UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT OF A COGNITIVE BOT FOR DATA SCIENCE TUTORING BASED ON A BIG DATA NATURAL LANGUAGE ANALYTICS PLATFORM

DANIEL CARLANDER-REUTERFELT GALLO

JUNIO 2019

TRABAJO DE FIN DE GRADO

Título:	Desarrollo de un Bot Cognitivo para el aprendizaje de Data Science basado en una plataforma de análisis de lenguje natural y Big Data
Título (inglés):	Development of a Cognitive Bot for Data Science tutoring based on a Big Data Natural Language Analytics Platform
Autor:	DANIEL CARLANDER-REUTERFELT GALLO
Tutor:	CARLOS A. IGLESIAS
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF A COGNITIVE BOT FOR DATA SCIENCE TUTORING BASED ON A BIG DATA NATURAL LANGUAGE ANALYTICS PLATFORM

DANIEL CARLANDER-REUTERFELT GALLO

Junio 2019

Resumen

La aplicación de técnicas de procesado de lenguaje natural para mejorar la interacción con usuarios humanos es una tendencia que no cesa de crecer. Los avances en computación cognitiva suponen una nueva manera de interacción con el usuario que permite el acceso a fuentes de información de manera más refinada.

El proyecto está centrado en el diseño e implementación de un bot conversacional integrado en una interfaz web accesible desde cualquier terminal. Su función principal es la resolución de dudas y preguntas relacionadas con el aprendizaje de técnicas de Aprendizaje Automático y Ciencia de Datos.

Para ello se ha desarrollado una interfaz conversacional basada en API externa, así como módulos desarrollados íntegramente. Incluye un módulo de pregunta-respuesta para el tema de este dominio específico. Además, para la persistencia se ha usado ElasticSearch como base de conocimientos.

El proyecto hace especial hincapié en el desarrollo de conversación ligera, porque supone un aumento de la usabilidad y el atractivo del proyecto. Además, se entrena un modelo de Aprendizaje Automático para detectar las intenciones del ususario.

Los resultados demuestran que, como se supuso, la implementación de conversación ligera crea una mayor puntuación del bot por parte de los usuarios. Además, es un proyecto innovador por integrar tecnologías "estado del arte" como son los chatbots y las plataformas e-learning.

Las implicaciones de este estudio presentan que la futura investigación y desarrollo de estas plataformas puede suponer una mejora de los métodos de aprendizaje aplicable a todas las áreas.

Palabras clave: Computación Cognitiva, ElasticSearch, Dialogflow, Ciencia de Datos, Aprendizaje Automático, Chatbot, Agente Conversacional

Abstract

The application of natural language to improve the interaction of human users with information systems is a growing trend in the recent years. Advances in cognitive computing enable a new way of interaction that accelerates insight from existing information sources.

In this project, a modular cognitive agent architecture for question answering featuring social dialogue (small talk) improved for a specific knowledge domain is designed and developed. The proposed system has been implemented as a personal agent to assist students learning Data Science and Machine Learning techniques. To that end, a responsive web interface was developed. Also, a Knowledge Base was provided with all the necessary information.

The developed prototype has been evaluated to analyze how users perceive the interaction with the system. We claimed that the inclusion of social dialogue results in better responses and engaging experiences to users.

In the end, the evaluation results that support our hypothesis are presented, as well as thoughts in future work.

Keywords: Cognitive Computing, Machine Learning, Conversational Agent, ElasticSearch, Dialogflow, e-learning

Agradecimientos

Me gustaría dar las gracias a mis padres por haberme ayudado durante mi vida. Gracias a ellos en gran parte he llegado a ser lo que soy.

Gracias al Grupo de Sistemas Inteligentes por haberme dado la oportunidad de desarrollar este proyecto y en concreto a mi tutor Carlos A. Iglesias por orientarme y guiarme en el proceso.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Context

The use of virtual assistants has grown in the last few years, boosting the research and development on the topic. The biggest firms have invested in these technologies with projects such as Google Assistant[1], Siri[2] and Amazon Alexa[3].

Many of these agents are now included in mobile devices such as iPhone and Android phones, which everyone owns nowadays. It is also possible to purchase specific platforms such as Google Home, Apple's Homepod and Amazon Alexa. As shared with The Verge, Amazon sold 100 million Alexa devices in the four years it has been live[4]. So it is clear that it is in the rise.

Conversational agents have evolved from very simple systems that were very guided into rather complex functionalities including Natural Language Understanding and Machine Learning Techniques which allowed them to be more flexible in maintaining a conversation.

[1] https://assistant.google.com

[2] https://www.apple.com/siri/

[3] https://developer.amazon.com/alexa

[4] https://www.theverge.com/2019/1/4/18168565/amazon-alexa-devices-how-many-sold-number-100-million-dave-limp

Everyday more businesses include Chatbots as a way to interact with the consumer to answer requests and FAQs. Natural language interfaces (NLI) increase user satisfaction and can help to find the information needed in a more comfortable way than other less sophisticated and time-consuming search interfaces [2].

In order to understand natural language, techniques such as grammars, statistics or lexical analysis are required to identify the relevant information in an organized way.

According to IBM, *Cognitive computing refers to systems that learn at scale, reason with purpose and interact with humans naturally* [3]. It follows the steps in human reasoning such as observation and examination of data, analysis of that data and interpretation.

Like humans, cognitive system can use already known information to deduce new meaning in other data based on context [4]. By having the advantage of computational power, a system like this can be even more successful than a human in this kind of task. Though they do not really understand the meaning like humans do, they insights these systems provide can be really useful. As they grow in time, it is expected that they gain abilities such as sensing and awareness [5].

Cognitive computing applications should be adopted for learning purposes [6] because:

- It can strongly enhance student's performances in computer science classes
- *Studying cognitive computing behavior can lead to significant results in AI related studies.*
- *Using a cognitive computing layer for digital interactions with students can enhance their performances and ease the teachers' job in managing classes and learning materials.*

Compared to other traditional e-learning training, chatbots generate a more positive response from the users [7]. There are advantages in this type of learning that need to be realized in many schools, such as interaction, active learning and sociability [8].

In a conversation, speech act types describe the purpose or function of a certain instance of speech [9]. This provides an insight into the conversation that can be useful to the chatbot in order to react in one way or another. Virtual assistants must be able to mimic the process of detecting speech acts and classifying them [10]. This kind of classification requires examples of speech correctly classified following a scheme [11] from which to learn. To achieve this understanding the use of Big Data analysis techniques is required.

1.2 Project goals

The aim of the project is to develop a cognitive bot that allows the user to ask Data Science related questions and get an appropriate answer. The aim is for it to also be able to engage in small talk to give a more human interface.

To achieve this, Natural Language Analytics tools are availed. A machine learning model will be trained to understand user intentions in speech. A more advanced system will be developed to attend Question Answering (QA). Thus, the project goals can be listed as follows:

- Respond to the user queries with the desired information
- Retrieve the necessary information from the data source.
- Classify Speech Act types correctly
- Handle small talk effectively and as human as possible.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is the following:

Chapter 1 is the introduction of the project. It provides the context in which the project is developed and what the main goals are.

Chapter 2 explains the technologies that have been used to support all the work and the reason why they have been chosen.

Chapter 3 describes the architecture of the project including the multiple modules it is composed of.

Chapter 4 describes a series of use cases of the product.

Chapter 5 includes the evaluation of the product.

Chapter 6 includes the conclusions, the goals achieved and the future work.

CHAPTER 2

Enabling Technologies

In this chapter, the technologies that have been used throughout the project are introduced and detailed. These will be for the most part programming languages, libraries and mathematical models because the project involves web and server architectures and machine learning techniques.

2.1 Python Libraries

The majority of the development was made with the help of Python because of its versatility. It's a language with a very expressive and intuitive syntax which gives the importance to the idea. *With additional basic tools, Python transforms into a high-level language suited for scientific and engineering code that's often fast enough to be immediately useful but also flexible enough to be sped up with additional extensions.* [12] There are lots of libraries that facilitate the tasks needed for this field of study. Among those libraries are:

- **Numpy**

  Numpy[1] is the fundamental package for scientific computing with Python. It is widely

[1] http://www.numpy.org

used when you need to go go beyond the structures included in Python. It includes tools such as N-dimensional Arrays which can be useful when managing more complex data structures. This library is also a dependency for the libraries explained next such as Pandas and Scikit-Learn.

- **Pandas**

  Pandas[2] is an open source, BSD-licensed[3] library providing high-performance, easy-to-use data structures and data analysis tools for Python. It is one of the most preferred and widely used tools in data munging if not the most used one.

  Among the tools it provides, the ones useful to us are DataFrames. A DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. It facilitates a generalization among different formats of datasets such as XML and csv, which are the most common. The other important aspect is that it makes it very easy to apply preprocessing to each instance of data with low complexity. These tools are very useful in managing the training data for a machine learning model.

- **Natural Language Toolkit**

  The Natural Language Toolkit (NLTK)[4] is a leading platform for building programs to work with human language data. It helps to categorize text, analyze linguistic structure, and more. The toolkit is efficient enough to support meaningful tasks, but to achieve a greater performance and to reduce complexity for runtime performance it would require implementations in a lower level language such as C or C++. This would make the library more difficult to understand.

  NLTK provides corpora resources as well as processing tools, all with extensive documentation. The dataset used in the Speech Act Classifier is included among its corpora resources. The principal language processing tools are also from this library. Among these tools are tokenization and stemming. Other important tools from this library are now explained:

  – *Part-of-Speech-Tagging.* POS Tagging is a typical NLP task which consists of assigning a proper tag to each word depending on the context and the function. Some examples of these tags are Determiner (DT) and Noun (NN). This kind of information can be very useful in the understanding of the meaning of a certain message.

[2] https://pandas.pydata.org

[3] BSD licenses are a family of permissive free software licenses, imposing minimal restrictions on the use and distribution of covered software.

[4] https://www.nltk.org

- *N-gram Similarity.* Unigram similarity is a widely used string similarity measure. As a word is compared the aim is to find common subsequences. The objective of N-gram similarity is to generalize the idea of unigram similarity to a number of n-grams, rather than just unigrams [13]. By doing this the context is taken into account.
- *Term Frequency Inverse Document Frequency. TF-IDF is an efficient and simple algorithm for matching words in a query to documents that are relevant to that query.*[14] Although it doesn't take into account relationships between words such as synonyms, TF-IDF is simple and efficient as well as effective in finding similarities.

- **Scikit-learn**

  Scikit-learn[5] is a free software module for machine learning built on top of SciPy, wich is an Python ecosystem for scientific and mathematical use. It features multiple of the most common classification, regression and clustering algorithms. It's very flexible for research and clean, while being very easy to use.

  It is designed to be compatible with the use of the Pandas DataFrames and Numpy arrays directly into the Machine Learning algorithms. Scikit-learn doesn't provide preprocessing tools, which is a very big step in the pipeline of machine learning. Because of this, the use of Pandas as a previous step in the pipeline is a very powerful combination.

  Scikit-learn comes with a wide variety of machine learning algorithms, supervised and unsupervised. It makes it possible to develop models in a few minutes. It also provides tools for model selection, such as Grid Search or vectorizers.

  `GridSearchCV` is a powerful tool used to fine tune parameters of a certain model. This means that the parameters used by an algorithm is usually selected with no prior knowledge of the impact on the model. By defining which parameters to try in a parameter grid, the algorithm tries all possible combinations of parameters and then performs a cross-validation to get a score and select the best parameters for the application at hand.

  A vectorizer is a tool usually used in Natural Language Processing applications. In this particular field it is used to transform a sentence into a vector. This works by first defining a dictionary (`['car', 'have', 'love']`). Then the sentence "I love my car" would be fitted into that dictionary in this way `[1, 0, 1]`.

[5] https://scikit-learn.org

- **Flask**

  Flask[6] is a licensed microframework for python. Multiple companies use Flask such as Airbnb, Netflix or Uber. It is also the choice for people who are learning because the are no limitations to getting a simple app or request handler up and running. It is considered more fit into the Python guidelines because the code is more explicit.

2.2 ElasticSearch

Elasticsearch[7] is a distributed, RESTful search and analytics engine. It is widely used because it allows the exploration of data at a great speed and scale. Companies like GitHub and Wikipedia use it [15].

ElasticSearch can be used for many types of search and analytics. Unlike other data storage technologies like relational databases which are structured, ElasticSearch works like a full-text database.

Full-text means that it includes complete texts of books, articles and other data that can be scrapped from the web. It makes possible *Full-text search*. This type of search examines all the words in a text for a specific search criteria. With little amounts of data or documents it is possible to scan all documents for the query. On the contrary, if the quantity of documents is very large, there is a need for indexing.

Indexing is the process of scanning the documents for search terms and generating an index (or concordance). Thanks to the index, when performing a search, the engine searches in the index and then into the document. This results in faster searches.

When performing Full-text search there is a term that does not appear in structured search. This concept is called Relevance. With a bug number of results from the query there is a need to know which of these results are more important. This relevance can be calculated by TF-IDF as previously explained, or another algorithm.

To better understand the inner workings we can draw some comparisons with common relational databases in 2.1.

[6] http://flask.pocoo.org

[7] https://www.elastic.co/products/elasticsearch

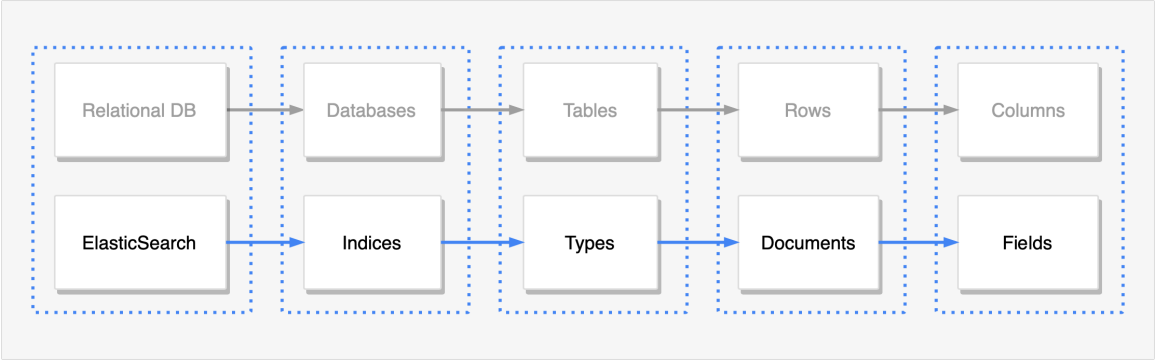

Figure 2.1: ElasticSearch vs. Relational Database

2.3 DialogFlow

Google's DialogFlow[8] is widely used to build conversational interfaces on top of other products and services. Following Google's traditional ease to use and backed by its machine learning products running in Google Cloud, DialogFlow is a very powerful tool for developing conversational agents.

Traditional interfaces usually require a very structured and predictable input to function. They leave no room for ambiguity and thus make these interfaces unnatural and difficult to use. There are endless ways of asking questions such as "What is the weather?". Responding to such a variety of queries requires a very robust system that takes into account all these possible cases.

However, thanks to its splendid handling of Natural Language Understanding (NLU), DialogFlow allows to achieve a broader conversational experience. It uses several techniques that only require to enter very specific training phrases and then learns what similar prompts with the same intent may look like.

By training the model with typical questions with the slots selected, as shown in Figure 2.2, DialogFlow can detect similar prompts and extract the slot that has the valuable information. After having recognized the slot, it can be handled however the user wants. It can be used as a search term in a database, a section in a form or anything imaginable.

When the slot has been recognized it can be sent to a Webhook. This is web service that receives a POST request from Dialogflow with the information it extracted from the query (If a slot has been matched or the intent). This is very useful in order to extend the functionality provided by Dialogflow, allowing to generate custom responses from an

[8]https://dialogflow.com

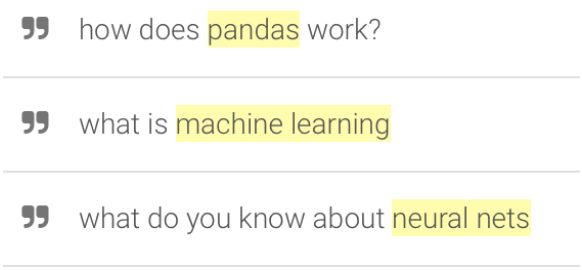

Figure 2.2: Slot Detection in DialogFlow

external API for example.

It also provides prebuilt agents to handle specific tasks such as small-talk, event managing and device commands. These come with multiple intents already trained with common phrases and responses.

2.4 Selenium

The purpose of *Selenium*[9] is to automate browsers. This power can be used to test a web service, automate web administration or as a web crawler. Selenium is open source and used by big enterprises to test their webs including Facebook and Google. The use of Selenium in this project is of web scraping.

Scraping is the process of automating the retrieval of data in the web. It involves fetching a certain web page and extracting data from it. To this end, there is a process of inspection of the DOM structure and then the extraction of information based on that structure.

2.5 Express JS

Express JS[10] is a web framework for Node.js. It's light and fast. It takes advantage of the Model View Controller (MVC) pattern. This means that it separates the logic, presentation and persistence.

- The **Model** contains the data structure and is independent of the user interface. It is responsible for managing the data from the application.
- The **View** is the representation of the information or any kind of interface to the user.

[9]https://www.seleniumhq.org
[10]https://expressjs.com

- The **Controller** uses the input from the user to present a view or to modify the model.

Express is fast and light-weight. It is the most starred web framework in npm [16]. It solves the problems that have developers reinventing the wheel any time they need to create a web interface, such as http parsing, cookie handling and sessions.

2.6 Docker

With the rise of the cloud and the growth of third-party hosting, Docker has became almost a standard in deploying to the cloud.

Docker[11] is designed to allow a developer to generate a low-weight virtual machine with all the components a service needs called a container. It allows the use of the same kernel as the system they reside in. This reduces size and increases performance. Then, platforms such as Google Cloud, Amazon Web Services or the GSI cluster can take the container and deploy it without any complications.

[11] https://www.docker.com

CHAPTER 3

Architecture

3.1 Introduction

This chapter covers the design phase of this project, as well as implementation details involving its architecture. Firstly, it presents an overview of the project, divided into several modules. This is intended to offer a general view of the project architecture. After that, each module is detailed separately and in much more depth. A visual look at the architecture is shown in Figure 3.1.

The system is intended to solve the problem of maintaining a conversation with the user while being able to solve doubts and questions. There are several steps involved in the process and are explained below.

The first step of the process is determining the Speech Act type from the user query, which is determined by the Speech Act Classifier. Then, depending on the output, it is passed onto the Small Talk module in the case that small talk is detected or into the QA Module in the case that a question regarding Data Science is detected.

The Question Answering (QA) Module needs to access a Knowledge Base which has been populated with data scrapped from the internet regarding the topic at hand. The

Knowledge Base is based on ElasticSearch.

Afterwards, the modules generate an answer to satisfy the user request. The answer is sent back to the user and a feedback is collected to evaluate and improve the model.

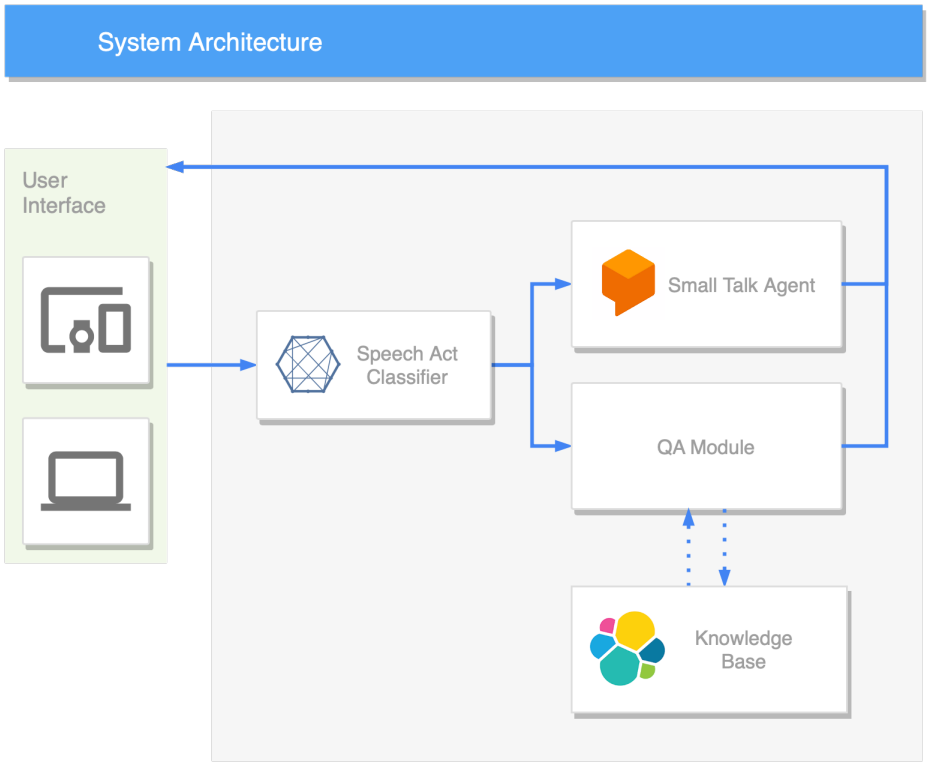

Figure 3.1: System Architecture

3.2 User Interface

The entry point of the whole architecture is the user interface. Having the book judged by its cover is unavoidable. Therefore it needs to be visually attractive as well as easy to use.

The Bot needs an identifier to generate a more personal relationship. Being a Cognitive bot and an intelligent one, with a combination of these ideas it was decided to be called *Just an Artificial Intelligence Cognitive Bot* (JAICOB) as seen in Figure 3.2.

A general purpose bot in contrast with Jaicob, would gain quality from a text-to-speech transformer, giving it a more human appearance. This is not the case of Jaicob because it is centered in answering documentation and programming related questions. The frequent use of acronyms and code examples in the answers would not make for a pleasant listening experience. Instead, the use of text is the best option in this case.

Figure 3.2: Jaicob

Due to the nature of the problem Jaicob solves, which is related to learning and study, the decision regarding the type of interface is important. Telegram is widely used to develop bots and takes away the problem of developing a platform and interface. It provides a full interface with buttons, custom keyboards and the ability to send images and videos. The problem with Telegram is that it is also used for general messaging purposes. This way, having the bot in Telegram elevates the risk of distraction of the user.

Instead, a Web based interface is a better option for a variety of reasons. It improves flexibility and the possibility of personalization. Also, it is universally accessible from any browser and reduces the probability of distraction by means of other messages from Telegram.

The interface is developed with ExpressJS based on botUI[1], a JavaScript framework to create conversational UIs. It provides tools such as buttons, and autocomplete which are used to develop the bot. It was modified to separate the responses from Dialogflow into different messages making a more human way of typing messages. The result interface from a Safari browser can be seen in 3.3

[1]https://github.com/botui/botui

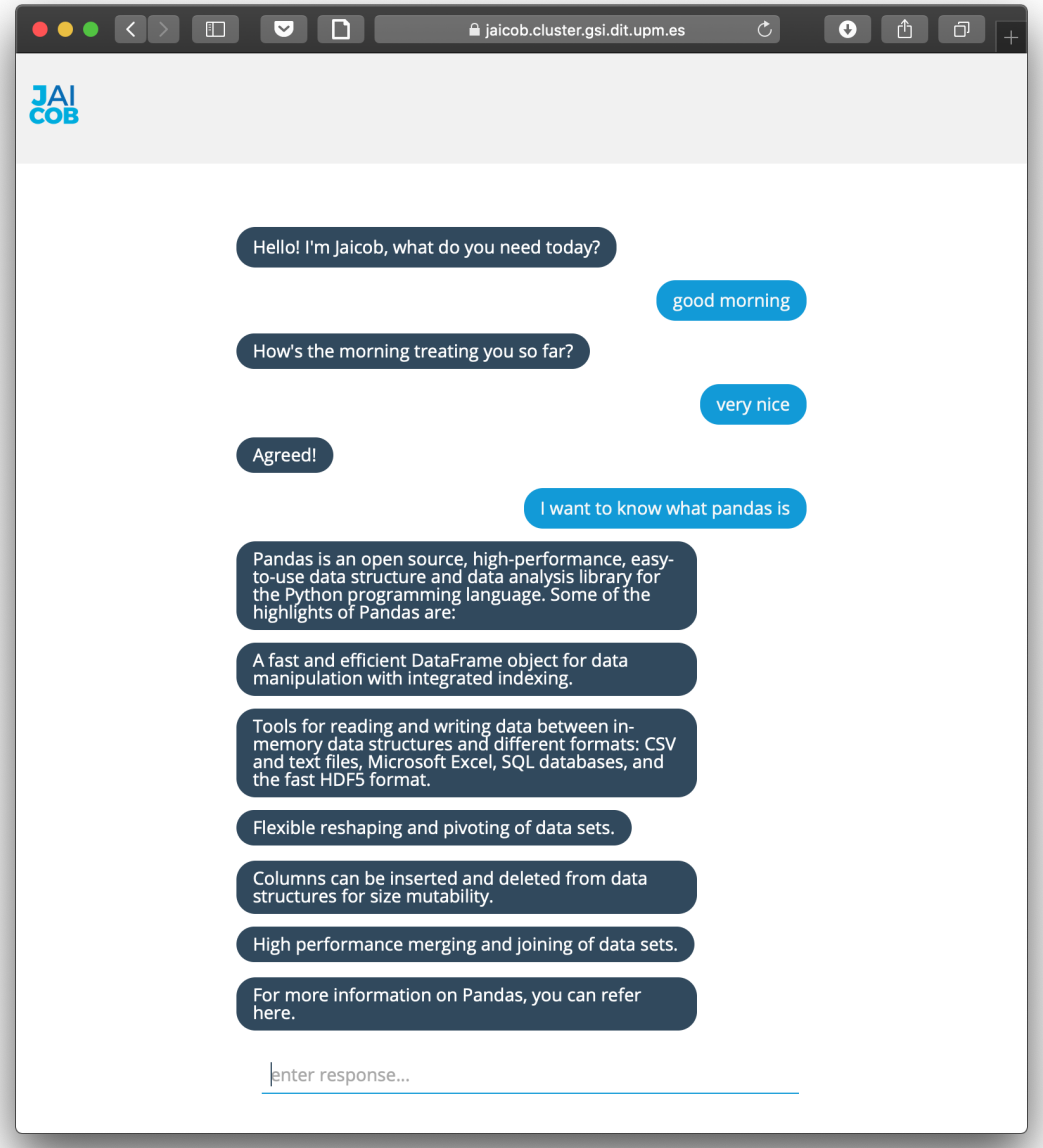

Figure 3.3: Jaicob user interface

3.3 QA Module

The Question Answering module comes into place when the user asks for a specific piece of information. These can range from doubt, a consultation or a documentation clarifications. It must be able to understand what the user is asking for to retrieve the information effectively.

The QA model is an adaptation of a simple Factoid Based Question Answering[2] which, while being very straightforward, obtains up to an 88.51% score in the Stanford Question Answer Dataset (SQuAD)[3]. Using natural language processing techniques, it answers to the question in near real time. This general purpose model is enhanced to attend specific cases to the task at hand, such as code examples.

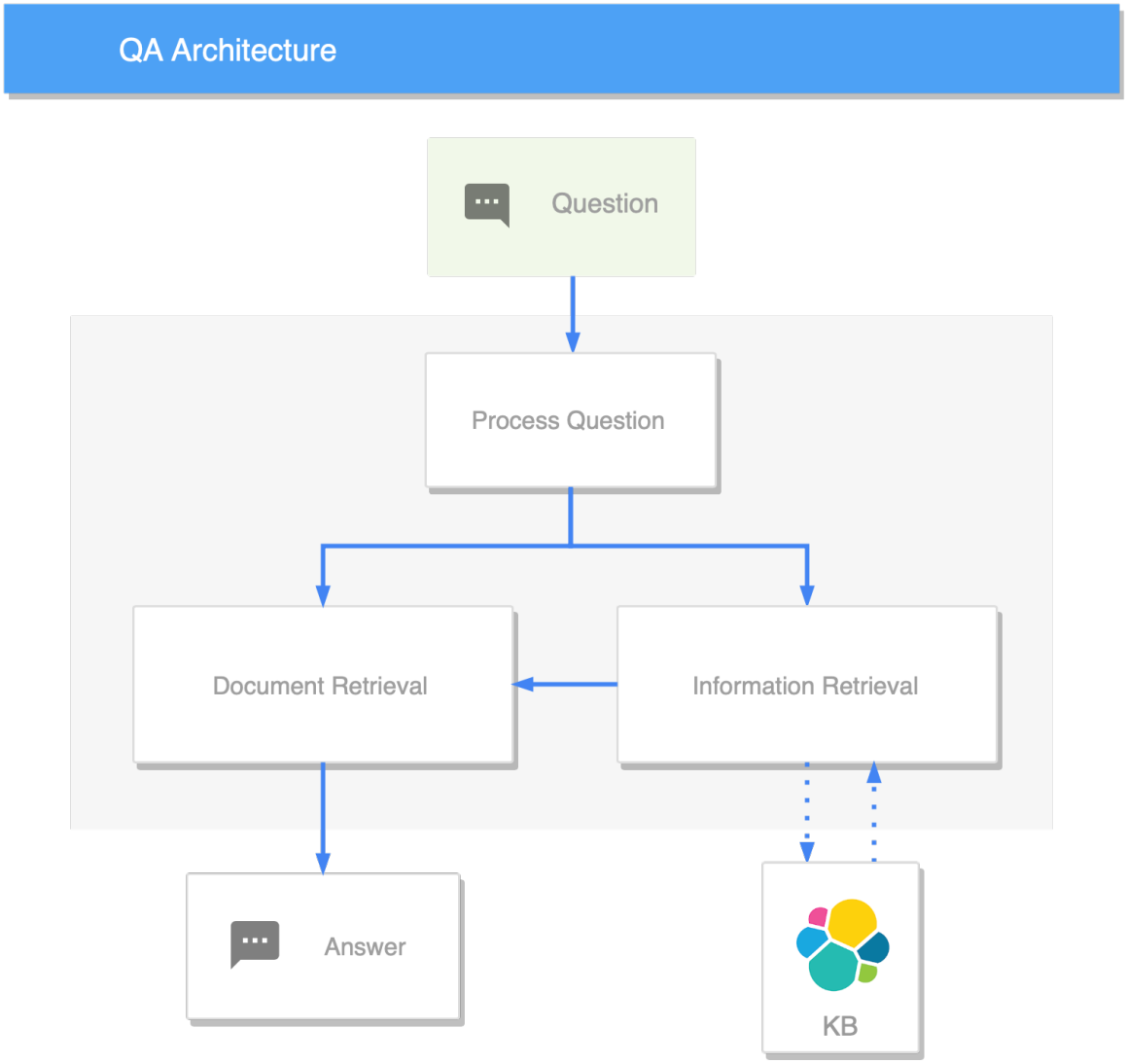

Figure 3.4: QA Architecture

The general view of the architecture is defined in Figure 3.4. This will help to understand how the modules interconnect before going into greater detail.

The ***Process Question*** module extracts the relevant information and intention of the question.

[2]https://github.com/vaibhawraj/Factoid-based-Question-Answer-Chatbot
[3]https://rajpurkar.github.io/SQuAD-explorer/

The ***Information Retrieval*** module extracts the relevant information from the Knowledge Base based on the parameters passed by the Process Question module.

The ***Document Retrieval*** module receives and parses the retrieved information so it matches the questions intent to finally generate an answer.

3.3.1 Process Question

This module receives a query as the input and returns a `ProcessedQuestion` object. The object contains the following information:

- **Question Type.** Depending on the type of question, it falls into one of five categories. These categories are WP (Who), WDT (What, Why, How), WP$ (Whose), WRB (Where) or COMPLEX.
- **Question Vector.** This consists of a python dictionary containing each relevant word (removing stop words and question type word) along with the frequency it appears in the query.
- **Answer Type.** By using the previous information the type of answer that best fits the query is determined. By making use of the other words in the query and POS tagging, it is classified into one of these categories:
  - YESNO
  - PERSON
  - LOCATION
  - DATE
  - DEFINITION
  - ORGANIZATION
  - QUANTITY
  - LINEAR MEASURE
  - EXAMPLE

3.3.2 Information Retrieval

This module receives the `question vector` and the `answer type` from the question processor as an input. The question vector is, in essence, a list of keywords ordered by

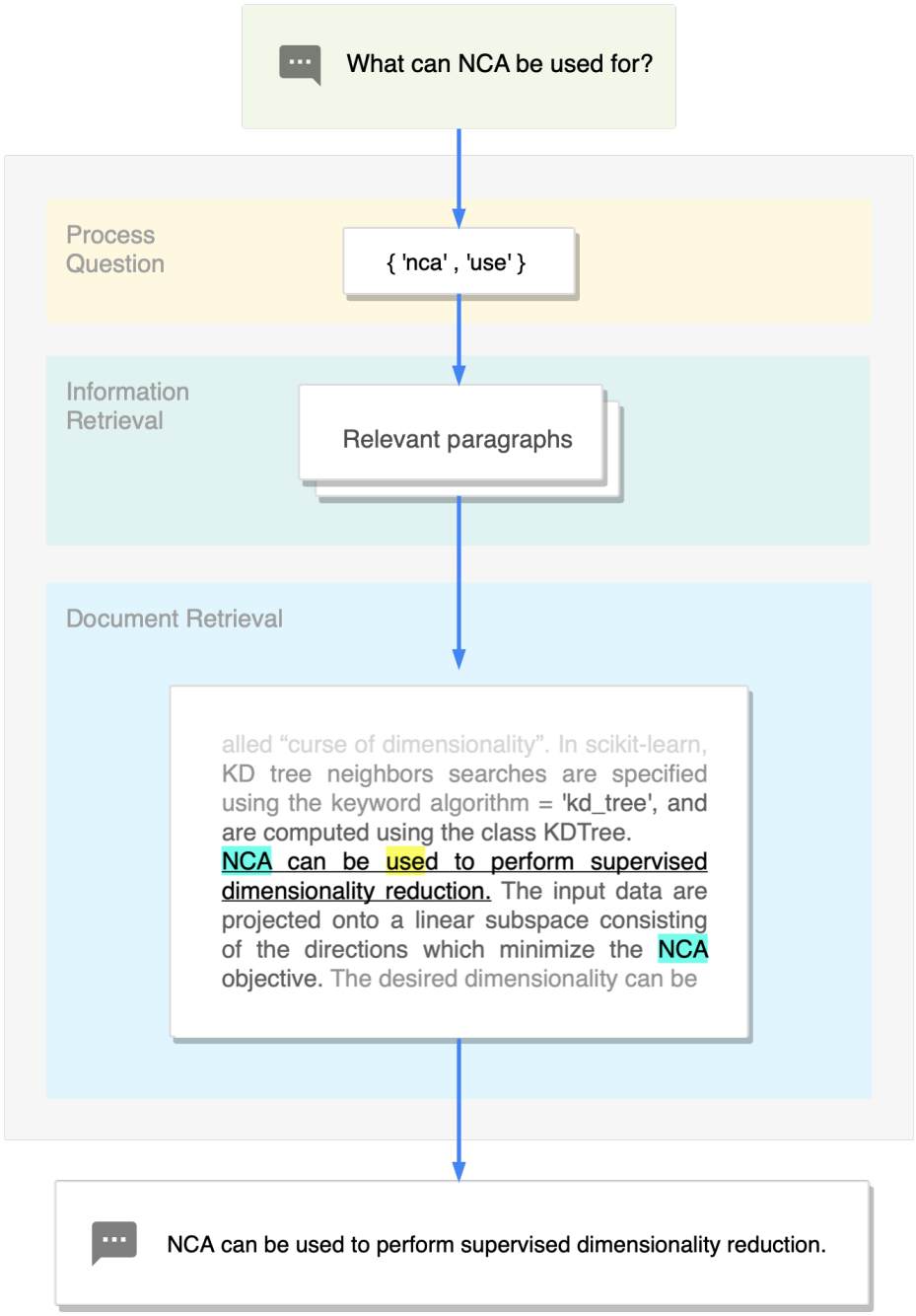

Figure 3.5: QA example

importance. Using this valuable information, an ElasticSearch query is generated. This is used to retrieve relevant documents and pieces of information that match the keywords. Because of the way the Knowledge Base is structured, which is explained later on, there are three main types of queries:

- **Definition Query.** These are those that fit the `definition` answer type.

- **Example Query.** These are designed for the `example` answer type.
- **Concrete.** All the other types of answer fall into this category.

3.3.3 Document Retrieval

This module implements a Document Retrieval Model. The main objective of the module is to parse the provided information into an answer that fits the intention of the question. A general view of the module pipeline is shown in Figure 3.6.

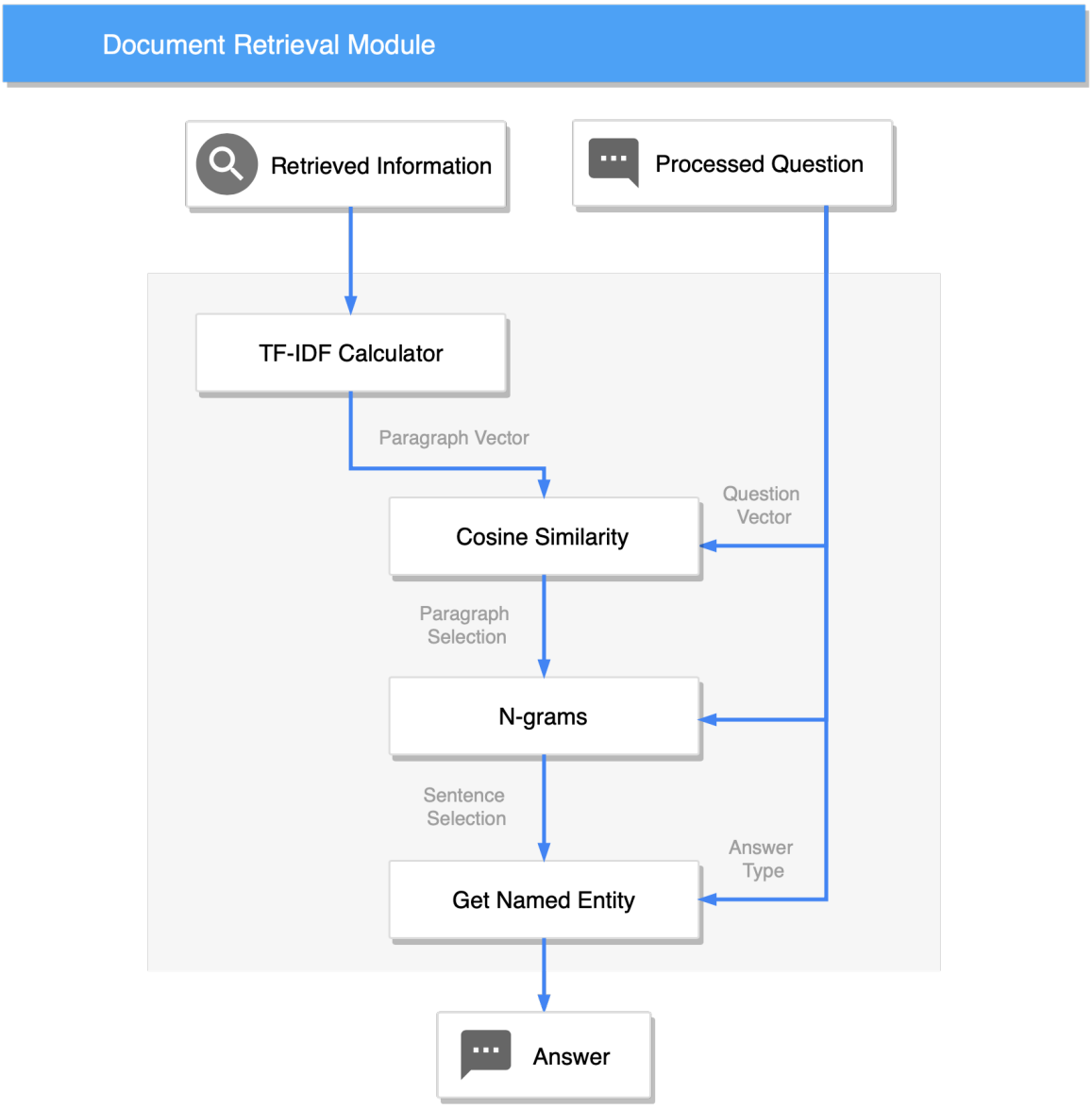

Figure 3.6: Document Retrieval Pipeline

First, it receives all the information collected by the Information Retrieval module. It

then computes the term-frequency inverse document frequency (TF-IDF) for every token of each paragraph.

After having processed all the paragraphs, the `Processed Question` is passed to the module. To find the answer among all the data, the algorithm searches and selects the most relevant paragraphs based on a cosine similarity[4] between the `Question Vector` and the `Paragraph Vector` previously computed.

Next, it gets the most relevant sentences using n-gram similarity between each sentence and the question. The program returns a list with relevant sentences a the default answer is selected as the most relevant sentence.

Finally, using the relevant sentences selection and the `Answer Type` the final answer is selected. This is done by searching in each of the sentence the named entity of `Answer Type`. Depending on the type of answer required, the algorithm tries to fit the sentence into a usual template and the sentence that best fits the template is the chosen one. In the case that no sentence fits well, the Default Answer (The one with the highest n-gram similarity) is sent as a response.

3.3.4 Definition Answering

When the `Answer Type` is of the *definition* type, the module searches in the Glossary index in the Knowledge Base. It searches for a match with the terms in the index. When a match is found, the corresponding definition is sent as an answer. Common questions of this type are:

- What is a neural network?
- Can you give me a definition of overfitting?

This module is implemented as a DialogFlow agent, with an intent that can recognize that the user wants a definition. The intent is trained with multiple training phrases that can be used to ask for a definition. It extracts a term as the slot. These slots are recognized thanks to an entity[5] defined as all the terms available in the Knowledge Base.

[4]The Cosine similarity is a way to measure the similarity between two non-zero vectors with n variables. If the cosine value of two vectors is close to 1, then it indicates that they are almost similar.

[5]An entity is the definition of a type of slot. It can be defined as a list with all the possible terms for that slot.

3.3.5 Example Answering

When the `Answer Type` is of the *example* type, there is a more complex type of search. There is a search across the documentation text to match the key words of the query. When a match is found, the corresponding code snippet is sent as a response with the appropriate format. Examples of these type of questions are:

- How is a dataframe defined in Pandas?
- How can I implement a k-fold using scikit?

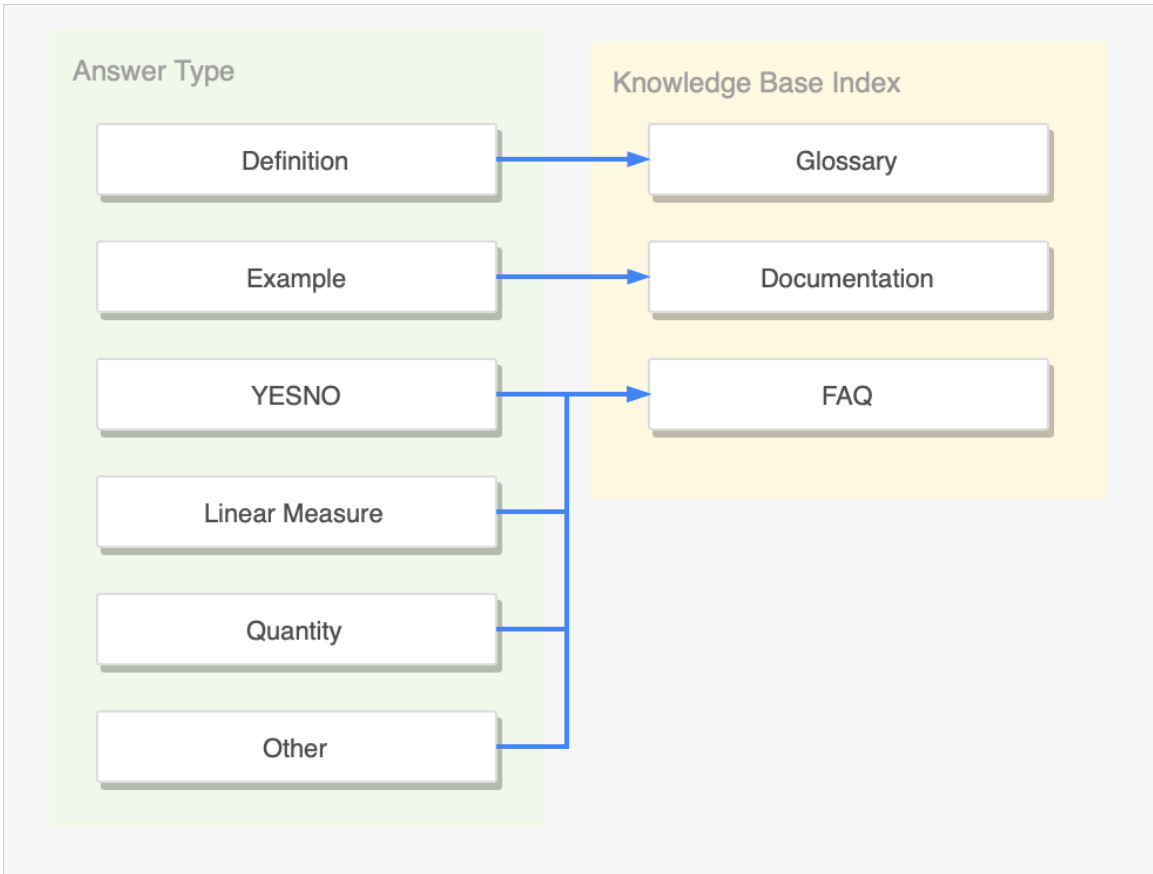

Figure 3.7: Knowledge Base selection based on Answer Type

This module is implemented as a DialogFlow agent with and intent trained to detect example queries. The slot in this case is more open, so there is no Entity defined. The example can be of any kind.

3.4 Knowledge Base

The knowledge base (KB) is the place where all the information used by the chatbot is stored and available to retrieve. This data is scraped from the web and is organized in order to be useful and effective. For this purpose ElasticSearch is used. It includes a very powerful and fast full-text search that helps to deliver the correct information while adding a low lag to the pipeline.

The Knowledge Base is adapted to the way the information retrieval modules work. There are three main types of information that are stored in the KB:

- **Glossary Terms.** It was observed that the main type of questions that people ask the chatbot are definitions of terms in the form of "What is a linear regression model?".
- **Documentations.** In order to retrieve examples and snippets of code the documentation of languages and libraries is the best source. They include explanations in plain text and examples in code.
- **Frequently Asked Questions (FAQ).** If the question can't be answered by any of the modules that handle the previous data, which is possible if the question is a comparison like "What is the difference between bias and overfitting?", there is an index with these types of questions and answers.

3.4.1 Scraping

The information stored in the KB is from public webs and available to scrap. According to the categories previously described, the sites that fit the necessities of the glossary are:

- **Big Data glossary.**[6] A list of terms regarding big data.
- **Machine Learning glossary.**[7] A complete glossary of machine learning and statistics terms and definitions.

These pages were organized by a different html class for each term and a different one for the definition. By identifying these classes, the term and definition are extracted and organized.

The documentation sites used to populate the KB are:

[6] https://bigdata-madesimple.com/big-data-a-to-zz-a-glossary-of-big-data-terminology/
[7] https://www.analyticsvidhya.com/glossary-of-common-statistics-and-machine-learning-terms/

- **Pandas Documentation.** Because the use of the Python Pandas library is widely used when developing machine learning models, it is very useful to have examples available of common implementations of data handling. This documentation is structured with brief descriptions with code examples.
- **Scikit-Learn Documentation.** Being the library used widely for Machine Learning purposes, Scikit examples of implementations is an obvious use case for the chatbot, and therefore a very important part of the KB.

For more complex questions, the use of a FAQ solves the problem. The site[8] used for this purpose is structured as a list of questions with the answers associated. It was selected because of the rich and adequate answer for the purpose of the project.

3.4.2 Structure

In order to fully take advantage of the ElasticSearch potential, the information must be correctly organized so it can be retrieved effectively. Taking into account the types of questions that can be asked and the modules used to implement the answering of each type of question, the structure of the database is as follows.

Regarding the indexes of the ElasticSearch node, which is the entrypoint of the search, the following are defined:

- **glossary.** It contains 358 instances of different terms. Each instance of this index is an object with the following attributes:
  - *Word* includes the term in the glossary.
  - *Definition* includes the definition of the term.
  - *Url* includes the source of the information.
- **scikit**. This index includes 225 chapters and subsections from the scikit user guide.
  - *Title* includes the title of the section.
  - *Text* includes the text of the section.
  - *Code Snippet* includes an array of the code examples in the section.
- **pandas**. This index contains 78 instances from the pandas documentation. It has the same structure as the scikit index.

[8] https://machinelearningmastery.com/faq/

  - *Title* includes the title of the section.
  - *Text* includes the text of the section.
  - *Code Snippet* includes an array of the code examples in the section.

- **faqs**. This index includes 99 instances of Frequently asked questions. It has this structure:
  - *Question* includes the question.
  - *Answer* includes the answer of the question

3.5 Small Talk Agent

3.5.1 Introduction

According to [17], a users satisfaction with a certain chatbot is influenced by various factors. By testing which of these factors were more influential, the results revealed that the human-likeness of the bot was very correlated with the users satisfaction.

Also, it was stated [2] that people were actually *inclined to send more than twice as many messages to chatbots with a human-like interaction compared to other people, contrary to our expectations and disconfirming the notion that people feel less confident or comfortable communicating with chatbots.*

Including a module to handle small-talk improves the human-likeness of the bot and makes it more fun and engaging. Instead of answering with the fallback answer, if the question isn't about the topic it will trigger the small-talk module to simulate human interaction and cleverness. Some examples of the behavior that the Bot can answer are collected in Figure 3.8.

3.5.2 Implementation

This module is implemented with Google's DialogFlow technology. There is a specifically trained agent to provide the desired output. This agent can detect more than 100 different intents.

Among these intents are some of the provided with the default Small Talk module and some custom ones. The intents are defined to fit the purpose of this project. For example, when asked what it can do, it responds with directions to ask questions about Data Science.

Figure 3.8: Small Talk Examples

3.6 Speech Act Classifier

3.6.1 Introduction

The main objective of the act of speaking is the transmission of an intention, which may or may not require a response or interaction. For example, if someone says "What time is it?" the intent is to get a response and the speech act is a question.

The task of speech act classification involves classifying a certain sentence into a set of predefined speech acts. Examples of these are questions, statements, greetings or insults. This is relevant to the project because in order to know what answer to give, it is useful to know the intention of the actor.

3.6.2 The Dataset

The dataset [18] used to train the classifier consists of 10567 posts from five different age-oriented chat rooms at an internet chat site. It is sanitized to protect user privacy. The posts were tagged using 15 post categories.

This dataset was selected because of its big size, which makes a more robust model. Also, it being hand-tagged contributes to reliability. There are other datasets that could have been selected, but they are too small in relation to the one used.

The examples shown in Table 3.1 reveal the complexity of the task: Sometimes more

Classification	Example
Accept	yeah it does, they all do
Bye	night ya'all.
Clarify	i meant to write the word may.....
Continuer	and thought I'd share
Emphasis	Ok I'm gonna put it up ONE MORE TIME
Greet	hiya hug
No Answer	no I had a roomate who did though
Other	0
Reject	u r not on meds
Statement	Yay...democrats have taken the house!
System	JOIN
Wh-Question	why do you feel that way?
Yes Answer	why yes I do, lol
Yes/No Question	cant we all just get along
Emotion	lol

Table 3.1: Post classification examples

than one category applies, a "Wh-question" does not start with "Wh" or a question does not have a question mark.

3.6.3 Preprocessing

In order to feed the data to the classifier it needs to be processed into something an algorithm can understand. The process to follow with each phrase is the following.

- **Tokenization:** It's the simple process of dividing a given sentence into a set of words. It is also common to use n-grams. This consists of grouping the words into subsets to take into account the context instead of treating each word independently. In this study it does not provide a significant insight and using simple tokenization reduces complexity.

- **Stemming:** In order to clean the samples into a unified form, lemmatization and stemming are the appropriate tool. It consists in finding the stem of the word , given a flexed form. The stem is what would be found in a dictionary. Comparing the results between processing the data with stemming and not doing it showed that stemming has a positive impact on the score of the model.

3.6.4 Feature Extraction

This process consists of transforming a preprocessed text into a numerical vector that can be understood by the machine in a simpler way. The tool to extract this features is called a vectorizer and the one used for this model is the `TfidfVectorizer` included in NLTK.

First, it creates a "dictionary" with all the input data, which is a vector of all the words found in the training data. Then, each instance of the data is fitted into this dictionary producing the output of a numerical vector which can be directly fed into the classifier.

3.6.5 Classifier

The last step of the process is finding the best model to classify the data. By training and evaluating some of the most popular classification algorithms, the best one is selected based on the score achieved by a K-Fold, a commonly used methodology which is explained later. This process can be automated by means of a grid search which finds the best parameters of a model the optimal way.

The classifiers tested are all implemented in the Scikit-Learn library. The ones that gave the best results were:

- **Decision Tree** uses a tree-like model of decisions. Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop.

- **Support Vector Classification (SVC)** uses support vector machines for a classification detection.

- **Multinomial Naive Bayes** implements the naive Bayes algorithm for multinomial models. The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

- **Random Forest** is an ensemble method that follows the bagging technique. It creates random subsets from the original dataset and then fits a decision tree into each of them with a random selection of features to decide the splits. The final prediction is an average of the predictions made by each tree.

  Since the classification of Speech Act is intuitively rule-based, a tree-type model is a good choice to consider beforehand.

3.6.5.1 Evaluation Metrics

The evaluation method of the model is very important to avoid biases. It must meet certain standards to know if it is needed to continue the search for the best model. There are a series of metrics that examined carefully and together will provide the necessary information on what the strengths and weaknesses of the model are.

In a multiclass prediction, the result on a test set is often displayed as a two-dimensional confusion matrix with a row and column for each class. Each matrix element shows the number of test examples for which the actual class is the row and the predicted class is the column. Good results correspond to large numbers down the main diagonal and small, ideally zero, off-diagonal elements.[1]

		Predicted class			
		a	b	c	
Actual class	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
	Total	120	60	20	

Table 3.2: Confusion matrix example [1]

Table 3.2 shows a numeric example with three classes. In this case the test set has 200 instances (the sum of the nine numbers in the matrix), and 88 + 40 + 12 = 140 of them

are predicted correctly, so the success rate is 70%. This measure of success is usually called Accuracy.

The metrics taken into account to the evaluation of each model are:

- **Accuracy** is the most intuitive because it is a ratio of correctly predicted observations to the total observations.
- **Precision** is the number of correct predictions divided by the number of total predictions made. Intuitively, a high precision for a class means that if our models predict that class, it is very likely to be true.
- **Recall** is the number of correct predictions divided by the total number of elements present in that class. Graphically, it is the value on the diagonal, divided by the sum of the values in the row. If recall is high, it means that our models manages to recover most instances of that class.
- **F1 score** is the harmonic mean of precision and recall. It reflects the combination of Precision and Recall if one of those is not specifically needed.

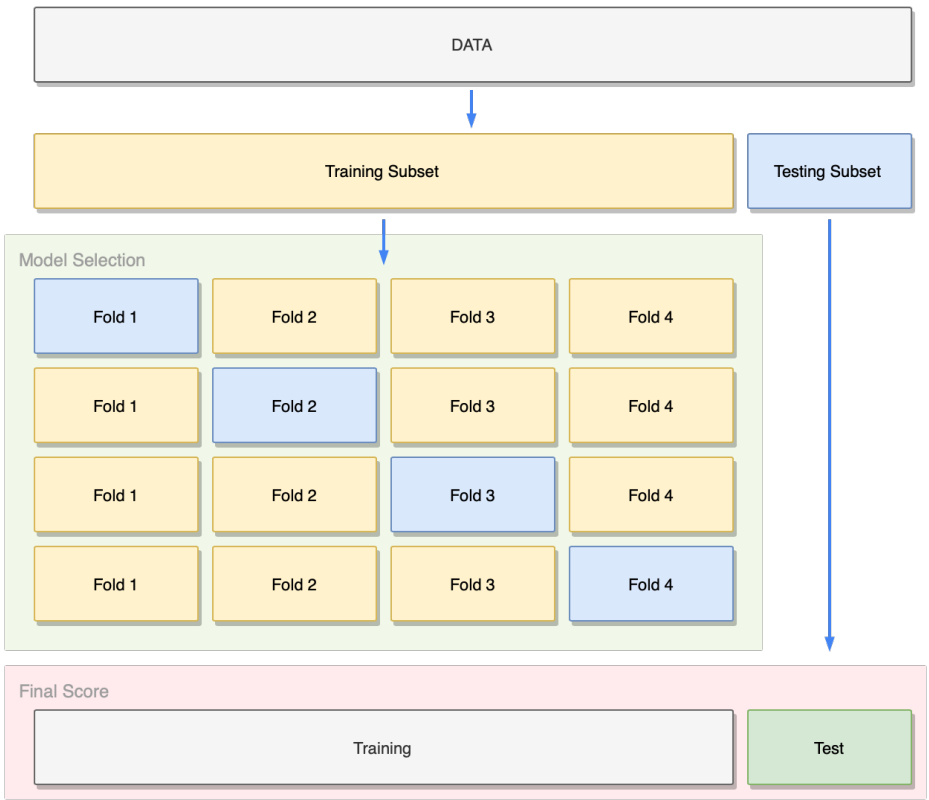

Figure 3.9: Cross-Validation and Hold out

When evaluating a dataset, the holdout method reserves a chunk of the data for testing while training the model with the rest. When the amount of data for training and testing is limited, the sample used for training or testing may not be representative. To solve this problem we repeat the process a number of times, until all the data has been used for testing (and training).

A k-Fold means that the data is divided in k parts. Then it iterates k times. Each iteration, one of the k parts is used for testing and the rest for training. This way misrepresentation is avoided. This method of evaluation is also called cross-validation.

3.6.5.2 Results

The scores following are calculated saving a fourth of the dataset for testing afterwards and using the rest to get this results.

Model	Accuracy	Precision	Recall	F1
Multinomial NB	0.685 (+/- 0.012)	0.75	0.71	0.66
Decision Tree	0.711 (+/- 0.005)	0.76	0.76	0.75
Random Forest	0.745 (+/- 0.005)	0.78	0.77	0.76
SVC	0.768 (+/- 0.007)	0.79	0.8	0.77

Table 3.3: Evaluation Scores

The scores in Table 3.3 are obtained by performing a 5-Fold and calculating the mean of the scores. The Support Vector Classifier model is the best and significantly different.

Using as the training data 3/4 of the dataset and the rest as testing data we obtain with the SVC a final accuracy score of 0.799.

CHAPTER 4

Use Cases

In this chapter a series of use cases of Jaicob are represented. The operation of the system is described from when the user query is typed to when the answer is displayed. A compilation of examples are shown in Appendix C

4.1 Small Talk intent

The queries used as an example as show in Figure 4.1 are:

- How are you today?
- What is your purpose?

The first step in the architecture is the Speech Act classifier, which recognizes the query as a *Greeting*, and therefore is forwarded to Dialogflow Small Talk.

Next in the process, Dialogflow uses its machine learning models to detect a certain type of small talk. The cases are:

- How are you today?: `smalltalk.greetings.how_are_you`

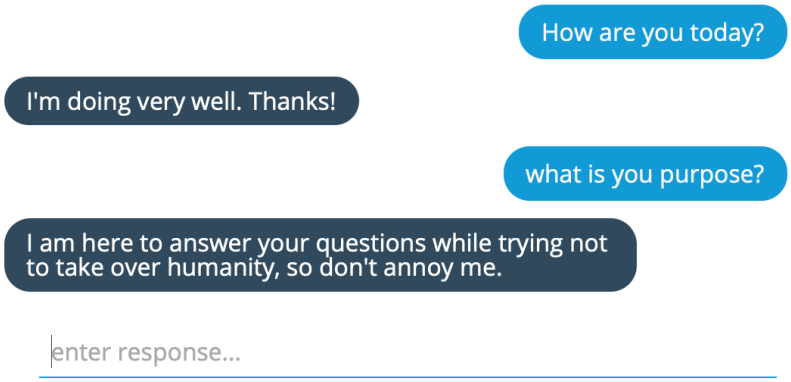

Figure 4.1: Small Talk Intent Use Cases

- What is your purpose?: `Get Purpose`

In both cases, Dialogflow generates a series of possible answers, and then selects randomly one of them. This is because if it always answered the same way it would seem more like a machine. This feature provides the feeling of a genuine conversation. It looks like this in the case of *How are you today?*

```
{
   "text":[
      "Doing great, thanks.",
      "I'm doing very well. Thanks!",
      "Feeling wonderful!",
      "Wonderful! Thanks for asking."
   ]
}
```

DialogFlow also returns information such as `intentDetectionConfidence` which is a percentage of security defined by the classifier model.

4.2 Definition Intent

To show how the intent of a definition request works, the query used is *What is elasticsearch?*

The first step in the architecture is the classification of Speech Act which returns Wh-question. Because of that it goes into the QA module and Dialogflow makes a match of the `Get Definition` intent with a `intentDetectionConfidence` of 1, which is the maximum possible. Then it uses the trained model to extract the information previously

defined as relevant. The intent is defined to match an Entity named topic. It is populated with all the terms and synonyms of the values in the Knowledge Base. Below is a response from Dialogflow with the parameters that were recognized.

```
{
   "queryText":"what is elasticsearch",
   "action":"show_info",
   "parameters":{
      "topic":"Elasticsearch"
   }
}
```

This is the information received by the webhook developed in this project. In the case of receiving the `Get Definition` intent, it uses the `topic` parameter to generate a query to elasticsearch.

The queries generated search for a match in the terms of the Glossary index in Elastic-Search:

```
{"query": {"match": {"Word": search_term}}}
```

ElasticSearch returns an array of hits ordered by a relevance score. Then, the first object of that list is selected and the definition of the term is returned to the user. This is shown in Figure 4.2

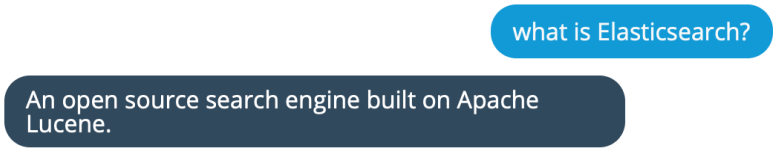

Figure 4.2: Definition Intent Use Case

4.3 Example Intent

The query in this intent is *how do you implement a svc in scikit?*. This is asked with the intention of receiving a code example of the problem stated. This also can be asked explicitly like *Give me an example of svc.*

As with the previous intent, the query enters the QA module. After that, Dialogflow recognizes the query as an `Get Example` intent with a `intentDetectionConfidence` of 0.75. The implementation of the intent detects the relevant information and parses it

into a parameter, which is forwarded to the Webhook.

```
{
   "queryText":"how do you implement a svc in scikit?",
   "parameters":{
      "any":"svc in scikit"
   }
}
```

The way the webhook works in this intent is by performing a search in the Knowledge Base with the parameter it received from Dialogflow. To generate the query it follows these steps:

- **Index detection**. First it searches the parameter for hints in which index it should search in. In the case of this example ("svc in scikit") it detects the term *scikit*, which is directly associated with the `scikit` ElasticSearch index. In the case that no hint is found, it performs a search across all indexes.

- **Text matching**. After having detected the index, a query is generated. In this case, a full text search [2.2] is performed.

- **Answer generation**. With the matches found, instead of returning the text where the match was made, the function returns the code snippet associated with the text in the case that it exists. The result can be seen in 4.3

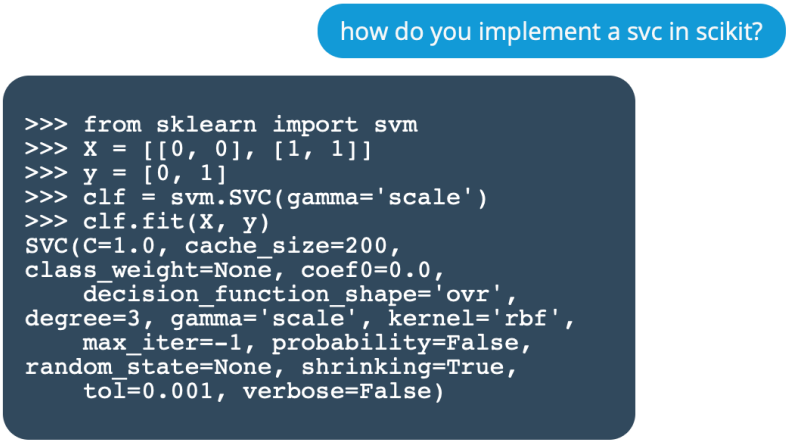

Figure 4.3: Example Intent Use Case

4.4 FAQ intent

The query used to illustrate how this intent works is *How do you differentiate standardization and normalization?*.

This intent was modified during the development of the project. Dialogflow launched a new feature called knowledge, that allowed to do precisely what was implemented. Therefore there was a migration to this solution. It required to upload the FAQ data. Then Dialogflow matches similar questions to the ones uploaded.

When entering the QA module, the query is detected as a knowledge intent (FAQ) with a `intentDetectionConfidence` of 0.809. After that, Dialoflow returns a list of the matched FAQs. The first one is *When should I standardize and normalize data?*. Then, the answer to that question is returned. As can be seen in Figure 4.4, the result is successful.

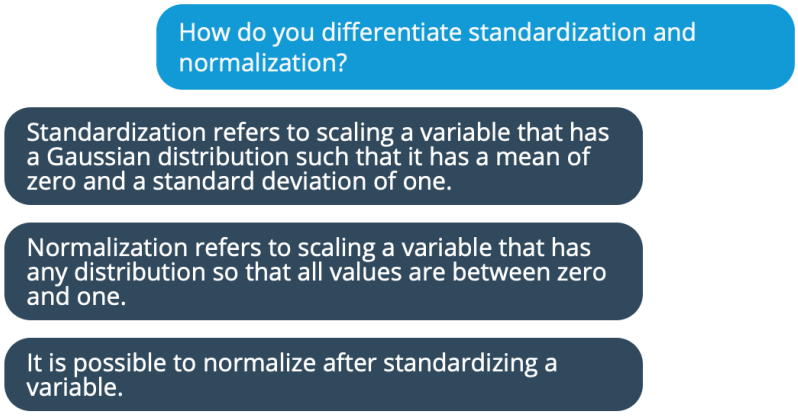

Figure 4.4: FAQ Intent Use Case

CHAPTER 5

Evaluation

In this chapter we will describe the evaluation of the project. Being a conversational interface, the way to test it is with real users because in a controlled environment, the are original thoughts from users that cannot be predicted.

5.1 Participants

The experiment was made with 29 participants. They were all of technical background. All of them were unaware of the the inner workings of Jaicob. They were asked to use the chatbot as a tool to answer any questions or doubts that may arise in the process of understanding Data Science related topics or of writing the corresponding code. They were asked to answer a small survey to get to know them.

The median of the ages of the participants is 22 years. 51% were studying the *Grado en Ingeniería de Telecomunicaciones* and the rest the Master or other technical studies.

About their technological background, 54% had developed and implemented something machine learning related. The rest had some basic knowledge.

5.2 Measurements

As explained in 3.5, small talk is an important part of the architecture of the chatbot. Therefore, before making the measurements the hypothesis that small talk makes a more attractive bot was stated.

Metrics about how innovative the product is and the reactions are also recorded. Also, there are some analytics related to the use of the product.

5.3 Results

About the user impressions the overall impression is positive as can be seen in Figure 5.1

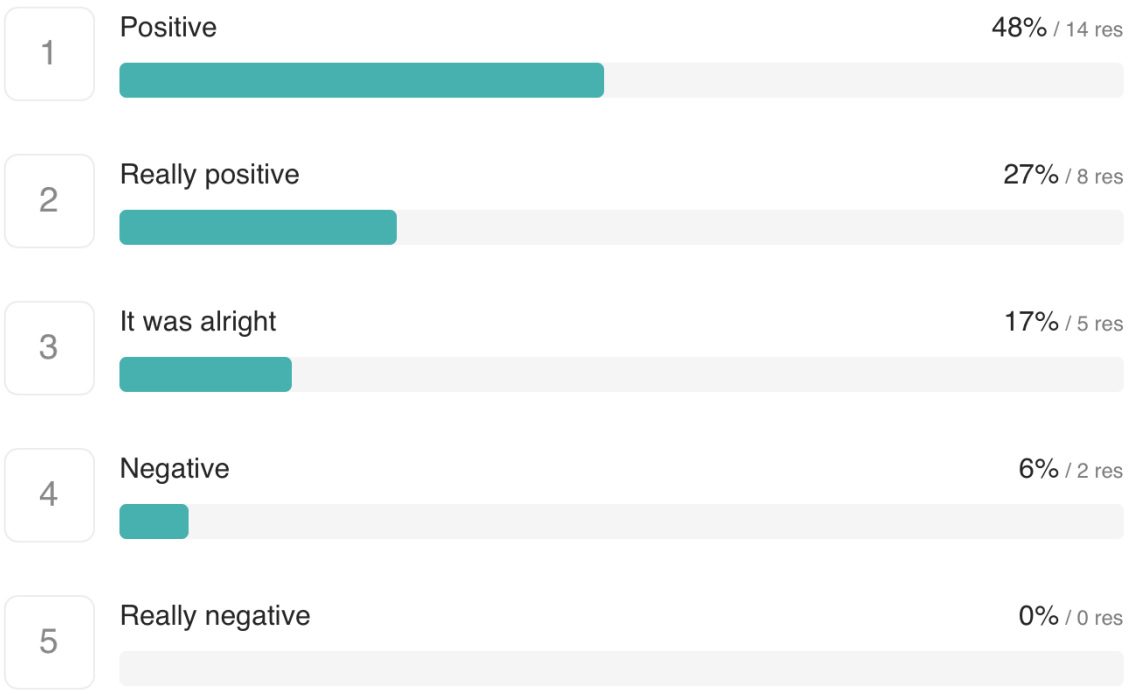

Figure 5.1: Overall Impression

The testers made an average of 15.86 queries per session. The distribution of intents in the sessions is shown in Figure 5.2. The graphic shows that the intent that matched most part of the queries was the one related to the FAQs, which is tagged in the image as Knowledge. After that, there is the Definition intent and then the example intent. Also, 10 to 12% of the queries resulted in small talk handling.

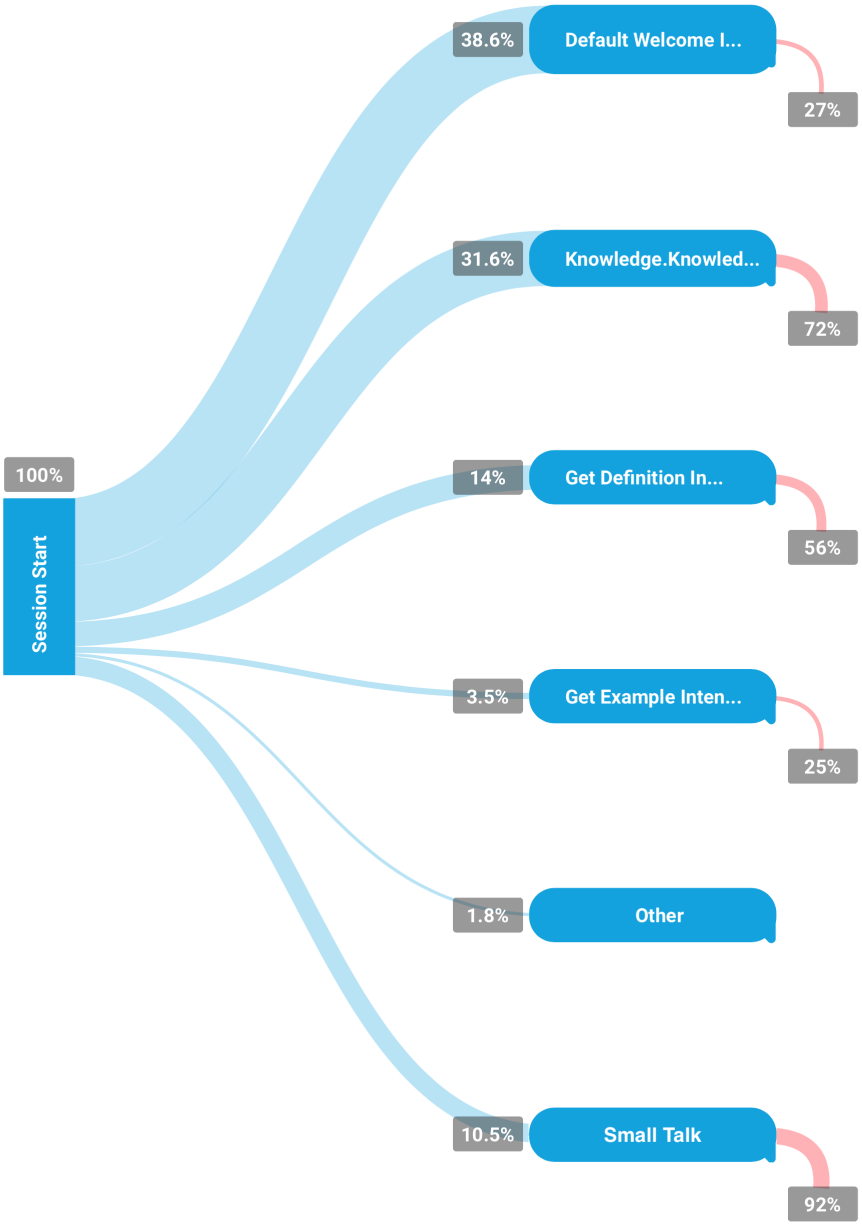

Figure 5.2: Intent Distribution

In order to confirm the hypothesis that was stated, the users were asked if they thought that small talk made Jaicob more attractive. The hypothesis is confirmed 100% as can be seen in Figure 5.3. This result must not be taken lightly, even though the sample of testers

is very small, because it confirms that the humanization of interfaces contributes to make a more positive experience to the users.

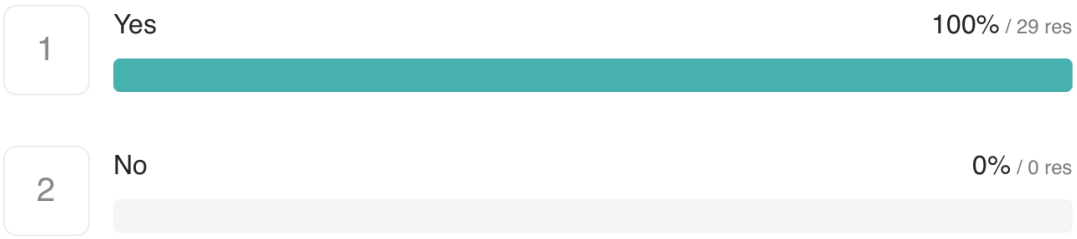

Figure 5.3: Small talk makes Jaicob more attractive

After that, the users were asked if the product would be used in a real scenario. In other words, how much they need the product. The results are displayed in Figure 5.4. Most of the users would use it from time to time, and 70% declare that they somewhat need it.

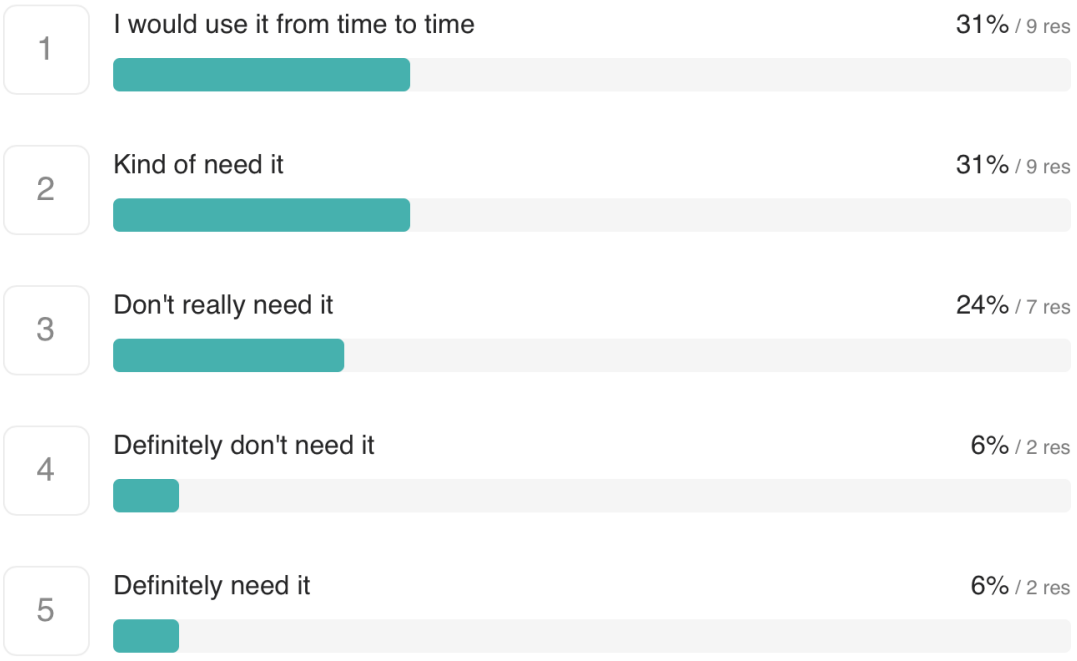

Figure 5.4: How much users need Jaicob

At last, the innovation level of the product reached a rating of 4.1 out of 5. This refers to how innovative the users feel that the product is. The distribution can be seen in Figure 5.5

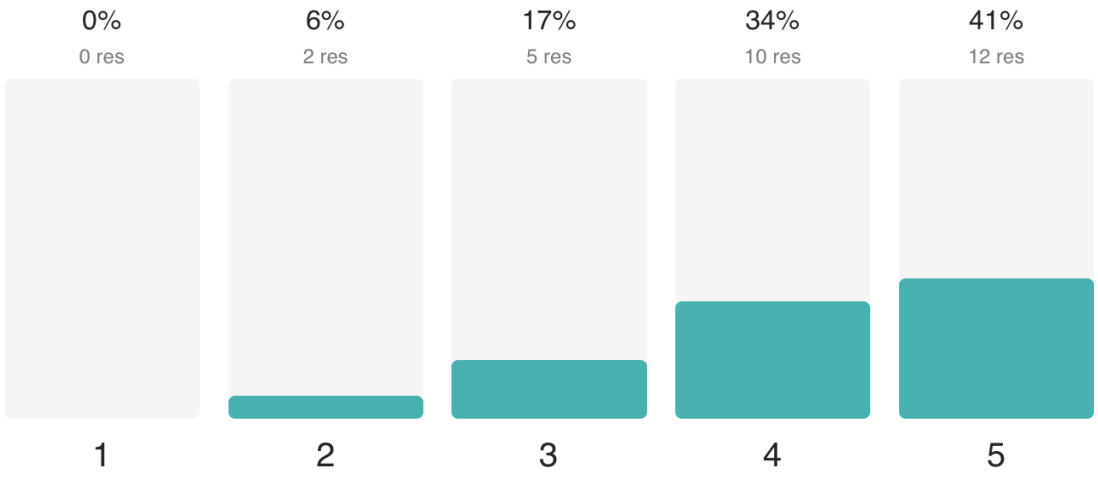

Figure 5.5: How innovative Jaicob is

To conclude, the results were satisfactory confirming all the hypothesis stated. It also shows that the project is innovative and that the users enjoyed the experience and feel that the product has a use in the world.

CHAPTER 6

Conclusions and future work

In this chapter the conclusions extracted from this project are described. The goals achieved will be explained, as well as the problems faced during the development. For the last part, thoughts about future work will be set forth.

6.1 Conclusions

In this thesis a conversational bot has been developed. The purpose of the project is to be used as a tool with a comfortable and usable interface with a human experience.

In the process of development, a series of modules were implemented:

- A speech act classifier to recognize the intents of the user.
- A small talk module to handle human-like interactions.
- A series of intent handlers to respond the user queries effectively.
- A Knowledge Base populated with information regarding the topic.

The project was evaluated with a small sample of users achieving very favorable results

and confirming the stated hypothesis that small talk makes the product more attractive.

6.2 Achieved goals

- Respond to the user queries with the desired information. The results expected from the products are the ones returned by it as it should be. This is done by means of a pleasant web interface.

- Retrieve the necessary information from the data source. A series of algorithms and parsing of search queries are developed in order to search and select the desired information from a data source.

- Classify Speech Act types correctly. A machine learning model based on real users training data is developed to recognize the users intentions correctly fit into a series of classes.

- Handle small talk effectively and as human as possible. A Dialogflow module is implemented to handle this type of speech and the results are very favorable regarding small talk.

6.3 Problems faced

The initial web interface was one already developed, but it was implemented in a old version of the Dialogflow API which becomes deprecated. In order to continue using the API there was a need to modify almost the whole module in order to be in the latest version. This was difficult because it is very different from the previous one and has several increases in security and identity. In the end, it was possible sacrificing the process time of the API, but it was necessary in order to be usable in the future.

6.4 Future work

- Increase of the Knowledge base

  At the end of the development of this project, the ElasticSearch database included three indexes: Glossary, Pandas and Scikit. This means that it can answer questions related to specific information about Pandas and Scikit, and general term definitions.

In order to achieve a broader reach in the areas of knowledge, it would be needed to place additional information in the Knowledge Base as well as the corresponding Dialogflow intents.

- Google Web Answers

  There is a feature that Google Search displays in a search when you ask for a question. It can be useful to use this feature to answer questions that the product is not able to respond.

- Multimedia in answers. The Knowledge Base includes information such as images and videos. In order to get richer examples and definitions, to implement answers with images can be very useful.

APPENDIX A

Impact of this project

This appendix reflects, quantitatively or qualitatively, on the possible impact of the developed project.

A.1 Social impact

The purpose of this project is to provide a useful tool to students that want to learn about machine learning. It can provide insights and resolve doubts about the tool used for machine learning. Therefore, the target user is mostly a student or anyone who wants to learn more effectively about data science and machine learning.

As these technologies evolve, more and more people will study these subjects. Therefore, the future impact of the project is promising and the affected groups will increase. The product will be accessible via a web interface (web browser) and is available globally.

In addition, the privacy and security of the users is guaranteed, as it is not required to provide any personal information, and the model does not store data related to the user identity. There is not any kind of risk to the user's health or well-being.

A.2 Economic Impact

In this section the possible economic impacts that users using the interface developed in this project may experience are assesed.

From the users perspective, it has no cost at all. It is a service provided with no monetary purpose but with the purpose of helping people build projects that can make the world a better place.

From the maintenance point of view, the system requires hosting. Hosting may cost money, but with the expected use it can be supported by the free tiers provided by cloud hosting services.

A.3 Environmental Impact

This section aims to define the main environmental impact of the development this system.

Computers and other information technology infrastructures consume significant amounts of electricity, adding a huge charge on our electricity networks and contributing to greenhouse gas emission. In addition to this consumption, the energy required for the cooling system associated with this equipment must also be added, which is the second main reason for the consumption of this equipment.

The main environmental impact of this project is the high consumption of the server where it is deployed.

A.4 Ethical Implications

In this section we will evaluate the ethical implications of such a project.

Chatbots have been controversial in the past years like with the case of machine learning in general. In the case of the project developed, being just a humanized mechanism to solve doubts, there is no risk of hurting a user in an ethical sense.

Jaicob also protects the users privacy by not storing data related to the user. It respects all laws related to this topic.

APPENDIX B

Economic budget

In this appendix we are going to make an adequate economic budget for the realization of this project. The main parts of this budget will be explained in the following sections.

B.1 Physical resources

The budget of the devices used to the develop this project is related to the computer used to implement all the necessary modules of the architecture. No special equipment was required.

The estimated cost of the computer used for the job is 1200€.

B.2 Human Resources

The salary of a person who can develop a project of these characteristic is based on the real value. A UPM scholarship of 160 hours and 1200€ was the case for the project.

The people involved in the evaluation of the product is also taken into account, by spending some time testing the features.

On the other hand, we must also consider the salary of a person who is in charge of software system maintenance. For this function you need a Telecommunication Engineer or Computer Engineer with knowledge of machine learning and NLP. The salary of a worker with this profile is approximately 24.000 e per year.

B.3 Licenses

This section includes the cost corresponding to the licenses of the software tools necessary for the development and deployment of the system carried out in this project. However, all the software used in this project is open-source, so the cost of software licenses is zero.

B.4 Taxes

One of the scenarios in which taxes related to a software engineering project must be taken into account is the case in which the final product is sold to an interested company. This sale is subject to a tax of 15% of the price of the product, as defined in Statute 4/2008 of Spanish law.

APPENDIX C

QA Corpus

In this appendix, a compilation of queries and answers typed by real users in the process of evaluation is presented. These are classified by intent type. It is important to notice that these answers are not hard-coded, but are the result of all the steps involved in the architecture as described in Chapter 3.

C.1 Small Talk

Question	Answer
What is you purpose?	You can ask me questions related with machine learning
How are you?	Doing great, thanks.
What is the weather like?	My purpose is beyond that type of questions. If you wanna talk about that ask Siri or Google
You are bad	I can improve with continuous feedback. My training is ongoing.

C.2 Definition Intent

Question	Answer
What is ElasticSearch	An open source search engine built on Apache Lucene.
I wanna know about pandas	Pandas is an open source, high-performance, easy-to-use data structure and data analysis library for the Python programming language. [...]
What does accuracy mean?	Accuracy is a metric by which one can examine how good is the machine learning model. [...]
Help me understand Deep Learning	Deep Learning is associated with a machine learning algorithm (Artificial Neural Network, ANN) which uses the concept of human brain to facilitate the modeling of arbitrary functions. [...]
What does IQR mean?	IQR (or interquartile range) is a measure of variability based on dividing the rank-ordered data set into four equal parts. It can be derived by Quartile3 – Quartile1.
I wanna know what the residual is	Residual of a value is the difference between the observed value and the predicted value of the quantity of interest. [...].
I wanna learn about tokenization	Tokenization is the process of splitting a text string into units called tokens. The tokens may be words or a group of words. It is a crucial step in Natural Language Processing.
What is scikit-learn?	Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. [...]

C.3 Example Intent

Question	Answer

How do I implement a SVC	`>>> from sklearn import svm`
	`>>> X = [[0, 0], [1, 1]]`
	`>>> y = [0, 1]`
	`>>> clf = svm.SVC(gamma='scale')`
	`>>> clf.fit(X, y)`

How can I code a cross validation	`>>> import numpy as np`
	`>>> from sklearn.model_selection import KFold`
	`>>> kf = KFold(n_splits=2)`

I want to build a bagging method	`>>> from sklearn.ensemble import BaggingClassifier`
	`>>> from sklearn.neighbors import KNeighborsClassifier as KNN`
	`>>> bagging = BaggingClassifier(KNN())`

How do i compute a confusion matrix in scikit	`>>> from sklearn.metrics import confusion_matrix`
	`>>> y_true = [2, 0, 2, 2, 0, 1]`
	`>>> y_pred = [0, 0, 2, 2, 0, 2]`
	`>>> confusion_matrix(y_true, y_pred)`
	`array([[2, 0, 0], [0, 0, 1], [1, 0, 2]])`

Show me an example of a dataframe	`>>> import pandas as pd`
	`>>> X = pd.DataFrame(`
	`... 'city': ['London', 'London', 'Paris', 'Sallisaw'],`
	`... 'title': ["His Last Bow", "How Watson Learned the Trick",`
	`... "A Moveable Feast", "The Grapes of Wrath"],`
	`... 'expert_rating': [5, 3, 4, 5],`
	`... 'user_rating': [4, 5, 4, 3])`

C.4 FAQ Intent

Question	Answer
How do I check my python version?	You can discover what version of Python you are using by typing a command line. On the command line, type: python -V You should then see the Python version printed. For example, you may see something like: Python 3.6.5
How do i know if im overfitting	It is possible to overfit the training data. This means that the model is learning the specific random variations in the training dataset at the cost of poor generalization of the model to new data. [...]
How do you differentiate standardization and normalization?	Standardization refers to scaling a variable that has a Gaussian distribution such that it has a mean of zero and a standard deviation of one. Normalization refers to scaling a variable that has any distribution so that all values are between zero and one. [...]
How do I calculate the accuracy of regression	You can't. Accuracy is a measure for classification. You calculate the error for regression.
How do I handle an imbalance in classes?	It is common to have a different number of observations for each class in a classification predictive modeling problem. This is called a class imbalance. [...] I recommend trying a suite of approaches to see what works best for your project.
What is the difference between "validation" and "test" sets?	A training dataset is used to train or fit a model. A test dataset has observations that do not overlap with the training dataset and is used to evaluate a trained model. Specifically, to estimate the skill of the model on a new data sample. [...]
Why learn machine learning?	This is an important question! Because machine learning provides techniques to learn a solution from historical examples for complex problems where it is intractable or infeasible to develop a manual solution.

Bibliography

[1] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[2] Jennifer Hill, W. Randolph Ford, and Ingrid G. Farreras. Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Computers in Human Behavior*, 49:245–250, Aug 2015.

[3] John E. Kelly. Computing, cognition and the future of knowing. *Whitepaper, IBM Reseach*, 2, 2015.

[4] Ying Chen, JD Elenee Argentinis, and Griff Weber. Ibm watson: How cognitive computing can be applied to big data challenges in life sciences research. *Clinical Therapeutics*, 38(4):688–701, Apr 2016.

[5] Rob High. The era of cognitive systems: An inside look at ibm watson and how it works. *IBM Corporation, Redbooks*, 2012.

[6] Mauro Coccoli, Paolo Maresca, and Lidia Stanganelli. Cognitive computing in education. *BIG DATA*, 12(2):15, 2016.

[7] Stewart Kowalski, Katarina Pavlovska, and Mikael Goldstein. Two case studies in using chatbots for security training. In *IFIP World Conference on Information Security Education*, pages 265–272. Springer, 2009.

[8] Patrick Bii. Chatbot technology: A possible means of unlocking student potential to learn how to learn. *Educational Research*, 4(2):218–221, 2013.

[9] Jaime Arguello and Kyle Shaffer. Predicting speech acts in mooc forum posts. In *Ninth International AAAI Conference on Web and Social Media*, 2015.

[10] Andrew Wood, Paige Rodeghero, Ameer Armaly, and Collin McMillan. Detecting speech act types in developer question/answer conversations during bug repair. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, page 491–502. ACM, 2018.

[11] Arthur C. Graesser and Natalie K. Person. Question asking during tutoring. *American Educational Research Journal, 31(1):104-137. Arthur C. Graesser, Natalie*, 1994.

[12] Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[13] Grzegorz Kondrak. N-gram similarity and distance. In *International symposium on string processing and information retrieval*, pages 115–126. Springer, 2005.

[14] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[15] Clinton Gormley and Zachary Tong. *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine*. O'Reilly Media, Inc., 2015.

[16] Azat Mardan. Related reading and resources. In *Pro Express. js*, pages 317–320. Springer, 2014.

[17] Mao Xuetao, François Bouchet, and Jean-Paul Sansonnet. Impact of agent's answers variability on its believability and human-likeness and consequent chatbot improvements. In *Proc. of AISB*, page 31–36, 2009.

[18] E. N. Forsythand and C. H. Martell. Lexical and discourse analysis of online chat dialog. In *International Conference on Semantic Computing (ICSC 2007)*, page 19–26, Sep 2007.

[19] Cristian Moldovan, Vasile Rus, and Arthur C. Graesser. Automated speech act classification for online chat. *MAICS*, 710:23–29, 2011.

[20] Miguel Coronado, Carlos A. Iglesias, Álvaro Carrera, and Alberto Mardomingo. A cognitive assistant for learning java featuring social dialogue. *International Journal of Human-Computer Studies*, 117:55–67, 2018.