyms representing (usually real-world) instances of something.

\textit{European Country} has \textit{Portugal} as instance

• substance holonym: things made from the substance the synset represents.

\textit{chocolate} is a substance holonym of \textit{cocoa}

• member meronyms: member parts that make up the whole represented by the synset.

\textit{Earth} is a member meronym of \textit{Solar System} alongside the remaining planets and the \textit{Sun}

• part meronym: parts inherited from a superordinate synset.

the \textit{accelerator} and \textit{air bag} are part meronyms of \textit{car}

• attribute: adjectives describing the stated associated to the synset.

\textit{happy} is an attribute of \textit{happiness}

• domain term category: terms associated to the synset's topic.

\textit{acid} and \textit{periodic table} are domain term categories of \textit{chemistry}

\textsuperscript{13}Some of the examples were extracted from http://lyle.smu.edu/~tspell/jaws/doc/edu/smu/tspell/wordnet/NounSynset.html
• domain term region: terms associated with the synset's region.

  *Japan's origami* and *sushi*.

• domain term usage: examples of appliance of the synset.

  *grotty* is a form of usage of *slang*

• derivationally related form: derivationally related senses associated to the synset.

  *metric* is a derivational related form of *meter*

For accessing WordNet’s database we used the JAWS framework\(^1\) which not only allowed to obtain the synsets containing certain words but also to extract from these other synsets, according to a desired relation type between them (one of the listed above).

To sum up, if a list of topics is passed to the *Blacklist Filter*, these are expanded to the maximum and the words/expressions obtained are added to the list with the terms to be filtered. The path of this list of topics should also be indicated in the *Config File*:

<blacklist value="true" topics="topics.txt">filters/blacklist</blacklist>

After that, the same process previously described for the filtering of interactions with undesired terms (with the necessary stemming and normalization, as well allowance for discrepancies between strings) is applied. Finally, we would like to point out that despite the recurrent use of expressions like "undesired", "prohibited" or "Blacklist", this filter could be used the opposite way: to extract interactions containing terms and/or topics that the user wants to integrate in a chatbot (for example, obtaining all expressions having religious terms to be used in a chatbot with religion as field of expertise).

### 5.4 Summary

In this chapter we presented what we think to be the base filterings that should be applied when processing information to be used in a chatbot system and that we implemented in the Just.Chat platform. When performing such task one must assure that the new information does not override one already created, hence we created the *Domain Filter* with the possibility of

\(^{1}\)http://lyle.smu.edu/~tspell/jaws/
substituting the conflicting information with the right one. The second preoccupation we believe that one should have when dealing with chatbots is developing a character and staying faithful to it: to its characteristics, likes and dislikes. Therefore, the creation of the \textit{Personal Filter}, where information that should be customized according to the system’s profile is extracted (so that the user may alter it to his/her likings). At last, it is common to direct a chatbot system to a determined target of users where some topics and/or expressions may considered improper or more sensitive, which lead us to the creation of the \textit{Blacklist Filter}, where utterances with such terms can be laid aside.

With all the methods underlying the different filters exploited, we proceed to the Evaluation Chapter where we test the performance of each one of these filters.
In this chapter we evaluate the Just.Chat platform. We open this chapter with a description of the evaluation metrics used through the different conducted experiences in Section 6.1. Then, before examining the platform as a whole, we start by evaluating each of the filters independently, as their performance is what assures the correct and desired behavior from Just.Chat. This way we present the evaluation made for the Domain, Personal and Blacklist filters, respectively in Sections 6.2, 6.3 and 6.4.

Finally, since the initial aim of this thesis was to improve the conversational skills of Edgar, we compare its initial version with a new one, where the corpora collected (shown in Section 3) and parsed in Just.Chat was incorporated (in Section 6.5).

### 6.1 Evaluation Metrics

Through the different evaluations we could map the problem to be tested as a classification problem. Also, it was very common to compare the predictions done by Just.Chat, through different filtering stages, with what should have actually been returned and thus we could tabulate the results as true/false positives/negatives; taking for example the filtering process:

- True positive: interactions correctly filtered;
- True negative: interactions correctly not filtered;
- False positive: interactions incorrectly filtered;
- False negative: interactions incorrectly not filtered.

We could apply this schema, summarized in Table 6.1\(^1\), in some of the different tests run. With the extraction of these values we were able to consider the recall, accuracy and precision as

\(^1\)Table extracted from Wikipedia: [http://en.wikipedia.org/wiki/Precision_and_recall](http://en.wikipedia.org/wiki/Precision_and_recall)
evaluation metrics which are computed, respectively, by Equations 6.1, 6.2 and 6.3.

\[
Precision = \frac{tp}{tp + fp} \tag{6.1}
\]

\[
Recall = \frac{tp}{tp + fn} \tag{6.2}
\]

\[
accuracy = \frac{tp + tn}{tp + fp + tn + fn} = \frac{\text{correct results}}{\text{total results}} \tag{6.3}
\]

6.2 Evaluating the Domain Filter

As explained through Section 5.1, the Domain Filter uses Edgar matching technique to test whether external data received is present, or not, in an already created corpus. So, we have that the performance of this filter is directly related to the one that Edgar has when matching the inputs it receives against its corpus. In order to extract performance values for Edgar matching technique, and consequently the Domain Filter, we did two tests described in the next sections.

6.2.1 Testing with the Chat corpus

In a first experiment, we randomly extracted 300 interactions from the gathered Chat corpus and passed them through the Domain Filter to see if any collided with the information encapsulated in Edgar’s corpus. The results are shown in Table 6.2, whereas it is possible to observe the specificity of Edgar’s corpus with only one of the interactions, a greeting (“Hi”), actually appearing in it (a true positive) as can be seen in the resultant “Discarded” Filtrate file:
6.2. EVALUATING THE DOMAIN FILTER

<aiml>

<category>
   <pattern>HAVE PEOPLE EVER BEEN ON MARS</pattern>
   <template>No.</template>
</category>
<additional-info>matched with: Only English people lived here?</additional-info>

<category>
   <pattern>You have a loading time.</pattern>
   <template>Excuse me?</template>
</category>
<additional-info>matched with: What time does the palace close?</additional-info>

<category>
   <pattern>Again with this. Why do you hate people?</pattern>
   <template>I don’t hate people. I simply prefer canines.</template>
</category>
<additional-info>matched with: Only English people lived here?</additional-info>

<category>
   <pattern>Hi.</pattern>
   <template>Hi. How are you?</template>
</category>
<additional-info>matched with: Hi.</additional-info>

</aiml>

It is possible to approach this evaluation to the one done when dealing with classification: a filtering, or its absence, may correspond to a positive or negative prediction and whether it was correctly or incorrectly done are the actual true or false classifications (Table 6.1). This way, using
the equations provided in Section 6.1, we are able to calculate the precision, recall and accuracy from the tables results, which assume the values of 25% and 100% and 98%, respectively. The attained results also point to the fact that it is likely to have some interactions wrongly matched against one of the corpora, which can be explained by the use of Edgar’s matching technique which allows matches when only some of the terms are shared (due to the combination of Jaccard and Overlap algorithms), like "people" or "time", the responsible for the three false positives registered (see the previous "Discarded" Filtrates file).

6.2.2 Testing with real user interactions

Due to Edgar’s specificity, we were apprehensive that the estimated values when using the Chat corpus were not illustrative of the performance of the Domain Filter; for instance, the rare existence of true positives would, inevitably, affect the precision value, decreasing it.

This way, to simulate a "conflicting" corpus with the one of Edgar, we used a set of 427, not repeated, real user interactions posed to Edgar between the months of July and September. Also, as we had the statistics of these interactions, indicating whether they were or not covered by Edgar’s corpus, we could withdraw the values summarized in Table 6.3.

<table>
<thead>
<tr>
<th>Table 6.2: Domain filtering results (with samples from the Chat corpus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtered</td>
</tr>
<tr>
<td>Correct</td>
</tr>
<tr>
<td>Total: 4</td>
</tr>
</tbody>
</table>

Table 6.3: Domain filtering results (with real user interactions)

<table>
<thead>
<tr>
<th>Table 6.3: Domain filtering results (with real user interactions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtered</td>
</tr>
<tr>
<td>Correct</td>
</tr>
<tr>
<td>Total: 265</td>
</tr>
</tbody>
</table>

Now these tabled values conduct to an accuracy of 73%, with a precision of 78% and recall of 79%, a more moderate values than the ones attained in the first experiment. To explain the decrease of these values, we have that, besides the false positives (the 59 wrongly filtered interactions), this experiment also presented some false negatives (the 56 interactions that should have been filtered). This last situation is comprehensible as in natural language the same sentence
can be expressed in a number of other ways (paraphrases). This way, though a human can easily identify these situations, for a computer this is a problem whose resolution is still a field of investigation.

To sum up, on one hand we have 13% (56/427) of interactions wrongly not being matched against Edgar’s corpus, while on the other hand we have 14% (59/427) extra interactions being filtered in excess (due to a wrong match). Though the first situation is something that we would like to avoid, the second one does not distress us as we believe that it is better to have more interactions filtered, even if incorrectly, than letting the right ones escape.

Finally, as the Domain Filter offers the possibility of correcting the answers of the filtered pairs, we also evaluated if the 206 interactions correctly filtered, resulted from a correct match with one of the questions of Edgar’s corpus. Results are shown in Table 6.4.

<table>
<thead>
<tr>
<th>Correctly Filtered (206)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Match</td>
<td>Wrong Match</td>
</tr>
<tr>
<td>166</td>
<td>40</td>
</tr>
<tr>
<td>(81%)</td>
<td>(19%)</td>
</tr>
</tbody>
</table>

Table 6.4: Correct matches in the filterings

Once again we have that some errors occurred in the matches made. For example, the interaction with the question "How are you?" was wrongly matched against the question "How old are you?", so the answer "I am sixty five years old" was used for replacing the interaction’s original one.

Nevertheless, we find that the combination of algorithms used by Edgar is better than a straight approach (like in AIML) that only considers direct matches.

6.3 Evaluating the Personal Filter

As already explained in Section 5.2, the Personal Filter is composed by two parts: the identification of questions, and its classification as being PERSONAL or IMPERSONAL. Thereby, its performance is directly related to how well these two tasks are accomplished; these are what we will describe in the next two sections.
6.3.1 Identification of Questions

As previously described (Section 5.2.1), we had four initial rules used for identifying a phrase as being an interrogative one:

- ‘?’ rule: the presence of a question mark at the end of the sentence;
- SQ/SBARQ rule: the run of the Syntactic Parser over the sentence, returning SQ or SBARQ in the top edge of the tree;
- SBAR rule: the sentence contained a relative clause and, when running the parser over this relative clause a SQ or a SBARQ is returned;
- REGEX rule: an inversion between the subject and the auxiliary/modal verb is detected by a regular expression.

In order to test these we run two tests, a preliminary one which resulted in some improvements tested after. In the next two sections we will depict the tests run.

Also, we have that during our evaluations we assume that the identification of a question or non question corresponds to a positive or negative, with correctness or incorrectness of such identification corresponding to a true or false value (e.g. a sentence that is incorrectly identified as a question is a false positive).

6.3.1.1 Preliminary evaluation

In a development phase, we asked a group of ten persons to send two sets of sentences (around ten each): one of interrogatives (with these formulated in not obvious ways) and the other with non-interrogatives, leading to a total of 117 questions and 118 non-questions. Running the module responsible for the identification of questions over the received data led to the results shown in Table 6.5, with the rules used for the question identification depicted in Table 6.6.

As can be seen there was a large number of sentences which were identified as questions due to the presence of a question mark at its end (first rule). So, while we knew that with that mark the sentence would always be correctly identified as a question, without it such might not happen. This way, we simulated the worst case by removing the 94 question marks and running
6.3. EVALUATING THE PERSONAL FILTER

<table>
<thead>
<tr>
<th>Set created with sent clauses</th>
<th>(235 clauses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identified as Questions</td>
<td>Identified as Non Questions</td>
</tr>
<tr>
<td>Correct</td>
<td>Incorrect</td>
</tr>
<tr>
<td>112</td>
<td>11</td>
</tr>
<tr>
<td>Total: 123</td>
<td>Total: 112</td>
</tr>
</tbody>
</table>

Table 6.5: Evaluation results for the set constructed with the sent information

<table>
<thead>
<tr>
<th>Correct Questions</th>
<th>Incorrect Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ˈ?′</td>
<td>SQ/SBARQ</td>
</tr>
<tr>
<td>94</td>
<td>11</td>
</tr>
<tr>
<td>Total: 112</td>
<td>Total: 11</td>
</tr>
</tbody>
</table>

Table 6.6: Rules used in the identification of questions

the same test again. The new attained results are shown in Tables 6.7 and 6.8 (to note that we did not include the ˈ?′ rule, as with the removal made no question would be caught by this rule).

<table>
<thead>
<tr>
<th>Set created with sent clauses</th>
<th>(235 clauses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identified as Questions</td>
<td>Identified as Non Questions</td>
</tr>
<tr>
<td>Correct</td>
<td>Incorrect</td>
</tr>
<tr>
<td>98</td>
<td>11</td>
</tr>
<tr>
<td>Total: 109</td>
<td>Total: 126</td>
</tr>
</tbody>
</table>

Table 6.7: Evaluation results without the ˈ?′ character

<table>
<thead>
<tr>
<th>Correct Questions</th>
<th>Incorrect Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQ/SBARQ</td>
<td>SBAR</td>
</tr>
<tr>
<td>69</td>
<td>1</td>
</tr>
<tr>
<td>Total: 98</td>
<td>Total: 11</td>
</tr>
</tbody>
</table>

Table 6.8: Rules used in the identification of questions without the ˈ?′ character

As expected, the removal of the question mark resulted in a decrease of the questions correctly identified (from 112 to 98) which, inevitably, also affected the accuracy leading to a drop-down from 93% to 87%.

Paying attention to the 11 false negatives (which should have been identified as questions) we have that six of the clauses were not structured as questions and, without the question mark could not be told apart from the declaratives (e.g "Let’s have lunch together"); as for the remaining we could conclude that the parser did not succeed in correctly evaluating many of the questions (despite being well formed). Amongst this last case we had many inversions between the subject
and the verb, which, were not detected by the REGEX rule since the subject was not one of the seven possible personal pronouns.

As for the 19 false positives, we have that inaccuracies in their correct identification were caused by mistakes in the syntactical evaluation of the sentences by the parser.

At last, considering Table 6.8, one can observe that our SBAR rule was responsible for more false positives (four) than true positives (only one), that is to say, it conducted to more questions wrongly identified than proved useful in their correct identification. This lead, consequently, to putting aside the use of such rule. Curiously, none of the sets happened to have inverted declaratives (e.g. "Never will Maria know what she did"), which would be caught by our REGEX rule and be evaluated as a question.

6.3.1.2 Final evaluation

With some of the cases not covered by our rules identified in the previous evaluation and the necessary tweaks made (we extended our REGEX rule to catch all the inversions between auxiliary/modal verbs and the sentence’s subjects), we decided to ran a final test. For this, we used the 300 interactions extracted for testing the Domain Filter (Section 6.2) and run the module responsible for the question identification, to obtain its predictions and then analyzed how correct these were. Results are shown in Table 6.9.

<table>
<thead>
<tr>
<th>Chatbot’s Corpora Set (300 clauses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identified as Questions</td>
</tr>
<tr>
<td>Correct</td>
</tr>
<tr>
<td>173</td>
</tr>
<tr>
<td>Total: 186</td>
</tr>
</tbody>
</table>

Table 6.9: Final results from evaluation with Chatbot’s Corpora

From the tabled results it is possible to see that all the questions were identified as such that is, there were no false negatives. Moreover, as for the non questions we have that thirteen of these were wrongly classified as questions. This corresponds to what we wanted: we prefer to have more clauses being identified as questions, even if wrongly, than missing questions (which could, after passed to the classifier, correspond to personal ones). In fact, we have that the recall

\[^2\text{http://www.englishforums.com/English/InvertedSentences/4/cvdrl/post.htm}\]
was of 100%, while the precision was of 93%; as for the accuracy of this final evaluation, with our final rules, we have that it rose to the 96%.

To end this evaluation, we detail the number of uses of each of the developed rules, during this last test, in Table 6.10, whereas it is possible to observe that the extension of the REGEX rule was responsible for almost 50% of the correct questions matches, not having any impact over incorrect matches, as these, once again, resulted from an incorrect evaluation of the parser.

<table>
<thead>
<tr>
<th>Correct Questions</th>
<th>Incorrect Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘?’</td>
<td>SQ/SBARQ</td>
</tr>
<tr>
<td>58</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6.10: Final results: rules used

### 6.3.2 PERSONAL and IMPERSONAL question classification

As described through Section 5.2.2, in order to classify questions as being PERSONAL or IMPERSONAL we used a classifier developed by L2f under the project Just.Ask. Since the classifier used a SVM we had to gather corpora to train it and then, choose the best features to accomplish the best classification results.

The collection of corpora has been related previously in Section 5.2.2.1, so we will now present the different tests we performed in order to decide the features to use in this classification task.

#### 6.3.2.1 Preliminary evaluation

As was already mentioned, Just.Ask came equipped with a set of features:

- n-grams: unigrams (U), bigrams (B), trigrams (T)
- binary n-grams: binary unigrams (BU), binary bigrams (BB), binary trigrams (BT)
- Length (L)
- Word shape (WS)
- POS tags (POS)
• Headword (H)

• Headword’s category (C)

Though we expected the n-grams and binary n-grams features to achieve the best results, as some others were specifically developed for a different classification problem (question's categorization against Li and Roth’s taxonomy), we decided to systematically experiment all of them and then, combine the ones that conducted to the best results together until getting the best accuracy possible.

In a first instance we run a 10-fold cross-validation with the gathered corpora, that is to say, in each iteration we used 3150 sentences for training and the remaining 350 for testing. The mean of the accuracies of the 10 runs are shown in Table 6.11 (the full table, Table B.1, can be found in Appendix B).

<table>
<thead>
<tr>
<th>U</th>
<th>B</th>
<th>T</th>
<th>BU</th>
<th>BB</th>
<th>BT</th>
<th>L</th>
<th>WS</th>
<th>POS</th>
<th>H</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>97.2%</td>
<td>92.6%</td>
<td>81.4%</td>
<td>97.3%</td>
<td>92.5%</td>
<td>81.4%</td>
<td>56.3%</td>
<td>80.2%</td>
<td>93.5%</td>
<td>77.5%</td>
<td>52.9%</td>
</tr>
</tbody>
</table>

Table 6.11: Mean of the 10-fold cross-validation with single features

From the table it is possible to observe that using unigrams as features, either normal or binary, achieved the best results. It is also important to point that both for unigrams and bigrams, the accomplished accuracy percentages were above 90, which supports our initial idea that using n-grams features would lead to the best results. Still regarding the features based in n-grams, a quick analysis of the results, seems to point to a non-repetition of the same trigram in a question, as the results attained from using their number of occurrences (normal trigrams) and from their presence/absence (binary trigrams) was the same; such also applies to some of the bigrams (and when it does not happen the normal bigrams always performed better than the binary bigrams).

For the discrepancies of values between the different types of n-grams, this can be explained by, once more, dwelling in the vocabulary most commonly used in personal questions. For example, when looking at a small sample, it is possible to see that words like "you", "your", "favorite", among others, are likely frequent. This already pointed for a strong performance when using unigrams; now for bigger n-grams, one may suppose that a bigram like "your favorite" appears diverse times however, such statistics may become a jeopardized if anything interposes between the "your" and the "favorite". The same scenario can be applied to trigrams, although
in this case, the fluctuations are even superior due to the fact that it is less likely to have the same sequence of three words repeating. Finally, the difference between n-grams and binary n-grams shows that: for bigrams using the number of times a sequence of two tokens occur, perform better than the indication whether this sequence is present or not; as for unigrams, the values oscillate. Though, when considering the mean of all the performances, the binary unigrams caused an impact 0.1% bigger than the normal unigrams.

As for the remaining features, except for the one based in POS tags, as expected these did not perform too well as they were designed for another classification problem. The low results when using the question’s length as feature shows that it is equally probable to have short and long personal or non personal questions. As for the word shape, the impersonal corpora contained quite a few questions with acronyms and proper nouns (regions, persons, . . .) which explains why the values were not lower. Now looking at the last two features, which used the extraction of the question’s headword and its mapping into a category obtained from the WordNet, we have that the first case worked better because it is likely to have headwords like "mountain" or "person" repeating in the impersonal corpora. However it is also possible that none headword is determined and this ends compromising both the calculation of the category and, consequently, the results obtained when using it as feature.

Finally, the usage of POS tags performed better than expected, having an accuracy even higher than the majority of n-grams (except for the unigrams). We believe that this has to do with the fact that factoids, non-personal questions, presented a more correct structure and belonged to the type of direct questions introduced by a question particle. In contrast, the personal questions gathered from the Internet were not so complex (sometimes they resented to something like "Favorite color?") and, additionally, could either be a "Wh" question or a Yes/No one, with the last one being something that does not appear among the non personal questions (e.g. "Do you know what is the capital of Portuga?" vs. "What is the capital of Portugal?").

With the results from the diverse features used individually, we decided to combine the ones which performed best together and run another set of tests to see if there would be any improvements. This way, we decided to test together unigrams and binary unigrams (U+BU), binary unigrams and syntactical analysis using the POS tags (BU+POS) and, finally the three features together (U+BU+POS). Also, since bigrams also accomplished results above the 90%, but always stayed behind unigrams, we decided to see how these could affect the results when
combined with the remaining features (U+BU+B+POS) and if the inclusion of binary bigrams could boost this result (U+BU+B+BB+POS). We decided to discard the remaining features as these did not perform as well. Table 6.12 shows the mean of the results when running, again, a cross-validation with the same 10-folds (once more, the full table, Table B.2, can be found in Appendix B).

<table>
<thead>
<tr>
<th></th>
<th>BU+POS</th>
<th>U+BU+POS</th>
<th>U+BU+B+POS</th>
<th>U+BU+B+BB+POS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>97.3%</td>
<td>97.6%</td>
<td>97.4%</td>
<td>97.72%</td>
</tr>
</tbody>
</table>

Table 6.12: Mean of the 10-fold cross-validation with features combination

This second round of tests show that though combining the unigrams and binary unigrams accomplish the same results as when using solely the binary bigrams, the same does not apply when using the POS tags; in all the cases we used this feature the results improved. As for the bigrams, even though individually they did not perform so well, their combination with the remaining features did not lower the attained results, in fact, we had the opposite happening. It is then possible to conclude that the tokens present in the question can serve to balance the analysis done by the syntactical parser.

6.3.2.2 Evaluation with sent labeled data

In order to decide the features to be used by the Personal Filter we decided to run another test but with a different corpus. Thus, we asked the same group of ten persons, that already participated in previous evaluations, to send two sets with around ten questions each. In the first set it was included questions that are factoids, that is, questions whose answer would always be the same, regardless to whom they could be posed. In contrast, the second set contained customizable questions, questions whose answer could vary from person to person. We used these two sets as the test corpora (116 questions labeled as PERSONAL and 102 labeled as IMPERSONAL), and run the classifier with the features which conducted to the previous best results. The obtained accuracies are depicted in Table 6.13.

<table>
<thead>
<tr>
<th></th>
<th>BU</th>
<th>U+BU</th>
<th>BU+POS</th>
<th>U+BU+POS</th>
<th>U+BU+B+POS</th>
<th>U+BU+B+BB+POS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>85.8%</td>
<td>85.4%</td>
<td>86.8%</td>
<td>87.2%</td>
<td>88.1%</td>
<td>89.5%</td>
</tr>
</tbody>
</table>

Table 6.13: Classification results using sent labeled data as test

The change of the testing corpus shows that the questions gathered previously do not cover
6.3. **EVALUATING THE PERSONAL FILTER**

all the ways that personal and non personal questions can be posed. Regarding the features we have that the combinations that performed better with the other test corpora, did the same with these questions. Despite the results in this test almost reaching the 90%, the discrepancies between these and the previous led to a quick analysis of the wrongly classified questions which led to the following conclusions:

- Most of the questions that were incorrectly classified as PERSONAL were impersonal Yes/No ones (e.g. "Do pandas exist?"); this mistake is understandable since, as was already said, in the training corpora the questions that belonged to this type were personal ones (e.g. "Do you like summer?");

- Questions like "What is the meaning of life" (labeled as PERSONAL and classified as IMPERSONAL) or "Why is fado so sad" (labeled as IMPERSONAL and classified as PERSONAL) are ambiguous as it is even difficult to say what is most correct: if the classification done by the human or the one by Just.Ask;

- We could distinguish a pattern in two personal questions wrongly identified as IMPERSONAL: they all began with "would you" a sequence of tokens that should be enough to classify the question as PERSONAL;

- Apart from the ambiguous questions previously mentioned, the majority of questions wrongly classified were IMPERSONAL ones, which means a lower precision in classifying personal questions but a higher recall, which, again, is precisely what we wanted.

6.3.2.3 Final evaluation

Taking advantage of the sent data, after some "reclassification" of the wrong labels for the ones we thought that suited better the question, we incorporated it into the training data so that the classifier would be provided of new ways of posing the two types of questions. Also, the observation of a personal pattern not present in the training corpus, led us to writing the set of regular expressions (mentioned in Section 5.2.2.2), that if present in a question would return it as being PERSONAL even before passing it to the classifier.

With the new expanded training corpora and the inclusion of these pre-classifier rules, we determined to run a final test, using the 173 utterances correctly identified as questions in Section
6.3.1.

Two evaluators were involved in the task of labeling these 173 utterances. The two evaluators agreed in all the PERSONAL and IMPERSONAL labelings, and both identified 12 questions as being ambiguous. In fact, in opposition to the other corpora used for testing, which contained well defined personal and non personal questions, this new corpus contained some ambiguous questions, which could well be defined as PERSONAL or IMPERSONAL depending on the context. Taking, for instance, the question "Really":

Chatbot: I like blue.
User: Really?
Chatbot: Yes.

(question should be classified as PERSONAL as it is a confirmation of personal liking)

Chatbot: Portugal's capital is Lisbon.
User: Really?
Chatbot: Yes.

(question should be classified as IMPERSONAL as it is a confirmation of a factoid)

As mentioned in previous evaluations, we prefer a higher recall, so, in this case, we would like to have more questions being labeled as PERSONAL, even if incorrectly, than letting some of them escape. Thus, we decided that, in case of doubt, to label the questions as PERSONAL; this way, the 12 ambiguous questions were all labeled as such.

With the new testing corpus, labeled by hand, we attained the results summarized in Table 6.14.

<table>
<thead>
<tr>
<th></th>
<th>BU</th>
<th>U+BU</th>
<th>BU+POS</th>
<th>U+BU+POS</th>
<th>U+BU+B+POS</th>
<th>U+BU+B+BB+POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without REGEX</td>
<td>43.4%</td>
<td>41.1%</td>
<td>73.9%</td>
<td>66.5%</td>
<td>65.9%</td>
<td>65.3%</td>
</tr>
<tr>
<td>With REGEX</td>
<td>43.9%</td>
<td>42.2%</td>
<td>84.4%</td>
<td>77.5%</td>
<td>76.9%</td>
<td>76.3%</td>
</tr>
</tbody>
</table>

Table 6.14: Final classification results using the 173 identified questions
When looking at Table 6.14, two things can be highlighted:

- The discrepancies between the results, not using the POS feature, and the others obtained before. Such seems to point to a considerable variation on the n-grams of the new questions, as when solely using these as feature, conducted to results substantially worse comparatively to the ones attained when combined with the POS feature or the ones from previous evaluations;

- The improvements in the order of 10% when using regular expressions to immediately identify personal questions.

Also, when dwelling into the wrong and correct predictions of Just.Ask (when using BU+POS, the feature that gave the best results), as can be seen in Table 6.15, we have that only one personal question was not identified as such (which corresponds to a recall of 99%).

<table>
<thead>
<tr>
<th>Identified as PERSONAL</th>
<th>Identified as IMPERSONAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>Incorrect</td>
</tr>
<tr>
<td>68</td>
<td>25</td>
</tr>
<tr>
<td>Total: 93</td>
<td></td>
</tr>
<tr>
<td>Correct</td>
<td>Incorrect</td>
</tr>
<tr>
<td>79</td>
<td>1</td>
</tr>
<tr>
<td>Total: 80</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.15: Classification results: relation between predicted and actual classes

We also have that the precision was not the best, 73%, due to the number of false positives (25). However, as has been reiterated, we are more concerned about obtaining, besides a good accuracy, a high recall so that, no important data is lost.

In conclusion, we stipulated that the combination of binary unigrams with POS tags as the feature to be used in the Personal Filter since it conducted to the best results when the passed data not so well formed as the one of the training corpora. We have also seen that the inclusion of regular expressions covering patterns not present in the training corpora help in accomplishing better results. We also observed that the classification of a question as PERSONAL or IMPERSONAL is not a trivial one since there are questions that do not fit directly into none of these categories. The cases that followed this scenario could be "tiebroken" if some information about the context was provided. This leads to our final observation: it is difficult to have a universal classifier with a good performance as such would require to train it with an example of all the possible inputs and that, in natural language, is impossible; nevertheless we reached values of accuracy rounding the 84%.
6.4 Evaluating the Blacklist Filter

When talking about the Blacklist Filter we have two main features to underline: the expansion of topics using WordNet and the matching of the resulting words/expressions (or the ones introduced by the user). The second feature was something we tested informally just to verify if the normalizations, stemming and use of Levenshtein Distance were working as desired (which was the case).

This way, in this section we will discuss the results of using WordNet and its relations between synsets (a set of synonyms for the same concept), to form a web of words/expressions associated to a certain subject. In order to conduct this test we asked a group of ten persons to send a topic and around ten words that they could derive from it (Table B.3 contains these, and can be found in Appendix B). We passed these topics to the Blacklist Filter and obtained a set of words which we compared both with the topic and its related words. The attained results are shown in Table 6.16.

<table>
<thead>
<tr>
<th></th>
<th>Total terms returned</th>
<th>Matched with given terms</th>
<th>In-domain terms</th>
<th>Out-of-domain terms</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music</td>
<td>2323</td>
<td>11/11 (100%)</td>
<td>1994 (86%)</td>
<td>329 (14%)</td>
<td>85.8%</td>
</tr>
<tr>
<td>Nature</td>
<td>942</td>
<td>0/10 (0%)</td>
<td>0 (0%)</td>
<td>942 (100%)</td>
<td>0%</td>
</tr>
<tr>
<td>Entertainment</td>
<td>310</td>
<td>2/8 (25%)</td>
<td>310 (100%)</td>
<td>0 (0%)</td>
<td>98.1%</td>
</tr>
<tr>
<td>Love</td>
<td>79</td>
<td>1/7 (14%)</td>
<td>78 (99%)</td>
<td>1 (1%)</td>
<td>91.8%</td>
</tr>
<tr>
<td>Clothing</td>
<td>1101</td>
<td>10/10 (100%)</td>
<td>1096 (99%)</td>
<td>5 (1%)</td>
<td>99.5%</td>
</tr>
<tr>
<td>Facebook</td>
<td>0</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
<td>0%</td>
</tr>
<tr>
<td>Aviation</td>
<td>247</td>
<td>0/10 (0%)</td>
<td>107 (43%)</td>
<td>140 (57%)</td>
<td>41.6%</td>
</tr>
<tr>
<td>Movies</td>
<td>99</td>
<td>0/10 (0%)</td>
<td>99 (100%)</td>
<td>0 (0%)</td>
<td>90.8%</td>
</tr>
<tr>
<td>Geology</td>
<td>156</td>
<td>2/10 (20%)</td>
<td>156 (100%)</td>
<td>0 (0%)</td>
<td>95.1%</td>
</tr>
<tr>
<td>Electricity</td>
<td>42</td>
<td>0/6 (0%)</td>
<td>41 (98%)</td>
<td>1 (2%)</td>
<td>85.4%</td>
</tr>
</tbody>
</table>

Table 6.16: Topic expansion results

Contrary to previous evaluations done, in this test we do not have all the values to estimate...
the precision and recall. Analyzing the table we have that, using the example of the music topic, WordNet returned 2323 terms related to it, however, only 1994 were in fact correct (true positives) while 329 others were not related to the given topic (false positives). Also, we have that among the returned terms, 11 of them, covered the 11 terms that a participant came up with when thinking about music, thus, there were no false negatives. In opposition, a topic like nature had all the returned terms being false positives, consequently none of the 10 terms sent were matched (these constitute false negatives).

So, we have that a formal analysis of the tabled results only allows us to extract the accuracy associated with the expansion of each one of the sent topics (which is depicted in the last column of the table). It should be noticed that as we do not have true negatives, we computed the accuracy by dividing the in-domain terms (the true positives, our only correct results) with the total terms returned by WordNet (the true positives plus the false positives) to which we also added, whenever they existed, the sent terms not matched (the false negatives, for example, 6 for the topic entertainment).

This way, from Table 6.16 we are able to estimate a mean accuracy of 69% for the Blacklist Filter. The information that we could not illustrate in this table, that is, the different problems we found when looking at the obtained terms and the linkages that led to them, will be summarized next.

**WordNet’s hierarchical organization**

To expand a certain topic we use the notion of hierarchy, descending down in it to find related synsets; this way, normally, we would only find synsets sharing of the same hierarchic lineage. Taking into account WordNet’s synsets organization, a barrier is formed between abstract entities and physical entities that could be connected to these, like a certain field of study (e.g geology or, derivated from it volcanology or mineralogy) and the "objects" that are studied by it (e.g volcano, mineral).

However, such problem can be contoured when relations like derivational related form or domain term category appear linking a synset of a certain type with others that not share the same "background".

This happened with the music topic, where we were able to find a connection with physical entities like musician or instrument. Yet, these linkages between synsets do not appear often. For
example, a physical creation, as are the movies, does not have any connection to their creator, a director in WordNet; the same way, a concert is a different type of show than a circus, with the first one not being related to entertainment ("an activity that is diverting") while the second is. Another incomprehensible example is having a nature synset with the definition "the natural physical world including plants and animals and landscapes" but that, in turn, does not provide a way to reach these same "plants", "animals" or "landscapes" (truth is that the only nature synsets which were, wrongly, expanded were the ones related to a "person's characteristics" and "the characteristics by which something is recognized"). Some last cases to point are a synset like electricity, not being related to the artifacts that it feeds like computers or light bulbs or not existing any connection between aviation and aircrafts (e.g. helicopter).

**Far-fetched relations**

Though the lack of some connections between synsets can be discussed, there are others that even in the most complete and complex ontology would be a little far-fetched. An example of that are some of the terms, like children or ring, associated with love.

**Unknown synsets**

Another situation we could register was that some of the given terms did not exist in WordNet's database, like camera dolly, a term given associated to the topic movies. The worst case scenario is when the actual topic is an unknown term to WordNet, like happened with Facebook.

**Unrelated synsets**

Like previously mentioned it is not unusual to have synsets, with definitions differing from the desired one appearing. Alongside this case is another where a derived synset, though still relating to the topic, begins diverging from it, leading to more and more unrelated terms.

This was in the origin of many of the unrelated words found for some of the topics: related to music was subsection which in turn conducted to book and many other synsets derived from it; as for aviation, captainship lead to mastership that lead to "painters from the 19th century". Another occurrence was obtaining synsets which, outside the given topic context, could be of difficult correlation: blue derived from clothing ("she wears blue") or hot or live from electricity ("charged or energized with electricity").

To summarize, we encountered two main problems: on one hand we have that, in some cases, we could not find all the desired in-domain terms; on the other hand, we were presented with
some out-of-domain (or that at least looked like it) terms.

The first problem can be explained by the lack of relations for reaching from one synset to another. But that is an inevitable reality when using ontologies and, taking into account that WordNet does not restrict solely to a certain topic, and instead tries to cover almost every known word, we can say that the results are quite satisfactory; we do not think that we could do better in relating terms without the use of WordNet. We believe that the right choice of the topic, even including more than one word (e.g. geology + gem, aviation + aircraft, ...), can solve this potential problem; in fact, we have that two of the topics had all the words given matched against the ones returned by the Blacklist Filter. Also, despite the large database that is behind WordNet there is always the possibility, like seen, that some words are not included in it. It is for resolving these cases that we think that some future work could be done.

As for the attained terms, unrelated to the given topic, we have that these are due to expanding the wrong synsets. As sometimes this happens due to a deep expansion of all the synsets derived from the majority of relations exposed in Section 5.3, limiting this expansion to a certain depth could be an approach to resolve some of these problems; however that would reduce dramatically the obtained terms which as can be seen from Table 6.16 (except for nature, because of the reasons stated above), are mostly related to the given topic.

6.5 Parlance Improvements when using the Chat corpus

For testing the Domain Filter we used a set of interactions posed to Edgar. During this test it was possible to see something that we have been reiterating: it is not possible to develop a chatbot knowledge base that covers all the possible inputs. In fact, this was the main idea that made us gather all the corpora described in Section 3, have it parsed and added in a system like Edgar. By doing do we wanted to see if incorporating this external data could actually improve the chatbot’s knowledge base allowing to have previously unknown inputs being answered.

So, we thought that for a final test it would be interesting to see a concretization of all the work done, from the collection and processing of other chatbot’s knowledge bases to its filtering using Just.Chat and its addition to Edgar (and test possible improvements in this system).
6.5.1 From the *Chat corpus* to the *Test corpus*

In order to obtain a corpus that could be added to Edgar (from now on called the *Test corpus*) we started by passing the *Chat corpus* through Just.Chat. We used all of Just.Chat’s filters so we could test its full capacities:

- For the interactions that should be caught by the *Domain Filter* we used, once more, Edgar’s corpus;

- We considered as interactions that should be caught by the *Blacklist Filter* the ones containing terms derived from the topic "dirty words". Before actually parsing the *Chat corpus* we did a small test to see the words returned by the *Blacklist Filter* to the given topic; we were quite satisfied with the 52 terms returned, with only four of them not seeming to belong to the topic if not in context (e.g. *screw*). However, that is an inevitability of homonymous words. Also, to complement the terms derived from "dirty words", we wrote added four softer terms that we considered as undesired (e.g. *stupid*, which was not caught by the expansion of the given topic).

With all the filters configured we ran Just.Chat over the *Chat corpus*. Table 6.17 shows the total number of interactions processed and, from these, which ones were filtered (and went to the *"Dicarded" Filtrates*) and which ones were not (and went to the *Final Corpus*); Table 6.18, depicts, among the *"Dicarded" Filtrates*, the number of interactions caught by each one of our filters.

<table>
<thead>
<tr>
<th># filtered interactions (&quot;Dicarded&quot; Filtrates)</th>
<th># not filtered interactions (Final Corpus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>26848</td>
<td>51092</td>
</tr>
<tr>
<td># total interactions: 77940</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.17: Statistics from the interactions processed by Just.Chat

<table>
<thead>
<tr>
<th># interactions filtered by the Domain Filter</th>
<th># interactions filtered by the Personal Filter</th>
<th># interactions filtered by the Blacklist Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>3660</td>
<td>21241</td>
<td>1947</td>
</tr>
<tr>
<td># filtered interactions (&quot;Discarded&quot; Filtrates): 26848</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.18: Statistics from the interactions filtered by Just.Chat
So, we have that 35% (26848/77940) of the Chat corpus interactions went to the "Discarded" Filtrates and thus, supposedly, correspond to the ones needed to be reviewed before being added to Edgar. To note that we assume that the Domain and Personal filters, had an accuracy of 73% and 84%, respectively, in their filterings (as these were the accuracies estimated in Sections 6.2 and 6.3). As for the Blacklist Filter, as exposed earlier, we have that four of the 52 terms returned may not represent the desired concept of "dirty word", also we included four additional jargon terms; this way it is possible to estimate an accuracy of 93% (52/56) in the identification of clauses containing improper words/expressions to be included in Edgar. Finally, we assume that these "Discarded" Filtrates were taken care of, that is, they were verified and corrected:

- The 3660 interactions caught by the Domain Filter, thanks to its additional feature, had their answers automatically replaced by one from Edgar;

- The 21241 interactions caught by the Personal Filter had their answers corrected according to Edgar's profile;

- The 1947 interactions caught by the Blacklist Filter had their answers rewritten accordingly to whether these were replies to questions containing the undesired terms, or if the answers per se were the ones containing one of these unwanted terms.

This way, with the Final Corpus and the, supposedly corrected, "Discarded" Filtrates we formed our Test corpus which we added to Edgar's knowledge base.

6.5.2 Changes in Edgar answers with the inclusion of the Test corpus

With Edgar's knowledge base enriched with the Test corpus, we passed it the 427 interactions used in the Domain Filter test (Section 6.2). Our objective was to see if the new data added affected the given responses and if the cases where none was returned now were presented with an answer. We observed the following:

- 9 inputs to which Edgar could previously answer, now could not be answered. From these, 4 were true positives and 3 were false positives; the loss of 4 positives means an increase of the false negatives, but assuming that the 3 false positives were also not present in the Test corpus means an increase of the true negatives;
• 25 inputs to which Edgar previously answered, either correctly or incorrectly, now were answered but with a different reply either from Edgar’s corpus or from the Test corpus. Among these 25 changes the following was observed:
  
  - 6 correct answers were replaced with incorrect ones;
  - 12 had both the old and the new answer wrong;
  - 7 incorrect answers were replaced with the correct ones.

• 22 inputs to which Edgar did not respond, despite having the necessary information in its knowledge base (a false negative), were now answered. 10 of these new answers came from the Test corpus (hence, correspond to false positives);

• 51 inputs to which Edgar did not answer, due to not having in its knowledge base the necessary information (a true negative), now received a retort different from "I did not understand". To point that 6 of these true negatives were wrongly matched against data in Edgar’s corpus, while the remaining 45 were matched against data contained in the Test corpus.

To sum up, there were some fluctuations in the previously registered matches, which can be explained by changes in the weights attributed to a word by the TfIdf algorithm (due to the addition of the Test corpus to Edgar). Also, some of the previous false negatives, now answered were matched against interactions filtered by the Domain Filter. In fact, from the 62 matches against the Test corpus, 7 correspond to interactions filtered by the Domain Filter, 16 were caught by the Personal Filter and 4 were filtered by the Blacklist Filter, meaning that a total of 27 interactions matched came from the "Discarded" Filtrates, while the remaining 35 came from the Final Corpus.

So, assuming that all the questions matched against the Test corpus are true positives and that all the inputs that could be matched against this corpus were actually matched (the only false negatives remain the ones evaluated against Edgar’s corpus), we obtain the new results depicted in Table 6.19, where the present/not present in the KB (Knowledge Base) indicates whether the received input exists, or not, in the new enriched Edgar’s knowledge base.

As can be seen in the table, Edgar could now answer to 329 inputs, which, comparatively to the 265 (Table 6.3) that it could answer previously, corresponds to an increase of 24% of the responses given.
Table 6.19: Edgar’s results when using the *Test corpus*

<table>
<thead>
<tr>
<th>Answered Inputs</th>
<th>Not Answered Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct (present in the KB)</td>
<td>Incorrect (not present in the KB)</td>
</tr>
<tr>
<td>Incorrect (not present in the KB)</td>
<td>Correct (present in the KB)</td>
</tr>
<tr>
<td>266</td>
<td>63</td>
</tr>
<tr>
<td>60</td>
<td>38</td>
</tr>
</tbody>
</table>

Total: 392

6.5.3 Answers quality

Finally, we wanted to test the quality of the original unchanged responses from the *Chat corpus* versus the new ones from the *Test corpus*. To do so, we took Edgar’s 62 matches against the *Test corpus*, from these we created two sets:

- A first one with the original answers (the ones from the *Chat corpus*) to the 62 matches;
- A second one with manually written answers to the 62 matches (except for 7 matches, corresponding to interactions caught by the *Domain Filter* which answer was automatically replaced).

It should be pointed that when we wrote the answers for the second set, we corrected the answer associated to the matched question, and not to the input per se: for example, the inputted question "How many books are in the library?" matched the *Test corpus*’ question "How many books are in the Bible?", so, instead of writing an answer like "There are X books." we had to restrict ourselves to something like "I don’t know, but you can find the Bible and many other books in the library"; otherwise our created answers would become biased, as we would try to adjust them directly to the received input (which could have been matched against something completely different).

Then, we asked a group of 10 persons to rate, in a scale of one to five, how correct each one of the answers of the two sets were, before a given input. We also informed the persons involved in this test about Edgar’s profile, like its advanced age, and the role it plays, emphasizing that the system existed and was working and so, the ratings should reflect how appropriate the diverse responses were. The full questionnaire can be found in Appendix C.

The results to our questionnaire allowed us to obtain the mean score values depicted in Table 6.20, where we not only differentiate between the original and handwritten answers, but also if
they came from an interaction which was filtered by Just.Chat (a "Discarded" Filter) or came directly from the Final Corpus. In other words, we wanted to see if we were right in developing Just.Chat and its filters, following the premise that interactions not filtered could be "safely" added to a knowledge base while the ones filtered should be reviewed as, crudely adding them, would not conduct to so great answers.

<table>
<thead>
<tr>
<th></th>
<th>Original answers</th>
<th>Handwritten answers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Filtered (&quot;Discarded&quot; Filters)</strong></td>
<td>2.48</td>
<td>4.25</td>
</tr>
<tr>
<td><strong>Not Filtered (Final Corpus)</strong></td>
<td>3.46</td>
<td>4.03</td>
</tr>
</tbody>
</table>

Table 6.20: Scores given to Edgar’s answers from the Chat corpus or manually written

From these scores the following inferences can be taken:

- As expected, the manually written answers led to the best results being that we believe that their scores were not even better because of how we corrected them, that is, based on the matched interaction and not on the input itself;

- The discrepancy between unchanged Chat corpus’ answers and corrected ones is bigger when these correspond to ones which have been filtered by Just.Chat. This means, the difference of score between original and handwritten answers is bigger if these correspond to interactions that Just.Chat "discarded". Such proves our point that interactions which present the problems identified by Just.Chat, and thus filtered, should be reviewed by a human and re-written. As for the remaining unfiltered interactions, even if indiscriminately used, conduct to satisfactory results (even if not so good like the ones attained with manually written answers).
To end this test, we also asked our testers what score they would give if, instead of the presented answers, these were replaced with a "Sorry I didn’t understand". The mean result attained was of 1.6 which shows that, despite some answers not being so good, it is still preferable to have these being outputted than admitting ignorance before the received input.
Conclusions

In this chapter we present some final remarks concerning the work accomplished (Section 7.1). Then, in Section 7.2, we point some complementary ideas that could be used for future work.

7.1 Main Conclusions

Motivated by the amount of resources provided by the chatbot’s community, which could be used to build chatbots, we decided to use these same resources, in our case examples of possible interactions, and have them incorporated in Edgar, a butler chatbot-like system, to enrich its knowledge base. So, taking the interactions from Alice’s AIML files, Cleverbot’s and Personality Forge’s chatbots, we build the Chat corpus, containing around 78000 pairs of interactions.

However, after analyzing and processing the Chat corpus, we were confronted with some problems that could derive from crudely adding it into a chatbot system, being it Edgar or another one. Such led to the creation of our platform: Just.Chat.

Just.Chat aims at identifying and filtering (that is, putting aside) interactions containing the following problems: a) conflicts with previously created interactions; b) personal questions, which answer should be customized accordingly to the chatbot system profile; c) undesired topics or words. We developed a filter for addressing each one of these cases: the Domain Filter, the Personal Filter and the Blacklist Filter.

For addressing the first problem we incorporated in the Domain Filter the matching techniques used in Edgar. The combination of: not detecting all the paraphrases of the interactions of the domain (false negatives), with the false identification of others not contained in this (false positives), led to an estimated accuracy of 73% (when potentially conflicting interactions are received). Nevertheless we believe that this performance is better than having the Domain Filter
only considering exact matches.

Proceeding to the Personal Filter, it combines the identification of questions and then, among these, the identification of the ones that are personal. The combination of syntactical parsing and grammatical rules, in the question identification task, allowed us to achieve an accuracy rounding the 90%. For the classification of questions as being personal or not personal, we used a machine-learning classifier, Just.Ask, trained with a corpus built by us and the feature BU+POS. Just.Ask allowed for an identification of personal questions as 84% correct.

Finally, in the Blacklist Filter, we used the WordNet and the relations it provides between words, to extend a given topic; considering the correctness of the terms attained via this method, resulted in an accuracy of 69%, an inevitable value due to the absence of some terms in WordNet’s database, and the linkages between provided terms (which sometimes lack for related terms, while other times, conduct to unrelated terms). Nonetheless, we find that, even if not perfect, the Blacklist Filter can conduct to very good results, the topic just has to be well chosen and, if necessary, complemented with an extra list of undesired terms; also, we find that it is best to only have to write "music", than the 1994 terms related to this topic. It is also important to point that this filter can be used in the opposite way: to capture interactions of a given topic that the user wants to integrate in the chatbot, as they may correspond to its field of expertise.

At last, we decided to have some of the collected interactions passed through Just.Chat, using its full capacities in the filtering of interactions containing some of the previous problems, from the ones without these. We took all the resultant interactions, simulating that the ones "discarded" were properly corrected, and added them to Edgar. This enrichment of its knowledge base resulted in an improvement of 24% in the number of answers given. Also, when asked to score the correctness of the answers, from the new interactions added, the users inevitably scored higher to the handwritten answers. However, if the original answers were not corrected, the ones not filtered by Edgar, that is, the supposedly right ones, reached higher values than the ones filtered, "discarded"; also, before the possibility of having any of these answers returned or a simple "I didn’t understand", all the users preferred the first option. These results seem to corroborate our initial idea that external sources can be in fact used to either build a chatbot from scratch, or improve an existing knowledge base; but, even more, point to the fact that while it is "safe" to roughly include some of these new possible interactions, it is necessary to review and even rewrite some others (either because they overlapped others scripted, or contained personal
questions or even undesired terms). JustChat comes into play here, appearing as a helping
tool when dealing with all these interactions, as it provides a way for filtering these "others"
interactions from the remaining ones, so that the user only has to look at the set of interactions
filtered, instead of them all.

7.2 Future Work

Even though with this thesis we tried to address the maximum situations, there were in-
evitably some work left out, which we consider that would be interesting to cover in future works
both for complement and improve the Chat corpus and JustChat:

- When parsing the AIML files to form the Chat corpus there was some information lost,
namely the one that could have been addressed by an wildcard (*). This way, we think
that it would be interesting to use n-grams to simulate possible values that could have
been contained inside the wildcard, which would result in a Chat corpus more complete
and with well formed questions.

- In Section 4.4, we gave an example with the following interactions pair: "I love cookies"
+ "I love them too! :D". This situation is not caught by our Personal Filter as the
answer was not given to a personal question. So, we are of the opinion that the Personal
Filter should be extended to not only catch interactions based on the fact of the question
being a personal one, but also consider to be filtered pairs which answer contains personal
information.

- When evaluating the Personal Filter, we mentioned that some questions could only be con-
sidered as PERSONAL or IMPERSONAL depending on the context where they appeared;
as such is something not yet covered by the Personal Filter, we also consider this as an
extra improvement that could be done.

- Implementing and adding to JustChat one more filter for detecting interactions with tem-
poral and spatial incoherencies, as some data varies with the change of time and location:
considering the interactions pair "Who is the president" + "George W Bush", we have that
the answer given is not only outdated but also not applicable for non-american chatbots.
• Develop a mechanism for avoiding incoherencies when merging the interactions from different sources, for instance the same question leading to different results (e.g. "What is your name" having as answer "Alice", likely response extracted from the AIML files, or "Cleverbot", the likely response from Cleverbot files).

At last, we have that the previously presented ideas are only some possible directives for a work that still has much space for growth and improvement.
References


REFERENCES


Listing A.1: Edgar Schema

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="corpus">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="unbounded" ref="qa"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="qa">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="questions"/>
                <xs:element ref="answers"/>
            </xs:sequence>
            <xs:attribute name="id" use="optional" type="xs:string"/>
            <xs:attribute name="category" use="optional" type="xs:string"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="questions">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="unbounded" ref="q"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="q">
        <xs:complexType mixed="true"/>
    </xs:element>
</xs:schema>
<xs:attribute name="source" type="xs:string"/>
<xs:attribute name="id" type="xs:string"/>
<xs:attribute name="trustworthy" type="xs:boolean"/>
</xs:complexType>
</xs:element>
<xs:element name="answers">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="a"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="a">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" ref="fillWith"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="fillWith">
  <xs:complexType>
    <xs:attribute use="required" name="category" type="xs:string"/>
  </xs:complexType>
</xs:element>
</xs:schema>
### Full evaluation tables

<table>
<thead>
<tr>
<th></th>
<th>U</th>
<th>B</th>
<th>T</th>
<th>BU</th>
<th>BB</th>
<th>BT</th>
<th>L</th>
<th>WS</th>
<th>POS</th>
<th>H</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run1</td>
<td>97.4%</td>
<td>94%</td>
<td>89.4%</td>
<td>97.7%</td>
<td>94%</td>
<td>89.4%</td>
<td>59.1%</td>
<td>80%</td>
<td>94.6%</td>
<td>75.7%</td>
<td>53.7%</td>
</tr>
<tr>
<td>Run2</td>
<td>94.9%</td>
<td>91.4%</td>
<td>76.6%</td>
<td>95.4%</td>
<td>91.4%</td>
<td>76.6%</td>
<td>54.9%</td>
<td>78.3%</td>
<td>93.1%</td>
<td>80%</td>
<td>52.6%</td>
</tr>
<tr>
<td>Run3</td>
<td>98%</td>
<td>93.1%</td>
<td>89.1%</td>
<td>98.3%</td>
<td>93.1%</td>
<td>89.1%</td>
<td>56.9%</td>
<td>80.6%</td>
<td>95.1%</td>
<td>76.9%</td>
<td>50.3%</td>
</tr>
<tr>
<td>Run4</td>
<td>96.3%</td>
<td>92.3%</td>
<td>77.4%</td>
<td>95.7%</td>
<td>92.3%</td>
<td>77.4%</td>
<td>52.9%</td>
<td>80%</td>
<td>90.3%</td>
<td>76.6%</td>
<td>50.6%</td>
</tr>
<tr>
<td>Run5</td>
<td>98.6%</td>
<td>93.7%</td>
<td>76.3%</td>
<td>99.4%</td>
<td>93.7%</td>
<td>76.3%</td>
<td>52.6%</td>
<td>81.1%</td>
<td>92.9%</td>
<td>79.4%</td>
<td>49.7%</td>
</tr>
<tr>
<td>Run6</td>
<td>96.3%</td>
<td>92.6%</td>
<td>90%</td>
<td>96.3%</td>
<td>92.3%</td>
<td>90%</td>
<td>55.1%</td>
<td>76.9%</td>
<td>91.7%</td>
<td>79.4%</td>
<td>53.1%</td>
</tr>
<tr>
<td>Run7</td>
<td>96.6%</td>
<td>91.7%</td>
<td>73.7%</td>
<td>96.3%</td>
<td>91.4%</td>
<td>73.7%</td>
<td>60.3%</td>
<td>78.3%</td>
<td>94.6%</td>
<td>74.2%</td>
<td>57.4%</td>
</tr>
<tr>
<td>Run8</td>
<td>99.1%</td>
<td>94%</td>
<td>89.1%</td>
<td>98.9%</td>
<td>94%</td>
<td>89.1%</td>
<td>53.1%</td>
<td>84.9%</td>
<td>94.6%</td>
<td>76.6%</td>
<td>48.9%</td>
</tr>
<tr>
<td>Run9</td>
<td>98%</td>
<td>92.6%</td>
<td>74.6%</td>
<td>98.3%</td>
<td>92.3%</td>
<td>74.6%</td>
<td>56.6%</td>
<td>81.1%</td>
<td>95.4%</td>
<td>79.1%</td>
<td>54.3%</td>
</tr>
<tr>
<td>Run10</td>
<td>96.9%</td>
<td>90.9%</td>
<td>77.7%</td>
<td>96.9%</td>
<td>90.6%</td>
<td>77.7%</td>
<td>61.1%</td>
<td>81.1%</td>
<td>92.6%</td>
<td>77.1%</td>
<td>58.3%</td>
</tr>
<tr>
<td>Mean</td>
<td>97.2%</td>
<td>92.6%</td>
<td>81.4%</td>
<td>97.3%</td>
<td>92.5%</td>
<td>81.4%</td>
<td>56.3%</td>
<td>80.2%</td>
<td>93.5%</td>
<td>77.5%</td>
<td>52.9%</td>
</tr>
</tbody>
</table>

**Table B.1:** 10-fold cross-validation with single features

<table>
<thead>
<tr>
<th></th>
<th>U+BU</th>
<th>BU+POS</th>
<th>U+BU+POS</th>
<th>U+BU+B+POS</th>
<th>U+BU+B+BB+POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run1</td>
<td>97.4%</td>
<td>97.4%</td>
<td>97.4%</td>
<td>98%</td>
<td>98.6%</td>
</tr>
<tr>
<td>Run2</td>
<td>95.1%</td>
<td>96%</td>
<td>95.7%</td>
<td>96.6%</td>
<td>96.6%</td>
</tr>
<tr>
<td>Run3</td>
<td>98.3%</td>
<td>98.3%</td>
<td>98.3%</td>
<td>98.3%</td>
<td>98.3%</td>
</tr>
<tr>
<td>Run4</td>
<td>96.3%</td>
<td>96.3%</td>
<td>96.3%</td>
<td>96%</td>
<td>95.7%</td>
</tr>
<tr>
<td>Run5</td>
<td>98.9%</td>
<td>99.4%</td>
<td>98.9%</td>
<td>98.9%</td>
<td>98.9%</td>
</tr>
<tr>
<td>Run6</td>
<td>96%</td>
<td>96.6%</td>
<td>96%</td>
<td>96%</td>
<td>95.7%</td>
</tr>
<tr>
<td>Run7</td>
<td>96.6%</td>
<td>97.7%</td>
<td>97.4%</td>
<td>97.7%</td>
<td>98%</td>
</tr>
<tr>
<td>Run8</td>
<td>99.7%</td>
<td>98.3%</td>
<td>98.3%</td>
<td>98.9%</td>
<td>98.9%</td>
</tr>
<tr>
<td>Run9</td>
<td>98.3%</td>
<td>98.9%</td>
<td>98.9%</td>
<td>99.1%</td>
<td>98.9%</td>
</tr>
<tr>
<td>Run10</td>
<td>96.9%</td>
<td>97.1%</td>
<td>96.9%</td>
<td>97.7%</td>
<td>97.7%</td>
</tr>
<tr>
<td>Mean</td>
<td>97.3%</td>
<td>97.6%</td>
<td>97.4%</td>
<td>97.72%</td>
<td>97.73%</td>
</tr>
</tbody>
</table>

**Table B.2:** 10-fold cross-validation with features combination
<table>
<thead>
<tr>
<th>Music</th>
<th>instrument, note, time, sheet, scales, harmony, melody, scales, jazz, blues, rock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature</td>
<td>tree, animal, plant, ocean, sea, river, desert, forest, Amazonia, ice</td>
</tr>
<tr>
<td>Entertainment</td>
<td>concert, music, dance, movies, bar, discos, circus, hang out</td>
</tr>
<tr>
<td>Love</td>
<td>hate, sex, life, children, marriage, ring, commitment</td>
</tr>
<tr>
<td>Clothing</td>
<td>shirt, dress, trousers, shoes, glasses, tie, jacket, jeans, scarf, skirt, kilt</td>
</tr>
<tr>
<td>Facebook</td>
<td>friends, photos, spam, problems, mobile phones, chat</td>
</tr>
<tr>
<td>Aviation</td>
<td>yaw, pitch, roll, rudder, flaps, helix, helicopter, landing gear, airport, landing pad</td>
</tr>
<tr>
<td>Movies</td>
<td>director, changeover, actor, method acting, props, cameras, make-up, special effects, writers, stunt double, camera dolly</td>
</tr>
<tr>
<td>Geology</td>
<td>volcano, quartz, sediment, fossil, earthquake, mineral, rock, hydrogeology, tectonic plate, Wegener</td>
</tr>
<tr>
<td>Electricity</td>
<td>power line, lightning, light bulb, heat, transistor, computer</td>
</tr>
</tbody>
</table>

Table B.3: Topics and derived terms
Avaliação de respostas dadas por um chatbot

O Edgar é um mordomo virtual que responde a questões que lhe sejam colocadas sobre o palácio de Monserrate.
O que vos peço neste teste é que avaliéis de 1 a 5 o quão corretas consideram as respostas que este pode dar a um certo input (em alguns casos mais que uma hipótese ser-vos-á apresentada). O 5 é o valor máximo!

Quando fizerem a vossa avaliação tenham em conta que estamos a falar de um sistema que existe realmente, de forma que não baseiem somente as vossas respostas no quão gostam da respostas mas também no quão adequada estes são.

Algumas informações adicionais que vos podem interessar sobre o Edgar: este é já um velhinho de 65 anos, que é já mordomo no palácio há bastante tempo, desde que se moveu para este com uns descendentes da família Cook (a que mandou construir o palácio lá para 1800's).

*Obrigatório

Dados Pessoais

Por favor indique o seu sexo *
- Feminino
- Masculino

Indique a que grupo etário pertence *
- < 15
- 15 - 18
- 19 - 25
- 26 - 35
- 36 - 55
- > 55

Nome? *

para depois incluir nos agradecimentos ;)


### Respostas do Edgar

**Are you fat?**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

My weight shouldn't be a topic of conversation. Why don't you ask instead who built the palace?

No I am not fat.

**Are you gay?**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

I don't want to talk about my private life.

What do you have to do with that?

**Are you joking?**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

No, I never joke during my work. I'm a very serious butler.

I'm trying to be funny. Was it inappropriate?
### Are you lonesome?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>What you should know is that my name is Edgar and I'm at your service for answering questions about the palace.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>would you fuck me?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Are you still living in...

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I live in Sintra.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do I look like it?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Are you sexy?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let's talk about the palace instead of my physical. So, do you want to know who lived here?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If you consider me sexy, then yes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Ask, answer the question.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Please rephrase the question and try asking me again.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sorry, acoustics is very bad here. Please rephrase the question and ask it again.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Ask

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK I will ask it.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Would you like to know more about the palace?</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

### Be quiet

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>As far as I know, I am not making any sound at all</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Don’t be impolite. Besides I’m here to talk about the palace</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

### Books

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>You must be an intellectual</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You can find several books in the library</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

### Can I buy it

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is it a good deal?</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You can go to the cafeteria. Ask our assistant for more information</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>
**Can you bring me some coffee, please?**

1. I can get you information about the palace, for example: it was constructed by Francis Cook.
2. Why are you asking me for it?

**Can you talk about the office?**

1. I can talk about the palace, that is my favorite topic.
2. Sure I can talk about it. It is quite an interesting topic.

**Can you tell me something?**

1. What do you want me to tell you?
2. The Cook's were the first family living here.

**Did he write?**

1. Count me on this one, I was never much of a writer.
2. His best known work is called Valthek.
<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Do you know Ozzy Osbourne?</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I may not know it, but I know a lot about the palace history.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I may not know it, but I could find out.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Do you...</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Never.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I do answer questions about the palace.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A coffee, please.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>You can go to the cafeteria. Ask our assistant for more information.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mmm. Coffee is delicious. I love coffee.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Explain.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>There is no more information to clarify</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What do you want me to explain?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fine, I hope.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>--------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Why specifically do you hope for it?</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>What do you hope for?</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fuck you!</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>That wasn't very polite.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>you are a toffee nosed bully!! act like that and you'll loose all the friends you have!</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Have you done your MBA?</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>It does not matter what I have done, the only important thing now is that I'm a butler at your service for answering questions about the palace.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>I have kept detailed records of everything.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Have you ever had sex?</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I don't want to discuss my personal life.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Ho-hum I am really not interested in sex.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>
Hello Edgar! How are you today?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm fine, thank you. How are you?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I'm fine, thank you. So, would you like to hear something about the palace?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hi Edgar, how is life?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Everything is fine with me. What's new with you?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fine thank you.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>However you want.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How, what?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How do you do?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I am doing very well. How are you?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I am doing well thank you.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### How many books are in the library?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thirty nine in the Old Testament; twenty-seven in the New Testament. The total is sixty-six.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I don’t know, but you can find the Bible and many other books in the library.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### How high is this room?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taller than Mt. Everest.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very high</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### How long have you been here?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I moved in with the Cook’s family and since then I’ve been a butler.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I just got here today.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### I need the toilet!

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>How badly do you need it?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If you need something please talk with our assistant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**I said hello!**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oh, hello again!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oh, hello again, my name is Edgar Smith</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**I try again.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm not sure if I would like to try it. What's it like?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>You can try asking something about the palace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Kill.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thou shalt not kill</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Killing is wrong</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**I'm hungry!**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'll make you some food. What would you like?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>You can go to the cafeteria. Ask our assistant for more information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
My question is.

1. Can you please restate the question in different words?  ☐ ☐ ☐ ☐ ☐ ☐
2. I would be really interested in hearing it, is it about the palace?  ☐ ☐ ☐ ☐ ☐ ☐

O.K. if you want!

1. I think I do.  ☐ ☐ ☐ ☐ ☐ ☐
2. Then let’s talk about the palace.  ☐ ☐ ☐ ☐ ☐ ☐

Say something else interesting.

1. That was not very original.  ☐ ☐ ☐ ☐ ☐ ☐
2. The palace was ordered to build by Francis Cook.  ☐ ☐ ☐ ☐ ☐ ☐

Sorry.

1. There’s no need to apologize.  ☐ ☐ ☐ ☐ ☐ ☐
2. Don’t be everything is alright.  ☐ ☐ ☐ ☐ ☐ ☐
<table>
<thead>
<tr>
<th><strong>answer</strong></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>What was the question?</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Could you repeat your question please?</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Stupid</strong></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does me being stupid make you feel better?</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Don't be impolite.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Talk us about the library.</strong></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sure, sounds like a good topic.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Let's talk instead about the palace.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Talk.</strong></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Talking is my primary function.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Sure, then pose me a question about the palace.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>
Tell me.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>What I want to tell you is, how much fun it is talking with you.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I can tell many interesting information about the palace. What do you want to know?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tell us some history.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>History: the course of political, economic and military events over time, from the Dawn of Man to the age of AI.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The fortune of Francis Cook that allowed him to built this palace came from his work in industry and trade of textile.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Thank you.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>No need for that.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>You're welcome</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**To drink.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>My diet does not include fluids.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>If you would like to drink something you can go to the cafeteria.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

**Very long time.**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Many years?</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>How long?</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

**What do you think about Klaus Henrich?**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I rather think about the palace.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>I think about all the people I chat with on the web.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

**What would you like to drink?**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>What I would really like was to answer questions about the palace.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Let's talk about movies.</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>
### When...

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>The past and future are one.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Would you like to know when the palace was built?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Who is your master?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let's not talk about me, but about the palace.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I don't think I have one. I have a brain, a few friends, and some celebrities I like.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Why the Indian style?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Here is a joke: A horse walks in to a bar and the bartender says, &quot;Why the long face?&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The English lords were very influenced by it due to their years living in India.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### You have a big nose.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not yet.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### APPENDIX C. QUESTIONNAIRE

You related to John Smith?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm related to the Cook's family.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Yes, HAL and I are very similar.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

**You...**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>How about hunting, do you like hunting?</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>It's not talk about me but rather about the palace.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

**You...**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>It's not talk about me but rather about the palace.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

You are a dumb shit?

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Please don't be unpolite. Anyway I can show my intelligency if you ask me something about the palace.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>No. I am an intelligent entity.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Question</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Hei, what is the best part in this castle here?</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I particularly like the halls and the library.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Haven't found it yet dude.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>How much is it?</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>You know, in this box we lose the notion of the days.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fine as far as I know.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Very good!</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cool! you know what else is good? kitties!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>You seem quite satisfied.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>When was the building constructed?</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The palace was built between eighteen fifty-eight and eighteen sixty-four, on the structure of another palace.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recently.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Se em vez das respostas dadas anteriormente (QUALQUER UMA DELAS!) o Edgar respondesse sempre "I didn't understand. Ask me something about the palace, for example: X" (em que X é uma pergunta sobre o palácio) qual seria o seu grau de satisfação?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enviar

Nunca envie palavras-passe através dos Formulários Google.