Chattuga – A meta-chatbot for the portuguese language

Mariana Gaspar Fernandes
mariana.gaspar.fernandes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2019

Abstract

Most works on question answering systems follow a similar approach: they start by gathering data, which is then pre-processed, classified and used in a system which replies queries in the domain of that data. In this work, we propose a plug and play (for the addition of corpora and agents and for the control of system modules) "collaborative" chatbot framework for the Portuguese language that has as basis the assumption that all agents can, potentially, answer all questions. We study classification approaches and similar systems and we base our architecture on that study. The system built in this work, receives a query from a user and redirects it to all chatbots integrated within it. These chatbots all provide their suggested answers to the user query. When the program finishes gathering all these answers it evaluates the agent's answers as well as their expertise in the domain of the question and combines that with the classification of the user's query intent to perform an answer ranking process. In both classification processes, both Machine Learning and rule-based approaches are combined. The evaluation of the program developed in the scope of this work is made with the metrics of Accuracy and Mean Reciprocal Rank, where the first is used for the query classification evaluation and the second is used for the system's answer ranking decision evaluation. Results show that our system has a strong classification component both for questions and answers and also that it is good at identifying specialist agents.

Keywords: Question Classification; Question and Answering systems; Conversational Agents; Machine Learning; Rule Based.

1. Introduction

This document is about a work named ChaTuga for the Portuguese language in the field of artificial intelligence, more specifically, in the area of Natural Language Processing (NLP) that intends to present a new plug and play architecture for a chatbot (also known as conversational agent) in which we will use several other conversational agents. Next we will present the motivation (section 1.1), objectives (section 1.2) and project overview (section 1.3) for this document.

1.1. Motivation

With the exponential increase in internet services, it has become clear that online support is very important for costumer satisfaction. One approach to this is to use conversational agents who can answer client's questions about a given service. There are chatbots that do this for a strict domain. However, for an online organization that has many different services, it would be interesting to have many chatbots: one for each service. For instance, an online retailer could have an agent that only knows about payment methods, other agent that has knowledge of technological products, another one that is a specialist in legal procedures, among others. Each have its area of expertise and they can work in parallel in providing online support for costumers. Not only is such an architecture useful for company's needs: a website or application with recreational and/or educational purposes could use such an architecture of multi-agents to answer user's questions about a multitude of subjects such as movies, personal matters, science, space, among others. One work that already integrates several conversational agents is described in [7]. However, this work does not take into account the fact that there may be some intersection between what two conversational agents can answer. For instance: an agent that retrieves answers from Wikipedia [1] can answer factoid questions, and an agent that retrieves answers from IMDB [2] can answer questions about cinema. Both can answer questions about cinema, although the agent that uses IMDB can potentially do it better since it is its domain of expertise. In this work we will tackle this issue by assuming that "all agents can potentially answer all questions". Then, a Coordinator will decide how to rank their answers based on their topics of expertise.

1.2. Objectives

For this thesis we have three main objectives: to have a system for the **Portuguese language** that works in a **plug and play** fashion in the way it adds agents and corpora and in the way it controls the modules of the system, and has a **Coordinator** controlling the different agents.

The focus of this work will not be on building conversational agents of different domains (those are already built and then integrated in our system), but rather on understanding a user's intention when he introduces a query in order to understand which answer among the ones returned by the various conversational agents is the most suitable. To perceive which answer is more

[1]https://www.wikipedia.org/ (Last accessed on: 25/11/2018).
[2]https://www.imdb.com/ (Last accessed on: 25/11/2018).

suitable to the user's query, we are going to use machine learning techniques, similarity techniques and rule based techniques. First, we classify the query with machine learning modules trained in the corpora that is added to the system using the plug and play module. The plug and play module also provides the system with conversational agents which will have specified their areas of expertise. With the classification of the query, we can compare it with the area of expertise of the agents. Secondly, depending on whether the query is personal or impersonal (information we obtain from the classification process), we use similarity measures, rule based approaches and/or machine learning techniques to determine to which extent the answers are close to the query. Lastly, if a corpora of answers is added through the plug and play module, we use machine learning techniques to classify the agent's answers and compare the classification of the answer with the classification of the query. The plug and play module also allows a control over which modules are used to rank the answers.

To ensure a coordination of the different techniques (machine learning techniques, similarity techniques and rule based techniques), we require an agent that is capable of retrieving from the combination of all these models, the ranking of answers that, according to the techniques, is the most correct one. This agent will be the Coordinator that was previously mentioned.

1.3. Project Overview

This work is divided as follows: , in Section 2 we describe some **agents** that are integrated in our system. Section 3 describes the corpora that we have used in this work to perform the training of the classification models. Section4 provides a guide through the implementation of this project. Section 5 explains what we have done to evaluate our work and the results of that evaluation. Lastly, Section 6 presents our conclusions.

2. Agents

The purpose of our work is not to build agents that answer questions, but rather to select an answer from those agents. There are already some agents developed in Instituto Superior Técnico (IST) that we can use in our work. Some of these agents were also used in TalKit. They are Talkpedia [14], Say Something Smart (SSS) [2], and Edgar [9].

- **Talkpedia** uses Wikipedia to retrieve answers to factoid questions adding a template to it, so it provides a complete answer instead of just pieces of information.

- **Say Something Smart** is a system that uses movie subtitles to to reply to an user interaction. In this work, the version of SSS used is the one built in a simultaneous master thesis to this work [16].

- **Say Something Smart AMA** is a system that uses the SSS agent framework with a corpus built with the Agência para a Modernização Administrativa (AMA)[3][4];

- **Edgar** is a virtual museum assistant in Palácio de Monserrate (Monserrate Palace) that has expertise about the palace. Edgar was at Palácio de Monserrate answering questions that users had about the palace.

- **Cheat** is an agent built in the context of this work and with some similarities to TalKit's rule based approach, that uses rules to answer queries. It answers salutations with the equivalent salutation (good morning with good morning, hello with hi or hello, etc.); answers or questions by replying the word after the or in the query; answers yes/no questions with yes, no or maybe. If it is none of the above it says that it is not its area of expertise.

3. Corpora

In this section, we describe some of the corpora that we used to perform the classification required for our work. We start by addressing the TalKit Corpora (Section 3.1), we proceed to talk about the Li & Roth corpus (Section 3.2), next we describe the Multieight corpus (Section 3.3) and we finish by talking about the SubTle corpus (Section 3.4).

3.1. TalKit Corpora

The TalKit corpora is composed of: the Question vs. Non-Question corpus which aims to classify an utterance as "question" if it is a question and as "NON_QUESTION" otherwise (Section 3.1.1); the Personal vs. Impersonal corpus which aims to distinguish personal questions from impersonal questions using the labels "PERSONAL" and "IMPERSONAL" (Section 3.1.2); finally, the Yes/No vs. Other corpus that uses the label "YN_QUESTION" when a question only requires a yes/no/maybe answer and the label "OTHER" otherwise (Section 3.1.3). The sentences in all the three corpora came from different sources, namely:

- from the translation into Portuguese [6] of the widely used Li & Roth corpus [12];

- from a manual Portuguese translation of parts of the corpora of a chatbot called Just.Chat [15];

- from the B-Subtle corpus, a corpus built from movies Subtitles, as described in [2].

In addition, some sentences were gathered from the web, created by the author of TalKit or suggested by Técnico Students at Taguspark, in a Natural Language course.

The proposed use of this corpora, is to use the Personal vs. Impersonal and the Yes/No vs. other corpus only if the utterance is first classified with the label "QUESTION" using the Question vs. Non-Question corpus.

In the context of this work, some corrections were made. The main corrections that were made were on the level of structure of sentences, grammar, ortographical mistakes, fixing incorrect labels, etc. The magnitude of the changes amounted to a Levenshtein[11] distance of 2306 for the question vs. non question corpus (section 3.1.1); 4455 for the personal vs. impersonal corpus

[3]https://www.ama.gov.pt/ (Last accessed on 02/01/2019).

[4]A public Portuguese institution that intends to bring some modernization to the administrative processes.

(section 3.1.2); and 78 for the yes/no vs. other corpus (section 3.1.3). These corrections and analysis of the corpora resulted in a paper [8] that was published on July 2019.

In the following we describe each one of the three corpora.

3.1.1 Question vs. Non-Question corpus

In this Section we give a brief description of the Question vs. Non-question corpus, namely, the different formulations of questions that were gathered, as well as some examples of the non-questions.

Questions :

Several types of questions were taken into consideration. Besides the usual *direct questions*, we also gave some room to *imperative sentences*[5] that constitute a request for information.

Regarding *direct questions*, the corpus contains several examples of the so called "Wh-questions", that is, questions that contain the keywords "quem" ("who"), "onde" ("where"), "porquê, porque" ("why"), etc. Examples of such questions can be found on [8]. It was also included in the corpus wh-questions that present a possibility, like an imagined scenario, and then inquire something with that scenario in mind (e.g., "Se o mundo inteiro estivesse ouvindo, que dirias?" – "If the whole world was listening, what would you say?"). In addition, questions whose answer can be a simple "Sim" ("Yes") or "Não" ("No"), including questions that are only one word (e.g., "Jantar?" – "Dinner?") were also addressed. Moreover, questions that contain two possibilities of answer separated by the connector "or" (choice questions), were also added to the corpus (e.g., "Do que gostas mais: factos ou ficção?" – "What do you like more: facts or fiction?").

In what concerns *imperative sentences* that constitute a request for information, or ask for a description or definition of something, several cases were included in the corpus. Some examples can be found on [8].

Finally, some cases where several questions are formulated in the same entry were also added. The reason for this is that sometimes people ask several questions related to each other in a row (e.g., "Se tivesses de escolher, qual animal de uma quinta gostarias ser? Porquê? Podes fazer o som?" – "If you had to pick, which farm animal would you like to be? Why? Can you do its sound?").

In summary, the utterances labeled as questions encompass: choice questions, wh-questions, yes/no questions and imperative sentences.

Non-questions The non-questions part of the corpus is constituted of sentences such as:

- "A ideia é os dez formarem um círculo de protecção em torno do possuído." – "The idea is that the ten form a circle around the possessed.";
- "O David precisa de ir ao lançamento de um filme." – "David needs to go to a movie launch.";

Some statistics about this corpus can be found on [8].

3.1.2 Personal vs. Impersonal Questions

In this Section we explain what can be found in the Personal vs. Impersonal corpus.

Personal Questions In European Portuguese, the way personal questions are formulated depend on who we are talking to. Hierarchy and age difference, among others, will lead to more formal/informal conversations. When two people engage in an informal conversation, the second person of the singular is usually used; otherwise the third person of the singular is employed. For instance, if we ask a friend if he likes to read, we would ask "Gostas de ler?" or "Tu gostas de ler?", but if we asked a person we do not know or has one of the aforementioned differences, we would ask "Gosta de ler?" or "Você gosta de ler?" (being the latter in a more Brazilian Portuguese style). In the English language all these questions translate to "Do you like to read?". In the corpus for personal and impersonal questions these cases were taken into account. Examples are (the first one is an example of formal speech, and the second of informal speech):

- "Diga algo que fez em criança que os seus pais não sabem." – "Say something that you did as a child that your parents do not know of.";
- "Diz 1 coisa que desejavas mudar em ti." – "Say 1 thing that you wish to change in yourself.".

Other examples of personal questions can be found at [8]. In conclusion, personal questions are questions about the interlocutor's personal matters, such as his opinions, feelings, memories, home city, friends, among others. If the questions are about the personal life of a person that is not an acquaintance of the interlocutor and if that question is not asking for an opinion, then it is not personal.

Impersonal Questions As for the impersonal questions, they are mostly factoid questions extracted from the aforementioned translation of Li & Roth corpus for Portuguese. Some examples include:

- "O que faz com que um tornado gire?" – "What makes a tornado turn?";
- "Quais são os dois países cuja costa faz fronteira com a Baía de Biscaia?" – "What two countries' coastlines border the Bay of Biscay?";

Some statistics about this corpus can be found on [8]

3.1.3 Yes/No Questions vs. Other

In this Section we explain what are Yes/No questions and we provide some examples of the questions of this kind that can be found in this corpus. We also present some examples of the questions that cannot be answered with a simple "yes", "no" or "maybe".

[5]Sentences that are an order, an instruction or a request to do something [1].

Yes/No Questions Examples of Yes/No questions are: “Lês muito?” (“Do you read a lot?”) and “Gostas de dançar?” (“Do you like to dance?”). Notice that, in the set of Yes/No questions, one can find questions constituted of one single word (e.g., “Pizza?”).

Other As to the questions labeled as other, they are similar to the ones presented in Section 3.1.1, excluding the Yes/No ones. Under the label “other” we can find questions such as ”Wh-questions”, imperative sentences, among others. Some examples (extracted from the corpus) are: “Indique um pesticida.” (“State a pesticide.”) and “Em que cidade se encontra a Basílica de São Pedro?” (“In what city is Saint Peter’s basilica located?”).

Some statistics about this corpus can be found on [8].

3.2. Li & Roth Corpus

The Li & Roth corpus [12] named UIUC dataset is a widely known corpus of factoid questions that contains 5500 questions manually labeled according to the taxonomy discussed in [12]. It contains English questions collected from the USC [10], 500 manually built questions, and 1403 from the conferences TREC 8,9 and 10.

Examples of utterances that are in this corpus are available at the website[6] that contains it.

This corpus is of special interest to our work due to the facts that it is publicly available, it has a considerable amount of data, it has a benchmark on classification and it was translated to the Portuguese language (as described in [6]). It constitutes a powerful tool for question classification.

3.3. Multieight-04 Corpus

The Multieight-04 corpus [13] was a corpus built in the context of CLEF 2004 QA track. The focus of this track was multilingual information access. One example of an application of this track, which is also mentioned in the paper cited above, is, for instance, automatic translation of texts. Having this purpose, there was a need for a multilingual corpus to support this task. From this need, the multieight-04 corpus was built having 700 questions translated to 8 languages (one of them being Portuguese) in a XML format. Each of these questions also has an answer (a very objective answer, containing only what was asked without proper sentence formation, for example: the question “Com quem se casou Michael Jackson?” (“Who is Michael Jackson married to?”), the answer is just “Lisa Marie Presley”) and an answer type which is what can be expected to be found on the answer out of 8 categories: person, location, organization, other, measure, time, object and manner.

For this work, slight adaptations were made to this corpus. First, we parsed the XML file to obtain only the Portuguese questions and answers as this work is focused solely on the Portuguese language. Then, using a model trained in the Portuguese Li & Roth corpus, we predicted the labels that the questions has within that taxonomy. Upon having the predicted labels, these were manually corrected when the author of this work considered that they were wrong. This produced a corpus that had both questions and answers and also labels. The interest of adding Li & Roth’s labels to this corpus stems from the fact that the questions encompassed within this corpus and its answers are factoid and thus, can be labeled with this taxonomy. As we already have a corpus with 5000 questions to train a model using this taxonomy, we can have a much more accurate label for a question. With the multieight corpus, we can now have labels for the answers and compare them with the labels for the questions obtained from the previously trained model with the Li & Roth corpus.

3.4. SubTle Corpus

The SubTle corpus [3] was a corpus built from movie subtitles, where the movies were selected with basis on IMDB[7] lists, and the subtitles for those movies were requested to OpenSubtitles[8]. The criteria for the selection of subtitles was their format and their average user rating. For our work, we are especially interested in question/answer pairs due to our answer ranking component. Upon some analysis of this corpus, we decided that it would be interesting to save the pairs I/R (Interaction/Response) that were a question and its answer. With this in mind, we used a model trained in the Question vs. Non-Question mentioned above, in Section 3.1, to predict which sentences were questions. We also used a rule that checked whether a sentence ended with a question mark or not. We only considered questions that were in the “Interaction” part of the subtitle, so that we could retrieve the “Response” part as being the answer to such question. Upon identifying all of the pairs I/R that were questions and answers, we wrote them in a new file forming a new corpus that contains pairs question/answer. We then made a subset of this corpus that contains only questions that were deemed personal by the model trained with the corpus mentioned above, in Section3.1.

4. Chattuga

In this section, we describe how this project is built. In section 4.1 is provided an overview of the whole system, in section 4.2 we describe the baseline of our work, in section 4.3 we talk about the plug and play module of Chattuga, in section 4.4.1 we refer the techniques we used to perform the task of classification, in section 4.4.2 we detail the module of answer classification and lastly, in section 4.5, we explain how the Coordinator decides on the answer ranking.

4.1. Chattuga Overview

Chattuga has three main components which are all managed by the Coordinator. These components are the plug and play module (section 4.3), the query classification module (section 4.4.1), and the answer classification module (section 4.4.2). The plug and play module is set up before the Coordinator starts. It gets the information from the configuration files that a technical user prepared. When the Coordinator starts, it sets up the agents. The query classification models are already trained when the Coordinator starts. After the

[6] https://cogcomp.seas.upenn.edu/Data/QA/QC/ (Last Accessed on: 30/10/2019).

[7] https://www.imdb.com/ (Last accessed on: 06/06/2019)

[8] https://www.opensubtitles.org/ (Last accessed on: 06/06/2019)

Coordinator sets up the agents, it prompts the user to insert a query. The user introduces a query, for example, the query "Como te chamas?". This query is routed through the query classification module which provides its predictions for the set of classification labels that this query should have according to the corpora at its disposal. With the base corpora that were detailed in the previous Section (Section 3), these labels would be: "QUESTION", "PERSONAL", among others. The Coordinator then proceeds to send the query to the agents and receive their answers. The answers and the query classification information are sent to the answer classification module which has several approaches to score the answers (detailed later, in section 4.4.2). The Coordinator uses this classification information, which includes the scores attributed by each module of answer classification to the agents answers and it combines them to perform an answer ranking of the answers. When all the information is gathered, the Coordinator provides the answers ordered according to their ranking to the user.

4.2. Baseline

The baseline of our work is the TalKit project (mentioned in Section 1). This work had a multi-agent architecture for query answering that redirected a query to an agent that was an expert in that type of query (for example: a query that was classified as "PERSONAL" was redirected to an agent expert in "PERSONAL" queries). It did not take into consideration the agent's answer. In Chattuga, it is assumed that all agents can, potentially, answer all questions. With that assumption, instead of having just one answer choice method, we have four, described in Section 4.4.2.

4.3. Plug and Play Module

A technical user that wishes to utilize Chattuga in his own tasks can do both external additions to the system, by using the plug and play module to add new agents and corpora, but also control its inner components by turning on and off the answer ranking modules available. In this section, it is detailed how this module works.

4.3.1 Agent Plug and Play Module

To add a new agent, the user simply needs to have a Python file (representing the new agent) that has a method which receives as argument a query and returns an answer to that query. This file must be encompassed in a folder that contains everything that is needed for the execution of the agent. This folder must be located in the "agents" folder of the Chattuga project. Upon performing all of these actions, the technical user must then proceed to add, in the "agents_config.xml" file, the required configurations. Those configurations include the name of the agent, the name of its dialog method, the folder where it is inserted, the labels in which it is an expert and the sentences it uses to "escape" a query (example: the agent replies saying it does not know the answer to a query). The inner representation of this configuration file within Chattuga is a dictionary containing the informations in that file.

4.3.2 Corpora Plug and Play Module

For the dynamic addition of corpora, the technical user needs to have in consideration some factors which are: (a) The type of hierarchy that the corpora being added has. In Chattuga, two types are accepted. The first one follows the same logic as the Li & Roth corpus where there are two labels for the same query and a classifier needs to first classify a query with the first label, and only then classify it with the second label using the classifier trained with the utterances labelled with the first label. The second follows the approach of the TalKit corpora where a label from one corpus indicates whether a second corpus is used (e.g. if a query is not labeled as "QUESTION" by a first corpus that distinguishes utterances that are questions from utterances that are not, it does not make sense to try to label it with a second corpus that classifies types of questions); and (b) The type of corpus being added to the system. For instance, the Monserrate corpus contains almost exclusively questions about Monserrate's palace and labels about Monserrate and it does not make sense to have a question labelled with this corpus if the query is about a generic domain. In the corpora configuration file, there are fields where the technical user should specify those factors (if the hierarchy is of sublabels, of subcorpus and if the corpus is generic or sepcific). Additionally, the technical user has to include the name and the path of the corpus, among other options.

4.3.3 System Plug and Play Module

The third plug and play's module purpose is to provide the possibility to the technical user to choose which methods it uses in the answer ranking process (explained in the next section). If the technical user considers that one of the modules is not useful or it even is prejudicial to his purposes he has this way to turn them off. As there are four available answer classification modules ((1) – query/agent label match module; (2) – personal answers module; (3) – impersonal answers module; (4) – query/answer label match module), the configuration file for the system has four elements, which correspond to the answer ranking methods available. To use one of them, the user just needs to write "True" inside the corresponding element. By default the system uses all three methods. They are all set to "True". Additionally, for the personal answers module (explained in the next section), as it has four available methods, the technical user needs to input in the attribute of the corresponding element, a string containing the methods or method which it intends to use separated by "|", in case it uses more than one.

4.4. Classification Module

In Chattuga, there are two main classification modules, which are the query classification module (detailed in section 4.4.1) and the answer classification module (explained in section 4.4.2).

4.4.1 Query Classification

Chattuga, without any modifications to the configuration files, uses Support Vector Machine (SVM) with a linear kernel, using unigrams or unigrams+bigrams as features (experiments conducted in this work have shown that these are the pipelines that scored higher in accuracy. These experiments are detailed later, in Section 5), trained with the Question vs. Non-Question, the Yes/No vs. other corpus, the Personal vs. Impersonal corpus, the Li & Roth corpus and the Monserrate corpus[9] to classify queries. The system also has handcrafted rules, to classify questions as “WH_QUESTION”, “OR_QUESTION” and “LIST_QUESTION”. This rule based classifier is very simple: it only looks for words that may indicate that we are in the presence of one of these questions such as “how” for “WH_QUESTION”; “Mention” for “LIST_QUESTION”; finally, “or” for “OR_QUESTION”.

4.4.2 Answer Classification

To understand, among a set of answers from different agents, which one is the most adequate to a given query, there are several approaches that can be followed. One is related to knowing the agent’s area of expertise (for example, the agent Edgar is an expert in all the labels that concern Monserrate’s palace). In this case, if the query is classified with a label that is related with that expertise, we could assume that an answer from that agent will be more valuable than the other agents’ answers. Another approach would be to have corpora with queries and answers labelled with the same set of labels. Having these corpora and using machine learning algorithms, one could obtain a classification on the query and on the set of agent’s answers and, thus, perform a direct matching. If the label of the query and the label of the answer matched, that answer could be considered more valuable. Without having this kind of data, considering that most labelled corpora available contain only queries, another kind of corpora could be used: a corpora that contains pairs query/answer. This kind of corpora could be used in two manners:

- If the corpora is very large, it could be used with neural networks to obtain a classification of whether that answer is adequate to that query or not.
- If we knew that the corpora had a specific characteristic, for instance, that it has only “PERSONAL” utterances, we could compare the agents answers when the user inserts a query that is considered “PERSONAL”, with the whole corpora using similarity measures. The most similar answer to that corpora would then be considered the most adequate.

In Chattuga, we use all the aforementioned strategies. The way in which we use them is detailed next.

Query Labels/Agent Labels Matching Module The first method of ranking the agent’s answers uses the information it has on the agent’s expertise. From the plug and play module explained before, the labels that each agent has an expertise on are drawn. These have a score that pertains to a measure of “how good” an agent is in answering utterances classified with a certain label. The query is also classified with a machine learning module to get its labels. Once we have both the query’s labels and the agent’s labels, we sum the scores of each agent’s label that matches the query’s label. This will render a final score that will be used to rank the answers. The Coordinator receives these scores and uses them to make a decision on the answer ranking.

Answer Labels/Query Labels Matching Module The module of answer labels/query labels matching is quite similar to the query classification module. The only thing that is done in this module is to use the answer corpora that are inserted in the corpora configuration file to train a model and perform a classification of the agent’s answers with that corpora. As the purpose of adding a corpora of answers is for it to have answers that are annotated with the same labels as one of the corpora of questions, it only gives a score above 0 to an answer if it is found to have labels in common with the ones found for the query in the query classification phase. If there are labels in common, the answer being analysed receives 1 point for each label it has in common with the query. For example, the query “Quantas janelas tem o palácio?” (“How many windows does the palace have?”) is classified with, among others, the label “MONSERRATE_INTERIOR” from Edgar’s corpus, and the answer “O palácio tem vinte e cinco janelas ogivais nos seus dois pisos principais e sete janelas mais pequenas ao nível do piso superior dos torreões.” (“The palace has twenty-five arched windows on its two main floors and seven smaller windows on the towers’ top floor.”) is classified with the same label. In this module, this answer would receive 1 point.

Impersonal Answers Module This module uses the multieight corpus described earlier, in section 3.3. However, as this corpus is very small, its accuracy cannot be relied to obtain a classification. For this reason, it was only used when a classification with rules could not be found. In this module, agent’s answers are iterated over to find clues to ascertain whether they have answered the question or not. To perform this operation, the query’s labels are used, more specifically, the ones that come from the classification with the Li & Roth taxonomy.

The first approach that this module tries is to use the Wikipedia api for Python[10] to send the ngrams of the answers to wikipedia and get a summary of the articles it finds. The impersonal answers module then searches for the fine category obtained in the classification of the query within that summary. The reason for this is that if, for example, a search is performed in Wikipedia about “Girassol” (“Sunflower”) (a plant), usually the summary returned contains the word “plant” which is one of the fine categories of the Li & Roth taxonomy. For fine categories that are more broad such as “cremat” that encompasses books, movies, plays, among others, a list of words to search was compiled (in the case of “cremat”, the aforementioned examples were included). If a

[9] Edgar’s corpus containing informations about Monserrate.

[10] https://pypi.org/project/wikipedia/ (Last accessed on: 27/10/2019)

user inputted the query "Que planta contém a palavra sol no seu nome?" ("Which plant contains the word sun in its name?") and it was classified correctly as "ENTY:plant", if an agent answered "É o girassol!" ("It is the sunflower!"), this module would be able to retrieve the word plant from the wikipedia summary and thus attribute the correct fine category to the answer. If this wikipedia method takes too long or is unable to find the category, the impersonal answer module proceeds to use another module based on the work in [17], which, depending on the coarse category, uses different methods which include using WordNet with the Li & Roth's fine category as hypernym and then look for words in the answer that are hyponyms of it; sing a named entity recognizer to extract entities and check if they are locations or humans as these are categories of the Li & Roth taxonomy; and using regular expressions and rules.

As we could not cover all methods with rules, when a question category is not found with this rule based classification, this module uses a SVM trained on the multieight corpus to perform the classification of the answer.

Personal Answers Module Working with the resources that we possess for the Portuguese language, that is, the corpora presented in Section 3, and due to the lack of corpora to distinguish personal answers from impersonal answers, i.e., corpora with answers labeled with "PERSONAL" and "IMPERSONAL", as we have for questions, which would allow us to compare the answers' labels with the question's labels, we, in this work, tried a method that uses similarity approaches. As we only contain a corpus with one example ("PERSONAL" answers) of answers, we decided to use an approach that simply compares each agent's answer with a corpus that contains personal answers (the subset of the SubTle corpus described in section 3.4). If a query is classified as "PERSONAL" and among the set of answers we can find one that is "PERSONAL" as well, we can give a higher score to that answer. We do this by using some similarity approaches to compare an answer to the corpus of personal answers, as if an answer is similar to a the aforementioned corpus of personal answers, it can be argued that the more similar it is, the most likely it is that that answer is personal. These similarity approaches are described next.

Word embeddings + Mean Embedding Vectorizer + Cosine similarity:
For this method and the following one, we used an implementation (with slight modifications) available online[11] to use word embeddings with Scikit learn functions. For this, the developer made two custom vectorizers, the **Mean Embedding Vectorizer** which computes the mean weights of word embeddings vectors and the **TF-IDF Embedding Vectorizer** which computes the mean values of the TF-IDF weights of the word embeddings vectors. In this section we will focus on the mean embedding vectorizer. This vectorizer simply computes the mean values of the word vectors of the word embeddings model (trained on the personal answers corpus). Upon transforming both the answers and the personal answers corpus, we compare the similarity between them using the cosine similarity.

Word Embeddings + TF-IDF Embedding Vectorizer for Embeddings + Cosine similarity:
Similarly to the previous approach, we use the aforementioned implementation which computes the mean values of the TF-IDF weights as well as the word vectors weights. With the vectors corresponding to the personal answers and the answers transformed with this vectorizer we perform the cosine similarity to determine the closest answer to the personal corpus.

Latent Semantic Indexing (LSI) similarity:
Another approach that can be used to test the similarity between an answer and the corpus of answers is **Latent Semantic Indexing**. Using gensim's implementation of this similarity approach, we first train the model with the corpus of answers, then we create a matrix containing all the similarities that came from the lsi model. Using this matrix, we then compare each agent's answer to this matrix and obtain a similarity for each one of them.

Regular TF-IDF Vectorizer + Cosine similarity:
The last and the default approach of our system is to use the regular TF-IDF vectorizer from Scikit learn, to get the vectors of weights to each word and then perform the cosine similarity. This method is the default because of its speed. It takes three seconds to compute four agent's answers as opposed to the more than ten seconds that the other methods take (they take a little over ten seconds for four agents using parallel computing, otherwise they would take more than that). As this method is fairly good at performing the similarity, we considered that it could be a default method considering the trade off time/effectiveness.

4.5. Coordinator's answer ranking decision

To return a ranked list of answers to the user, the Coordinator checks the system's configuration file to check which answer classification modules it can use and, in case the personal answers module is active, the similarity measure or measures it can use. With this information, the Coordinator calls these modules and sums the different scores it receives from each one of them. The answers are showed to the user ordered by their score, from highest to lowest. In case there is a draw, this information is given to the user as well.

5. Results & discussion

The Chattuga framework consists of several functionalities which connected make a meta-conversation system. As the focus of this work is not the agents that integrate it, in this evaluation we focus on its true functionality: how good is Chattuga in the **ordering of answers according to their relevance** and how good is it in the **integration of new agents and corpora**. Thus, we have three main characteristics on which we base this evaluation:

- **The query classification module** – In this mod-

[11] http://nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/ (Last accessed on: 09/09/2019)

ule, the user's intent is classified and it is further used to discern between the agent's answers. This helps the system in the **ordering of answers according to their relevance**.

- **The plug and play module** – This module has three parts: the agents plug and play part to add agents; the corpora part which is to add corpora; finally, the system part that controls which answer ranking methods are used. The first two parts of this module (agents and corpora addition) will allow the system in the **integration of new agents and corpora** and the third part (system) will help the system in the **ordering of answers according to their relevance**.

- **The answer ranking process** – This process is consisted of four modules of answer classification that each produce a score for each agent's answer which are then summed by the Coordinator. The system uses this information to perform the **ordering of answers according to their relevance**.

In sections 5.1, 5.2 and 5.3 we describe how the evaluation was performed in each of these modules.

5.1. Query Classification

To evaluate the performance of the different corpora (Question vs. Non-Question, Yes/No vs. Other, Personal vs. Impersonal and Li & Roth) we conducted experiences in the suggested partition of train and test corpus with several scikit-learn[5] algorithms of classification, all the vectorizers and with these together with the TweetTokenizer of NLTK [4]. We experimented with Unigrams, Bigrams, Trigrams and combinations of them. When possible we maintained the default parameters of the classifiers and of the vectorizers. The algorithm that achieved the best results in all the corpora was the SVM with linear kernel. The results are 100% accuracy for the Question vs. Non-Question Corpus and for the Yes/No vs. Other corpus and 82.4% accuracy for the fine categories of the Li & Roth corpus with unigrams as features, the vectorizer CountVectorizer and the classifier LinearSVC; 98.1% accuracy for the Personal vs. Impersonal corpus with unigrams+bigrams as features, the classifier SVC+linear kernel and the vectorizer TfIdfVectorizer; lastly, 90.5% accuracy for the coarse categories of the Li & Roth corpus with unigrams+bigrams as features, the classifier LinearSVC and the vectorizer TfIdfVectorizer. Additionally, using the algorithms, features and vectorizers found to be the best according to these experiences, we performed a cross validation with 10 folds on these corpora (for the cross validation, we used the whole corpus, that is, the test and train corpus together for each corpus). The results are for the Question vs. Non-Question corpus: 100.0%; for the Personal vs. Impersonal corpus: 97.0%; for the Yes/No vs. Other corpus: 97.0%; for the Li Roth corpus coarse categories: 86.0%; lastly, for the Li Roth corpus fine categories 78.0%.

The accuracies obtained show that the system has a strong module of classification for the labels in these corpora. Some of these corpora are quite small and the results obtained can be high due to that fact. The values obtained for the accuracy are quite similar to the one's obtained in TalKit [7]. However, a direct comparison cannot be done due to the fact that the cross validation in that work was done solely on the train corpus whilst the values presented in this work were obtained for the combination of the train and test corpus.

5.2. Evaluating the Plug and Play Module

To evaluate the first two parts of this module, we have considered a case study which is Edgar. We built the entire system which started by containing only the chatbots Talkpedia and SSS. We then decided to get a highly specific chatbot as is Edgar and add it to the system to see how it adapts. In section 5.2.1 we describe the experience of adding the agent Edgar, in section 5.2.2 we detail how it was to add corpora and finally, in section 5.2.3 we detail some experiences made with the parameters of the system configuration file.

5.2.1 Evaluating the Agents Plug and Play Module

The first thing that had to be done to integrate Edgar in Chattuga was to run it as a standalone agent. The agent was ran and, in doing that, the required steps to invoke the agent and what libraries were needed to run it were identified. When the agent was ran successfully and a conversation with it was held, an investigation had to be done to see how the agent could be adapted so that it would receive a single query instead of being always running and asking for more queries. This is because the only agent that should be asking the user for a query is Chattuga. After performing the required adaptations to Edgar's code, we had to build a wrapper in Python for Edgar's code as it was made in Java. The hardest part was performing all these adaptations to Edgar's code and understanding its architecture and its requirements. The construction of the Python wrapper, after having an example from the other two agents did not prove to be difficult. The actual part of the plug and play, which is to add the agent to the config file, was straightforward: all the required arguments were written in the config file and when the program was ran, the agent immediately started answering questions alongside the other agents.

5.2.2 Evaluating the Corpora Plug and Play Module

The original corpora of the Edgar project was built in a XML format where each file corresponded to a question about a different part of Monserrate's palace or about the butler itself (a different category of question). Chattuga requires a csv file or a txt file in a different format. The first step to integrate this corpus in the system was to transform it into this format. To do this operation, a parsing of the xml files was done. The questions and the category were retrieved and all the files of Edgar were combined in a single txt and a single csv file. The path for these files and the remaining informations required by the plug and play module were added to the corpora config file, including that this corpora is of type "specific" due to its restrict domain. Chattuga was ran after the configurations were added to the corpora config file and the labels of this corpora were promptly returned for the queries that were introduced as part of the experiment.

5.2.3 Evaluating the System Plug and Play Module

To evaluate this module, we introduced a query that we knew the agent Edgar could answer, and then, with the same query we tried different combinations of settings in the system config file. In this way, we can assert how well this module works. The query that was first inserted in the system was “Quantas janelas tem o palácio?” (“How many windows does the palace have?”). This led the system to choose Edgar’s answer as the best one with a score of 3.4 (2 points from the query-agent label match module; 0.5 points from the impersonal answers module; 1 point from the query-answer label match module). Disabling the modules one by one (this is done by setting them to False in the config file), the scores varied as expected, that is, if we disabled the query-agent label match module Edgar’s score would drop 2 points, if we disabled the impersonal answers module the score would decrease 0.5, if we disabled both, 2.5, and so on and so forth.

Another setting that can be changed in this config file is the similarity measure or measures that are to be used to obtain a score to personal answers. The first experiment that was conducted was to use all similarity measures (to perform this experiment and the following experiments, we only activated the personal module as that was the focus of these experiences) and then run the system with the query “Como te chamas?” (“What is your name?”). All combinations of similarity measures were then tried and the scores showed that all measures chose the same answer as the best one, although the scores were different depending on the similarity measure. It also showed that this module was working as expected as the scores varied with different combinations of similarity measures.

5.3. Evaluating the Answer Ranking Process

One of the most important jobs of Chattuga is its ability to detect among a set of agent’s answers, the one or ones that are more relevant to a query. In information retrieval, it is studied the retrieval of relevant documents to a given query. These techniques are mostly used in search engines to order the results found according to their relevance. As the system discussed in this document has an analogous function in the sense that it intends to order its results (answers) according their relevance, to evaluate the Chattuga’s answer ranking capabilities, we have decided to compute the Mean Reciprocal Rank[18] which is one of the metrics used in the field of information retrieval to evaluate a system’s effectiveness in ordering the retrieved documents according to their relevance. This metric will help us establish the system’s performance benchmark. As we do not have a corpus that establishes the relevance of each answer to each query inserted in the system, we have asked three users to rank the answers given to the system according to a scale of 1 to 5. We have randomly selected 100 queries from various sources. The queries that we used to perform the evaluation of Chattuga’s answer ranking process were randomly sampled queries (25 impersonal corpus questions; 25 personal corpus questions; 10 Edgar’s corpus questions; 10 AMA’s corpus questions; 10 queries from the non-questions corpus; 5 queries such as “bom dia” (“good morning”), “boa tarde” (“good afternoon”), etc.; 5 yes/no corpus questions; 5 or questions; 5 list questions.) We then took these ratings given by the annotators as the correct ones (to calculate the Mean Reciprocal Rank) and calculated the Mean Reciprocal Rank between the rating of each annotator and the rating the system gave to each answer. If there is more than one answer graded with a one, we just compute the reciprocal rank for the first one to be rated with a one. The results for each annotator are: Annotator 1: 68.2%; Annotator 2: 63.1% ; Annotator 3: 51.0%.

From the answers selected as the most relevant for each query out of the 100, some were answers stating that the chatbot was unable to find an answer to that particular query. Examples of such answers are “Não sei responder a isso” (“I do not know how to answer that”), “Essa não é a minha área de especialidade” (“That is not my area of expertise”), among others. The number of queries to which such answers were deemed more relevant per annotator were: Annotator 1: 46; Annotator 2: 57; Annotator 3: 62. And for Chattuga they were 65.

These results show that for half (or more) the queries selected, both the annotators and the system were unable to find a satisfactory answer among the ones returned by the different conversational agents inserted in the system. The answers in which there was more disagreement between annotators and system were the ones provided by Talkpedia and by the Cheat agent. This is due to the fact that in the agents config file, Talkpedia was given a big score in the expertise of the labels “IMPERSONAL” and “QUESTION”. This agent, however, was not able to deliver any viable answer to any of the queries. Regarding the Cheat agent, due to the fact that it is based in very simple rules without “artificial intelligence”, it was given a low score in its expertise on the labels “OR_QUESTION” and “YN_QUESTION” causing it to have a low score against the more intelligent agents.

5.4. Discussion

With this evaluation, we have concluded that Chattuga has a strong query classification module due to the high accuracy achieved in all the main corpora. Some of these accuracies are quite high which can be caused by two factors: either the amount of data is insufficient and with low diversity causing the algorithm to easily learn the patterns of the data or it is not very hard to distinguish the utterances of the different labels for a computer algorithm. In what concerns the plug and play module, there are some difficulties inherent to adding a new agent and corpora but these cannot be surpassed as they are related to the fact that systems and corpora are built in different ways and, to be integrated in another, some adaptations have to be made. However, it did not took a great implementation effort to make such modifications and after these were made, the only thing that remained to be done was to add some configurations in the corpora and agent’s configuration files. Lastly, the answer ranking module evaluation shows that, when the agents can provide an answer, the ranking process behaves as expected by giving the right scores to the right answer.

6. Conclusions

In this work, we proposed a different approach to multi-agent query answering where it is assumed that all agents can, potentially, answer all queries. To build such an architecture, the system is composed of a set of modules that contribute to a final choice on the answer ranking process. Additionally, this architecture is plug and play in regard to (a) external components: more agents and corpora can be smoothly integrated in the system; and (b) internal components: specific system modules to rank answers can be disabled. The modules proposed and used in this work are (a) the query classification module which provides the labels for the user inserted query; (b) the answer ranking modules: one that takes into account the agent's expertise; one that does a classification of answers to queries labelled as "IMPERSONAL"; one that uses similarity measures to compare answers to queries labelled as "PERSONAL" to a corpus of personal answers; lastly, one that uses machine learning to classify answers with labels and matches it to the queries labels; and (c) the plug and play module which has three sub-modules: agents, corpora and system. These modules all provide a score which is the basis for the answer ranking decision.

References

[1] B. Aarts. *Oxford Modern English Grammar*. Oxford University Press, 2011.

[2] D. Ameixa. Say Something Smart - ensinando um chatbot a responder com base em legendas de filmes. Master's thesis, Instituto Superior Técnico, 2015.

[3] D. Ameixa, L. Coheur, and R. A. Redol. From subtitles to human interactions: introducing the subtle corpus. Technical report, Tech. rep., INESC-ID (November 2014), 2013.

[4] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[5] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[6] Â. Costa, T. Luís, J. Ribeiro, A. C. Mendes, and L. Coheur. An english-portuguese parallel corpus of questions: translation guidelines and application in smt. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA), 2012.

[7] C. Dias. TalKit - Desenvolvimento de um Sistema de Diálogo para Português. Master's thesis, Instituto Superior Técnico, 2015.

[8] M. G. Fernandes, C. Dias, and L. Coheur. Distinguishing Different Classes of Utterances - the UC-PT Corpus. In R. Rodrigues, J. Janousek, L. Ferreira, L. Coheur, F. Batista, and H. G. Oliveira, editors, *8th Symposium on Languages, Applications and Technologies (SLATE 2019)*, volume 74 of *OpenAccess Series in Informatics (OASIcs)*, pages 14:1–14:8, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[9] P. Fialho, L. Coheur, S. Curto, P. Cláudio, Â. Costa, A. Abad, H. Meinedo, and I. Trancoso. Meet edgar, a tutoring agent at monserrate. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66. Association for Computational Linguistics, 2013.

[10] E. Hovy, L. Gerber, U. Hermjakob, C.-Y. Lin, and D. Ravichandran. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*, 2001.

[11] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[12] X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[13] B. Magnini, A. Vallin, C. Ayache, G. Erbach, A. Peñas, M. de Rijke, P. Rocha, K. Simov, and R. Sutcliffe. Overview of the clef 2004 multilingual question answering track. In C. Peters, P. Clough, J. Gonzalo, G. J. F. Jones, M. Kluck, and B. Magnini, editors, *Multilingual Information Access for Text, Speech and Images*, pages 371–391, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[14] P. Mota. LUP: A Language Understanding Platform. Master's thesis, Instituto Superior Técnico, 2015.

[15] M. Pereira. Just.chat - dos sistemas de pergunta/resposta para os chatbots. Master's thesis, Instituto Superior Técnico, 2015.

[16] J. Santos. Say Something Smart 3.0:A Multi-Agent Chatbot in Open Domain (Under Preparation). Master's thesis, Instituto Superior Técnico, 2019.

[17] J. Silva. QA+ML@WikipediaGoogle. Master's thesis, Instituto Superior Técnico, 2009.

[18] E. M. Voorhees. The trec-8 question answering track report. In *In Proceedings of TREC-8*, pages 77–82, 1999.