Universidade do Minho
Escola de Engenharia
Departamento de Informática

António Pedro Mesquita Barbosa

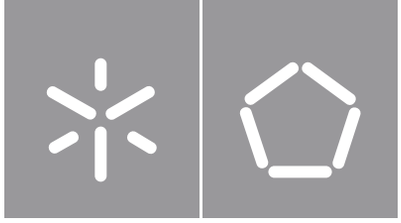
Chatbot Development to Assist Patients In Health Care Services

Outubro de 2020

Chatbot Development to Assist Patients In Health Care Services

António Pedro Mesquita Barbosa

UMinho | 2020



Universidade do Minho
Escola de Engenharia
Departamento de Informática

António Pedro Mesquita Barbosa

Chatbot Development to Assist Patients In Health Care Services

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho efetuado sob a orientação de
Professor Doutor Joaquim Macedo
Professora Doutora Olga Craveiro

Outubro de 2020

Copyright Notice

Acknowledgements

First and foremost, I would like to thank my supervisors Professor Joaquim Macedo and Professor Olga Craveiro for all their availability and support throughout the development of the dissertation. With their expertise and dedication, they were undoubtedly the main key to learn and guide the course of this work.

In recognition, I would also like to thank Carla Vieira, Carmen Rute, Gilda Oliveira, Patrícia Novo, Sara Sousa and Sílvia Mendes, nurses in the Orthopaedic Department, for their crucial contribution of the medical information document.

During this five-year journey, I have to thank my friends whom our life paths have fortunately crossed. To Filipe, for the adventures and always being present, even though you might not have been in Portugal. To José, for our complicity and sharing of musical knowledge. To Rita, for dealing so well with my petty rudeness at my lowest times.

Naturally, a special thanks to David, for being the friend and partner in all aspects of my life, for the long work nights, the fresh moments and memories. A sincere thank you, Miguel, for our long-term friendship, endless support and our trips in Europe.

A huge shout-out to my ESN and Erasmus friends for being patient enough to bear with my supposedly stressful life with a pinch of craziness. Yet trust me in all the work done in recent years—a heartfelt thank-you.

Last but certainly not least. Family, I would not dare to describe what you have done for me since day zero. I am deeply grateful to my parents, António and Teresa, my brother João, my grandmothers, Angelina and Teresa, and my cousin Joana.

Statement of Integrity

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of Universidade do Minho.

Braga, 20th October 2020

Resumo

Dados de alta qualidade sobre tratamentos médicos e de informação técnica tornaram-se acessíveis, criando novas oportunidades de E-Saúde para a recuperação de um paciente. A implementação da aprendizagem automática nestas soluções provou ser essencial e eficaz na elaboração de aplicações para o utilizador para aliviar a sobrecarga do sector de saúde. Atualmente, muitas interações com os utentes são realizadas via telefonemas e mensagens de texto. Os agentes de conversação podem responder a estas questões, fomentando uma rápida interação com os pacientes.

O objetivo fundamental desta dissertação é prestar apoio aos pacientes, fornecendo uma fonte de informação fidedigna que lhes permita instruir-se e esclarecer dúvidas sobre os procedimentos e repercussões dos seus problemas de saúde. Este propósito foi concretizado não apenas através de uma plataforma Web intuitiva e acessível, composta por perguntas frequentes, mas também integrando um agente de conversação inteligente para responder a questões.

Para este fim, cientificamente, foi necessário conduzir a investigação, implementação e viabilidade dos agentes de conversação no domínio fechado para os cuidados de saúde. Constitui um importante contributo para a comunidade de desenvolvimento de chatbots, na qual se reúnem as últimas inovações e descobertas, bem os desafios actuais da aprendizagem automática, contribuindo para a consciencialização desta área.

Palavras-chave: Cuidados de Saúde, Processamento de Linguagem Natural, Aprendizagem Automática, Chatbot, Perguntas e Respostas

Abstract

High-quality data on medical treatments and facility-level information has become accessible, creating new eHealth opportunities for the recuperation of a patient. Machine learning implementation in these solutions has been proven to be essential and effective in building user-centred applications to relieves the burden on the healthcare sector. Nowadays, many patient interactions are handled through healthcare services via phone calls and text message exchange. Conversation agents can provide answers to these queries, promoting fast patient interaction.

The underlying aim of this dissertation is to assist patients by providing a reliable source of information to educate themselves and clarify any doubts about procedures and implications of their health issue. This purpose was achieved not only through an intuitive and accessible web platform, with frequently asked questions, but also by integrating an intelligent chatting agent to answer questions.

To this end, scientifically, it was necessary to conduct the research, implementation and feasibility of closed-domain conversation agents for healthcare. It is a valuable input for the chatbot development community, which assembles the latest innovations and findings, as well as the current challenges of machine learning, contributing to the awareness of this field.

Keywords: Healthcare, Natural Language Processing, Machine Learning, Chatbot, Question Answering

Contents

List of Figures

List of Tables

List of Listings

List of Acronyms

AGATA Automatic Generation of AIML from Text Acquisition.

AI Artificial Intelligence.

AIML Artificial Intelligence Markup Language.

ALICE Artificial Linguistic Internet Computer Entity.

ANN Artificial Neural Network.

AP Average Precision.

API Application Programming Interface.

B2B Business-to-Business.

B2C Business-to-Consumer.

BERT Bidirectional Encoder Representation from Transformers.

BiLSTM Bidirectional Long Short-Term Memory.

BM25 Best Matching 25.

BOW bag-of-words.

CMS Content Management System.

CPU Central processing unit.

EM Exact Match.

FAQ Frequently Asked Questions.

FCT Foundation for Science and Technology.

GPU Graphics processing unit.

GUI Graphical user interface.

HF HuggingFace.

HITL Human-In-The-Loop.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

IaaS Infrastructure as a Service.

IDF Inverse Document Frequency.

IoT Internet of things.

IP Internet Protocol.

IPA intelligent personal assistants.

IR Information Retrieval.

JSON JavaScript Object Notation.

LSTM Long Short Term Memory.

mAP Mean Average Precision.

ML machine learning.

NER Named Entity Recognition.

NLG Natural-language generation.

NLP Natural Language Processing.

NLU Natural Language Understanding.

PaaS Platform as a Service.

POS part-of-speech.

QA Question Answering.

R&D Research and Development.

RAM Random-access memory.

REST Representational state transfer.

RNN Recurrent Neural Network.

RQ Research Question.

SDK Software development kit.

SE Software Engineering.

seq2seq Sequence-to-sequence.

SLR Systematic Literature Review.

SQL Structured Query Language.

SQuAD Stanford Question Answering Dataset.

SSD Solid-state drive.

SSH Secure Shell.

SSL Secure Sockets Layer.

TF Term Frequency.

TF-IDF Term Frequency - Inverse Document Frequency.

UI User Interface.

UX User Experience.

VM Virtual Machine.

XML Extensible Markup Language.

Introduction 1

> "I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."
> — Turing (1950)

This chapter introduces the purpose of this dissertation by given context, goals and outcomes on the topic of assisting patients through conversation agents. Moreover, the document structure is described according to chapters.

1.1 Context and Motivation

Currently, given the recent improvement in health-related information, an enormous amount of high-quality data on medical treatments and healthcare providers has become accessible, leading to new opportunities for a patient's recovery. Modern technological advances have boosted these opportunities' development and release, integrating seamlessly into the health ecosystem (Øvretveit et al., 2007).

These eHealth applications have been used to lessen the pressure on the healthcare sector by establishing a change in a patient's behaviour in daily life either under the supervision of a healthcare professional or in stand-alone use to promote self-management (Bohr and Memarzadeh, 2020).

In the same scope, machine learning shows robust performance and is mostly enforced in healthcare, providing a solution for lowering the rising cost of medical care and serving to build better communication between patient and health professional.

As can be seen nowadays, health care services are dealing with many patients interactions by phone calls. After hospital discharge, patients are facing some unknown

signs and symptoms, and for that reason, they do not know that they are normal or not. Consequently, the services are contacted often by the patients with their concerns. In fact, many of the patient questions have a known procedure.

Here is the motivation on both sides to work on, i.e., to relieve the repetitive work of the health professionals as well as to clarify quickly and credibly the consistent doubts that each patient has.

For this purpose, with the guidelines provided by doctors and nurses, it is possible to understand what are the urgent needs and reliable information feasible to combine with technology. As previously stated, the combination of machine learning methods in the hospital context can result in a viable product to apply in the recovery and palliative care.

Taking all the above into consideration, the outcome expected is a then a product that should incorporate an intuitive application with medical information, together with an automatic system able to process questions from patients in real-time.

Here is where the chatbot field enters to solve those system requirements (Amato et al., 2017; ter Stal et al., 2020). For the questions, the answers can be given by conversation agents. They take the work off from the health services, allowing a quick interaction with patients—this means a real benefit for both participants (Dale, 2016).

Although the first computer program with a chatbot characteristics dates back to 60s (ELIZA) (Weizenbaum, 1983), only in recent years, chatbots have been the focus of many research, and now they are beginning to be widely used by companies (Shum et al., 2018).

Chatbots try to understand user's needs and then give them useful information or assist them to perform a specific task. Indeed, a chatbot is a computer program that interacts with users using natural language, in a specific domain. The conversation is

due to Natural Language Processing (NLP) paradigm, using modules such as Natural Language Understanding (NLU) for question answering (Huang et al., 2007; Galitsky, 2019).

Conclusively, regarding the research area, it is required to evaluate the feasibility of implementing a question answering conversation agent for healthcare, focusing on those benefiting from self-learning and natural approaches in a closed-domain.

1.2 Objectives and Goals

The main goal of the work is to **develop an application to assist patients clarifying any doubts as quickly as possible**, whether it is finding out more information, or answering some questions.

To achieve the purpose of the dissertation, the following tasks must be carried out:

- Write state of the art about algorithms, and tools used in chatbot building;
- Develop a platform to make available the main Frequently Asked Questions (FAQ), with a special concern to a simple and user-friendly interface;
- Build a chatbot application with a self-learning approach and integrate it into the application;
- Evaluate the chatbot performance and the user satisfaction of the solution developed;
- Write a paper on chatbots for healthcare for a scientific conference/journal.

1.3 Work Outcomes

During the development of this dissertation, a wide variety of work has been produced. Along with all the learning and research associated, there are mainly three tangible results of this work.

The most important is the minimum viable product that presents the full-stack application (Chapter 5) integrating the chatbot and FAQs about palliative care [1]. The project developed is live in `aidbot.vercel.app` as well as the backend running on `188.166.67.79`.

The developed data collection can be seen as a contribution since it is available for the community [2]. The dataset has the medical information from the health professionals' original document, plus the real users' questions retrieved for the experimental phase (Chapter 6).

Lastly, the submission of a full-paper manuscript (Barbosa et al., under review) have been conducted about the unbiased and concise Systematic Literature Review (Chapter 3) to special issue in the "Pattern Analysis and Applications" Journal. The status is "Under Review" at the time of delivery this dissertation and notification of acceptance will be given in December 2020.

1.4 Document Structure

The dissertation is organised as follows. Chapter 2 reviews the state-of-art of chatbots which presents a historical context, as well as general design features and development platforms.

1 https://github.com/apedrob/thesis-project
2 https://github.com/apedrob/thesis-data

Next, Chapter 3 outlines the adopted Systematic Literature Review, a methodology for the literature review. It presents a synthesis of the main findings and gaps, concluding with recommendations for further research.

Chapter 4 proposes a high-level, modular architecture and decisions regarding the chatbot and the platform. The modular development and implementation of the conversation agent and web application are described in Chapter 5.

Concerning the setup, deployment and experiment, Chapter 6 exposes the evaluation and viability of the proposed system. Lastly, Chapter 7 draws the concluding discussion, along with viable future work.

State of the art 2

This chapter presents the historical context of chatbots. Then, it examines how the conversation agents are classified and where each one is useful, followed by a general architecture of a conversational system. Next, it discusses the platforms that can be used to build a bot and compare them for the features they present.

2.1 Historical Background

A Chatbot is a computer program that can engage with a person using natural language throughout voice commands or textual dialogue, behaving as a conversational human partner.

Alan Turing, cryptoanalyst, pioneer of computer science and artificial intelligence, proposed the question ”Can machines think?” in Turing (1950). Instead of following a traditional approach to answering this highly ambiguous question, he reformulates this problem with a test which he called “Imitation Game”, latter Turing Test (Figure 1).

While exists a dispute about Turing’s intention (Moor, 2003; Traiger, 2003), the general interpretation (Saygin et al., 2000) of this game consists of three players - an interrogator, a machine and a human. The interrogator aims to determine which one of the two players is the human, solely from the answers of a written question-answer test.

Fifteen years after Turing, Weizenbaum (1966) came up with ELIZA, an early natural language conversational program, the first able to pass the Turing Test. Joseph Weizenbaum developed ELIZA to replicate the behaviour of a Rogerian psychotherapist.

The conversation model involves *“the identification of keywords, the discovery of minimal context, the choice of appropriate transformations, generation of responses in the absence of keywords, and the provision of an editing capability for ELIZA scripts”* (Weizenbaum, 1966).

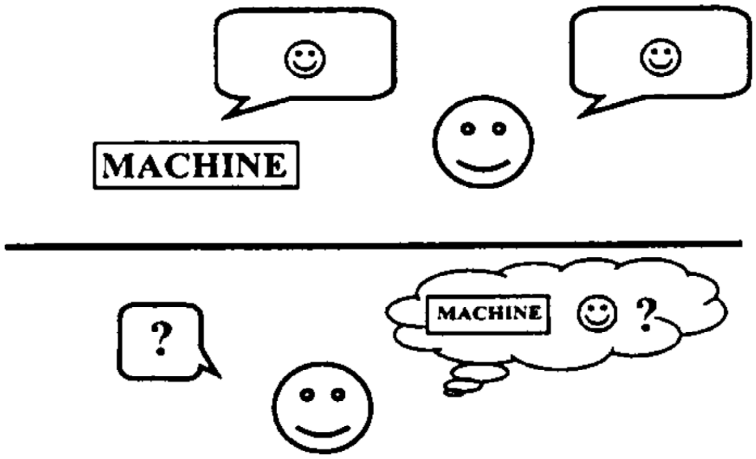


Figure 1: General interpretation of the Imitation Game (Saygin et al., 2000)

Essentially, when the input sentence matched the pre-defined knowledge script, ELIZA would answer by rephrasing the input, i.e., answering questions with questions (Figure 2).

Although the processing involved had been purely superficial, with these techniques, it induced an illusion of understanding (Weizenbaum, 1976; Thomas et al., 1995).

ELIZA sparked engagement in Artificial Intelligence (AI) research, serving as a base for numerous upcoming chatbots with its rule-based approach and pattern matching techniques.

Colby (1975) work on PARRY marks an impactful step in simulating typed conversations by creating a bot which simulates a person with paranoid schizophrenia. The program embodied a more complex conversational strategy (Cerf, 1973) than ELIZA since it resembles an own internal emotional state with a paranoid behaviour. Boden (2008) describes PARRY as "*ELIZA with attitude, where the attitude was systematically grounded in its anxieties and beliefs*".

Men are all alike.

IN WHAT WAY?

They're always bugging us about something or other.

CAN YOU THINK OF A SPECIFIC EXAMPLE?

Well, my boyfriend made me come here.

YOUR BOYFRIEND MADE YOU COME HERE

He says I'm depressed much of the time.

I AM SORRY TO HEAR YOU ARE DEPRESSED

It's true. I am unhappy

DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY

I need some help, that much seems certain.

Figure 2: Extract of ELIZA's typical conversation in Weizenbaum (1966)

The first Loebner Prize competition occurred in 1991, underwritten by Dr Hugh Loebner, the National Science Foundation, and the Sloan Foundation. This annual competition in artificial intelligence awarded up to $100,000 to the computer programs judged to be the most human-like by passing an unrestricted Turing Test (Epstein, 1992).

Even the work in Mauldin (1994) that originated the term "ChatterBot" was an entry in this contest. Although the main objective was to advance the state of AI research, this competition had been criticised in Shieber (1994) as inappropriate given the current level of technology.

However, ever since, chatbot development with different goals including other functional features arose. Artificial Linguistic Internet Computer Entity (ALICE) appeared in 1995, invented by Dr Richard Wallace.

Even though it is based on ELIZA, Alicebot acts as a general humanoid instead of having a specific role with a more extensive natural language sample data collection provided by

the internet (Wallace, 2009). ALICE uses a markup language called Artificial Intelligence Markup Language (AIML), an Extensible Markup Language (XML) language designed for developing pattern matching stimulus-response chatbots (Epstein et al., 2009).

Nowadays, due to significant progress in AI and wireless technologies, have emerged new types of conversational systems.

Social chatbots, such as XiaoIce (Zhou et al., 2018) or the awarded Mitsuku (Worswick, 2019), have an impactful social role by establishing an emotional connection with users. Those chatbots act as a long period virtual companion through multiple approaches, e.g, text, voice and vision (Shum et al., 2018).

Another area is the tremendous investment in developing intelligent personal assistantss (IPAs) that are intended to answer a broad array of questions the users might have, usually deployed on mobile devices. As an example, Apple's Siri, Microsoft's Cortana, Google Assistant, and Amazon's Alexa.

2.2 Classification and Types of Chatbots

Conversation agents can be used for multiple objectives. There are several potential implementation variations which lead to a different classification. For example, Business-to-Business (B2B) and Business-to-Consumer (B2C) or group and personal bots (Radziwill and Benton, 2017).

Sansonnet et al. (2006) affirms that a Conversation Agent embodies three functions: comprehension, competence, and presence. The combination of these requirements can lead to different types of chatbot.

According to Nimavat and Champaneria (2017), a text-based chatbot can be classified, as shown in Figure 3, on the following criteria: Knowledge Domain, Service Provided, Goal and Response Generation Method.

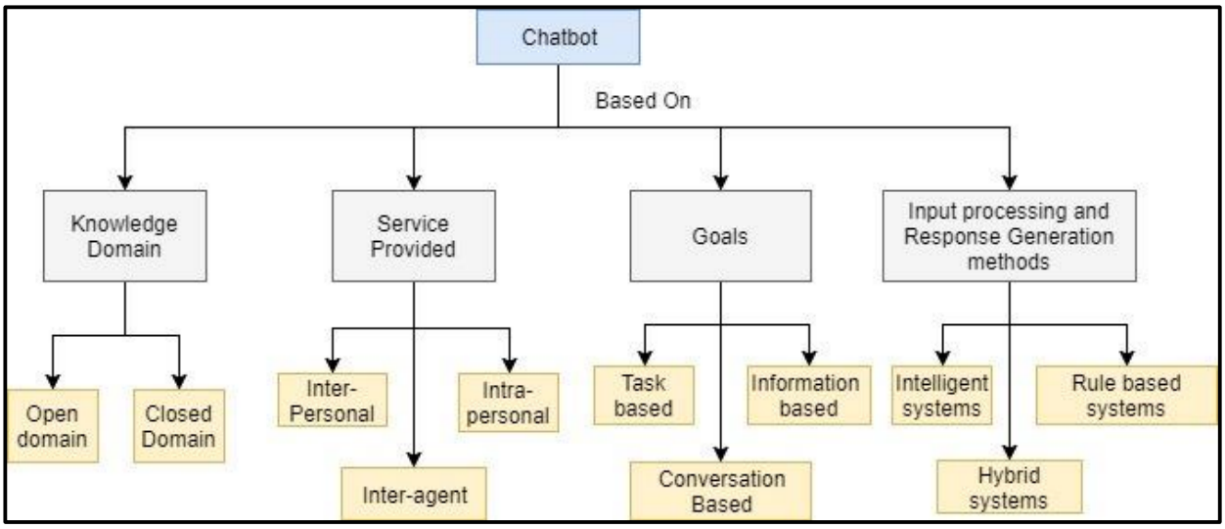


Figure 3: Chatbot Classification in Nimavat and Champaneria (2017)

Regarding the Knowledge Domain principle, chatbots are, mostly, closed-domain meaning they are trained and focused on a particular area of expertise, for instance, a restaurant booking bot. Contrarily, open-domain agents can replicate a conversation, talking about general topics acting as a chit-chat bot (Csaky, 2019).

Chatbots can differ in emotional proximity to the user, this parameter is called Service Provided. An interpersonal bot falls under a service domain, its aim is to give information to the user, such as booking assistants or FAQ bots; however, intrapersonal agents exist in a personal area, acting as companions with a unique personality to the user, e.g., calendar managers or behavioural therapist bots.

Yet, the inter-agent class can be defined by Internet of things (IoT) communication, where two or more systems have their services linked and integrated. The emerging SingularityNET, described in Goertzel et al. (2017), is an example of inter-agent communication within a network.

In general, conversation agents are classified into three classes on what they want to achieve (Nimavat and Champaneria, 2017): informative, retrieving information stored from querying a database; chat-based, replicating a human being with continuous conversation and task-based, their actions are under a specific task following predefined events.

An intelligent system handles the method of processing inputs and designing responses when uses NLU to comprehend the user's query. In a rule-based system (Mauldin, 1994), parsing and pattern matching methods are used to create rigid answers but with limited and fixed outcomes.

The main difference between these types of system is the existence of a self-learning algorithm in the intelligent system to process and generate the response, such as the usage of an Artificial Neural Network (ANN) (Lu et al., 2008).

Although, both methods can coexist, creating a hybrid system with rules to manage the conversation flow and machine learning to provide responses.

Recent studies (Hussain et al., 2019) shows that following an overall chatbot classification can be subjective and deprecated to the scope of a specific use owing to the fact that the chatbot field is very dynamic due mainly to the emerge of new technologies. Consequently, this paper proposes a universal chatbot classification focusing and converging on the goals, as demonstrated in Figure 4.

The authors have formulated and proposed a suitable classification of chatbot within the research goals for this context. This proposal centers the knowledge on until where its domain can expand and how a conversation can be designed (Figure 5):

- Knowledge Area = Broad $\oplus$ Restrict

- Conversation Design = Rule-based + Artificial Intelligence

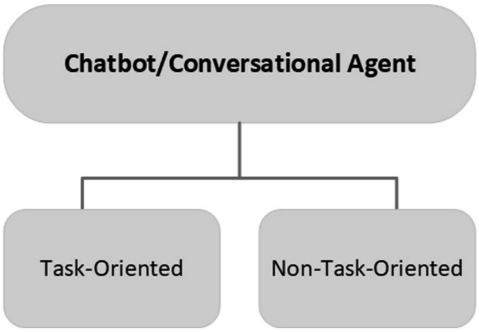


Figure 4: General Classification of Chatbots in Hussain et al. (2019)

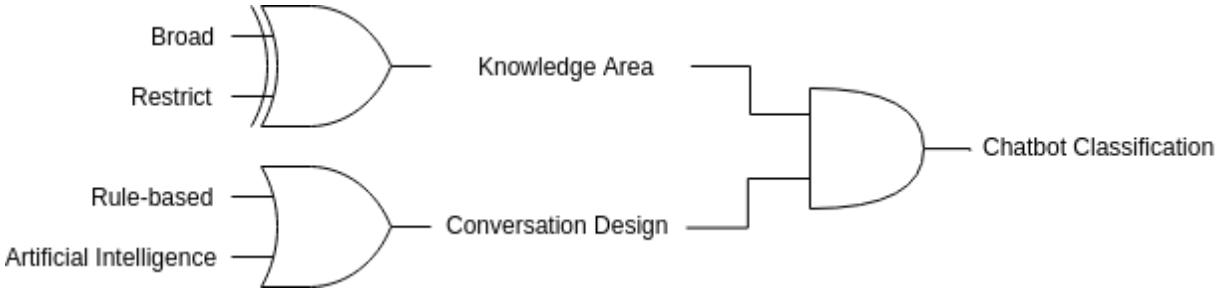


Figure 5: Chatbot Classification Proposal

Concerning the Knowledge Area criteria, restrict chatbots are aimed to help the user with a particular task and dealing with specific scenarios, acting as goal-oriented chatbots within a closed domain. However, broad chatbots can carry out extended, unstructured and inconsequential conversations in open domains, simulating a human-human interaction.

The Conversation Design principle is equivalent to the Response Generation Method criteria (Nimavat and Champaneria, 2017), where the design approaches can be either using a rule-based engine or retrieval/generative-based models.

Although, these design techniques can overlap, it differs from the broad/restrict area of knowledge where they cannot coexist in the conversational system.

2.3 Chatbot Architecture

As described above, exists multiple conversation agents with a combination of different parameters. However, chatbots follow a general structure, a pipeline (Figure 6).

This flow (Nimavat and Champaneria, 2017) consists of four stages: the input is processed into machine representation, the entities are extracted and intent is discovered. With this input understanding, possible responses are generated and filter to give the most proper answer to the user.

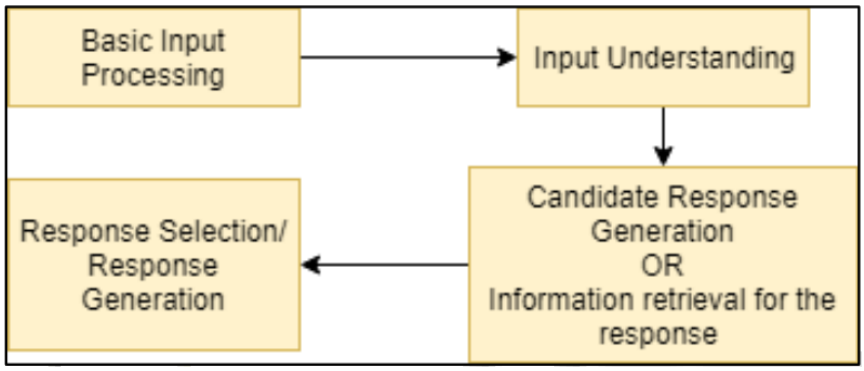


Figure 6: Chatbot Architecture in Nimavat and Champaneria (2017)

Ramesh et al. (2017) presents a more complex chatbot structure and its design principle can be extended to chatbot development. Therefore, the system is divided into three constituent modules, as illustrated in Figure 7:

- Responder: manage and monitor the data transfer between the user and the classifier, functioning as an interface amongst the user and chatbot.

- Classifier: normalise and filter the input, transferring it to the graph master through segmentation into logical components. Plus, it does graphmaster output processing and database syntax instructions handling.

- Graphmaster: serving as the brain of the bot by organising all the contents which is responsible for pattern matching algorithms and storage.

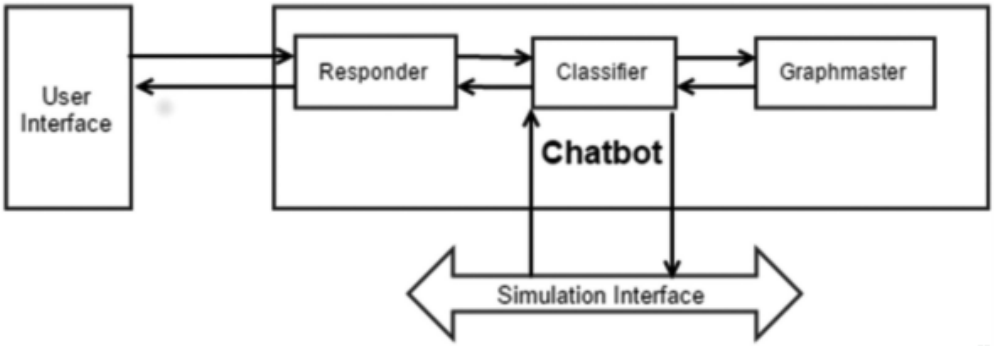


Figure 7: Chatbot Architecture in Ramesh et al. (2017)

2.4 Approaches to Chatbot Development

There are several different approaches for modelling restrict or broad chatbot. Such methods may overlap in these two main chatbot types. This section explores those approaches that have been used in the development of chatbots since the very early chatbot ELIZA (Weizenbaum, 1983). These approaches may be divided into three main categories (Hien et al., 2018), as illustrated in Figure 8: rule-based, retrieval-based, and generative-based.

2.4.1 Rule-Based Approach

In a rule-based approach, a bot answer questions relying on specific rules in which it is trained. While developing such bots is relatively straightforward, the bot is unsuccessful in answering questions if the user input pattern does not match the trained rules.

Besides, it is very time consuming to write rules for different scenarios and it is difficult to write rules for every possible scenario. Bots can handle simple queries but neglect

complex inputs, therefore, it could not pass the Turing test (Turing, 1950; Mauldin, 1994).

2.4.2 Retrieval-Based Approach

Conversation agents are trained on a set of questions and their likely outcomes with retrieval-based models (Hien et al., 2018). The bot can identify the most accurate answer from a repository of all possible answers for each and every question and ultimately deliver the output answer.

The heuristic can be as basic as matching a rule-based expression or as complex as ensembling machine learning classifiers. However, the bot can not generate new responses in which the language and grammar are pre-determined, leading to not processing syntactically incorrect phrases.

Therefore, it can be used when the data is limited and the knowledge area is restricted to a few conversation scenarios such as booking systems, FAQ systems or other systems that fetch information.

2.4.3 Generative-Based Approach

Generative models differ from rule-based models in a sense that they can generate new answers, not relying solely upon predefined responses. It makes them smarter, as they take the question word by word and produce the responses.

However, they are more prone to mistakes, since they take into account spelling and grammar. That way, such models need to be trained more carefully in order to make them better at overcoming such errors.

Once trained on a large amount of processed data, they outperform rule-based models, as they can adapt to complicated and unforeseen queries. These systems often use NLP and NLU algorithms to process input and generate sentences (Lu et al., 2008).

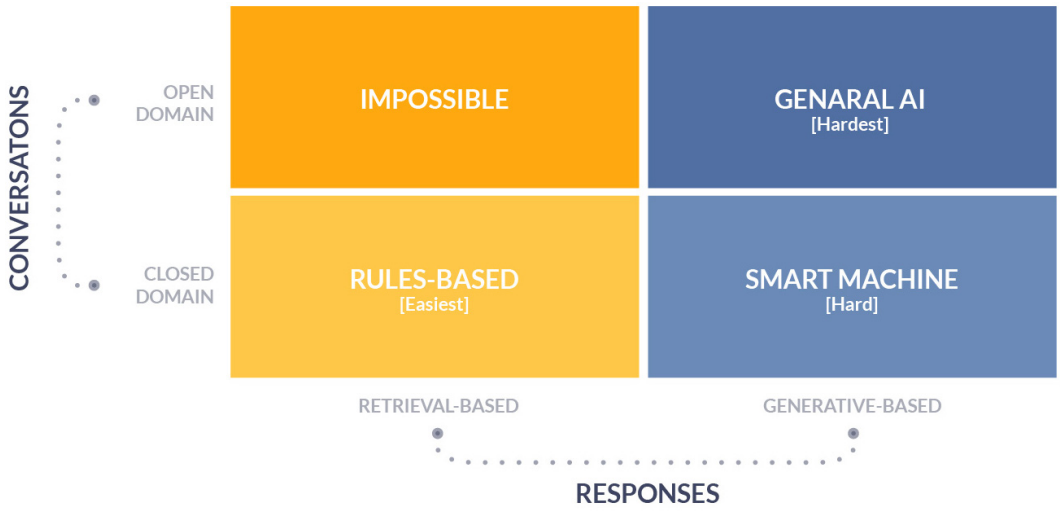


Figure 8: Chatbot Conversation Framework in Bobriakov (2019)

2.5 Chatbot Development Platforms

Through the increasing development of chatbots, the implementation from scratch of its architecture has been facilitated with the help of emerging platforms that identify and classify the purpose and context of the particular input from a range of possibilities. This identification is defined by using **Utterances**, **Intents** and **Entities**.

An Utterance is the text introduced by the user that the bot needs to interpret. It has assigned a single intent but can include many entities or none at all.

An Intent represents the task or action of a user's utterance.

An Entity represents a unit of data you want to be extracted from the utterance that provides a specific context for an intent.

For example, a user types the following utterance: “show me the weather in Braga for tomorrow”, the user’s intent is to know the weather, and the entities are “Braga” and “tomorrow”. Intents are given a name, often a verb and a noun, such as “getWeather”. Entities are given a name, such as “place” and “dateTime”.

Chatbot development platforms can be divided into two categories (Bobriakov, 2019):

- **Frameworks** are a convenient package with a standard way to develop a generic chatbot workflow. Similar to any other software (e.g. web application) frameworks, chatbot frameworks provide tools and utilities with the aim of automating the overhead associated with common chatbot development tasks. Usually, they are implemented with a specific programming language and some of those frameworks have also hosted and interactive development environments to make creating and deploying bots.

- **Services** are high-level cloud-hosted platforms, usually providing a Graphical user interface (GUI) for interactive chatbot logic development, incorporating predefined NLP functionalities, and supporting a Representational state transfer (REST) Application Programming Interface (API) connection.

In this research, the authors chose to use frameworks instead of services allowing more versatility and control not only in the architecture and modeling the chatbot as proposed but also designing the final solution.

The following subsections explain some of the most reliable and well-known frameworks (Braun et al., 2017): Microsoft Bot Framework, Botkit, and Rasa.

2.5.1 Microsoft Bot Framework

Microsoft Bot Framework is a Microsoft's Software development kit (SDK) for building intelligent agents. The central component Bot Connector is responsible for integrating and developing a robust bot logic.

Although it comes in the form of a C# project template, all the structure is REST-based which can be used to deploy and host in any application or website. Regarding natural language processing and understanding, this framework features LUIS.ai as an optional component in order to give a sense of humanity to the bot. In addition, the framework supports translation to more than 30 languages.

2.5.2 Botkit

Botkit is an open-source Node.js based SDK, recently acquired by Microsoft. Botkit is the leading developer tool for building chatbots with custom integrations for major messaging platforms such as Facebook, Microsoft Teams and Slack.

The framework has straightforward workflows structured in a clear and concise way, it is well-documented and has plenty of chatbot samples to explore. However, this platform does not have NLP functionalities but can be handled by integrating with existing or custom NLP resources through middleware.

Furthermore, Botkit is self-hosted and provides a web-chat plugin capable to embed on any website.

2.5.3 Rasa

Rasa is an open-source Python-based machine learning framework for building contextual assistants. There are two main components where separate implementation is possible: NLU and Core.

Rasa NLU is responsible for natural language understanding, assesses what the user wants and captures key contextual information (Braun et al., 2017). It can also run as a service for another framework via REST endpoints.

The Core component determines the communication flow, choosing the next best answer or action based on conversation history. It allows a complex conversation, trained by supervised machine learning algorithms.

The chatbot is self-hosted, storing all the data and components internally, and can be integrated smoothly into applications written in Python.

Systematic Literature Review 3

The chapter is organised into four main parts: The first outlines the methodology for the literature review as follows. The second part describes the Systematic Literature Review (SLR) methodology, along with all the steps taken from the Research Questions (RQs) definition to data extraction.

Then, it presents a synthesis of the main findings and gaps, together with recommendations for further research. Finally, the limitations and conclusions from this SLR are provided.

3.1 Methodology

A thesis dissertation must satisfy several academic requirements, such as performing a literature review. This examination provides a proper in-depth analysis, critical evaluation and statistical knowledge about the research area (Fink, 2019). It is essential to justify the relevance of the proposed work, avoid duplicating results and also find gaps in other published research to further investigation.

One of the most conventional approach to carry out extensive research of the literature is through a SLR. This type of study has a well-defined methodology to identify, evaluate and interpret all the relevant work within the ongoing research (Keele et al., 2007).

Since there are several examples of SLR in multiple disciplines (Barcelos and Travassos, 2006; Boren and Moxley, 2015; Dybå et al., 2006), it is relevant to comprehend how can the correlation between SE and Medicine, in which systematic reviews were first developed, influences the choice of this methodology.

Budgen et al. (2006) examined the similarity between software engineering research practices and those of other domains through experimental practices, subject types

and blinding procedures. Table 1 compares SE experimental methodology with other disciplines.

This analysis concluded that software engineering differs significantly from the traditional medical arena because SE does not conduct randomised controlled trials or to undertake double-blinding and human meddling influences all results of experiments.

Table 1: Comparing SE methodology with other fields in Budgen et al. (2006) (1 is agreement, 0 is disagreement)

Field	*Comparison with SE*
Education	0.83
Empirical Psychology	0.66
Clinical Medicine	0.17

Therefore, this research follows a strict directive to carry out a systematic literature review only for software engineering research, as firstly stated by Keele et al. (2007) and adopted in Amara et al. (2016); Gonçalves et al. (2019). The methodology above is a guideline to efficiently assess and select existing studies suitable for specific research, covering three phases of an SLR, i.e., planning, conducting and reporting the review.

The main reason for undertaking the systematic literature review in this research is the obligation to follow a predefined search strategy where assess the literature and makes the authors identify and report not only research that does not support the study hypothesis but also research that confirms it.

In other words, the literature reviewed is less likely to be biased with little scientific value, synthesise efficiently the existing data concerning a method or technology and identify any gaps for further investigation in current research. However, it requires significantly more effort than traditional literature reviews.

The approach follows six steps, shown in Figure 9: define the research questions, stipulate literature sources and search string, select relevant studies, assessing the gathered studies, extract data from those studies and synthesise the selected data.

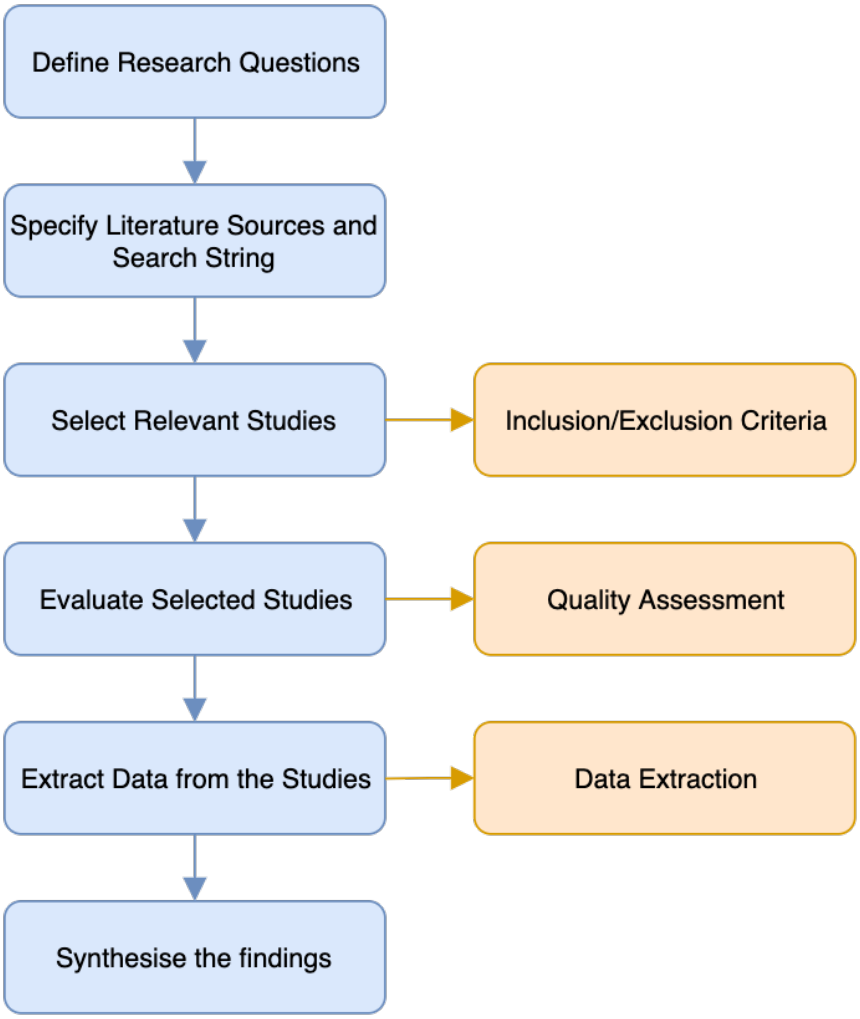


Figure 9: Steps for conducting the SLR

3.2 Research Questions

The purpose of this review is to identify the state of the art of developing a text-based chatbot, specifically for medical assistance. As a result, the RQs established are the following:

RQ1: What is the standard classification used in chatbots?

Nowadays exists diverse types of chatbots due to the innovation and dynamisation in this area. The intent of this RQ is to identify a general classification for those various mixed chatbots.

RQ2: Which architecture should be used for a task-based chatbot?

Likewise to classification, several architectures have been established over the last years, combining different parameters. This question aims to understand which chatbot structure is best suited under a specific task.

RQ3: What are the approaches used for modelling a closed-domain conversation agent?

Chatbots can be broadly divided into two groups: closed-domain chatbots, focusing in a particular area of expertise, and open-domain chatbots, which can have general-purpose discussions with the user. As a result, different methods are used when modelling a conversation agent.

Therefore, this RQ serves to identify the best development approach within the environment of this work.

RQ4: Which frameworks are used for building a chatbot?

With the purpose of building a chatbot, several platforms facilitate its implementation from scratch. The goal of this RQ is to search and compare the existing frameworks, so then choose the most suitable tool to build the solution.

RQ5: How can self-learning algorithms improve chatbots?

When processing and generating responses, self-learning algorithms can be used instead of traditional parsing and pattern matching techniques. Thus, these question's scope is to uncover how and where can those machine learning (ML) algorithms improve the chatbot performance.

RQ6: How can an interpersonal chatbot be tested and evaluated?

Undoubtedly, the best way to assess a chatbot is to test it with human interaction. Nevertheless, numerous resources such as individuals, materials, and time are required for this process.

Therefore, simulations and grading systems must be used. This RQ aims to find which metrics can be used to test and evaluate a conversation agent.

This SLR collects knowledge related to medical technology, such as developing health chatbots, in addition to these six research questions.

3.3 Literature Sources and Search String

Using the previously established RQs, in this section, a search string is constructed and applied in established research sources.

The search of studies in this SLR is outlined in major literature databases in order to curate not only reliable studies, but also extensive sources: ACM Digital Library, IEEE Xplore, Science Direct and PubMed.

ACM Digital Library (Online Database) is a research, discovery and networking platform containing a full-text collection of articles and bibliographic literature directed exclusively on the area of computing. IEEE Xplore (Online Database) is a research database of articles on computer science, electrical engineering and electronics and related fields. Science Direct (Online Database) is a freely available platform of peer-reviewed literature covering scientific, technical, and medical research. PubMed (Online Search Engine) is a free platform promoting the search and collection of literature on biomedicine and life sciences from MEDLINE, life science journals, and online books with the goal of improving health both internationally and individually.

After setting the research sources, the search string has been constructed using the method proposed in Keele et al. (2007); Amara et al. (2016):

1. Extract the main search terms from the Research Questions (Table 2);
2. From studies previously reviewed - seeds, collect additional search terms;
3. Find synonyms and alternative spellings for each main search term;
4. Build a search string connected with booleans operators; OR to attach synonyms, and AND to join main search terms.

Despite this pre-established method, the authors deem necessary to add boolean NOT in order to set apart misleading search terms.

Table 2: Search main terms with synonyms and alternative spellings

Main Terms	*Synonyms and alternative spellings*
chatbot	dialogue system, conversational agent
task	task-based, closed-domain, interpersonal, inter-personal, chat-oriented
text	text-based
voice	speech, talk, voice-based

This review incorporates key search terms such as “chatbot”, “task” and “text”. The main word “voice”, however, is considered as an excluded term.

The resulting search string is the following: (chatbot OR dialogue system OR conversational agent) AND (task OR task-based OR closed-domain OR interpersonal OR inter-personal OR chat-oriented) AND (text OR text-based) AND NOT (voice OR speech OR talk OR voice-based).

Various sources of literature have different search syntax when combining with Boolean complexity. Herewith, it is necessary to rewrite the string above for each research source (Table 3). Some limitations were discovered in certain search engines after iterations of the search string in the several sources: the boolean NOT will discard some studies which have the main term "voice" referred in the full text and did not support that many booleans in the search string.

As a result: ACM search has been conducted within the abstract paper; ScienceDirect string has been reduced to seven booleans including main terms and primary synonyms; since PubMed is a medical database, a broad search string has been given to acquire more research.

3.4 Studies Selection

This SLR used the following search process to guarantee that all relevant studies are identified. First, searching web engines and online databases. Then, searching manually for research events procedures and individual journals. Lastly, scanning the reference lists of all found articles to avoid missing any interesting study.

At this stage, even with a restrictive search string, a collection of 120 papers was identified following this search method mainly due to the vast and increasing chatbots

Table 3: Search string formatted for each literature source

Literature Source	*Search String*
ACM Digital Library	(chatbot OR "dialogue system" OR "conversational agent") AND (task OR task-based OR closed-domain OR interpersonal OR inter-personal OR chat-oriented) AND (text OR text-based) NOT (voice OR speech OR talk OR voice-based)
IEEE Xplore	(chatbot OR "dialogue system" OR "conversational agent") AND (task OR task-based OR closed-domain OR interpersonal OR inter-personal OR chat-oriented) AND (text OR text-based) AND NOT (voice OR speech OR talk OR voice-based)
ScienceDirect	(chatbot OR "conversational agent") AND (task OR "chat-oriented") AND (text OR "text-based") NOT (voice OR speech)
PubMed	(chatbot OR "dialogue system" OR "conversational agent") AND (task OR task-based OR closed-domain OR interpersonal OR inter-personal OR chat-oriented) AND (text OR text-based)

research. For the purpose of eliminating duplicate papers and filtering relevant studies from the collection, a selection method was created using the following inclusion and exclusion criteria based on the screening approach in Amara et al. (2016).

Inclusion criteria

- If a study has available a journal and a conference version, only the journal version will be used;

- If a study has many published versions, only the most recent one is held;

- If a study appears in more than one source, only one copy is included.

Exclusion criteria

- Studies that do not report on primary data such as review articles. Only research articles are kept;

- Studies that do not consider text-based chatbot;

- Studies that do not address a closed-domain architecture;

- Studies that do not handle English and Latin languages.

This selection method followed three recursively steps: first screening of the article title, then abstract filtering and finally a selection on the full-text (Figure 10). These steps are based on Mateen et al. (2013) which concluded that selecting via a titles-first approach may be more efficient than screening titles and abstracts together.

After applying those inclusion and exclusion criteria in the selection steps, a list of 17 papers were filtered and ready for quality assessment.

3.5 Quality Assessment

Conducive to assess the research selected, based on Amara et al. (2016), a quality evaluation checklist was created and applied to each study (Table 4).

The first question is evaluated depending on the source of the paper. The CORE (2006) classification is added whether it is a conference or workshop paper with the values: A* (2); A (2); B (1.5); C (1);. If this CORE ranking does not exist, the rating from the QUALIS (Sucupira, 2014) ranking shall be calculated with the following values: A1-A2 (2); B1-B3 (1.5); B4-B5 (1); If either the paper does not have a rating on any of the two indexes, the value will be: No ranking (0). When the paper is a Journal article, Journal Citation Reports (Clarivate, 2017) is used with the scores: Q1 (2); Q2 (1.5); Q3(1); Q4 (0.5); No JCR (0).

From the second to ninth question, it should have one of the following values: Yes/Both (1); Partially/Only One (0.5); No/Neither (0).

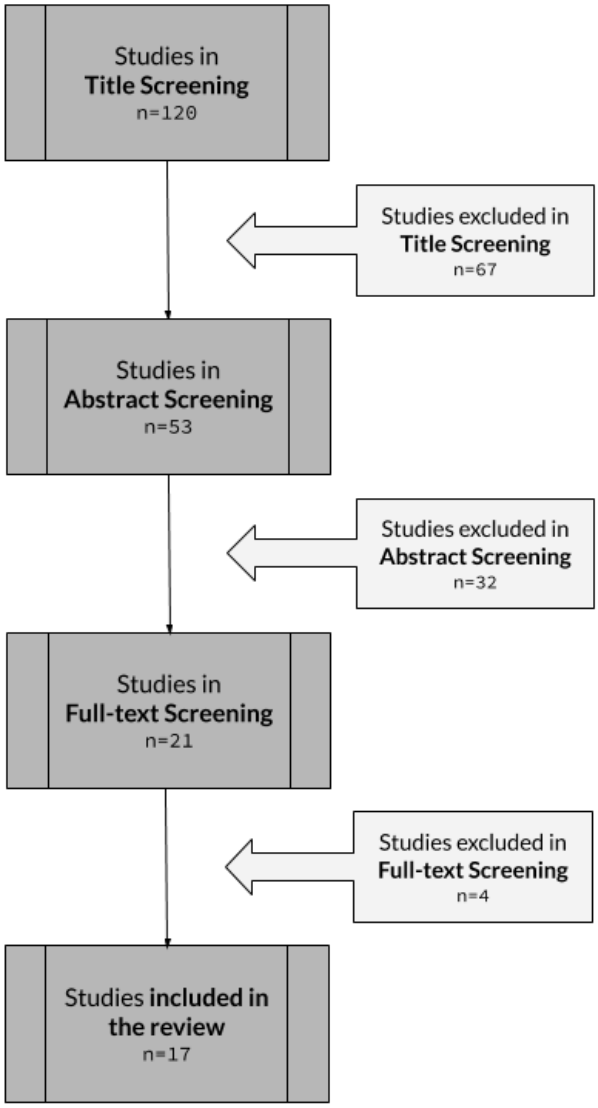


Figure 10: Flow diagram of the selection method through article screenings

Every research which has an overall quality score of less or equal to four will be excluded. A final list of 8 papers was obtained at the end of this process. The selected studies are labelled from S1 to S8, showing respective assessment scores in Table 5.

Table 4: Assessment Criteria Checklist

Question	*Score*
Is the research published in a recognised journal or science conference?	• CORE ranking • JCR ranking
Is there a clear statement of the research's aim?	• Yes • Partially • No
Does the study consider rule and retrieval based approaches?	• Both • Only one • Neither
Does the study apply to healthcare or the medical field?	• Yes • No
Are the findings set out and presented clearly?	• Yes • Partially • No
Have other researchers cited the paper?	• Yes • No

Table 5: Quality assessment scores

Study	*Reference*	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*	*Q6*	*Total*
S1	Hien et al. (2018)	1	1	0.5	0	1	1	*4.5*
S2	Krassmann et al. (2019)	1	1	0.5	0	1	1	*4.5*
S3	Nigam et al. (2019)	1.5	1	1	0	1	1	*5.5*
S4	Shridhar et al. (2019)	2	1	0.5	0	0.5	1	*5*
S5	El Zini et al. (2019)	2	1	0.5	1	0.5	0	*5*
S6	Ropero et al. (2012)	2	0.5	0.5	0	0.5	1	*4.5*
S7	Beaudry et al. (2019)	2	1	0	1	0.5	1	*5.5*
S8	Xu and Zhuge (2020)	2	1	0.5	0	1	0	*4.5*

3.6 Data Extraction

The main objective at this stage is to obtain information from each study that has been retrieved. Thus, to fetch essential data from those research papers, the following extraction parameters were defined:

- Study identifier

- Name of authors

- Title of the paper
- Year of publication
- Paper's type (Journal, conference/workshop proceedings)
- Quality assessment score
- Classification admitted
- Architecture considered
- Approaches used to model chatbot
- Development platforms used
- Self-learning algorithm(s)
- Strategies applied under a healthcare context
- Methods used to evaluate the solution proposed

3.7 Data Synthesis

Synthesising the data collected aims to summarise and report the relevant results of the analysed research. With this, this phase answers each one of the research questions established from the extracted data above and identifies any possible research gaps and recommendations.

3.7.1 Studies Overview

In a broad perspective, eight papers were reviewed and discussed wherein they ranged not just in the venue type (conference or journal), but also in the literature source and published year, as shown in the Figure 11.

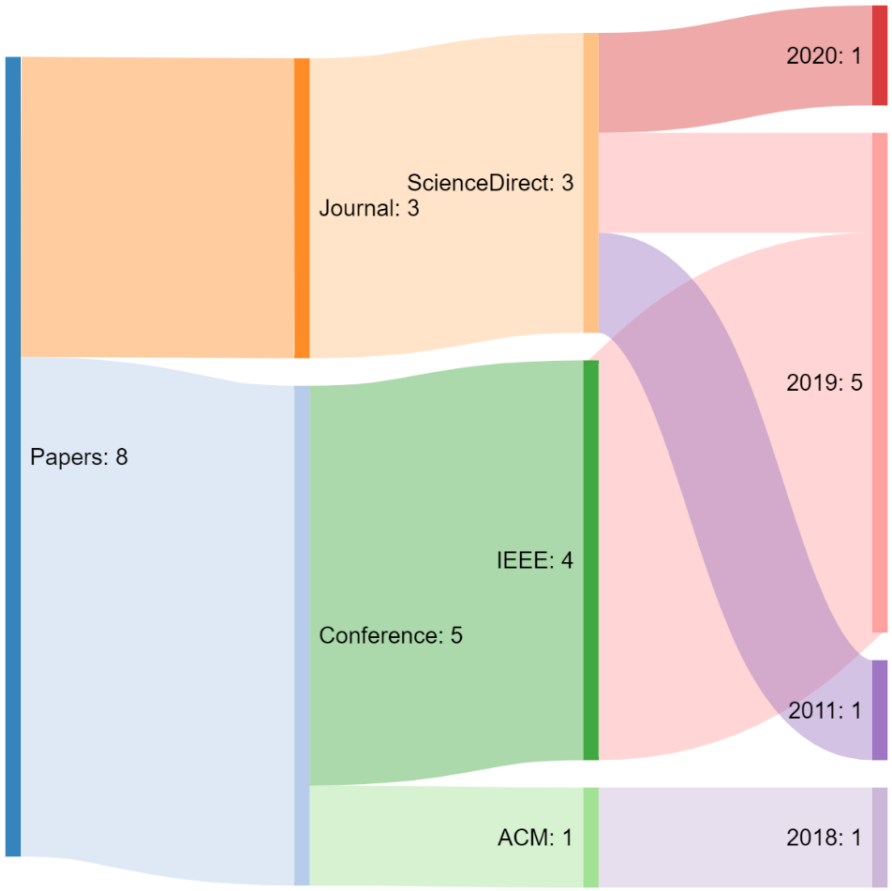


Figure 11: Type, source and date of selected papers

Journal publications are obtained from Science Direct, while most conference papers are retrieved from the IEEE. This discrepancy can be explained by the great involvement, recognition and exposure of IEEE conferences, as well as by ScienceDirect's vast group of scientific fields and journals.

Throughout the systematic review and by this meta-analysis, it has been identified that most articles have been published over the last decade. This verification is explained by the growth and exponential importance of the intelligent agent research and development, and the increasing versatility and adaptability of this technology in other scientific fields, such as medicine. As can be confirmed in the Figure 12, since 2010, the search interest in the term chatbot has increased eightfold.

Therefore, one can conclude that the research on this topic is volatile, recent and constantly evolving. This proves the rising need for up-to-date literature reviews, as this one.

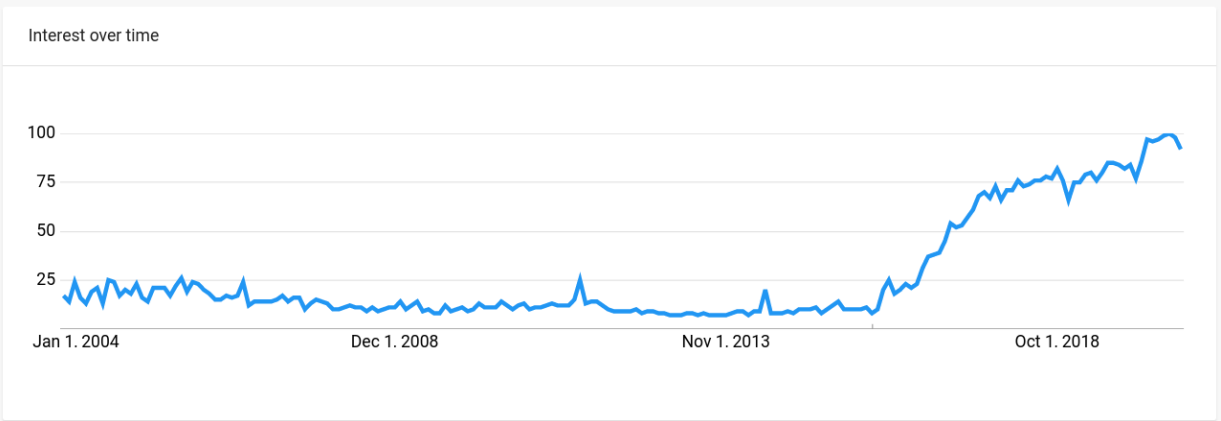


Figure 12: Chatbot search interest over time (Google)

This review also assesses the approach model of the papers towards the Natural-language generation (NLG) algorithm, whether if it is retrieval or rule-based. For this feature, it is shown to be proportionally distributed among the eight articles (Figure 13).

A retrieval-based system (Singhal et al., 2001) is relatively recent, mainly using machine learning and deep learning classifiers as the heuristic. Which is why more recent publications are tackling this system. In contrast, the rule-based approach has been used and researched since the early chatbot ELIZA (Weizenbaum, 1983). Therefore, the use of the rule-based model can be explained in the 2011 paper by its temporal context.

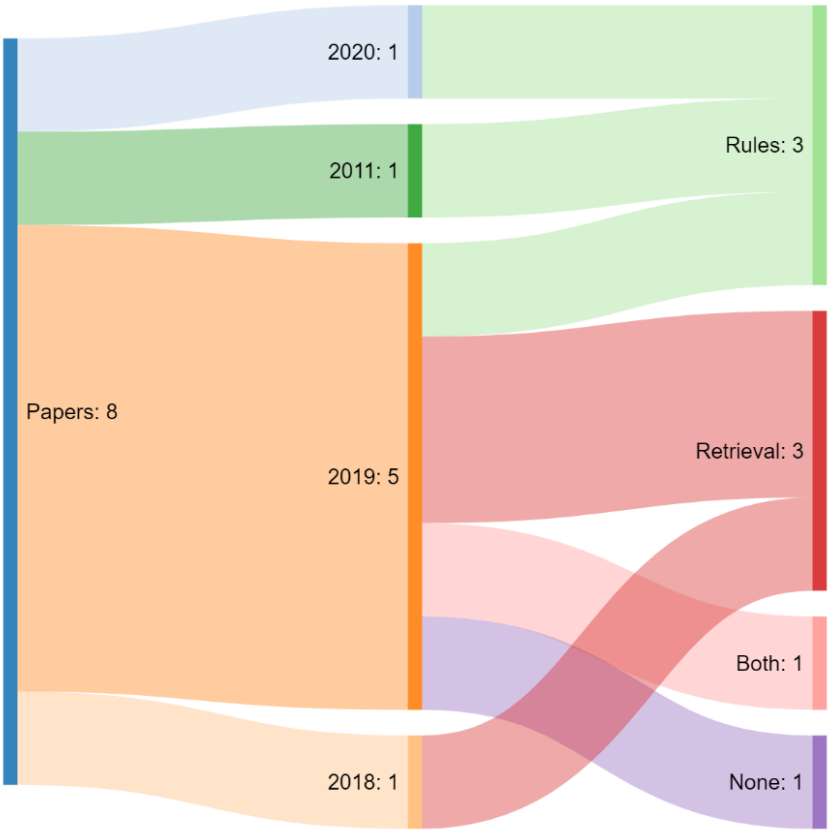


Figure 13: Correlation between published date and algorithm approach

Nevertheless it can be observed that this very principle is being explored by further current research.

Hence, it is feasible to conclude that the two approaches are currently being researched and implemented, which reinforces the initial interest to study and apply both of them.

In the health research context, two articles have been obtained regarding the chatbot's applicability in this domain. There has also been a lack of research with all those characteristics throughout the review. For example, no papers taken from PubMed (an excellent source for this type of research) were deemed suitable or with enough standard to overcome the iterative selection process.

3.7.2 Research Questions Findings

Table 6: Research questions addressed

Study	*RQ1*	*RQ2*	*RQ3*	*RQ4*	*RQ5*	*RQ6*
S1		✓	✓	✓		✓
S2		✓	✓			
S3			✓		✓	
S4				✓	✓	✓
S5	✓		✓		✓	
S6			✓			
S7				✓		✓
S8	✓	✓	✓			

RQ1: What is the standard classification used in chatbots?

S5 explains that a QA natural language system has two main approaches: restricted domain QA and open domain QA. The former is very limited and uses specific terminology, however the latter is wider in scope.

S8 shows that question-answering (closed-domain) chatbots can be classified by their representation of knowledge: structured, semi-structured and unstructured. The first requires a lot of manual work to retrieve data, and the domain knowledge limit border is fragile. Systems using question-answer pairs, such as deep learning applied to vast pair sets or extracting answers from a wide range of dialogs, can be considered to have semi-structured knowledge. Free-text-based QA system has an unstructured area of knowledge, as it extracts answers from a large collection of plain texts and often requires external understanding to deal with semantics in text.

RQ2: Which architecture should be used for a task-based chatbot?

S1 proposes a general structure composed of two modules: User Message Analysis and Response Generation (Figure 14). The analysis component extracts the message aim

(intent) using text classification, and context information (entities) through a Named Entity Recognition (NER) system. The second part is responsible for building and giving a response to the user based on the information retrieved above.

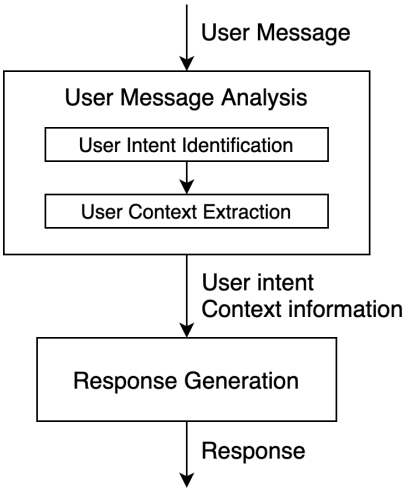


Figure 14: S1 general chatbot structure Hien et al. (2018)

S2 presents an architecture focused on its knowledge base extraction. This process is divided in five phases: pre-processing, cleaning and leaving only textual information; extraction of keywords using Voyant Tools to identify the most frequent ones; selection of keywords, the expert on the subject handily pick those with more potential to correspond to the corpus; selection of sentences, a text file is upload to Automatic Generation of AIML from Text Acquisition (AGATA) system which scans and select sentences with the keywords chosen, and, sentences formatting, the expert can edit and format each outputted sentence. By the end of this flow, an AIML file is generated and sent to the chatbot interpreter.

S8 outlines the following broad design for QA chatbots: Question Analysis, which classifies questions into several types (Five Ws and How); Information Retrieval, which narrows the range of candidate responses in the set of texts; and, finally, Answer Extraction, which selects answers from candidate responses.

RQ3: What are the approaches used for modelling a closed-domain conversation agent?

S1 explains the three models used to produce responses: Pattern-based, Retrieval-based, and Generative. It also affirms that using a generative approach is the smartest. However, it encountered many difficulties when building and training because it needs a vast set of data in order to achieve good performance. S1 prefers to use a retrieval-based model, explaining its flexibility when compared to a pattern approach.

S2 proposes a rule-based approach based on a software solution AGATA that extracts chatbot knowledge bases from a text corporation through AIML. The study consisted of converting the contents of two textbooks into AIML and sending them to a knowledge base for the chatbot. Based on this technology, the chatbot retrieves data using a high-level program that extracts them into an XML file and it provides the user with the information as requested in the message.

S3 follows a hybrid approach using Recurrent Neural Network (RNN) models to predict the intent category and rules for identifying the subcategories and then it retrieves the entities through Named Entity Taggers. Stanford CoreNLP is used to train those taggers with different vocabularies.

S5 seeks retrieval through deep learning, and domain-specific embedding models. From a deeper view the embedding is obtained using two baseline models: bag-of-words (BOW) and an information retrieval model; this allows to create vector representations of words on medical documents instead of relying on pre-trained embedding words.

S6 proposes a new approach based on fuzzy rules logic combining with Support Vector Machine for information extraction. This system design gives flexibility when dealing with a set of abundant, ambiguous, and imprecise data.

S8 presents a Semantic Link Network based QA system, a self-organised semantic model that describes and stores semantic objects and links within texts. This approach extracts

answers from a range of texts (answer range) which induce better text understanding and support patterns for answering questions.

RQ4: Which frameworks are used for building a chatbot?

S1 solution is built on the Dialogflow framework, which has embedded machine learning algorithms and external interaction channels such as Facebook Messenger.

S4 compares its intent classifier performance with existing NLU services and platforms in the market. There are three services from big companies – Dialogflow (Google), Luis (Microsoft) and Watson (IBM) -, platforms from growing start-up, such as Recast and Botfuel, as well from open-source projects like Rasa and Snips.

S7 text messaging chatbot was built with Motion.ai platform (HubSpot company) and Twilio for external message communication service.

RQ5: How can self-learning algorithms improve chatbots?

S3 applies a deep learning model to category and subcategory predictions which allows a slot-gating mechanism, a network for modelling the relationship between intents and slots. This multi-staged method outperforms any preceding state-of-the-art algorithms used for comparative analysis.

S4 explains that a deep networks approach learns more efficiently the features present in the dataset and have better performance when compared to less complex machine learning. However, this model is unsuited for small datasets and does not account for out-of-vocabulary words. So, to overcome these challenges, the paper proposes the idea of Subword Semantic Hashing that uses a hashing method to represent subword tokens.

S5 acknowledges that deep learning has been popular either in the open or closed domain, as it eliminates the burden of extraction of syntactic and semantic features due to the general adoption of a response selection in many deep networks. Also, it integrates Long Short Term Memory (LSTM) cells to learn embedding sentences which collect long-term memory dependencies.

RQ6: How can an interpersonal chatbot be tested and evaluated?

S1 validates its approach through the performance of the User Message Analysis component. The practical experiment has a static number of user intents (13) and context information (3). Each experiment is applied 10-fold cross-validation, which obtains its F-score, used to evaluate this solution. The final results are very satisfactory, with an average F1 between 82 and 97.

S4 assesses its intent classification method's performance using F-score. To initialize randomly and obtain the average score, performs 10 runs per experiment. The final results shows that the algorithm is accurate, versatile and fast, outperforming some of the giants in the field.

S7 system feasibility study was conducted through a six-month intervention using text messaging with thirteen adolescents with chronic conditions in the paediatric domain. Also, surveys were given to assess the acceptability and motivation of its chatbot.

3.7.3 Summary of Findings

This research lists widely acknowledged features and design elements for conversational agents, showing their coverage in the literature. Overall, it is clear that this area is still under developing and nascent, causing controversy on a general and effective approach to design.

Chatbots can be classified mainly for their representation of knowledge and how is structured. Although, it is important to include how the design approach influence this knowledge; if it follows a pattern, a retrieval or generative model. In order to identify the intent and entities of a user message, some existing NLU services can be helpful such as Dialogflow, Botfuel or Rasa.

Through the architectures presented in these papers, it has been evident that there are similarities in developing a task-based chatbot, more-specific QA agents. Therefore, the authors aggregate the several revised modules and propose a generalistic architecture relying on a retrieval-based approach aiming in applying them in the research problem. It is a three-part architecture as shown in Figure 15: firstly, analyses the user's message where extracts the intent and the entities to have the question context; through this given context and QA data previously stored (ideally in pairs), it is extracted and generated several candidates for the response; and finally, the end-user response selection is performed.

A retrieval-based system may apply self-learning in solving several problems including: new intents, lack of entities, incorrect answers, out-of-vocabulary word embeddings and lumpy conversation flow. Although pattern-based heuristics can generate positive outcomes, based on papers, a self-learning approach incorporates a deep learning model, e.g. Recurrent Neural Networks with LSTM cells (Sutskever et al., 2014b) can collect data dependencies which can enhance the conversation flow or generate data augmentation.

Reinforcement learning techniques can be combined with these models to further improve the chatbot, which can generate more interesting responses regarding the reward function (Mnasri, 2019; Li et al., 2016). Depending on the scope, this idea may encompass a Human-In-The-Loop (HITL) learning methodology, which means humans are actively involved in training, tuning and testing of the bot.

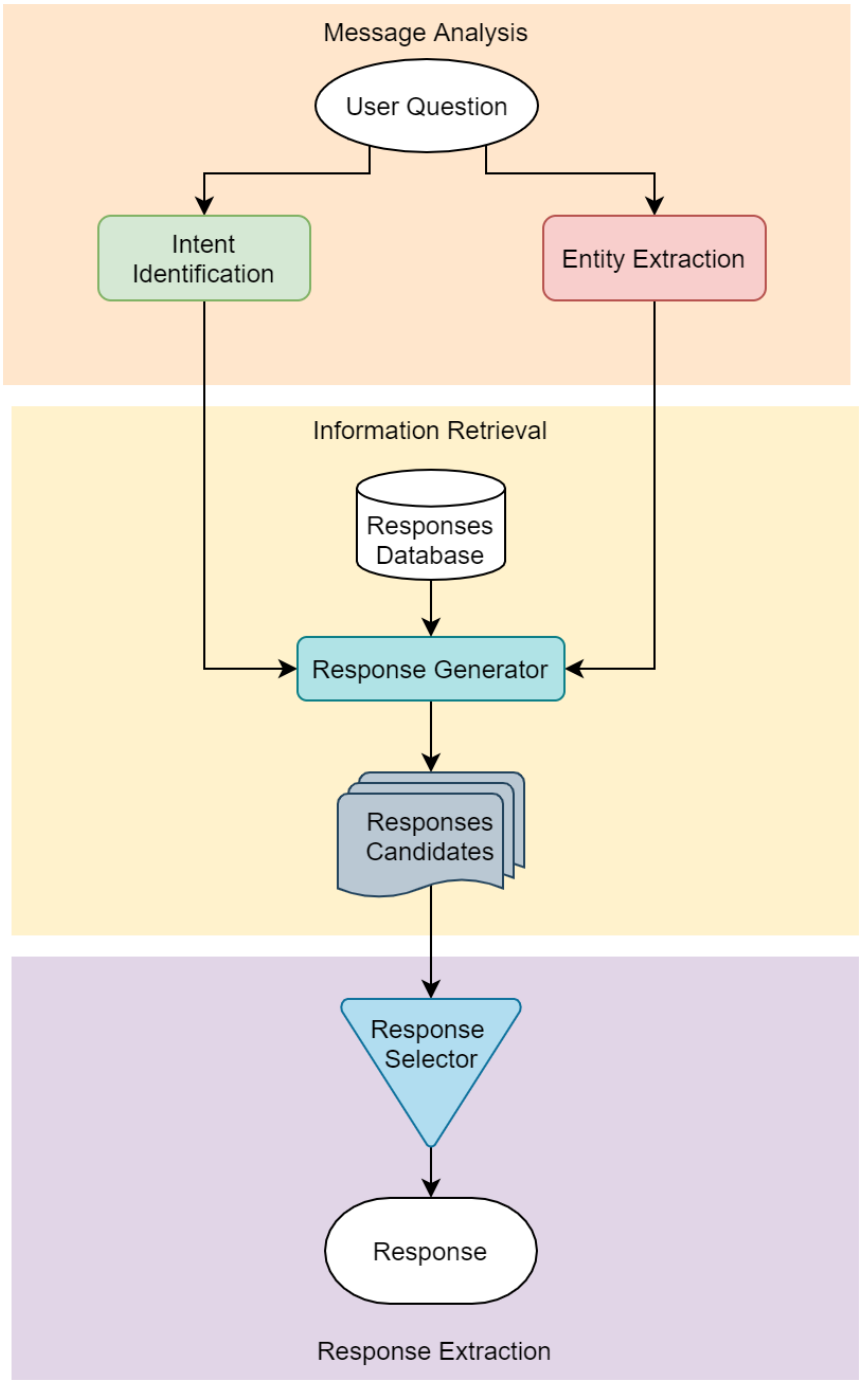


Figure 15: QA chatbot architecture proposal

An interpersonal chatbot can be assessed using F-score metric in different components, for example, using Figure 15 as a root, a performance evaluation can be conducted in

the message analysis, information retrieval or response extraction modules. However, in order to evaluate if the system is feasible, it is crucial for the real application in the healthcare context to do a macro user testing with humans and retrieve their feedback.

3.8 Limitations

The conducted SLR clearly examines the current state of research on the topic of chatbots not only in healthcare but also in a general context. However, the following limitations must be considered.

The obligation of excluding articles through the Quality Assessment step can lead to removing exclusive and out-the-box papers that can be important to evaluate new and innovative approaches. Moreover, the necessity to adapt the Search String to the numerous literature sources. This can lead to differentiations on the paper's characteristics, leading a heterogeneous literature review.

About the domain, this SLR only focuses on the application of healthcare in the academic field. More useful ideas for this topic can be provided by considering practical concepts and applications or other non-educational settings.

3.9 Conclusion

This research presents a Systematic Literature Review on the problem of developing a text-based chatbot, specifically in medical context. This SLR, provides chatbot development community a clear overview of the research and appliance on the health domain.

Considering the extracted findings and research gaps, the authors recommend the chatbot development and eHealth communities to explore opportunities for future work researching the effect of design features regarding the chatbot actions in relation to

the characteristics of the user. Given the lack of a considerable number of relevant approaches analysing the presented research problem, the authors suggest repeating studies focusing on the effects of design features in a general or other context for the eHealth context.

4 Architecture and Solution Design

This chapter presents a high-level solution design that integrates the conversation agent into the application. Next, it explains the decisions regarding the architecture of the chatbot and its development approach based on previous research.

4.1 System Overview

As previously mentioned, this project has two parts which require planning and decision making in its development and implementation:

- Build a health question-answering conversation agent;
- Develop an application to make available the main FAQ and to enable interaction with the chatbot.

Figure 16 gives a high-level architecture overview of this project solution to retrieve answers of an input question on a specific health domain in the shortest possible time.

The user question is received by the web application that requests an answer from the server. In the backend, the chatbot core is running and processes the query in the Data Storage, which has the QA document indexed. Then, it is returned to the server, which sends the extraction response to the web application, displaying it to the user.

Regarding medical knowledge, a Portuguese hospital provided a document with questions and answers on various procedures, surgeries and pathologies related to orthopaedic and traumatology specialities. However, this material has to be analysed and processed in order to be more machine-processable and machine-understandable.

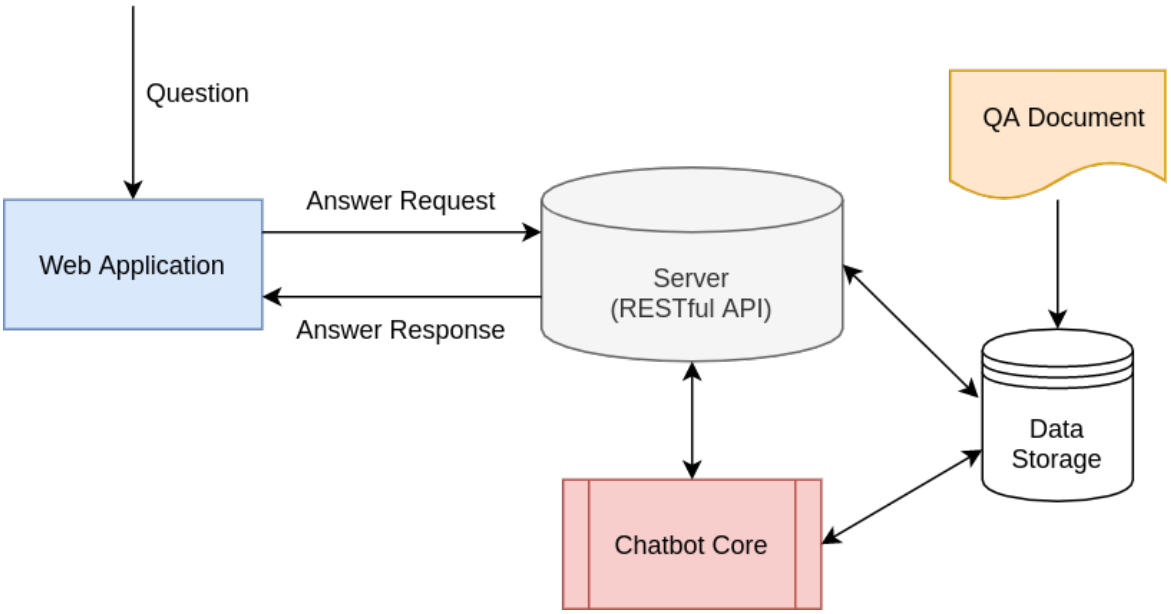


Figure 16: Solution Design Overview

The task of developing the medical FAQ platform and integrating with the chatting agent was decided to create it for the Web, taking advantage of its flexibility and accessibility to all types of users. The development of this platform is built on a modern JavaScript framework, which it is discussed in the next chapter. The authors designed and validated the User Interface (UI) and User Experience (UX) of the web application to meet the particular concern for a user-friendly and straightforward interaction.

One of the most disputing challenge to build this chatbot is dealing with the Portuguese language and specific medical areas with unique keywords. From this perspective, and since this is a research work, instead of using the chatbot development services to abstract all the requirements, this system adopts a low-level approach. By doing so, it provides complete understanding and control over the algorithms built to practice, tune and optimise and be installed on its own server by keeping all the components in-house. The elements and architecture belonging to this system are explained in the following section.

4.2 Chatbot Components

To fulfil the stipulations described in Chapter 1, since the solution is a health FAQ chatbot, it should be a closed domain system that will narrow in scope and focus on a specific context and should deal with a semi-structured and small set of medical data.

From the design approaches discussed in previous chapters, the system can be knowledge-based, information retrieval-based and generative-based. The system proposal exploits a hybrid design that harvests responses from both the paradigms, retrieval and generative. In short, the system retrieves QA pairs from a wide range of sets as well as extracting answers from all the information available.

Regarding the architecture, the Systematic Literature Review (Chapter 3) proposes a general QA chatbot architecture (Figure 17) based on three main parts: Message Analysis, Information Retrieval and Response Extraction.

Adapting this scheme and connecting with the approach described above, the intended architecture for this system consists of two main components: Retriever and Reader.

Essentially, this Retriever-Reader design has the following workflow:

1. The user's Question is queried in the Retriever;
2. Retriever selects the top K documents (QA pairs) most relevant to the Question;
3. Each QA pairs is provided to the Reader;
4. Reader returns the highest-rated units of text that answer the Question.

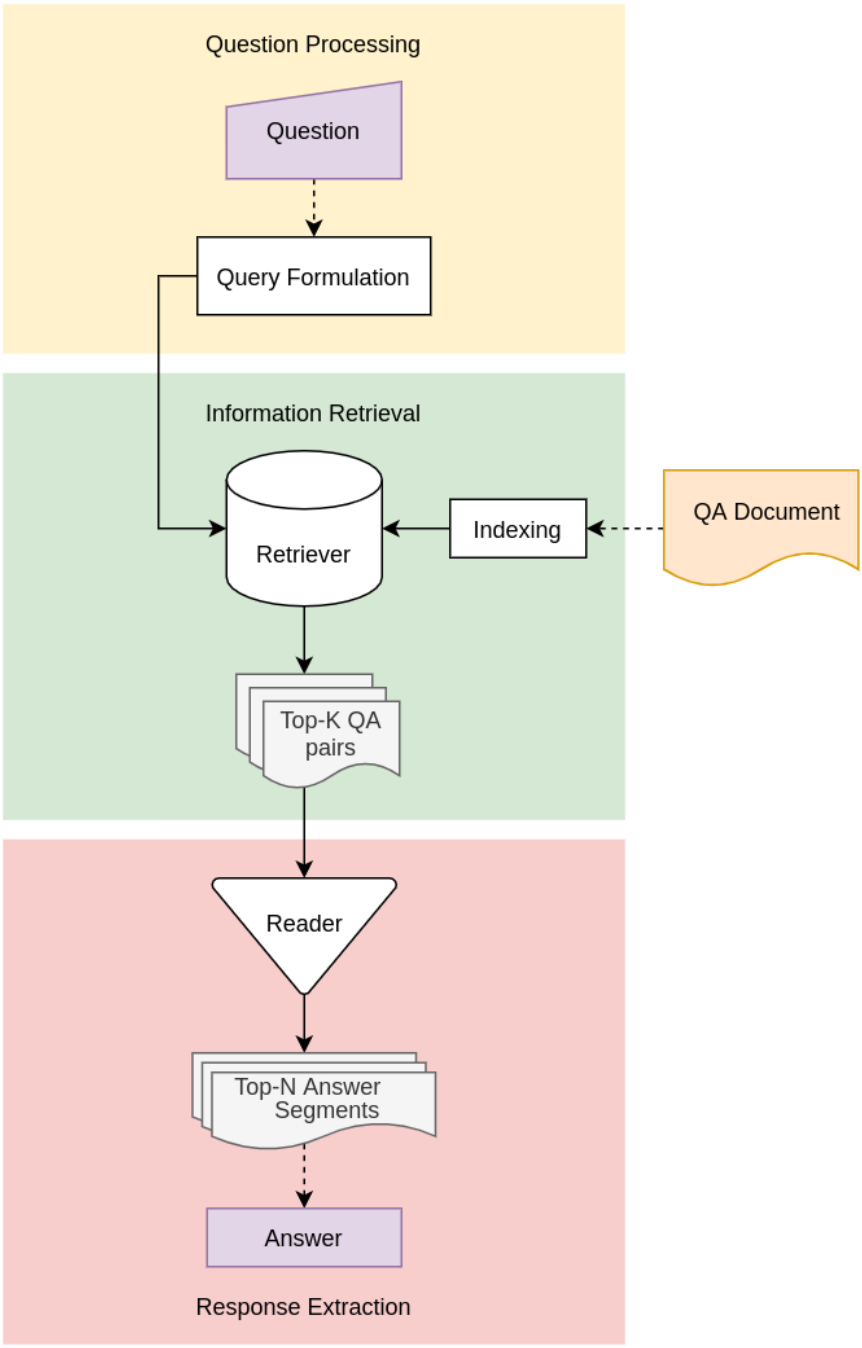


Figure 17: Chatbot Core Architecture

4.2.1 Information Retrieval - Retriever

The Retriever component has two main tasks: it is responsible for processing the user query and for identifying the most similar and relevant QA pairs that can answer the input question.

Accordingly, the Retriever system handles the contextual and semantic understanding of the input to match and rank possible responses by using similarity metrics, to be explored in the following chapters.

Since both input and database are always processed, this system is responsible for storing the whole corpus, making the query search easy and fast.

The most straightforward implementation would return the top-K answer-containing context from which the Reader can extract responses. Can turn more sophisticated by breaking more the existing answers and filter them, for example, by a health issue. Logically, the performance of the Retriever affects heavily the Reader, which in turn influences the final response.

4.2.2 Response Extraction - Reader

From the retrieval pool of top-K FAQ candidates, the Reader takes multiple passages of answers as input and returns top-N answer segments with corresponding confidence scores. In essence, the purpose of the Reader is to apply textual comprehension algorithms to the FAQ corpus for answer extraction.

Similar to the design approaches, those reading comprehension algorithms follow in two categories: feature-based and neural-based.

Feature-based response extraction can involve rule-based schemes, regex pattern matching, or a suite of NLP models, such as part-of-speech (POS) tagging and NER, designed

to identify properties that the ability to comprehend the information and produce a correct answer properly.

Under the neural understanding, current models extract robust responses by selectively learning semantic embeddings between the question and answers, rather than relying on keywords and patterns. Requires language understanding to deal with semantics to fully interpret the corpus and produce an accurate response using similarity functions.

A deep networks approach, relying on prior research, learns more efficiently the features present in the corpus and has better performance when compared to less complex machine learning models. Neural network models that perform well in this area are Seq2Seq models and Transformers (Sutskever et al., 2014a; Vaswani et al., 2017). These models are going to be used and explained later on in the Implementation chapter.

Implementation 5

This chapter discusses the implementation of the system. For clarity, it follows the natural order of the chatbot architecture from the moment the QA document and the question are entered, until the answer reaches the user via a web-based UI. At the end of this chapter, it is presented a summary of the modular implementation that has been developed, with final comments on each component, together with a diagram overview of the implementation.

5.1 Document and Database

This section represents the processing given to the QA document, which represents a stand-alone component in the chatbot architecture, as can be seen in the extraction part in Figure 18. This process happened before being indexed by Elasticseach, explained in the next section.

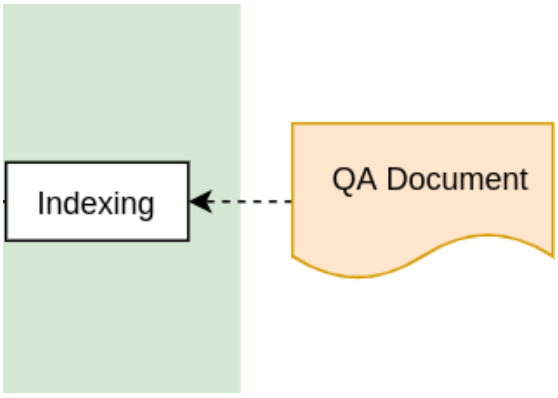
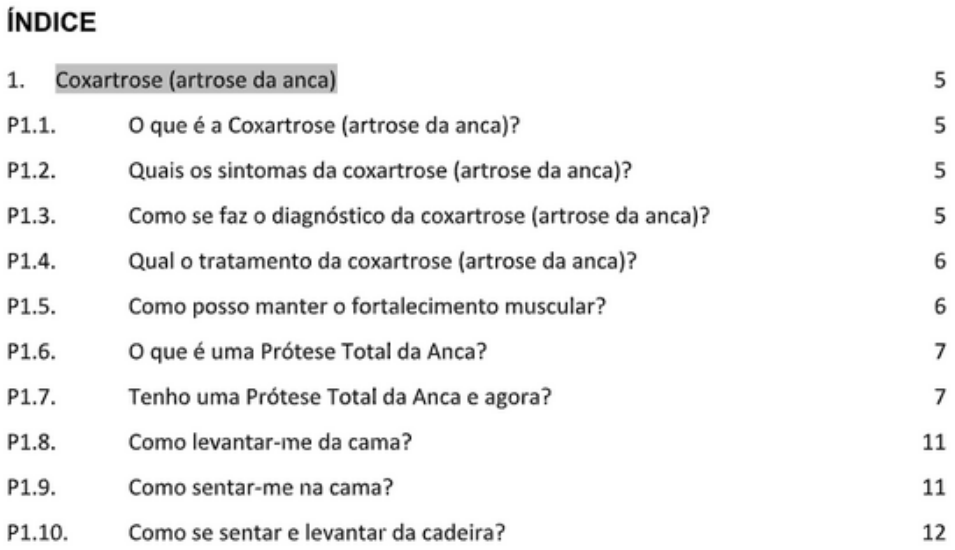


Figure 18: Extraction of the QA document in the chatbot architecture

Health professionals provided documentation to serve this project. The given medical information concerns pathologies and surgeries in orthopaedic and traumatological scope. The document (Figure 19) received is 45 pages long and is organised by health issue which contains generalised questions and lengthy answers to them.

It compiles 83 questions with their respective answers, and if broken down by the medical pathology are: 29 (hip arthrosis); 24 (knee arthrosis); 8 (femoral fracture); 6 (external fixators); 4 (plaster casts) and 12 (spine fracture). The format of the document is in the .docx extension, which is the filename of the Microsoft Word documents.



ÍNDICE

Figure 19: Extract of the Document Index

It is therefore evident the need to process and tweak it in such a way as to achieve the two objectives in this work: Build a static Frequently Asked Questions page segregated by health illnesses. Feed the Retriever and, as a result, provide knowledge to the chatbot.

The document presents sections of question-answer pairs. An extract of those sections can be observed in Figure 20.

By processing it, this original document can be transposed into a JSON-like file. Thus, it is easily read through dictionaries with Python, the choice for its processing.

P1.2. Quais os sintomas da coxartrose (artrose da anca)?

O sintoma que surge em primeiro lugar é a dor que se localiza normalmente na região inguinal (virilha), mas também pode localizar-se na coxa, nádega, região lateral da anca e até mesmo no joelho.

Com o desenvolvimento da doença pode surgir rigidez articular (diminuição da amplitude articular) que se manifesta pela dificuldade no agachamento, no calçar os sapatos ou meias e na alteração da marcha.

P1.3. Como se faz o diagnóstico da coxartrose (artrose da anca)?

É possível fazer o diagnóstico através de exame médico e de uma radiografia.

Figure 20: Document extraction with two QA sections

The dictionary transformation is record-oriented (`[{column -> value}, ... , {column -> value}]`), presenting an array of dictionaries with question and answer pairs, as can be seen from the code extract presented in the listing 1. Also, a "context" key is added to the QA dictionary to connect the referred information to the proper medical pathology, e.g., *coxartrose* (hip arthritis). With this change, the format becomes machine-understandable, serving both the web application and the chatbot knowledge base.

```
[
    ...
    {
      "question": "Como se faz o diagnóstico da coxartrose (artrose da anca
         )?",
      "answer": "É possível fazer o diagnóstico através de exame médico e
         de uma radiografia.",
      "context": "coxartrose"
    },
```

```
8     {
9       "question": "O que é uma Prótese Total da Anca?",
10      "answer": "A prótese total da anca é constituída por duas partes: uma
            que encaixa na cavidade da bacia e a que vai substituir a
           extremidade do fémur.",
11      "context": "coxartrose"
12    },
13    ...
14 ]
```

Listing 1: QA JSON File Sample

The next decision is the database storing, determining where and how to store this document file. Light-weight options are a possibility for fast prototyping, for example, a Structured Query Language (SQL) or In-Memory storage.

However, since the main purpose is to look up for information, it makes sense to use the Retriever as a database system, making easier and more efficient to query it. The choice will therefore be to use the Elasticsearch that it is presented and explained in the next section.

5.2 Elasticsearch

Locating in the chatbot architecture outlined in chapter Chapter 4, this section corresponds to the Message Analysis and Information Retrieval parts, represented in Figure 21.

Nowadays, the task of information retrieval ranges from traditional sparse vector space models, such as Elasticsearch, to dense representations of embedded and encoded questions and text. Despite the current research effort in this field, the sparse vector

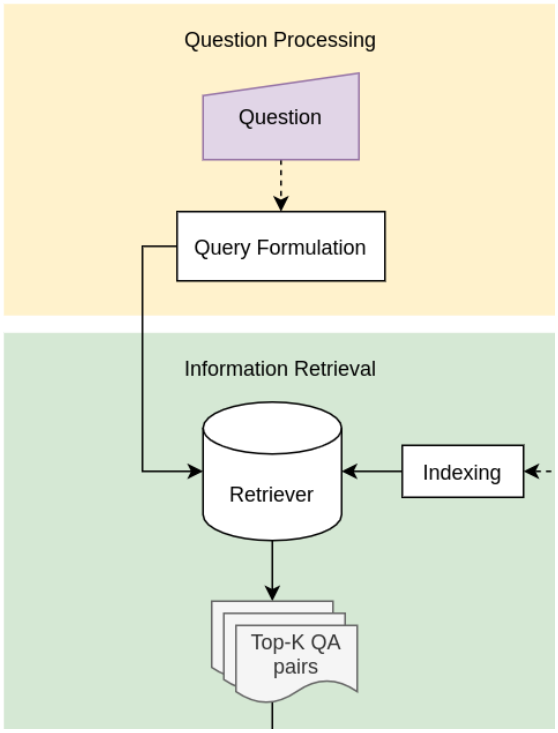


Figure 21: Message Analysis and Information Retrieval in the QA chatbot architecture

algorithms are performing significantly well, and only recently, these embedding-base systems are shortly outperforming, yet little to nothing for non-English languages.

On this ground, in the context of the present work, Elasticsearch (Coviaux, 2019) will serve as an easy-to-use framework for document retrieval, using the features covering full-text matching search.

Elasticsearch is a versatile open-source search engine based on the Apache Lucene library that can manage all forms of data-including textual, numerical, geospatial, organised and unstructured information.

Being built on top of Lucene, it reduces the latency significantly from when a document is indexed to become available to search. It is designed to scale with a robust set of features, a rich ecosystem and a diverse client library list, supporting several program-

ming languages such as Python. Others advantages are being distributed, simplifies data ingest, visualisation and reporting.

Elasticsearch calls an Index the collection of Documents that have common characteristics related to each other. Those documents are stored as JSON data, meaning that each document correlates a collection of keys with their corresponding values.

The data structure used is called an inverted index (Andersen, 2018b), which lists every unique word and identifies where each word appears in all the documents, allowing a fast full-text search. The Table 7 presents an example of a high-level inverted index representation.

Table 7: Example of an Inverted Index representation. Source: Andersen (2018b)

Term	**Document #1**	**Document #2**
best	×	
carbonara		×
delicious		×
pasta	×	×
pesto	×	
recipe	×	×
the	×	
with	×	

During the indexing process, meaning, when a document is added into an index, the document's text fields undergo through an analysis process, as shown in Figure 22. After that, it is stored and builds an inverted index to gain a near real-time data search. By full-text fields, it is referring to the fields with the type text, and not keyword fields, which are not analysed.

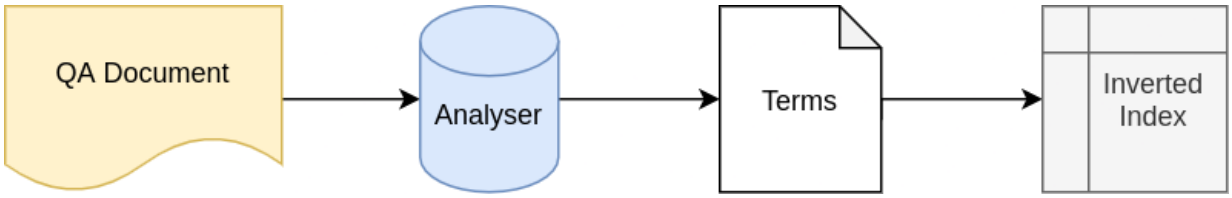


Figure 22: Elasticsearch pipeline

A customisable pipeline entitled as Analyser conducts the analysis method (Andersen, 2018a). An Analyser is composed of three main sequential parts: character filters, a tokeniser, and token filters (Figure 23).

First, a character filter can be optional, and it receives the input text and changes it by adding, removing or replacing characters. It can be used, for example, to correct misspelt word or strip down Hypertext Markup Language (HTML) tags form the original text.

The character-filtered output suffers a tokeniser transformation which consists of splitting up the string into individual tokens (or terms). The standard tokeniser uses grammar-based techniques as tokenising on whitespaces and punctuation, which can be extended to the Portuguese language.

Finally, the array of terms outputted are passed to the token filtering which operates on the tokens to add, remove or modify them. The usual token filters are lowercasing terms and removing stopwords.

The results from the analysis are actually what is stored in the index where a document is added. In other words, when running a search query against an index, it is merely looking for a post-processed representation that is stored in the inverted index, not the raw input document itself.

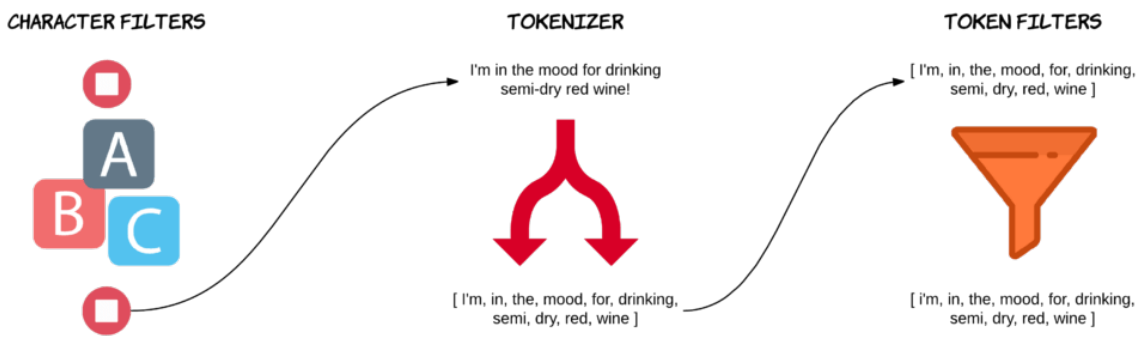


Figure 23: Elasticsearch Analyser flow. Source: Andersen (2018a)

These components that transform the input stream can be configurable granting full control over the analysis process. Although the tokeniser and lowercase filter work in a multilingual context, with the specific case of creating an analyser for the Portuguese language, it is necessary to adjust the token filters for Portuguese stopwords (e.g., "a", "com", "de", "em", "para", et al.). Also, the Snowball stemming algorithm was added to mould derived words to their root form.

The mapping, i.e., the schema for the documents portray the structure of the FAQ document which had been processed into QA pairs (Listing 2). The keys "question" and "answer" are text fields which allows to apply the above-described analyser and be queried on. The property "context" represents a keyword, useful for query filtering to retrieve QAs about a specific health issue.

```
    index_config = {
        "settings": {
            "analysis": {
                "filter": {
                    "stemmer": {
                      "type": "snowball",
                      "language": "Portuguese"
                    },
                    "stop": {
                      "type":       "stop",
```

```
                    "stopwords":  "_portuguese_"
                },
            },
            "analyzer": {
                "pt_analyzer": {
                    "type": "custom",
                    "tokenizer": "standard",
                    "filter":[
                        "lowercase",
                        "stop",
                        "stemmer"
                    ]
                }
            }
        },
        "mappings": {
            "dynamic": "strict",
            "properties": {
                "question": {"type": "text", "analyzer": "pt_analyzer"},
                "answer": {"type": "text", "analyzer": "pt_analyzer"},
                "context": {"type": "keyword"}
            }
        }
    }

```

Listing 2: Index and Analyser configuration to store the document

When a query (q), as shown in Listing 3, is performed on the questions (Q) of the documents (QA pairs) stored will compute a q-Q similarity score based on how well the documents match the query.

```
    query = \
        {'query': {'bool': {'should': {'match': {'question': question}},
         'filter': {'term': {'context': context}}}}}

```

Listing 3: Seach query construction for q-Q similarity

Earlier versions of Elasticsearch use the similarity metric TF-IDF (Luhn, 1957; Jones, 1972) to retrieve the score of a request. The essence behind TF-IDF relies on two main factors to determine if a document is similar to the query:

- **Term Frequency (TF)**: how many times the term is in a given document; it gives a higher score if a term is more present - Eq. (1).

$$\text{tf} = \sqrt{\text{Number of times the term is in a given document}} \tag{1}$$

- **Inverse Document Frequency (IDF)**: measures how many documents which the term appears; better score if the word is less present in the entire corpus - Eq. (2).

$$\text{idf} = 1 + \log\left(\frac{\text{Total number of documents}}{\text{Number of documents that contain the term} + 1}\right) \tag{2}$$

As a consequence, the most meaningful words, i.e., those that are rated higher, are typically frequent in one document but are rare in the whole index.

More recently, Elasticsearch uses BM25 (Robertson and Zaragoza, 2009) in newer models. The ranking function BM25 improves TF-IDF by estimating the relevance of documents to a given search query. It is based on the probabilistic retrieval model which states that the relevance score represents the probability a user will consider the result to be relevant (Turnbull, 2015).

Despite both methods use the Term Frequency, the main difference is that there is no maximum for TF-IDF. In BM25, the impact of TF increases; however, logarithmically approaches a value - Eq. (3).

$$\text{BM25} = \sum \text{idf} * \frac{\text{tf} * (1 + k)}{\text{tf} + k * \left(1 - b + b * \frac{\text{document length}}{\text{average document length}}\right)} \tag{3}$$

In Figure 24, it is clear that BM25's term frequency grows quickly at first but then has a maximum that never gets reached, in contrast, TF-IDF overgrows without limitation, as long as a term is more and more present, the score will be higher. For example, removing stopwords would be an important task not to influence the TF-IDF score. As a result of this improvement, their frequency does not have a significant impact on the BM25 score.

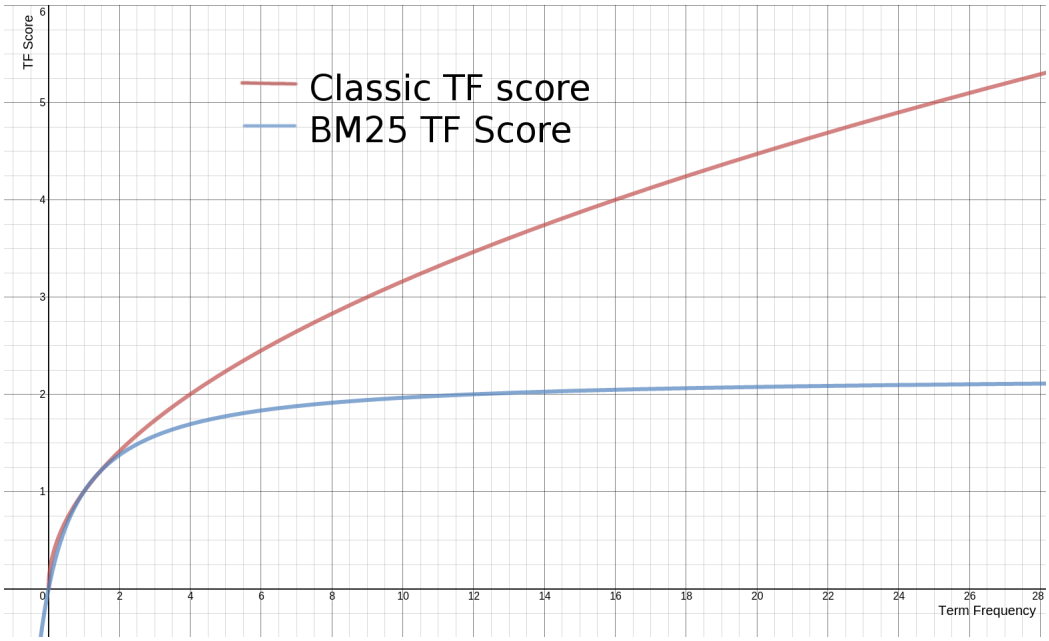


Figure 24: Term Frequency comparation between TF-IDF and BM25. Source: Turnbull (2015)

5.3 Transformer

As stated in the Architecture and Solution Design chapter, the Reader employs a deep network approach, and it is the main component in Response Extraction section, as shown in Figure 25.

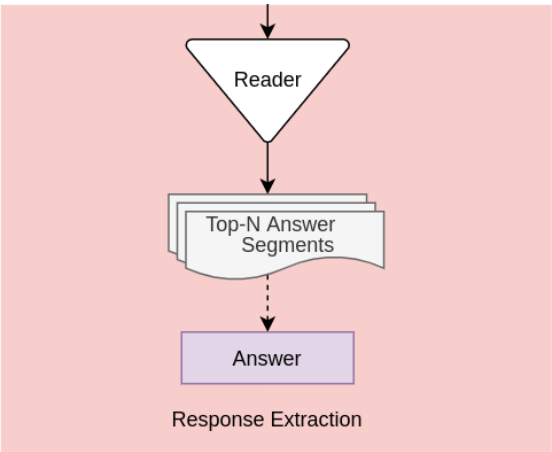


Figure 25: Response Extraction in the chatbot architecture

The system stands on the intuition that the question and its answer are semantically similar. An introductory context to the existing paradigms is first presented before explaining the model used in the Reader component, complementing the findings of previous research.

Sequence-to-sequence (seq2seq) models, proposed in two groundbreaking papers (Cho et al., 2014; Sutskever et al., 2014a), are deep learning models that have gained widespread success in this kind of tasks, such as machine translation, text summary, and image captioning. Even Google in 2016 began using this kind of model in production.

A seq2seq model essentially takes a sequence of words, processes each other sequentially and produces another series of words. The model, under the hood, consists of an encoder and decoder, as shown in Figure 26. The encoder processes each word of the input phrase and compiles it into a vector (called a context). The decoder receives the context

from the encoder and produces the output phrase word by word. Both the encoder and decoder usually are RNN, and the words are transformed into an array of numbers (context) by a word embedding algorithm which captures semantic information of them.

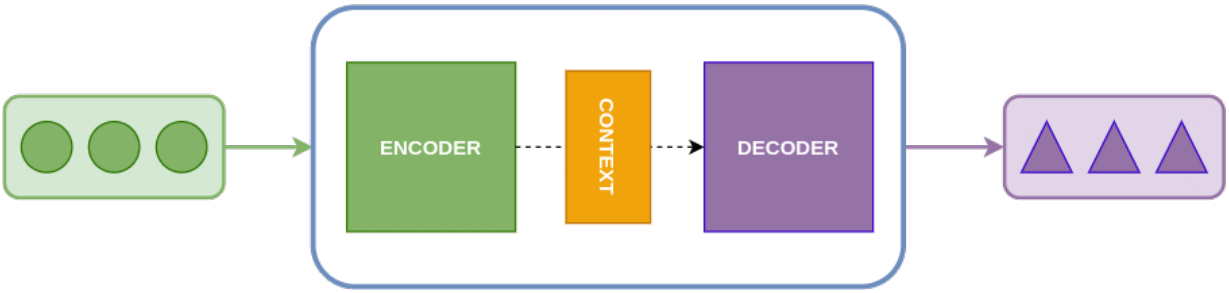


Figure 26: General Sequence to Sequence model

For instance, a Bidirectional Long Short-Term Memory (BiLSTM) (Chen et al., 2017a) approach, a popular seq2seq model, can be used for a question and answer system. Given a question and a set of answers, BiLSTM builds embedding representations for both the question and answers independently, and then utilise cosine similarity to measure where the most likely the correct answer span starts and ends (Chen et al., 2017b).

The Transformer model was introduced in 2017 in the cutting-edge paper "Attention Is All You Need" (Vaswani et al., 2017). Similar to the above approach, Transformers are an encoder-decoder architecture designed to deal with sequential data such as natural language phrases. However, it significantly improves the state-of-the-art without using any recurrent or convolutional layers, only attention mechanisms, as the paper title indicates.

Attention-mechanism involves answering the question "what part of the input phrase should it focus on?". Each word has an output attention vector that shows how relevant that word is compared to the other words in the same sentence. In other terms, it captures the contextual relationship between words in a phrase and gives the decoder context-sensitive information between them.

Therefore, Transformers do not require the processing of sequential words one at a time. Due to this functionality, this architecture is much faster to train and easier to parallelise, a fraction of time and cost to train compared to previous state-of-the-art models. The Transformer architecture is represented in Figure 27.

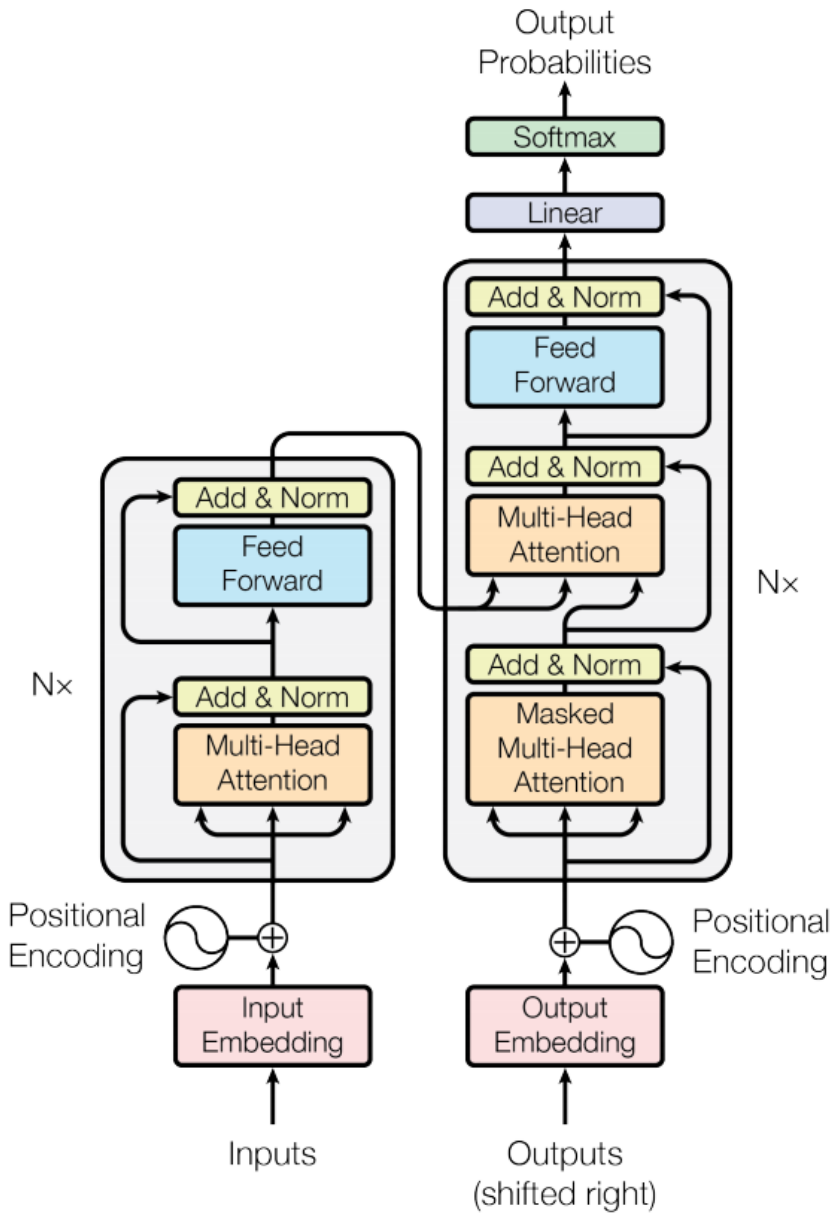


Figure 27: Transformer model architecture. Source: Vaswani et al. (2017)

The core of the natural language models in this architecture is the Multi-Head Attention layer, which helps identify how each word are most aligned with one another and improves the representation of every word using contextual traces. The output from the last layer is inserted into the Linear Layer and Softmax (Bridle, 1990) function, which normalises and distributes probabilities for each response span — all probabilities should add up to one.

Notably, the Transformer system is currently revolutionising the entire field of NLP. These architectures have begun to outperform people on some key NLP benchmarks (Wang et al., 2018), including answering questions. However, not all Transformer models ought to perform the task of question answering.

BERT, proposed by Devlin et al. (2018) (Google team), is one of the most popular Transformer models, displaying significant improvements compared to previous state-of-the-art results on a wide variety of NLP tasks such as next-sentence prediction, question answering, reading comprehension, sentiment analysis and paraphrasing.

BERT aims to provide pre-trained models for extracting contextualised word embeddings. Consequently, these models can be fine-tuned for NLP tasks without interference with the architecture; in the case of the current research, the purpose is for question answering.

In brief, the model is divided into two key sections:

1. a pre-trained transformer serving as a language model;
2. a component fine-tuned for the task at hand, which is shown in the following Figure 28.

Since BERT is an unsupervised model, it is essential to have an enormous amount of question answering data for this task-specific tuning. For example, the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) consists of 100,000 natural

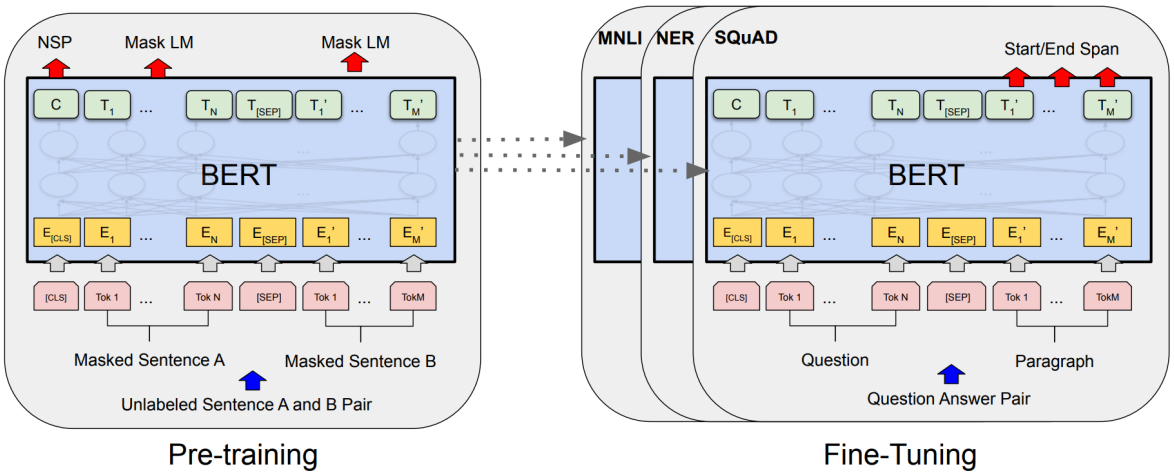


Figure 28: Overall pre-training and QA fine-tuning procedures for BERT. Source: Devlin et al. (2018)

question-answer pairs on 500 Wikipedia articles. Therefore, the neural answer extractor pre-trained is tuned end-to-end by using the previous version SQuAD 1.1 dataset to concentrate on the base task of span prediction.

However, there is a language barrier; the SQuAD data collection is only available in English, and the aim here is on the Portuguese language.

The alternative is to use a Portuguese translation of the SQuAD dataset. The conversion was carried out automatically using the Google Cloud API. The output model was retrieved from Romero (2020).

Having considered that teaching from scratch in Portuguese for this type of architecture is extremely costly. Conversely, fine-tuning is indeed very inexpensive. In conclusion, the system adopted is a BERT multilingual pre-trained model, fine-tuned in the QA dataset, the Portuguese translation of SQuAD.

The first approach was using an existing QA library, the Deepset's haystack (Pietsch et al., 2020). Deepset is a German NLP startup that maintains open-source software and libraries for question answering at scale.

Haystack is crafted in a modular way and employs any models trained with Transformers. With this, it served as a customisable high-level approach, developer-friendly way to mould the system with an **Elasticsearch Retriever** and a **BERT/Transformer Reader**, as represented in Listing 4.

```
from haystack.database.elasticsearch import ElasticsearchDocumentStore
from haystack.retriever.sparse import ElasticsearchRetriever
from haystack.reader.transformers import TransformersReader
...
document_store = ElasticsearchDocumentStore(index)

retriever = ElasticsearchRetriever(document_store)
reader = TransformersReader(model)

finder = Finder(reader, retriever)
prediction = finder.get_answers(question, top_k_reader=top_N)
```

Listing 4: Implementation code using Deepset's haystack library

The approach above referred was not flexible and versatile enough to meet the needs of the designed solution. It did not provide enough adaptability in indexing and querying Elasticsearch, dealing with Transformer architecture and, mostly, using Portuguese language. Moreover, since the present work is carrying out academical research, the authors follow a lower-level QA framework, which leads to HuggingFace's Transformers (Wolf et al., 2019).

HuggingFace (HF) with this library provides features to customise a model easily and build an architecture suiting the needs for this project. It exposes the models internally

as consistently as possible and can be used independently for quick operations. HF's Transformers is backed by the two most common deep learning libraries, PyTorch and TensorFlow, with smooth integration between them.

The library API can be applied for preprocessing data, training and fine-tuning, model sharing, and support for multilingual models. Besides, it supports multiple use-cases with different kinds of models and offers excellent documentation. In this case, the solution can be implemented with the invariably referred architecture employing an autoencoding model, BERT, on a question-answering task.

The `pipeline()` wrapper has the leverage to efficiently use the available pre-trained models, using the following task identifier: "question-answer". Furthermore, it has the power to evaluate the model and configurations using an own SQuAD-format test dataset, explained in the next chapter.

Thanks to the integration with HuggingFace's model hub, it is possible to load custom models, including those that have been fine-tuned for the question-answer task. In other words, the community exchange thousands of pre-trained models, along with the few that perform Portuguese QA - which will be adopted.

In a nutshell, using the primary python library `elasticsearch`, a Retriever is formed with the index configuration and query presented in the previous section. Retriever results go into the prediction function which uses the `QuestionAnsweringPipeline` from HuggingFace's Transformers library. With a judicious heuristic, returns the final answer to the user's question.

Through the connection between these components, a business logic is developed feeding the web application on the background. The following section introduces and describes how this middle ground implementation is constituted.

5.4 Backend / Application Programming Interface

As stated above, there must be a connection between the answer request to a question given in the web application interface and the entire chatbot components model. For that purpose, a middleman service is created which uses Hypertext Transfer Protocol (HTTP) requests to access and use data, i.e., a REST API.

Apart from all the chatbot core code being written in Python, this language is long used for developing lightweight web apps. Moreover, due to its extensive usage in machine learning, it is incredibly convenient to package models and to expose it as a service.

There are several remarkable frameworks, such as Flask, Django, Falcon (Zwerschke, 2020). Although Flask was the most used for this purpose, a relatively new framework came with improvements compared to its predecessors and was quickly adopted and gained significant traction - this is the case of FastAPI, built on top of Starlette, brings useful features to this industry.

This framework is intuitive and straightforward to code, provides excellent documentation and includes security and authentication (Ramírez, 2020). It also produces OpenAPI documentation using the interactive Swagger UI. Thanks to `Pydantic` integration, it implements type declaration and data validation to define the schema. In case of using Docker for deployment, there is an official Docker image which has an auto-tuning mechanism to adjust quickly to written code. The backend deployment is discussed in depth next chapter.

As noted in the Figure 29, the backend project can be viewed in a three-part breakdown: the chatbot core code, the knowledge base and routers. The `core` directory represents the chatbot components, which are glued together through the finder file. The folder `db` has the JSON-like document with questions and answers. In order to HTTP request

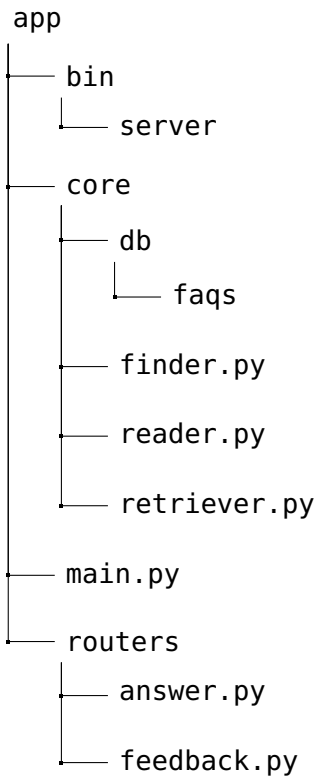


Figure 29: List contents of the backend directories

data, in this case, through GET / POST methods, the API must declare path operations to its router - the files in `routers` represent those API endpoints paths.

When executing the `bin/server`, the platform will initialise by including the routers and creating an instance of Elasticsearch middleware due to the toolkit Starlette. This triggers the loading of the transformer model and the indexing of the FAQ document.

Thereafter, the system is ready to receive and process an HTTP request efficiently. For instance, if there is a request for an answer, a JSON is sent as a body to the relative path `api/answer` with the pathology and the user's question. Since all components have been configured beforehand, this request calls the `query` of the Finder class, as represented in the Listing 5. This function calls the `search` function of the Retriever class that constructs the Elasticsearch query and retrieves top K documents. Afterwards,

these documents are passe to the `predict` function of the Transformer Reader. Finally, through the Reader class, it returns the top N answers from the received documents.

```
class Finder:
    ...
    def query(self, question, context, topK, topN):

        retriever_results = self.retriever.search(
            question_text=question,
            context_kw=context,
            n_results=topK)

        passages = retriever_results['hits']['hits']
        docs = []
        for passage in passages:
            doc = {
                'id': passage['_id'],
                'score': passage['_score'],
                'answer': passage['_source']['answer'],
                'question': passage['_source']['question'],
            }
            docs.append(doc)

        answers = self.reader.predict(question, docs, topN)
        return answers
```

Listing 5: Finder's `query` function that glues the Retriever and Reader

5.5 Web Application

The platform's frontend was developed in Next.js, a React framework, which supports server-side rendering in contrast with the more common Create React App which supports client-side rendering.

This framework seems fit because of the features present, which are simple configuration with automatic compilation and bundling. A hybrid renders with build or request time and has a fast, reliable hot-loading editing experience.

Moreover, the framework has smooth integration with Vercel - the deployment platform created by Next.js team. The next chapter discusses this deployment.

There are two different paths inside the application. Users are presented with a page when accessing the web site. This page is called Landing Page and contains all the primary information about the orthopaedic service to get the user attention, as shown in the Figure 30.

Figure 30: Web Application Landing Page

The second one is the FAQs page, as can be seen below in the Figure 31. For the chosen use-case, we utilised the same QA data that powers the chatbot. It presents real questions and answers to clarify the possible user doubts about six orthopaedic pathologies which are: *Coxartrose* (hip arthrosis); *Gonartrose* (knee arthrosis); *Fratura do fémur* (femoral fracture); *Fixadores Externos* (external fixators); *Aparelhos gessados* (plaster casts); *Fratura de coluna* (spine fracture).

Figure 31: Frequently Asked Questions Page

Plus, for the chatbot integration, there is a custom widget created for this purpose. The widget is fixed on the lower right-hand side of the screen so that it is always easily accessible and visible for the user to chat. The component features direct text message exchange, custom styling and quick buttons. These characteristics are useful to create a rational conversation flow and to choose the orthopaedic pathology correctly, as highlighted below in Figure 32.

All of the editable text is present on the file-based Content Management System (CMS), powered by Forestry.io. This system has rich UI with an editor-friendly allowing developers and editors to work synchrony in a unified workflow and toolset. Forestry.io is Git-backed, meaning that every committed change triggers a new production image which contributes to continuous deployment.

All of the files are in the `cms` directory which accesses them and treats them as editable regular files from any Javascript based app. For now, every piece of content (text,

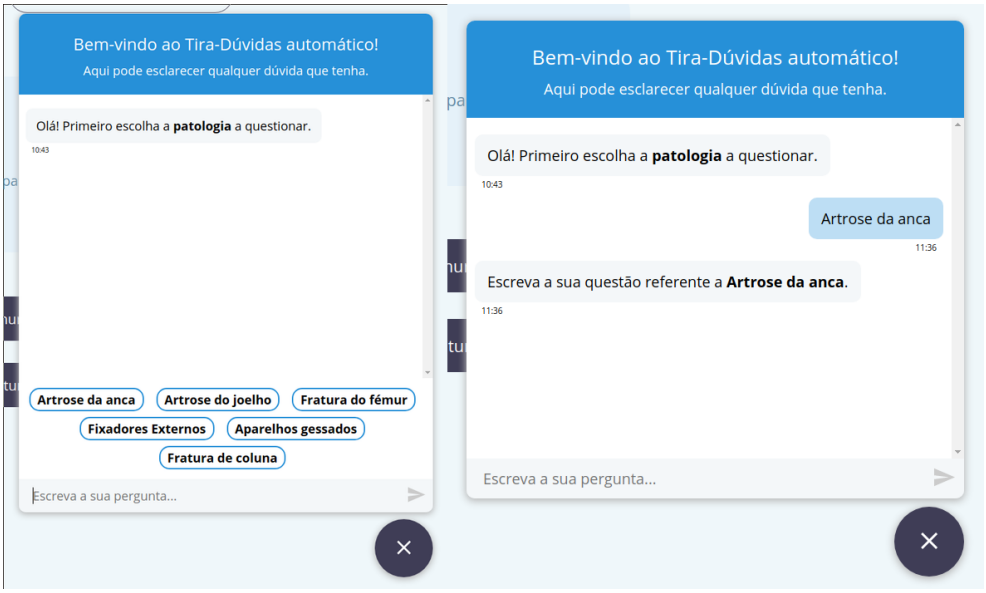


Figure 32: Chatbot Widget

descriptions and question-answers) is there, so non-technical people can edit it as they see fit.

All described implementations are based on the principle that all content and code belong to those who implement it. Furthermore, no conversation is recorded; only the feedback contributed. These eliminate the complications related to data privacy that is currently widely addressed worldwide, thus complying with existing standards such as HIPAA. Therefore nothing is stored in databases in the cloud out of reach, keeping all the components in-house. It is a compliant application that is infeasible when a third party shares medical and sensitive data.

5.6 Feedback Learning

One of the aims in the present work is to introduce a self-learning approach that gradually adapts the model to understand human questions better and to provide more trustworthy answers. This idea can incorporate a HITL methodology, as discussed in

the findings in Chapter 3, which can actively involve humans in training, tuning and/or testing the conversation agent.

With that goal in mind, the users can express their happiness about the task success by having them interact with the dialogue system. In the case at hand, a multiple-choice yes/no is presented to the users to give feedback if the answer given could address their question correctly. This feedback is sent to the backend using `api/feedback` endpoint to fine-tune the model.

As explained in the previous chapter, fine-tuning a model on a SQuAD dataset enables the model to learn the task of question-answering. Then, an additional fine-tuning on a specialised QA allows a knowledge transfer learning, ensuring better performance in that technical domain.

For the purpose of fine-tuning, HuggingFace provides a helpful script called `run_squad.py` that fines-tunes a Transformer model on a SQuAD dataset. The feedback given to the backend by the user is stored, and then it generates an own SQuAD-format dataset. Afterwards, it runs the script to tweak it with that new data collection.

6 Experiment and Results

This chapter describes the deployment of the system developed, together with its testing environment composed of evaluation metrics and a labelled dataset. The scores of the implemented modules are then presented and discussed.

6.1 Setup and Deploy

The repositories created throughout the project regarding the web application, the REST API, and the NLP model applied are available with open access at Github. After having these projects running smoothly in the localhost, it was time to set it up in a production mode inside a cloud environment.

In this project, the frontend and backend require distinct approaches, despite being deployed in the cloud, which will be explained below.

FRONTEND The first approach to deploying the web application was using Netlify, a cloud solution with a developer focus on the frontend. Netlify could instantly build and deploy the static website automatically from Github with free own hosting, including Secure Sockets Layer (SSL).

However, at the time of development, the Next.js was not supported by Netlify, which implies additional configuration and tweaks to work correctly. For this reason, it was changed for Vercel, another cloud platform for static and dynamic websites and web applications.

Vercel is created by the developers of Next.js and has first-class Next.js support and optimization. Besides having the same features as Netlify, it is the easiest way to deploy the Next.js project to production. The web application is deployed in the following link: `aidbot.vercel.app`.

BACKEND Several cloud paradigms can deploy this kind of REST API service, which depends on the use case and the tools employed. Generally, five main cloud service types are used to compare and thereby obtain the most suitable for the required use case.

Table 8 portrays this comparison, from which one can conclude that there are two ways forward. Either take a Platform as a Service (PaaS) approach, which lets the platform take care of the hosting and project infrastructure. Alternatively, go forward with an Infrastructure as a Service (IaaS) provider, allowing full control to configure and administer the system.

Heroku was the first strategy used to deploy the REST API that uses the FastAPI framework. This PaaS platform runs the applications in virtual systems known as dynos that can be configured according to users' needs. Heroku users can add resources through horizontal or vertical scaling.

There were two requirements to be able to push the project to Heroku: a file called `Procfile` that lists the process types in the application, and a `requirements.txt` configuration file that gives an insight into what dependencies and specified Python packages are needed to run this project.

PyTorch's machine learning framework, essential for the work developed, has a conflict with Heroku, which does not run the Graphics processing unit (GPU) versions, solely the Central processing unit (CPU)-only ones. It was fixed by adding those versions' download links as a standalone line in the dependency list.

Next, two vital problems emerged in the attempt to deploy in Heroku. Firstly, Elasticsearch was not running, which could be resolved by attaching a paid add-on or another dyno. Moreover, as explained in the previous chapter, many resources are consumed by running the NLP models, exceeding the existing 512MB Random-access memory (RAM).

Table 8: Types of Cloud delivery models

Service Type	*Use cases*	*Providers Examples*
Software as a Service (SaaS)	• Web-based software, on-demand software and hosted software • Limited or none user-specific application configuration settings	Salesforce, Google Apps, Dropbox, MailChimp
Platform as a Service (PaaS)	• Manage app resources such as hosting, scaling, monitoring, and infrastructure • Connect with database systems, such as PostgreSQL, Redis or MongoDB	Heroku, UpCloud, AWS Elastic Beanstalk, Azure App Service, Google App Engine
Infrastructure as a Service (IaaS)	• It provides a robust computing infrastructure for the configuration, administration, and monitoring of systems. • Full control of the systems and unlimited flexibility	DigitalOcean, Amazon EC2, Google Cloud, Microsoft Azure
Functions as a service (FaaS)	• Webhooks to respond to HTTP triggers • Mobile backend operations	AWS Lambda, Google Cloud Functions, IBM OpenWhisk
Containers as a service (CaaS)	• Deployments and integration into hosting environments of container-basedapplications • Use of container orchestration platforms such as Google Kubernetes, Docker Swarm, Rackspace Carina.	Red Hat OpenShift, Amazon ECS, Docker Enterprise, Google Kubernetes Engine (GKE)

Therefore, the solution was vertical scaling the dyno for more power and horizontal scaling to run Elasticsearch. That would mean a huge increase in the monthly cost, out-of-reach for the authors.

Knowing an IaaS provider might be a more cost-effective solution, there was this tradeoff of simplicity and cost in which it was necessary to decide between which to abdicate. Since the beginning, the present work explores and values flexibility and customizability; the decision was simple. It was to explore the IaaS approach and use DigitalOcean as the cloud provider for this project.

DigitalOcean Droplets are Virtual Machines (VMs) based on Linux that runs on top of virtualised hardware. Each Droplet created is a new Solid-state drive (SSD) cloud server with a dedicated Internet Protocol (IP) and root access via Secure Shell (SSH). The Droplet created for this work has the following specifications: 4 CPUs, 4GB RAM, 80GB SSD disk, Ubuntu 20.04 distribution and an European datacenter.

In the local environment, Elasticsearch was running through a Docker container. So, it was chosen to use containers in every project piece - API and Elasticsearch - in order to achieve a stable production environment. A Docker container is a lightweight, standalone software unit that packages code and all its dependencies, so the application runs from one computing environment to another quickly and reliably.

Docker Compose is a tool used for defining and running this multi-container Docker application. Compose uses a YAML file to configure the application's services. The file content is illustrated in Listing 6. It was essential to making them both running in the same host network, in order to be possible to communicate on the correct ports and defining a to the application API a `restart:always` with a `depends_on:[elasticsearch]` to make sure is built in order, and their communication is established.

```
1 # Version of docker-compose
2 version: "3"
3
4 # Containers we are going to run
5 services:
6
7   elasticsearch:
```

```
    image: elasticsearch:7.6.2
    container_name: elasticsearch
    environment:
      - http.cors.enabled=true
      - http.cors.allow-origin=*
      - discovery.type=single-node
    volumes:
      - /usr/share/elasticsearch/data
    ports:
      - "9200:9200"
    network_mode: "host"

  app:
    image: aidbot-api:lastest
    container_name: aidbot-api
    restart: always
    build: .
    network_mode: "host"
    depends_on: [elasticsearch]
```

Listing 6: `docker-compose.yml` configuration file

After that, accessing the VM with SSH root access, the project was pulled from Github and set it running. However, in order to allow outgoing connections, the firewall needs to be set up. Through the firewall configuration tool `ufw`, the firewall had been enabled and written a rule to allow the service HTTP on port 80. The project backend is live on the following public IP `188.166.67.79`.

6.2 Evaluation Metrics

Evaluation is required to design a conversation agent accurately. However, quantifying the success of a question-answering system can be a perplexing task.

Since the solution architecture follows a modular architecture which has two main components: Retriever and Reader. It is fundamental to assess them individually in order to understand and yield an overall performance of the proposed system. There-

fore, this section looks into the evaluation metrics that can illustrate each component performance.

6.2.1 Retriever

As mention in the previous chapter, the Information Retriever uses the search engine Elasticsearch, which relies on the BM25 algorithm, to pick out the best document results for a given question. Nevertheless, it must be understood if the system is working accurately and what an excellent retriever tool resembles.

In the conventional field of information retrieval, several metrics are used to measure the relevance of queried results, essentially based on accuracy and recall measurements (Lioma et al., 2017). Specifically, in the context of answering questions, the evaluation metrics scientifically accepted and used are: Recall and mAP.

RECALL Conventionally, the metric Recall (often called Sensitivity) in Information Retrieval represents the fraction the relevant documents to the user query which are successfully retrieved - Eq. (4). In this case, instead of counting the relevant passages, it follows a binary classification. In other words, since the knowledge base is a FAQ document, the Retriever's Recall outlines the percentage of questions for which the final response segment appears in one of its top K documents returned by the Elasticsearch query.

$$\text{Recall} = \frac{|\{\text{ relevant documents }\} \cap \{\text{ retrieved documents }\}|}{|\{\text{ relevant documents }\}|} \tag{4}$$

MEAN AVERAGE PRECISION (MAP) Compared to the Recall, this metric differs in the sense that the results retrieved contain a notion of ranking.

Therefore the Mean Average Precision values the Retriever not only the returned documents in which there is the fragment of response, but it also scores, within the retrieved documents, those that contain the answer higher up than those non-answer-containing documents, i.e. ranking them properly.

For a better explanation, let's break down the definition of Average Precision for Information Retrieval. The Average Precision (AP) is defined in the below Eq. (5) - where Retriever returns K documents and m of those contains the correct answer, and rel(n) is just a binary indication of whether the nth passage has the right response or not.

$$AP@K = \frac{1}{m} \sum_{n=1}^{K} \Big(P(n) \text{ if the } n^{th} \text{ item contains the answer } \Big) = \frac{1}{m} \sum_{n=1}^{K} P(n) * \text{rel}(n) \quad (5)$$

Consider the following example: the Retriever returns K=3 documents in which only one contains the correct answer. Then there are three feasible scenarios of how this calculation would proceed, represented in Table 9.

Table 9: Average Precision in three different scenarios

Scenario	**Binary Array**	**Precision@n's**	**Average Precision@K**
A	[1, 0, 0]	[1/1, 0, 0]	(1/1) * [(1/1) + 0 + 0] = 1
B	[0, 1, 0]	[0, 1/2, 0]	(1/1) * [0 + (1/2) + 0] = 0.5
C	[0, 0, 1]	[0, 0, 1/3]	(1/1) * [0 + 0 + (1/3)] = 0.33

Therefore, Scenario A is the most scored, because it has the answer-containing document rank in the first place, relative to the others. While average precision is measured on a per-query result, the mean average accuracy for all queries is just the average of every AP.

6.2.2 Reader

Just as similar to the Retriever, there are several metrics to evaluate the Reader on a Transformer model. As it is a system of questions and answers based on datasets such as SQuAD, two metrics are usually used (Rajpurkar et al., 2016), both ignoring punctuation, spaces, and capital letters: Exact Match and F1 score.

These metrics are computed individually for each question-answer pair. Therefore the results measured for the model are averaged over the individual scores.

EXACT MATCH This metric assesses, as the name implies, whether the predictive answer is exactly the same combination of characters as the true answer. EM is a strict all-or-nothing rule, not having a character results a score of zero. Overall EM means the percentage of predicted answers which match the gold answer exactly.

F1 SCORE F1 score is commonly used metric for this type of problem. It is a equal combination between precision and recall. The precision is the counting ratio of words shared in the number of words in the prediction answer, and recall is the number of shared words to the total of words in the ground truth - Eq. (6).

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

Thus, this metric is based on the number of words shared between the prediction and the actual answer. Accordingly, the overall F1 score can be interpreted as the average overlap between predicted and gold responses.

6.3 Data Collection

The present chapter focuses on the case study of the system developed, which needs to be evaluated for its validation.

As indicated throughout the development of this work, to evaluate appropriately and correctly, it is necessary to create a testing environment with humans and retrieve their feedback.

However, in terms of cost, time and availability, it is infeasible to run user testings to validate the model every time after a new change is made in the system.

While the current system is capable of providing such an evaluation, the previous section presents metrics that guide us to a modular assessment of the application. This is performed only through labelled examples.

With the need to have a proper dataset to test, a questionnaire was created and given to 17 people within an age range between 20 and 60 years old. This form places the participant in several hypothetical scenarios, each one presenting themselves with every pathologies that the system handles. For example, the user is presented with the hypothesis of having arthrosis on his hip, and he needs to wonder what reasonable questions he wants to see answered. This way, it is possible to generate real questions in order to test the system correctly.

After the closing of this survey, 216 questions were collected. The following step was to transform these real questions into somewhat usable for the Retriever and Reader's testing and evaluating applying their metrics. These questions have been processed, filtered and divided into several files, sorted by medical pathology. As it turns out, some of them have more questions than others, but in total, there are 163 questions.

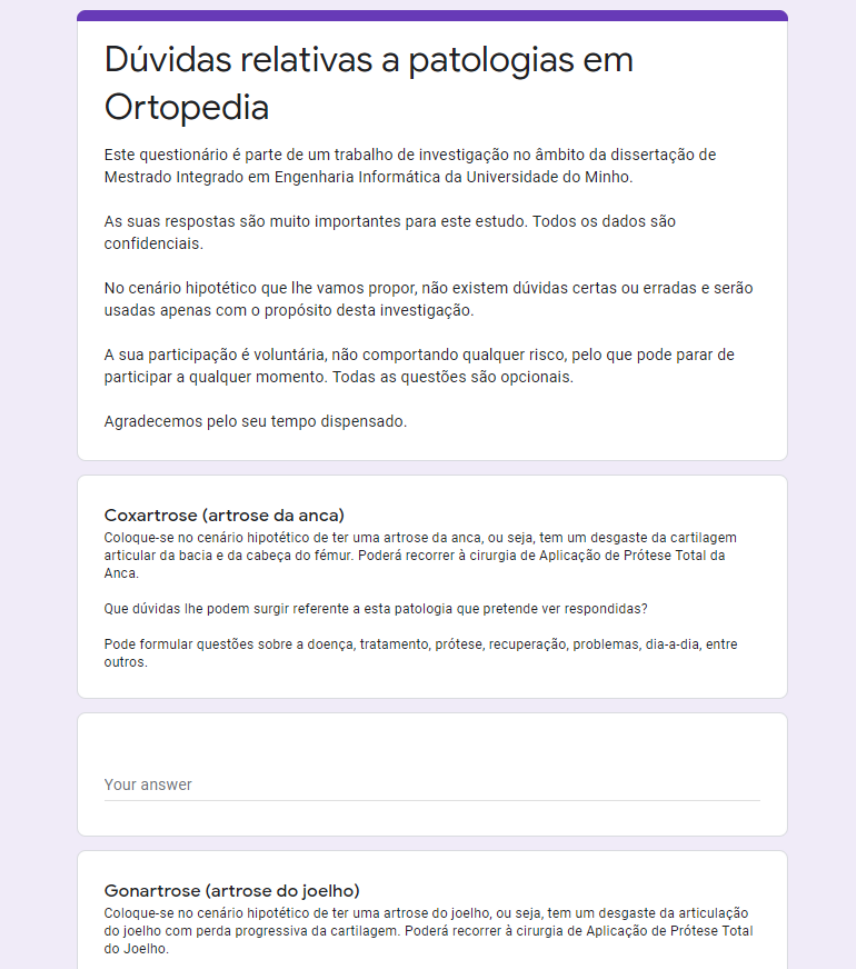


Figure 33: Screenshot of the user questionnaire

The primary purpose has always been to create a versatile generalised dataset which could be used not only for testing but also for training and possibly fine-tuning the application.

Next, it is necessary to annotate and connect the questions received to health professionals' original document that provides knowledge to the system. This process of creating question-answer annotations can be very time-consuming, as it is done manually to link each question to the documents and potential extractions answers.

The authors have carried out this lengthy and exhaustive procedure. Yet, in this way, such investment can be asserted with considerable payoff since it can be evaluated with

genuine questions, and it is flexible and independent on the types of testing that can be applied.

Initially, an intuitive and manual similarity connection was established between users' questions and knowledge base questions. Hence, already it can be stated that there is enough material to evaluate the Retriever, in which, through the aforesaid metrics, it examines if the questions are similar.

Having those several questions correlated, it is time to extract the answer to each one of the 163 questions. This response should be present in the answer document associated with the knowledge base question, already associated with the user's question.

This process can be classified as pretty extensive, and for that purpose, the deepset's Haystack Annotation platform was used to aid the organisation and compilation during this manual annotation process of labelling the answer to the question. Figure 34 illustrates how the platform was involved in this task.

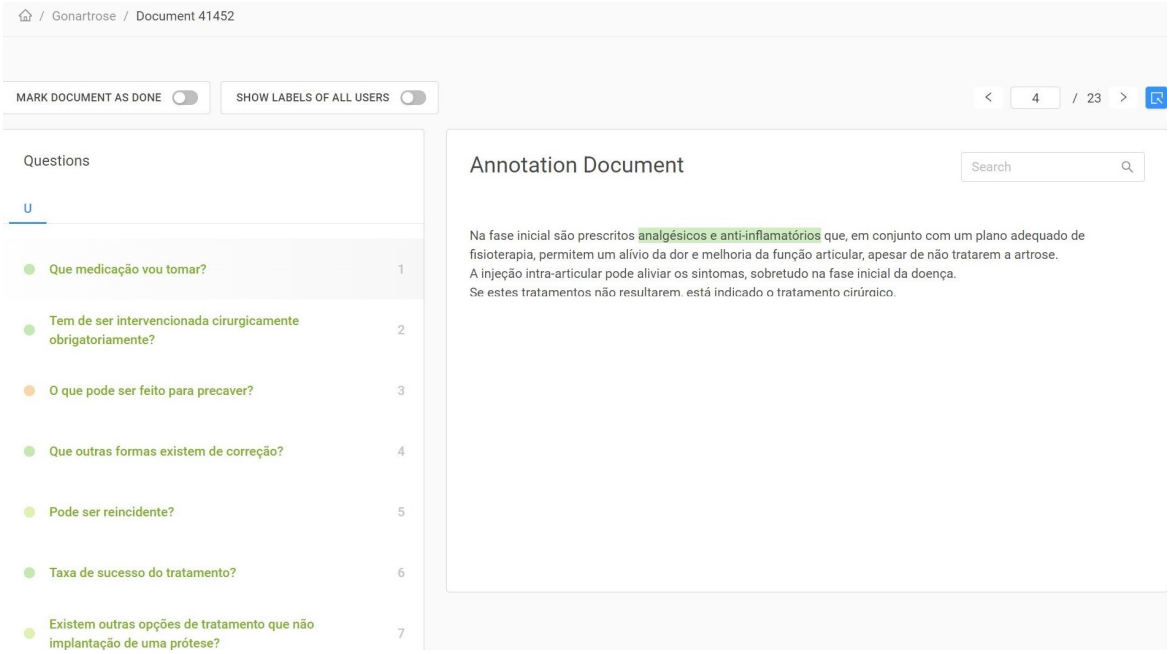


Figure 34: Haystack Annotation screenshot showing users' questions and a labelled answer

As previously mentioned, taking into account Transformer Reader is trained in the SQuAD dataset, its evaluation will also be simplified by supplying examples in the SQuAD-format. Haystack Annotation platform has this logic built-in, allowing to export the labelled dataset in this format.

Thus, one can conclude, for the testing samples, there is now an own SQuAD-style QA pairs created from real human questions within the original document given by health professionals.

6.4 Discussion and Summary

For evaluating the components, it requires two elements: performance metrics and labelled question-answering examples. Once several datasets are prepared for testing and the metrics established, it is time to proceed with the evaluations' results.

The Retriever component has defined a function called `evaluate_retriever`, which iterates through the testing examples that have been created. This function allocates the users' questions over search queries and, for each, it retrieves a predefined number K of documents that have a reference to its base question. This identification is then compared to the one previously assigned to the given question and produces a binary array with the hits. Next, it applies the Recall and mAP metrics mentioned above that evaluate the quality and performance of Elasticsearch.

Figure 35 presents the Recall results in comparison to the number of documents retrieved. As expected, the higher the number of documents, the greater the impact on the evaluation of the Elasticseach Retriever. However, a certain over saturation can be observed when marginally increased.

There is also a discrepancy in the scores between the different pathologies, due mainly to the corpus size; that is, the larger the corpus, the gradual decrease in the score

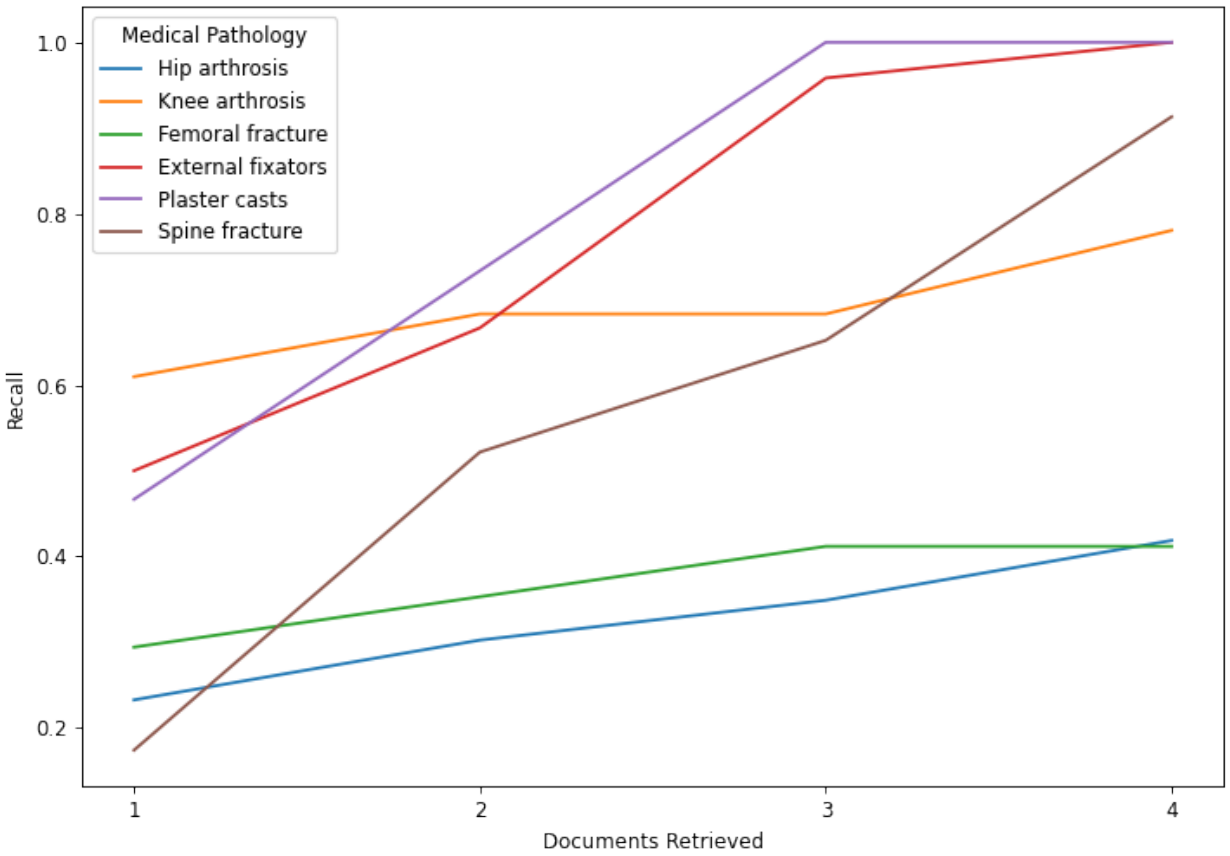


Figure 35: Recall scores over the documents retrieved

is observed. As is the case of hip arthrosis, which is founded on 29 questions and answers, i.e. documents, in contrast to the plaster or external fixators who have 4 and 6 documents respectively. In addition, users' questions are more generally and daily related questions instead of certain exceptional situations as foreseen in the knowledge base. Further documents restricting to particular occasions are not so useful for those human questions, which can deceive the Retriever when compared only with the question text.

The same can be noted in Figure 36 for the mAP metric, when retrieving K=3 passages and separating by the health issue. However, there is a meaningful difference when compared to the Recall, which is these values are lower because the ranking of the relevant documents enters into the calculations, thus making a more real sense of

accuracy. Calculating the average across all pathologies, the mAP value for K=3 documents is 0.5055 (50.55%).

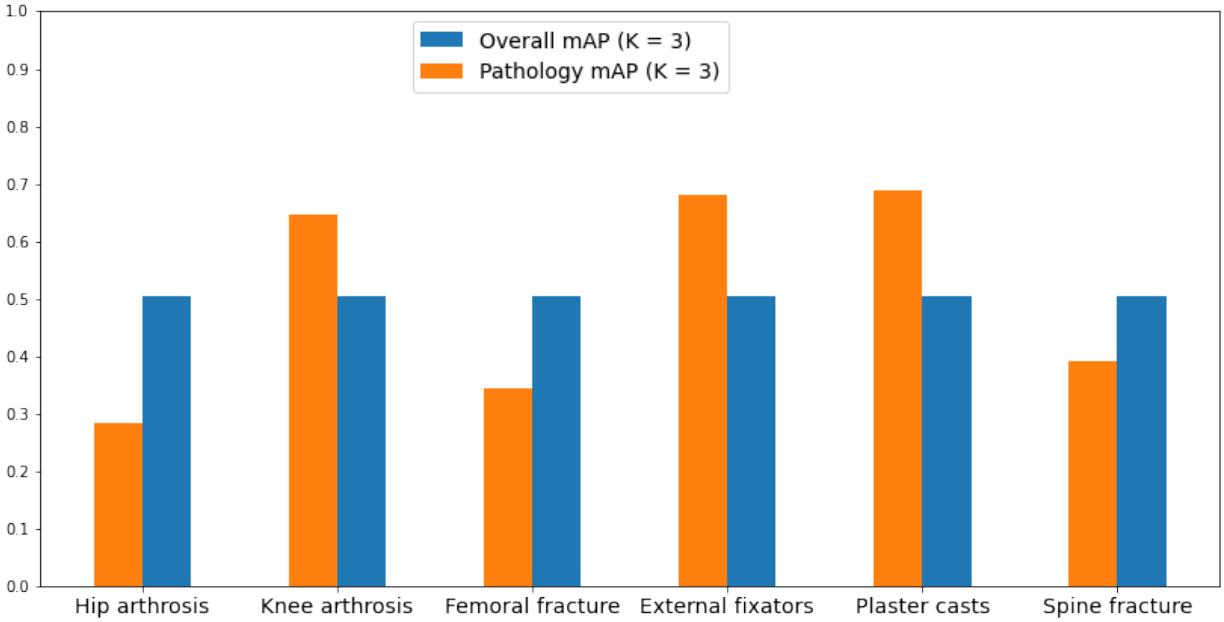


Figure 36: Overall vs Pathology mAP when retriving K=3 documents

Furthermore, it is essential to note that the dataset only contains only one possible answer for each question received, thereby differs from the other datasets with the same intention, making the comparison between them incommensurable. With only one answer, it means that only one document is relevant against all the retrieved ones.

For the Reader, HuggingFace provides the Processors library to facilitate the processing for NLP tasks, turning the examples into features for the model to work with; in this case, let's use SQuAD Processors and import the `squad_evaluate` method that dictates the model evaluation through the calculation of F1 and Exact Match.

Similarly to the approach in Chapter 5, the full system evaluation is performed under the Finder class that connects the Retriever to the Reader. For this purpose, a method `evaluate_finder` has been coded which combines the `query` function, shown in Listing 5, and the above `squad_evaluate` function.

While the increase in the number of documents retrieved may imply a greater evaluation on this component, it can have implications for the overall performance of the whole system, as the Reader not only becomes slower but now has to work on more text.

This leads to the need for defining an optimum K for each pathology addressed. Then, as presented in Listing 7, to obtain the several metrics' values, the value of k was iterated with every health issue.

```
...
contexts = ["coxartrose","gonartrose","fermur","fixadores","gesso","coluna"
    ]
all_results = []

for cxt in contexts:
    k_results = []

    for ik in range(1,5):
        examples = processor.get_dev_examples(filename=f"questions_{cxt}
    _test.json")

        results, examples, predictions = evaluate_finder(
            reader.predict,
            examples,
            index_name=index_name,
            context=cxt,
            topk=ik,
            )

        k_results.append(results)

    all_results.append({ "context": cxt, "scores": k_results })
```

Listing 7: Evaluation through iterating the number of documents retrieved

The results of this iteration have been compiled in Table 10. In the Reader, these results correspond the matching with the highest-ranked answer (N=1), since only one is returned to the user.

Table 10: EM and F1 scores considering K documents retrieved

Medical Pathology	K = 1		K = 2		K = 3		K = 4		K = 5	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Coxartrose (Hip arthrosis)	33.3	42.5	27.8	39.9	**27.8**	**40.4**	27.8	39.	27.8	39.4
Gonartrose (Knee arthrosis)	11.9	35.3	**19.1**	**40.1**	19.1	34.8	19.1	31.4	16.7	27.1
Fratura do fémur (Femoral fracture)	23.5	39.3	35.3	50.2	**41.2**	**51.**	**41.2**	**51.**	35.3	44.
Fixadores Externos (External fixators)	29.2	35.6	29.2	41.4	**33.3**	**48.6**	25.	40.2	16.7	26.3
Aparelhos gessados (Plaster casts)	20.	31.3	26.7	40.8	**33.3**	**57.1**	20.	45.5	20.	45.5
Fratura de coluna (Spine fracture)	17.4	23.2	26.1	42.3	21.7	41.6	**30.4**	**57.2**	**30.4**	**57.2**
Average	22.6	34.5	27.4	42.5	**29.4**	**45.6**	27.3	44.0	24.5	39.9

One can confirm that the implications discussed above are present in these results, because it improves upon the number of passages which the Retriever returns; however, it is up to a certain saturation point which from there can become a noise, deceiving the excepted final answer.

Briefly, in most pathologies, the best value of K is 3. Moreover, evidently, given the metrics' meaning, the Exact Match presents a value lower than the F1 score.

The values in bold represent the best ones within the same line. Looking at the whole picture, the EM metric varies between 19.1% and 41.2%, and the F1 score is between 40.1% and 57.2% in the different health conditions. As previously mentioned in the

Retriever evaluation, the gap between these values is based on the same reasons: the corpus size, answers quality of the documents, specific answers and users' questions.

Finally, by aggregating all the results and calculating their average, with the K being 3, the best and final scores of the system are: Exact Match with 29.4% and F1 score with 45.6%.

The authors are very pleased with these results. There are obviously improvement marks in both components. Nevertheless, analysing all the complexity of the system, the use of the Portuguese language and the handwork done in the testing dataset, it can be affirmed as a reasonable pioneer study within this medical area, with recent models that frequently neglect other languages besides English.

Conclusions 7

This chapter presents the conclusions of the study outlined in this dissertation. A summary of the primary purpose of this work is provided. The achievements and limitations which occurred during the development of the work are also highlighted.

It is further presented the value proffered to the scientific community and the contributions for the chatbot development field. Finally, it sets out several recommendations for feasible future work.

7.1 Synthesis of the Work

In this dissertation, taking into account the burden on the healthcare sector and the non-continuous monitoring of the patient, the outcome expected is a tool to assist patients, clarifying any doubts as quickly as possible about surgeries, procedures and implications of their health issue.

Apart from the technical specifications, the most fundamental goal of this work is to create and deliver a reliable source of information for patients to inform themselves and answer their concerns. All these should be intuitive and accessible, promoting a stand-alone use.

The above, as referred in Chapter 4, would be achieved by two independent modules. Through not only implementing a real-time chatting agent to answer questions, but also by having a practical and responsive web application, with Frequently Asked Questions.

Inferring upon this task, the creation of a minimum viable product was carried out which denotes as the final result of this dissertation. This version is presented with enough features to be used by early users, who can then provide feedback on future product development.

At this seed stage, the discussion and implementation were successfully applied to orthopaedic and traumatological pathologies. For the knowledge base for the system, health professionals gave a FAQ document on these scopes. Further engagement with these professionals was expected; however, due to the COVID-19 pandemic and overburdened health services, it was impossible to have them more involved.

The conversation agent was built with a full QA system with off-the-shelf parts using ElasticSearch and a Transformer architecture, BERT. The main point of having a two-stage QA system is to narrow the large search space for extracting the answer efficiently.

It was challenging to validate and obtain high performance by three factors. First, it is the most disruptive element in this project which is the use of the Portuguese language. There are not many studies and trained models, especially to treat specific terminology as what happens in the area of healthcare.

Next, it has a narrow document, with broad questions. Therefore, it makes the knowledge base limited to feed the entire model as well as its answer extraction. As a third factor, it is necessary to apply techniques of question generation to create a testing dataset. The annotation of the answers can be unbalanced since it is manual, which represents a direct influencing factor on the score, given the evaluation metrics.

The full-stack web application was developed using React and FastAPI. All the features elaborated have been from the chatbot backend, the user interface design, an intuitive FAQs section, up to the chatbot integration widget. Despite some roadblocks in the deployment due to pricing and application architecture, the full project is live.

In summary, these are the specific project milestones that have been successfully concluded:

- Comprehensive review of all the literature concerning chatbots and their practical applicability in the health area. General state-of-the-art has been made, as well as

the implementation of a systematic methodology that presents an unbiased and strict up-to-date overview of this topic;

- Specification and implementation of a health question-answering conversation agent. Plus, it enables unsupervised self-learning through fine-tuning the model from the users' feedback on the answers given;

- Development of a full-stack platform, taking into account the user experience and integrating the chatbot as a widget;

- Evaluate the system performance. However, instead of retrieving their satisfaction, the users were involved in creating the testing dataset by generating questions;

- Submit a full-paper manuscript with the Systematic Literature Review (Chapter 3) for a Journal on machine learning applications for healthcare.

Finally, looking at the objectives proposed in this dissertation and the milestones presented above, it can be concluded that they have been successfully achieved. Nevertheless, there is still plenty of room to explore and improve the performance to increase user satisfaction in using this system.

7.2 Dissertation Contributions

A wide variety of work has been developed during the development of this project. The main contributions to the scientific community and the chatbot development area are discussed here.

Besides the state-of-the-art on chatbots, it presents a Systematic Literature Review (Chapter 3), a strict and predefined methodology about textual chatbots in healthcare. Moreover, a full-paper manuscript containing this methodology, entitled "Systematic

Literature Review on Text-Based Chatbots for Healthcare" was submitted for a Special Issue in the Journal "Pattern Analysis and Applications" (Barbosa et al., under review).

This journal is evaluated in Q2 by Clarivate (2017) ranking and it was for the special issue "Computer Vision and Machine Learning for Healthcare Applications". The status at the time of delivering this dissertation is "Under Review". Even though the submission was made in August 2020, the acceptance notification will be given in December 2020.

As the final result of this dissertation, the web platform can be totally used by patients with ease of access in desktop or mobile [1]. Plus, it can be managed by non-technical people, i.e., edit, add or remove information, questions and answers. The backend offers versatility to be used on a wide range of applications, exposing an API to make it easier to merge or even to integrate in different applications on other systems. The project is live in `aidbot.vercel.app` with the backend running on `188.166.67.79`.

During the Data Collection phase in Chapter 6, a dataset was created in SQuAD format with labelled examples [2]. The content has not only the FAQs given by health professionals, but also the users' questions with its answers labelled. This collection is available for the community, believing it can be useful for more work and improvements, not only for Portuguese, but to increase the importance of machine learning and NLP application can have in the healthcare sector.

The authors attended the 26th National Congress of Internal Medicine, taking place in Braga, between 27th and 30th of August 2020. Although it was very specific in the medical scope, topics such as artificial intelligence applications in healthcare and medical technology were discussed in its papers, posters and talks.

1 https://github.com/apedrob/thesis-project
2 https://github.com/apedrob/thesis-data

Additionally, this work was the basis for an Research and Development (R&D) project application in the grant "RESEARCH 4 COVID 19", given by Foundation for Science and Technology (FCT). The aim was to reorient the work to meet the needs of the National Health Service in response to the COVID-19 pandemic. So, it was an adaptation to help patients and caregivers through a web platform and a chatbot containing reliable real-time information. However, despite all the work and time allocated to the application, the project was not funded and did not proceed.

7.3 Future Work

The purposes of this project have been achieved. Yet, throughout this work, recommendations for future contributions have emerged to improve the performance and validity of the established system. Accordingly, these suggestions proposed to study are explained below.

Taking into consideration that the NLP field is blistering, the research must continue as new models and datasets emerge that push the boundaries of question-answering.

In the research scope, the suggestion is watching closely two types of models. Those that combine search over large corpora with answer extraction, since, in this project, the Reader is dependent on the Retriever's returns. Moreover, such models that can infer an answer, not by extracting from one document but gathering several pieces of data from multiple documents - in other words, connecting the QA task with Text Summarisation.

In applying a question-answering system, several techniques in information retrieval should be examined for a more pragmatic chatbot. Methods for query expansion can be explored, such as entity enrichment and contextual synonym could conceive more answer-containing passages.

Passage ranking is another practical example which describes a broker as re-ranking the Retriever's documents to filter noisy content, possibly yielding more relevant documents to the Reader. Additionally, Feedback Learning section in Chapter 5 could perhaps lean towards this approach.

Besides the implementation aspect, the initial document was narrow when compared to the vast possibilities of questions and information that a patient could ask. Involving health professionals in this work could be more meaningful - improvements and enlargement on the knowledge base as well as experiments by formulating trustworthy testing dataset and tuning through their feedback.

Sometimes palliative care is devalued in the modern world. With the context of COVID-19 pandemic, there has been a shift in mentality, routines, and, above all, in innovations. Today, after worldwide compulsory confinement, it has become clear that society was unprepared for the full applicability and necessity of technologies, especially those related to Health. Finally, now, digital healthcare is growing, but still not enough to deal with possible modern hazards.

This is the call to action of this dissertation, the development and innovation must aim at the growth of society, literate technologically and, first and foremost, aim at the well-being for us all - because, in the end, Health is our most precious asset.

Bibliography

Amara, Sofiane, Macedo, Joaquim, Bendella, Fatima, and Santos, Alexandre. Group formation in mobile computer supported collaborative learning contexts: A systematic literature review. *Journal of Educational Technology & Society*, 19(2):258–273, 2016.

Amato, Flora, Marrone, Stefano, Moscato, Vincenzo, Piantadosi, Gabriele, Picariello, Antonio, and Sansone, Carlo. Chatbots meet ehealth: Automatizing healthcare. In *WAIAH@ AI* IA*, pages 40–49, 2017.

Andersen, Bo. Understanding analysis in elasticsearch (analyzers), May 2018a. URL https://codingexplained.com/coding/elasticsearch/understanding-analysis-in-elasticsearch-analyzers. Last accessed on Sep 2020.

Andersen, Bo. Understanding the inverted index in elasticsearch, May 2018b. URL https://codingexplained.com/coding/elasticsearch/understanding-the-inverted-index-in-elasticsearch. Last accessed on Sep 2020.

Barbosa, António, Macedo, Joaquim, and Craveiro, Olga. Systematic literature review on text-based chatbots for healthcare. *Pattern Analysis and Applications: Computer Vision and Machine Learning for Healthcare Applications*, under review.

Barcelos, Rafael Ferreira and Travassos, Guilherme Horta. Evaluation approaches for software architectural documents: a systematic review. In *CIbSE*, pages 433–446, 2006.

Beaudry, Jeremy, Consigli, Alyssa, Clark, Colleen, and Robinson, Keith J. Getting ready for adult healthcare: Designing a chatbot to coach adolescents with special health needs through the transitions of care. *Journal of pediatric nursing*, 49:85–91, 2019.

Bobriakov, Igor. A comparative analysis of chatbots apis, 2019. URL https://activewizards.com/blog/a-comparative-analysis-of-chatbots-apis/. Last accessed on Dec 2019.

Boden, Margaret A. *Mind as machine: A history of cognitive science*. Oxford University Press, 2008.

Bohr, Adam and Memarzadeh, Kaveh. The rise of artificial intelligence in healthcare applications. In *Artificial Intelligence in Healthcare*, pages 25–60. Elsevier, 2020.

Boren, Suzanne Austin and Moxley, David. Systematically reviewing the literature: Building the evidence for health care quality. *Missouri medicine*, 112(1):58, 2015.

Braun, Daniel, Mendez, Adrian Hernandez, Matthes, Florian, and Langen, Manfred. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 174–185, 2017.

Bridle, John S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Soulié, Françoise Fogelman and Hérault, Jeanny, editors, *Neurocomputing*, pages 227–236, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. ISBN 978-3-642-76153-9.

Budgen, David, Charters, Stuart, Turner, Mark, Brereton, Pearl, Kitchenham, Barbara, and Linkman, Stephen. Investigating the applicability of the evidence-based paradigm to software engineering. In *Proceedings of the 2006 international workshop on Workshop on interdisciplinary software engineering research*, pages 7–14, 2006.

Cerf, Vincent. Rfc0439: Parry encounters the doctor, 1973. Last accessed on Jan 2020.

Chen, Danqi, Fisch, Adam, Weston, Jason, and Bordes, Antoine. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017a. URL http://arxiv.org/abs/1704.00051.

Chen, Qian, Zhu, Xiaodan, Ling, Zhen-Hua, Wei, Si, Jiang, Hui, and Inkpen, Diana. Recurrent neural network-based sentence encoder with gated attention for natural language inference. *CoRR*, abs/1708.01353, 2017b. URL http://arxiv.org/abs/1708.01353.

Cho, Kyunghyun, van Merrienboer, Bart, Gülçehre, Çaglar, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

Clarivate. Journal impact factor - journal citation reports, 2017. URL https://clarivate.com/webofsciencegroup/solutions/journal-citation-reports/. Last accessed on Jan 2020.

Colby, Kenneth Mark. *Artificial paranoia: a computer simulation of paranoid process*. Pergamon Press, 1975.

CORE. Core rankings, 2006. URL http://www.core.edu.au/conference-portal. Last accessed on Jan 2020.

Coviaux, Quentin. Optimization of the search engine elasticsearch. Master's thesis, Universitat Politècnica de Catalunya, 2019.

Csaky, Richard. Deep learning based chatbot models. *arXiv preprint arXiv:1908.08835*, 2019.

Dale, Robert. The return of the chatbots. *Natural Language Engineering*, 22:811–817, 09 2016. doi: 10.1017/S1351324916000243.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Dybå, Tore, Kampenes, Vigdis By, and Sjøberg, Dag IK. A systematic review of statistical power in software engineering experiments. *Information and Software Technology*, 48(8): 745–755, 2006.

El Zini, Julia, Rizk, Yara, Awad, Mariette, and Antoun, Jumana. Towards a deep learning question-answering specialized chatbot for objective structured clinical examinations. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2019.

Epstein, Robert. The quest for the thinking computer. *AI magazine*, 13(2):81–81, 1992.

Epstein, Robert, Roberts, Gary, and Beber, Grace. *Parsing the Turing test*. Springer, 2009.

Fink, Arlene. *Conducting research literature reviews: From the internet to paper*. Sage publications, 2019.

Galitsky, Boris. *Developing Enterprise Chatbots: Learning Linguistic Structures*. 01 2019. ISBN 978-3-030-04298-1. doi: 10.1007/978-3-030-04299-8.

Goertzel, Ben, Giacomelli, Simone, Hanson, David, Pennachin, Cassio, and Argentieri, Marco. Singularitynet: A decentralized, open market and inter-network for ais. *Thoughts, Theories & Studies on Artificial Intelligence (AI) Research*, 2017.

Gonçalves, Fábio, Ribeiro, Bruno, Gama, Oscar, Santos, Alexandre, Costa, António, Dias, Bruno, Macedo, Joaquim, and Nicolau, Maria João. A systematic review on intelligent intrusion detection systems for vanets. In *11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT'2019)*, pages 28–30, 2019.

Google. Chatbot search interest over time. URL `https://trends.google.com/trends/explore?date=all&q=chatbot`. Last accessed on Apr 2020.

Hien, Ho Thao, Cuong, Pham-Nguyen, Nam, Le Nguyen Hoai, Nhung, Ho Le Thi Kim, and Thang, Le Dinh. Intelligent assistants in higher-education environments: The

fit-ebot, a chatbot for administrative and learning support. In *Proceedings of the Ninth International Symposium on Information and Communication Technology*, pages 69–76, 2018.

HIPAA. Hipaa compliance checklist. URL https://www.hipaajournal.com/hipaa-compliance-checklist/. Last accessed on Jan 2020.

Huang, Jizhou, Zhou, Ming, and Yang, Dan. Extracting chatbot knowledge from online discussion forums. pages 423–428, 01 2007.

Hussain, Shafquat, Sianaki, Omid Ameri, and Ababneh, Nedal. A survey on conversational agents/chatbots classification and design techniques. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 946–956. Springer, 2019.

Jones, Karen Sparck. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

Keele, Staffs et al. Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical report, Ver. 2.3 EBSE Technical Report. EBSE, 2007.

Krassmann, Aliane Loureiro, Flach, João Marcos, da Silva Grando, Anita Raquel Cestari, Tarouco, Liane Margarida Rockenbach, and Bercht, Magda. A process for extracting knowledge base for chatbots from text corpora. In *2019 IEEE Global Engineering Education Conference (EDUCON)*, pages 322–329. IEEE, 2019.

Li, Jiwei, Monroe, Will, Ritter, Alan, Galley, Michel, Gao, Jianfeng, and Jurafsky, Dan. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Lioma, Christina, Simonsen, Jakob Grue, and Larsen, Birger. Evaluation measures for relevance and credibility in ranked lists. In *Proceedings of the ACM SIGIR International*

Conference on Theory of Information Retrieval, ICTIR '17, page 91–98, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450344906. doi: 10.1145/3121050.3121072. URL https://doi.org/10.1145/3121050.3121072.

Lu, Wei, Ng, Hwee Tou, Lee, Wee Sun, and Zettlemoyer, Luke. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792, 2008.

Luhn, H. P. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957. doi: 10.1147/rd.14.0309.

Mateen, Farrah, Oh, Jiwon, Tergas, Ana, Bhayani, Neil, and Kamdar, Biren. Titles versus titles and abstracts for initial screening of articles for systematic reviews. *Clinical epidemiology*, 6:89–95, 03 2013. doi: 10.2147/CLEP.S43118.

Mauldin, Michael L. Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In *AAAI*, volume 94, pages 16–21, 1994.

Mnasri, Maali. Recent advances in conversational nlp: Towards the standardization of chatbot building. *arXiv preprint arXiv:1903.09025*, 2019.

Moor, James. *The Turing test: the elusive standard of artificial intelligence*, volume 30. Springer Science & Business Media, 2003.

Nigam, Amber, Sahare, Prashik, and Pandya, Kushagra. Intent detection and slots prompt in a closed-domain chatbot. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 340–343. IEEE, 2019.

Nimavat, Ketakee and Champaneria, Tushar. Chatbots: An overview types, architecture, tools and future possibilities. *IJSRD-International Journal for Scientific Research and Development*, 2017.

Øvretveit, John, Scott, Tim, Rundall, Thomas G, Shortell, Stephen M, and Brommels, Mats. Improving quality through effective implementation of information technology in healthcare. *International Journal for Quality in Health Care*, 19(5):259–266, 2007.

Pietsch, Malte, Soni, Tanay, Moeller, Timo, and Chan, Branden. Haystack - neural question answering at scale. https://github.com/deepset-ai/haystack, 2020.

Radziwill, Nicole M and Benton, Morgan C. Evaluating quality of chatbots and intelligent conversational agents. *arXiv preprint arXiv:1704.04579*, 2017.

Rajpurkar, Pranav, Zhang, Jian, Lopyrev, Konstantin, and Liang, Percy. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL http://arxiv.org/abs/1606.05250.

Ramesh, Kiran, Ravishankaran, Surya, Joshi, Abhishek, and Chandrasekaran, K. A survey of design techniques for conversational agents. pages 336–350, 10 2017. ISBN 978-981-10-6543-9. doi: 10.1007/978-981-10-6544-6_31.

Ramírez, Sebastián. Fastapi documentation, 2020. URL https://fastapi.tiangolo.com/. Last accessed on Dec 2020.

Robertson, Stephen and Zaragoza, Hugo. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

Romero, Manuel. Bert portuguese finetuned with squad in pt, 2020. URL https://huggingface.co/mrm8488/bert-base-portuguese-cased-finetuned-squad-v1-pt.

Ropero, Jorge, Gómez, Ariel, Carrasco, Alejandro, and León, Carlos. A fuzzy logic intelligent agent for information extraction: Introducing a new fuzzy logic-based term weighting scheme. *Expert Systems with Applications*, 39(4):4567–4581, 2012.

Sansonnet, Jean-Paul, Leray, David, and Martin, Jean-Claude. Architecture of a framework for generic assisting conversational agents. In *International Workshop on Intelligent Virtual Agents*, pages 145–156. Springer, 2006.

Saygin, Ayse Pinar, Cicekli, Ilyas, and Akman, Varol. Turing test: 50 years later. *Minds and machines*, 10(4):463–518, 2000.

Shieber, Stuart M. Lessons from a restricted turing test. *arXiv preprint cmp-lg/9404002*, 1994.

Shridhar, Kumar, Dash, Ayushman, Sahu, Amit, Pihlgren, Gustav Grund, Alonso, Pedro, Pondenkandath, Vinaychandran, Kovács, Gy¨orgy, Simistira, Foteini, and Liwicki, Marcus. Subword semantic hashing for intent classification on small datasets. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2019.

Shum, Heung-Yeung, He, Xiaodong, and Li, Di. From eliza to xiaoice: Challenges and opportunities with social chatbots. *Frontiers of Information Technology and Electronic Engineering*, 19, 01 2018. doi: 10.1631/FITEE.1700826.

Singhal, Amit et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

Sucupira, Plataforma. Qualis periódicos, 2014. URL https://sucupira.capes.gov-.br/sucupira/. Last accessed on Jan 2020.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014a. URL http://arxiv.org/abs/1409.3215.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014b.

ter Stal, Silke, Kramer, Lean Leonie, Tabak, Monique, op den Akker, Harm, and Hermens, Hermie. Design features of embodied conversational agents in ehealth: a literature review. *International Journal of Human-Computer Studies*, page 102409, 2020.

Thomas, Peter J et al. *The social and interactional dimensions of human-computer interfaces*. Cambridge University Press, 1995.

Traiger, Saul. Making the right identification in the turing test. In *The Turing Test*, pages 99–110. Springer, 2003.

Turing, Alan. Computing machinery and intelligence. *Mind*, LIX:433–460, 01 1950.

Turnbull, Doug. Bm25 - the next generation of lucene relevance, Oct 2015. URL https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/. Last accessed on Sep 2020.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Wallace, Richard S. The anatomy of alice. In *Parsing the Turing Test*, pages 181–210. Springer, 2009.

Wang, Alex, Singh, Amanpreet, Michael, Julian, Hill, Felix, Levy, Omer, and Bowman, Samuel R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018. URL http://arxiv.org/abs/1804.07461.

Weizenbaum, Joseph. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

Weizenbaum, Joseph. Computer power and human reason: From judgment to calculation. 1976.

Weizenbaum, Joseph. Eliza - a computer program for the study of natural language communication between man and machine (reprint). *Commun. ACM*, 26:23–28, 01 1983. doi: 10.1145/357980.357991.

Wolf, Thomas, Debut, Lysandre, Sanh, Victor, Chaumond, Julien, Delangue, Clement, Moi, Anthony, Cistac, Pierric, Rault, Tim, Louf, Rémi, Funtowicz, Morgan, Davison,

Joe, Shleifer, Sam, von Platen, Patrick, Ma, Clara, Jernite, Yacine, Plu, Julien, Xu, Canwen, Scao, Teven Le, Gugger, Sylvain, Drame, Mariama, Lhoest, Quentin, and Rush, Alexander M. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Worswick, Steve. Mitsuku wins 2019 loebner prize and best overall chatbot at aisb x, Dec 2019. URL https://aisb.org.uk/mitsuku-wins-2019-loebner-prize-and-best-overall-chatbot-at-aisb-x/. Last accessed on Jan 2020.

Xu, Bei and Zhuge, Hai. The influence of semantic link network on the ability of question-answering system. *Future Generation Computer Systems*, 108:1–14, 2020.

Zhou, Li, Gao, Jianfeng, Li, Di, and Shum, Heung-Yeung. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, (Just Accepted): 1–62, 2018.

Zwerschke, Chris. Python web frameworks, 2020. URL https://wiki.python.org/moin/WebFrameworks. Last accessed on Dec 2020.