Sara Filipa Rodrigues Martins Inácio

DYNAMIC VIRTUAL ASSISTANTS

1 2 9 0 UNIVERSIDADE Ɖ COIMBRA

1 2 9 0

UNIVERSIDADE Ɖ COIMBRA

Sara Filipa Rodrigues Martins Inácio

DYNAMIC VIRTUAL ASSISTANTS

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, advised by Professor Hugo Gonçalo Oliveira and by Professor Catarina Silva and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.

January 2022

Faculty of Sciences and Technology

Department of Informatics Engineering

Dynamic Virtual Assistants

Sara Filipa Rodrigues Martins Inácio

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. Hugo Gonçalo Oliveira and by Prof. Catarina Silva and presented to the
Faculty of Sciences and Technology / Department of Informatics Engineering.

January 2021

1 2 9 0

UNIVERSIDADE Ð
COIMBRA

This page is intentionally left blank.

Acknowledgements

First and foremost, I would like to thank my two extraordinary advisors, Prof. Hugo Gonçalo Oliveira and Prof. Catarina Silva, who have never left my side this past year and a half. Even when I was feeling lost, they always found a way to cheer me up and encourage me to keep going and always do my best. Thank you for you patience with me, and for being available everytime i needed, even when it was not your obligation. Thank you, from the bottom of my heart.

I would also like to thank our beloved department, DEI, that always made me feel at home. Especially, I have to thank all the professors that have been a part of my academic journey, and that were always there to help.

A heartfelt thank you to all my friends, who have stuck with me since day one, and who have been by my side through the good and the bad. If we celebrated, we celebrated together. If we cried, we cried together. Thank you, my close friends Vera, Caio, Martinho, Alex, Sofia, Maria and Rodrigo, for your constant support and for all the memories we made, especially during late nights in the department, accompanied by a project, blankets, lots of coffee and laughter.

Last but not least, a huge thank you to my family, from the bottom of my heart. To my parents, that not only gave me the opportunity to study, but also that always encouraged me to do more and better. To my boyfriend Mário, that has always supported me no matter what, and that, even in the worst days, had the most patience to talk me though it and never let me give up. To my little brothers, who always wanted me home but understood why I had to leave. Lastly, to the rest of my close family, a sincere thank you for accompanying me during the past 6 years, and always cheering me up. To all of you, thank you and I love you.

This page is intentionally left blank.

Abstract

Dialogue Systems have been progressing over the last few years, becoming more and more common and sought-after, for instance when it comes to reproducing actions usually performed by real persons, such as responding to questions regarding a specific domain. However, the more complex the conversational agents, the more resources necessary to create and maintain them. As such, our work explores different alternatives for the development of a conversational agent, capable of answering questions related to a given domain in Portuguese. We focus on two types of documents to be part of the domain, structured lists of question-answer pairs and collections of plain text documents. In addition, we evaluate and compare the alternatives whilst considering significant aspects such as time consumption when adapting to the domain, average answering time and answer quality, and implementation effort. We experiment with traditional IR, fine-tuned neural language models, and the combination of both. When using domains composed by lists of question-answer pairs, our main conclusion is that Whoosh, a text indexing IR engine, is the fastest to adapt to the domain and retrieve answers, whilst presenting a high percentage of quality answers. When using collections of documents as the domain, a combination of the neural language model BERT with IR-based Whoosh, used for pre-selecting the most relevant documents, also retrieves a high percentage of quality answers, whilst not requiring a great time consumption or demanding configuration.

Keywords

Dialogue Systems, Conversational Agents, Natural Language Processing, Question Answering, Information Retrieval, Encoder-decoder Model.

This page is intentionally left blank.

Resumo

Sistemas de Diálogo têm vindo a progredir nos últimos anos, tornando-se cada vez mais comuns e procurados, por exemplo para reproduzir funções habitualmente realizadas por pessoas, como responder a perguntas relativas a um domínio específico. No entanto, quanto mais complexos os agentes conversacionais, mais recursos são necessários para os criar e manter. Assim, o presente trabalho tem como objetivo explorar diferentes alternativas para o desenvolvimento de um agente conversacional, capaz de responder a perguntas relacionadas com um determinado domínio em português. Temos em conta dois tipos de documentos para formar o domínio, listas estruturadas compostas por pares de pergunta-resposta e coleções de documentos de texto simples. Para além disso, avaliamos e comparamos as alternativas, considerando aspectos importantes como o tempo dispendido na adaptação ao domínio, o tempo médio de resposta e a qualidade das respostas, e o esforço necessário durante a implementação. Fazemos experiências com IR tradicional, modelos neuronais de linguagem e a combinação dos dois. Ao utilizar domínios compostos por listas de pares pergunta-resposta, a nossa conclusão princpial é que o Whoosh, um mecanismo IR de indexação de texto, é o mais rápido na adaptação ao domínio e na recolha de respostas, apresentando ao mesmo tempo uma percentagem alta de respostas de qualidade. Ao usar coleções de documentos como domínio, uma combinação do modelo neuronal de linguagem BERT com o Whoosh, usado para pré-selecionar os documentos mais relevantes, também apresenta uma alta percentagem de respostas de qualidade, sem consumir muito tempo ou exigir uma configuração complicada.

Palavras-Chave

Sistemas de Diálogo, Agentes Conversacionais, Processamento de Linguagem Natural, Resposta a Perguntas, Recuperação de Informação, Modelo Codificador-decodificador.

Contents

Acronyms

FAQs	Frequently Asked Questions. 2
IR	Information Retrieval. 2, 9
NLP	Natural Language Processing. 1, 5
NLU	Natural Language Understanding. 1
QA	Question Answering. 10
RNN	Recurrent Neural Network. 13

This page is intentionally left blank.

List of Figures

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

Over the course of the past years, meaningful advancements in the field of Artificial Intelligence were accomplished. Amongst such advancements were dialogue systems, progressing substantially from conversational agents focused on emulating human-to-human conversations to agents capable of assisting users accomplishing tasks, for instance in the shape of virtual assistants, such as Apple's Siri, Amazon's Alexa, Google Assistant or Microsoft's Cortana.

Such evolution caused conversational agents to become an ordinary aspect in everyday life, gaining interest with the passing of time, presumably due to the multitude of functionalities these systems provide. Functionalities which range from placing orders[1] and planning outfits[2] to banking related services[3] including balance checking and payments or transfers processing.

Natural Language Understanding (NLU) derives from Natural Language Processing (NLP), a field that gives computers the ability to understand and process natural language. NLU aids computers understanding and interpreting language by decomposing and classifying speech in relevant elements, such as intents and entities. Whilst intents represent an action the user wants to perform by extracting the purpose or goal behind that utterance, entities specify and retrieve parameters from such utterance required to execute the action.

As previously mentioned, conversational agents have become a sought-after approach, as far as impersonating functions generally performed by real persons, such as responding to questions. NLU platforms such as Google Dialogflow, Microsoft LUIS and Altice Labs BOTSchool are a popular approach to the development of systems focusing on question answering, as they offer an intuitive interface that allows for an uncomplicated construction of such systems.

However, the aforementioned platforms require the specification of intents, entities and respective responses regarding the domain intended to be learnt. Particularly, Natural Language Understanding platforms demand that examples of inputs and corresponding outputs are defined, often manually, so that the respective agents can be trained for responding to interactions as naturally as possible and correctly answering questions on the target domain.

Having an input with an intent not yet defined would result in the non-recognition of

[1]https://www.chatbotguide.org/starbucks-bot
[2]https://www.chatbotguide.org/h-m-bot
[3]https://www.chatbotguide.org/santander-uk-bot

the utterance's intent, leading thus to a default or fallback one, possibly not responding to the user's question or stating instead that the input was not understood. Furthermore, conversational agents derived from such platforms are not able to evolve independently, from new unrecognized inputs for instance, having therefore to constantly rely on manual labor for improvement.

Not only does the manual effort associated with the creation and maintenance of conversational agents through NLU platforms escalate with the size and diversity of the desired domain, as, in addition, it must be applied whenever a new agent is created, or the domain altered, leading to an intensive requirement of both human and financial resources.

With the intensive requirement of resources, the use of conversational agents from such platforms may become an unfeasible option. This leads to the main purpose of our work, which focuses on the study and comparison of distinct alternatives to the creation of conversational agents capable of answering questions related to a given domain.

This work can thus be divided into three significant phases: (i) **research**, where different state-of-the-art approaches are studied and assessed on their capability of answering questions related to a given domain; (ii) **configuration**, where the approaches deemed as most relevant are implemented and configured to receive a collection of data as the domain and answer questions related to that same domain; (iii) **evaluation**, where each approach is used for answering a set of pre-defined questions and several metrics are computed from their answers, to later compare the different approaches on crucial aspects such as answer quality, answering time and expended manual effort related to the configuration.

Although most available state-of-the-art approaches are designed to be used with English, our focus lies in the Portuguese language. The approaches to be configured and evaluated must accept a collection of documents on the target domain and posed questions in Portuguese, as well as provide the answers in that same language.

In addition, we consider two types of documents for the approaches to receive: Frequently Asked Questions (FAQs), i.e., a list of question-answer pairs; and plain text, i.e., a collection of documents containing raw text. As long as it belongs to one of the types, the approaches should be able to receive a domain composed by any subject. For instance, we had to our disposal a domain related to Portuguese economic activity and several domains provided by the Altice company, containing information about telecommunication equipment they supply.

Following the three phases we defined as the course for our work, we began by researching the currently available state-of-the-art approaches, which lead us to a set of approaches that can be divided into three categories: (i) traditional Information Retrieval (IR); (ii) fine-tuned neural language models; (iii) a combination of (i) and (ii).

With the approaches selected, we configured each one to receive both types of documents, i.e., FAQs and raw text, and to answer a posed question at a time, in Portuguese language. We then submitted the approaches to an experimentation and evaluation phase, where each was given a domain and related questions to be answered. This process was repeated for each domain we had, and all retrieved answers were stored and later used to compute different metrics.

Finally, we compared the results obtained by the different approaches on each domain type, focusing on critical aspects such as answer quality, measured by the evaluation metrics, average answering time, and subjectively assessed expended manual effort.

A traditional IR based approach proved to outperform the remaining approaches when answering questions related to a domain composed by a list of FAQs, as not only it was the least time consuming, but also the one with highest percentage of correct answers and answer quality scores.

The same IR-based approach was also our selection to answering questions related to a collection of plain text documents, when combined with a state-of-the-art neural language model. This combination, where the first pre-selected the most relevant documents and the latter used relevant documents as context to retrieve an answer for a given question, achieved a high percentage of quality answers, whilst not requiring a demanding configuration or great time consumption.

In addition to the information regarding the configuration and performance of the different approaches, we also created a software package containing all the alternatives to the creation of conversational agents fully configured to receive either a list of FAQs or a collection of plain text documents and answer questions related to the given domain, in Portuguese. The code of the package is available in github[4].

This package has the purpose of helping those interested in our work to test the various implemented approaches and check the distinct types of answers they provide, as well as allowing to see results with domains of different subjects. We also hope that it helps those who are trying to implement or improve a Portuguese question-answering agent, as it provides a set of already configured options.

One platform we aspire to significantly contribute to is Altice's BOT-School[5]. This NLU platform, built to simplify the design and integration of conversational agents into applications, reveals several limitations, mentioned briefly in the forthcoming chapter 3. Thus, we hope to contribute to Altice company with not only the knowledge we obtained but also with the ready-to-use approaches configured for answering questions given a Portuguese domain containing FAQs or raw text, some of which may be integrated in the formerly built platform in order to further improve it.

Lastly, part of our work is reported in a scientific paper (Inácio et al., 2021), published in the proceedings of the 27th Portuguese Conference on Pattern Recognition. Our paper is focused on the approaches we configured and their performance when used for answering questions related to a domain composed by a collection of plain text documents. It also disclosed the results presented by each approach, as well as our main conclusions.

The remainder of this document is structured as follows. In the upcoming chapter, relevant concepts related to the field of language processing such as Natural Language Processing, Vector Semantics, Information Retrieval and Question Answering are briefly explained, as they form the basis of the present work. Additionally, Neural Networks and Transformer-based models are also concisely mentioned in chapter 2. The third chapter introduces and analyzes related work. Rule-based, corpus-based, information retrieval-based and sequence-to-sequence model-based dialogue systems are covered in chapter 3, whilst referring the state-of-the-art. Chapter 4 describes in detail the followed architecture, the data used throughout experimentation and the selected approaches. The configuration details for each approach are also depicted in this chapter. Chapter 5 gathers all information regarding the conducted experimentation with the different approaches and the results obtained through evaluation, as well as the retrieved conclusions. The sixth and last chapter presents the main conclusions and mentions possible future work.

[4]https://github.com/NLP-CISUC/PT_QA_Agents.git
[5]https://botschool.ai/

This page is intentionally left blank.

Chapter 2

Background

This work is dedicated to exploring several approaches to the construction of a Portuguese question-answering conversational agent, in order to retrieve useful conclusions. In particular, we focus on obtaining knowledge about the performance of each approach when it comes to answering questions in Portuguese language related to a given domain, composed by a structured list of FAQs or by an unstructured collection of raw text, distributed by one or more documents.

Prior to commencing any sort of development, we believe that it is important to understand a few relevant concepts. Thus, the following section provides a brief description of the concepts we considered to be relevant for the understanding of the upcoming work.

We start with a concise explanation of the field of Natural Language Processing, along with a description of its subfields Vector Semantics and Information Retrieval, as they form the basis of how systems process and understand language. Having our focus on question-answering agents, we address the subfield of Question Answering.

We then discuss Neural Networks, as nowadays many language processing tasks involve them, doing so with a modest description of a few important aspects within the field. Finally, as within the set of state-of-the-art approaches we consider in this work are neural language models, we sought fit to also discuss Transformers network architecture. To close the section, we mention the most relevant features of the two well-known transformer-based models GPT-2 and BERT.

2.1 Natural Language Processing

Natural Language Processing (NLP) is derived from major fields such as linguistics and computer science, created to allow computers to understand human language and be able to use it to communicate. NLP focuses on aiding computers with the accomplishment of human language-related tasks, including human-machine communication or text and speech processing (Jurafsky and Martin, 2021).

The achievement of this type of tasks can prove to be challenging, as computers expect to receive and process structured data, which human speech is usually not. In addition, human speech can often be ambiguous, when the same sentence may have more than one correct interpretations. An example of pragmatic ambiguity is a sentence often used to call a taxi, where *call me a taxi* could have two meanings, asking to hail for a taxi or to be called taxi. The sentence *I saw the man with the binoculars* is an example of structural ambiguity, where it could mean i saw a man by using some binoculars or that i saw a man wearing binoculars. Another type of ambiguity is semantic ambiguity, where, for example, the phrase *I spent my evening at the bank* could either mean i spent the evening at the building bank, or alongside the river.

Consequently, language processing applications explore the knowledge of language at several levels, including morphology, syntax, semantics, pragmatics and discourse. The high frequency of ambiguity in human speech leads most language processing-related tasks to focus on resolving ambiguity towards one or more of the aforementioned linguistic knowledge levels.

Morphology is the capability of understanding that words are composed by meaningful component parts, such that one individual word may have several variations. For instance, Figure 2.1 shows how the words *door* and *doors* should be interpreted by the computer as the same individual word, with the first carrying the component singular and the latter the component plural.

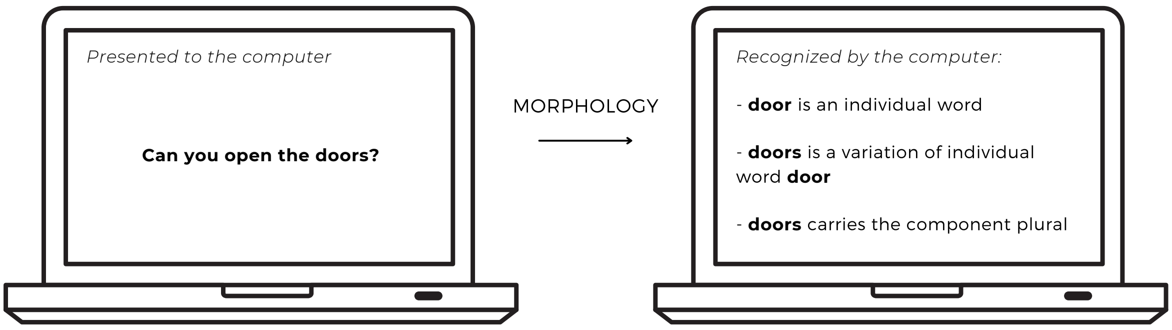

Presented to the computer
Can you open the doors?
MORPHOLOGY
Recognized by the computer:
- door is an individual word
- doors is a variation of individual word door
- doors carries the component plural

Figure 2.1: Morphology Example

As for syntax, it represents the comprehension of structural relationships between words, required to order or group them correctly. This structural knowledge is required for the computer to properly join together the correct words and form an appropriate answer. Figure 2.2 contains an example of two sentences composed by the exact same words, where *Yes i can open the door* makes sense, and *Door open I yes can* does not, due to the words positioning.

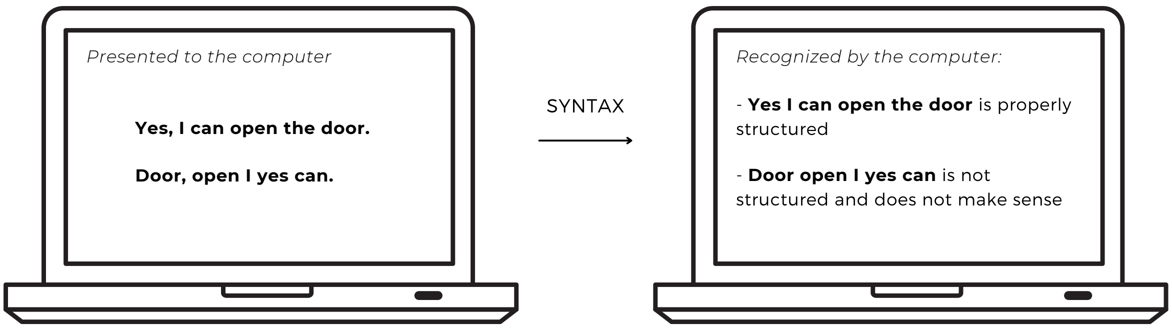

Presented to the computer
Yes, I can open the door.
Door, open I yes can.
SYNTAX
Recognized by the computer:
- Yes I can open the door is properly structured
- Door open I yes can is not structured and does not make sense

Figure 2.2: Syntax Example

Semantics can be described as the awareness of the meaning of each word, and can be divided in two fields: (i) **lexical semantics**, that represents the meaning of all words and relations between them; (ii) **compositional semantics**, that defines the meaning of words when combined with others. An example presenting the difference between the fields of semantics is displayed in Figure 2.3, where, given the same sentence, lexical semantics identify the meaning of each word separately, e.g. *close* and *door*, and compositional semantics identify the meaning related to a combination of words, in this case a temporal endpoint represented by the set of words *by the end of the day*.

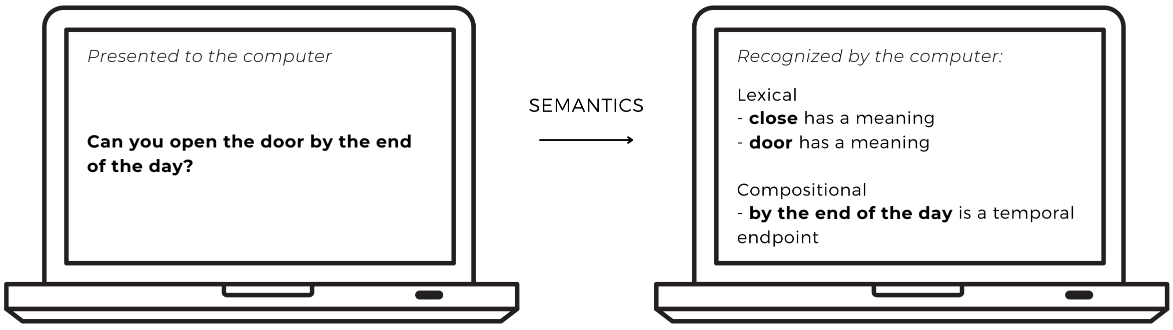

Presented to the computer
Can you open the door by the end of the day?
SEMANTICS
Recognized by the computer:
Lexical
- close has a meaning
- door has a meaning
Compositional
- by the end of the day is a temporal endpoint

Figure 2.3: Semantics Example

Pragmatics goes beyond the meaning of the words, and portrays the recognition of actions behind sentences. For instance, as can be seen in Figure 2.4, several sentences with many similar words can have very different meanings, ranging from an affirmation to a request of information.

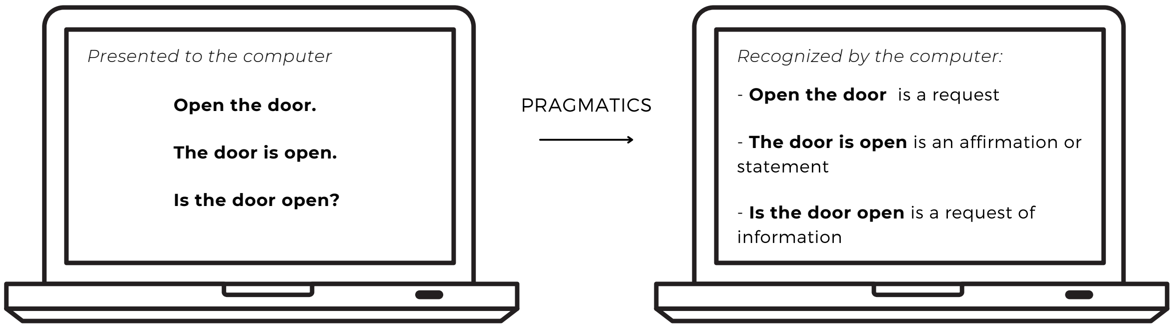

Presented to the computer
Open the door.
The door is open.
Is the door open?
PRAGMATICS
Recognized by the computer:
- Open the door is a request
- The door is open is an affirmation or statement
- Is the door open is a request of information

Figure 2.4: Pragmatics Example

Finally, discourse is is a combination of several sentences, as opposed to solely considering the current utterance. An example is presented in Figure 2.5, where the computer associates the expression *that day* with *Friday*, mentioned in the previous utterance.

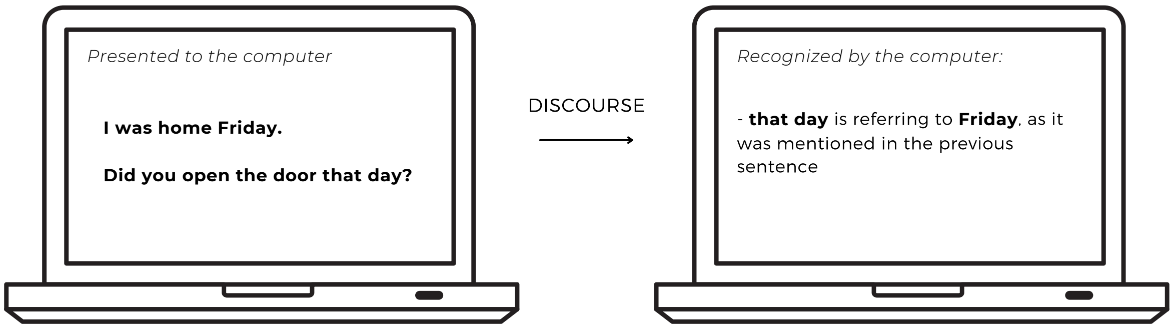

Presented to the computer
I was home Friday.
Did you open the door that day?
DISCOURSE
Recognized by the computer:
- that day is referring to Friday, as it was mentioned in the previous sentence

Figure 2.5: Discourse Example

The different levels of knowledge can be represented by theories or models derived from the fields of computer science, mathematics, and linguistics. Such theories or models are then used by algorithms with the purpose of transforming unstructured data to a format understandable by the computer.

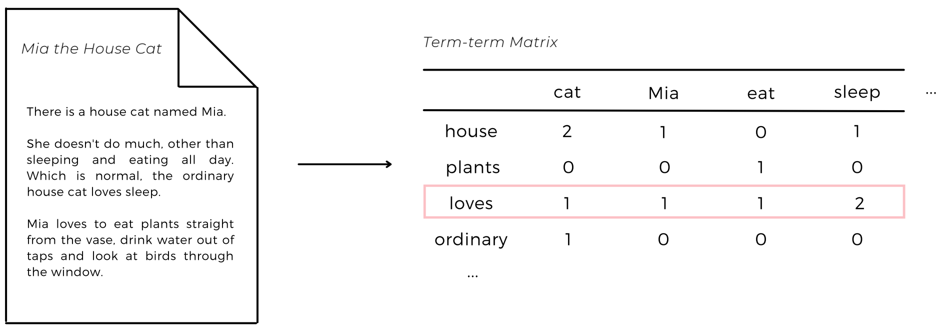
2.2 Vector Semantics and Information Retrieval

Vector semantics is a field that allows to represent words according to their distribution in collections of text (Jurafsky and Martin, 2021). As the Distributional Hypothesis (Harris, 1954) states, words having identical neighboring words, occurring in similar grammatical environments and contexts are likely to have a similar meaning.

This field combines the definition of a word according to its prior usage with that same word's meaning, through a point in some multidimensional semantic space, also known as a vector. The vectors representing the meaning of words can also be referred to as embeddings (Jurafsky and Martin, 2021).

Embeddings can be based on co-occurrence matrices, such as term-term or term-document matrices. Whilst term-term matrices represent words through the number of times they appear next to other words, term-document matrices represent words through their occurrence frequency in a given set of documents. Both matrices grant the vector representation of each word in the vocabulary, through the respective row.

Term-term matrices represent words through the frequency of their occurrence next to other words. The rows and columns labels of these matrices both contain all the words in the vocabulary, thus having each cell represent the number of times two distinct words co-occur in some context. This context is most commonly defined as a brief window of words around the term in question. A term-term matrix example is shown in Figure 2.6, where the resulting vector of the word *loves* is circled.



Mia the House Cat

There is a house cat named Mia.

She doesn't do much, other than sleeping and eating all day. Which is normal, the ordinary house cat loves sleep.

Mia loves to eat plants straight from the vase, drink water out of taps and look at birds through the window.

...

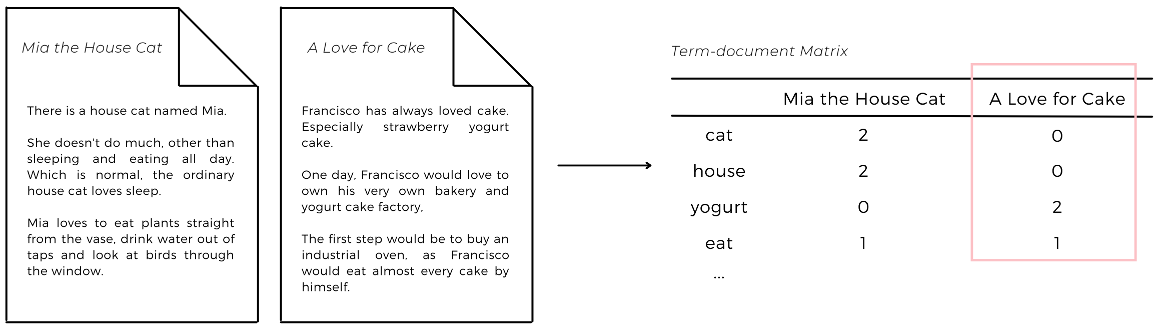
Term-term Matrix

	cat	Mia	eat	sleep	...
house	2	1	0	1	
plants	0	0	1	0	
loves	1	1	1	2	
ordinary	1	0	0	0	
...					

Figure 2.6: Term-term Matrix Example

In addition to representing words, term-document matrices also represent documents with vectors. The words representation is done through the number of their occurrences in each document, and the documents representation is done through the frequency of each word in the vocabulary occurring in it.

In this type of matrices, the rows labels contain all the words in the vocabulary and the columns labels contain the documents, having as such each cell represent the number of occurrences of a word in a certain document. An example of a term-document matrix with two documents is presented in Figure 2.7, where the vector representing the document *A Love for Cake* is circled. Note that, similarly to the previous example, each row of the matrix is a resulting vector for the respective word.



Mia the House Cat

There is a house cat named Mia.

She doesn't do much, other than sleeping and eating all day. Which is normal, the ordinary house cat loves sleep.

Mia loves to eat plants straight from the vase, drink water out of taps and look at birds through the window.

A Love for Cake

Francisco has always loved cake. Especially strawberry yogurt cake.

One day, Francisco would love to own his very own bakery and yogurt cake factory.

The first step would be to buy an industrial oven, as Francisco would eat almost every cake by himself.

Term-document Matrix

	Mia the House Cat	A Love for Cake
cat	2	0
house	2	0
yogurt	0	2
eat	1	1
...		

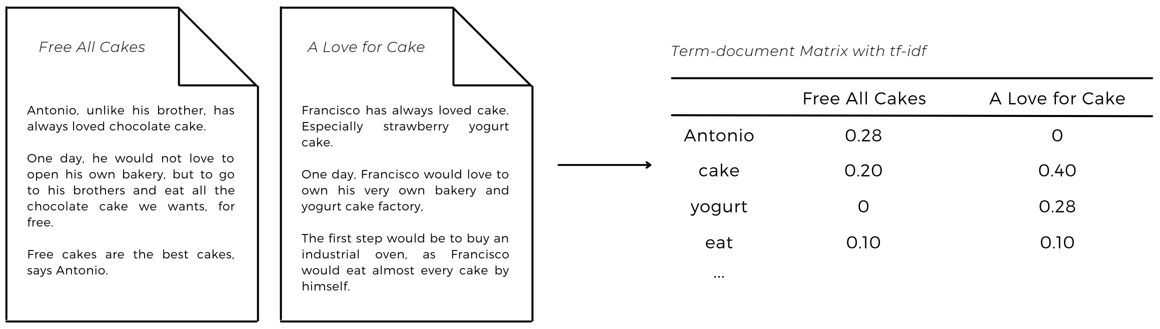
Figure 2.7: Term-document Matrix Example

Information Retrieval (IR) can be formally described as the task of finding and retrieving a document from a collection of documents that best matches a provided query (Manning et al., 2008). As two similar documents tend to be composed by identical words and term-document matrices supply a vector representation of each document through the number of times each word occurs, documents with similar words are bound to have approximate vector representations. Thus, term-document matrices were originally defined to serve the purpose of IR, due to each column providing a vector representation of a document.

In traditional IR vectors are sparse, due to having the same dimension as the vocabulary. Each position of the vector has an associated weight, thus far being considered as the frequency of occurrence of the word in a given document. However, instead of the weights in a vector simply being the frequency, there are different metrics to calculate them.

One of them is tf-idf, a metric that combines the frequency of a word with its relevancy. Words that occur in only a few documents of the whole collection are considered as more relevant than others that appear in most documents, as they can be useful in the discriminating the few documents from the rest of the collection. Words that appear in fewer documents are considered as more relevant and assigned a higher weight.

Figure 2.8 shows an example of a term-document matrix with tf-idf weights, where the words *Antonio* and *yogurt*, only appearing in one document, show a higher weight than the word *eat* that appears in both. Furthermore, the word *cake*, appearing in both documents, has a higher wait associated with the second document, due to it appearing more often in that document.



Free All Cakes

Antonio, unlike his brother, has always loved chocolate cake.

One day, he would not love to open his own bakery, but to go to his brothers and eat all the chocolate cake we wants, for free.

Free cakes are the best cakes, says Antonio.

A Love for Cake

Francisco has always loved cake. Especially strawberry yogurt cake.

One day, Francisco would love to own his very own bakery and yogurt cake factory.

The first step would be to buy an industrial oven, as Francisco would eat almost every cake by himself.

Term-document Matrix with tf-idf

	Free All Cakes	A Love for Cake
Antonio	0.28	0
cake	0.20	0.40
yogurt	0	0.28
eat	0.10	0.10
...		

Figure 2.8: Term-document Matrix with tf-idf Example

Having the vectors representing words or documents, it is possible to measure the similarity between them. A well-know method to do so is calculating the cosine of the angle formed between the vectors, where the higher the cosine, the more similar the vectors.

2.3 Question Answering

Question Answering is a original NLP task, first approached by (Phillips, 1960), with a text-based algorithm that performed a simple parsing of the question and sentences in the document, and then looked for possible answers.

Another early approach was presented by (Simmons et al., 1964), a more complex system that would form a query from the words in the question and then retrieve candidate answer sentences from the given document. The question would then be parsed with each candidate sentence and the one whose structure better matched the question structure would be retrieved as the answer. For instance, as depicted in Figure 2.9, the question *What does Francisco eat?* would lead to the answer *Francisco eats cake.*, as that answer shares the same structure as the posed question, of the subject *Francisco* being dependent of *eat*.

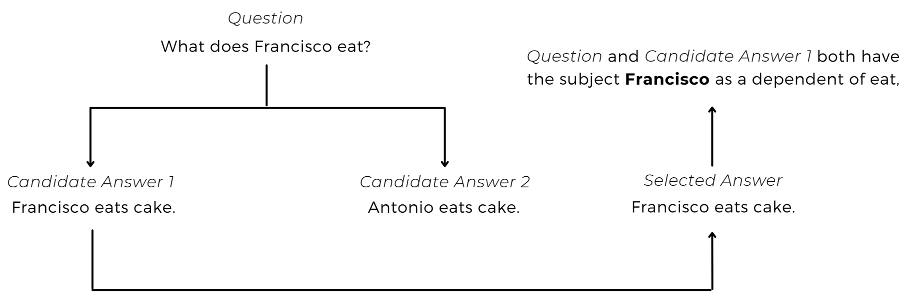


Figure 2.9: Early QA Approach Example

By definition, Question Answering (QA) systems focus on providing answers to questions posed in natural language (Jurafsky and Martin, 2021). As these are usually factoid questions, such systems tend to reply with brief passages, commonly containing simple facts. QA systems can be divided in two categories: (i) **IR-based**, that search textual information for a suitable answer, and (ii) **knowledge-based**, that build a representation of the question and later query databases.

2.3.1 IR-based Systems

IR-based question answering systems rely on textual information, e.g. a collection of documents, to answer a posed question, as they explore the available information to retrieve an adequate answer (Kolomiyets and Moens, 2011). The pipeline of these systems is typically composed by three crucial phases: question processing, passage retrieval and ranking, and answer extraction. Figure 2.10 shows the three phases of the pipeline.

In the question processing phase, a query formed by relevant keywords is extracted from the posed question. This query is then used to find documents potentially containing an appropriate answer, process which is accomplished with IR-based techniques.

Occasionally, systems can perform query reformulation, by rephrasing the question to an equivalent declarative statement. In addition, some systems are capable of extracting more than keywords from the posed question, thus extracting complementary information

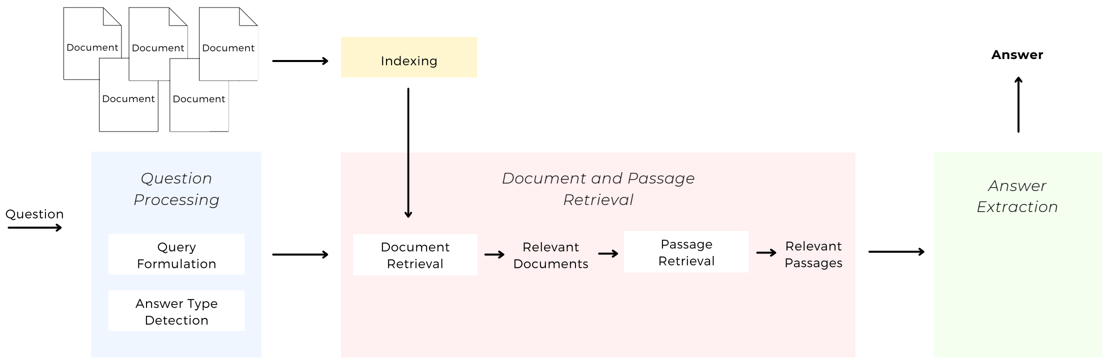


Figure 2.10: Three Main Phases of IR-based QA Systems

such as an answer type, a named entity that allows to categorize the answer and restrict the search for the response. An example of posed question and extracted information is presented in Figure 2.11.

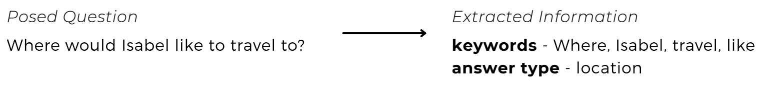

Posed Question
Where would Isabel like to travel to?
Extracted Information
keywords - Where, Isabel, travel, like
answer type - location

Figure 2.11: Information Extraction Example

When in the document and passage retrieval phase, the query formulated in the previous phase is passed through an IR-based engine, which retrieves a set of relevant documents ranked by their level of relevance. However, as most replies are intended to be composed by brief statements and not whole documents, systems may divide the top-rated documents in segments, such as paragraphs or sentences.

A straightforward approach to passage retrieval is to send every relevant passage to the succeeding answer extraction phase. Another possible approach is to take advantage of the aforementioned complementary information in the query, that allows for a classification of the retrieved passages and, consequently, for the possibility of discarding the segments not containing the answer type in question.

Furthermore, supervised learning techniques can also be used to rank the significant passages, utilizing features such as the number of question keywords in the segment. Figure 2.12 shows an example of passage elimination, where the candidate passage *would like to try something new.* is discarded due to not containing the identified answer type, *location.*

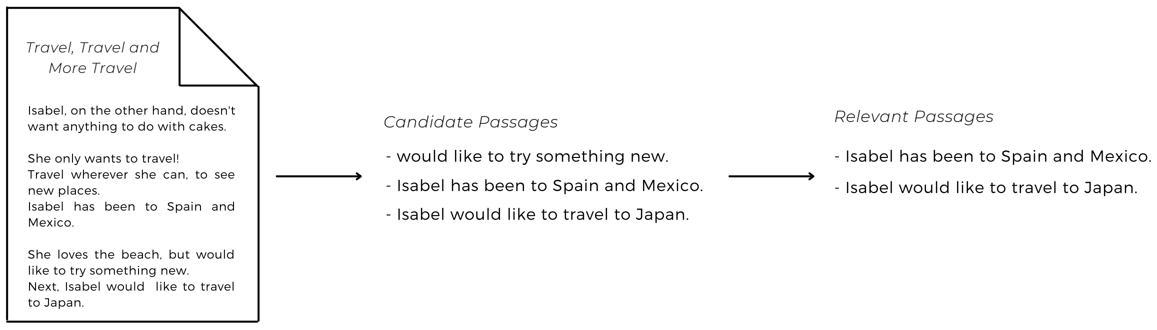

Travel, Travel and More Travel
Isabel, on the other hand, doesn't want anything to do with cakes.
She only wants to travel!
Travel wherever she can, to see new places.
Isabel has been to Spain and Mexico.
She loves the beach, but would like to try something new.
Next, Isabel would like to travel to Japan.
Candidate Passages
- would like to try something new.
- Isabel has been to Spain and Mexico.
- Isabel would like to travel to Japan.
Relevant Passages
- Isabel has been to Spain and Mexico.
- Isabel would like to travel to Japan.

Figure 2.12: Passage Elimination Example

The final phase of question answering is the extraction of a suitable answer from the most relevant passages. In theory, a suitable answer would be a passage or a segment of a passage containing the information necessary to answer the posed question.

An algorithm capable of identifying the question's previously defined named entity on a candidate passage can be considered as a forthright approach to extracting the answer, as it returns the segment of the passage carrying the corresponding answer type. An example of answer extraction from a candidate passage carrying several keywords and an answer type corresponding to the posed question is presented in Figure 2.13.

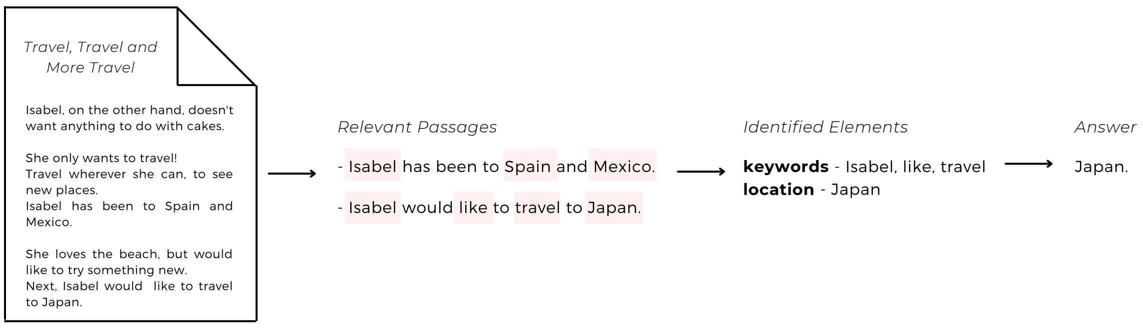

Travel, Travel and More Travel
Isabel, on the other hand, doesn't want anything to do with cakes.
She only wants to travel!
Travel wherever she can, to see new places.
Isabel has been to Spain and Mexico.
She loves the beach, but would like to try something new.
Next, Isabel would like to travel to Japan.
Relevant Passages
- Isabel has been to Spain and Mexico.
- Isabel would like to travel to Japan.
Identified Elements
keywords - Isabel, like, travel
location - Japan
Answer
Japan.

Figure 2.13: Answer Extraction Example

Nevertheless, many segments in passages tend to not have a specific answer type, which requires the use of more advanced algorithms, usually based on supervised learning. Supervised learning can be described as the task of learning how to map from a new input to a correct output, based on information formerly provided, i.e., several inputs being associated with the corresponding output (Jurafsky and Martin, 2021).

Feature-based answer extraction is an example of a supervised learning approach that trains a classifier to distinguish whether a given passage contains a suitable answer to the question or not (Jurafsky and Martin, 2021).

Neural answer extraction is another interesting approach to extract answers. This approach relies on the principle that a suitable response for a question is semantically similar to the question itself. As such, algorithms based on a neural network approach compute an embedding for the whole question as well as an embedding for each token of the candidate answer. Subsequently, they calculate the similarity between the question and each passage word in context, to then select the passage segments whose embeddings are most similar to the question embedding.

2.3.2 Knowledge-based Systems

Knowledge-based systems construct a semantic representation of the question and later use it to query structured facts-filled databases (Jurafsky and Martin, 2021). To create the question's representation, i.e., to define its logical form, several methods can be utilized, ranging from rule-based to semi-supervised.

When logical form relations in semantic representations of questions are considerably recurrent, rule-based methods can be beneficial, as a set of defined rules would be enough to extract relations from the questions.

However, when data is available for supervision, such as an list of pairs composed by questions and the related logical form, it becomes favorable to take advantage of the data and assemble a system capable of mapping new questions to their logical forms.

The creation of training collections containing questions and the respective meaning representation can be arduous. In addition, supervised datasets are usually not capable of ensuring coverage to the broad diversity of forms questions can take. Thus, semi-supervised methods are utilized to provide textual redundancy, from text retrieved from the web or from databases containing multiple questions carrying the same meaning, for instance. This allows for the creation of new pairs of questions and respective logical forms, whilst learning to map the question with its correct meaning representation.

It is important to mention that the type of QA system to use is not exclusive, as it is possible to simultaneously use IR-based and knowledge-based methods, depending on both the textual information and structured databases to find a suitable answer for a posed question.

2.4 Neural Networks

A neural network is an interconnection of neurons that processes information, that is, a network composed by neural units with each receiving a vector of input values and producing a single output value (Hassoun et al., 1995). Every neural unit calculates the output value by multiplying the input values by a weight vector, adding a bias, and applying a non-linear activation function.

In a neural network, neurons are arranged in layers, such that neurons in a layer are connected only to neurons in the previous and following layers. The more layers a network has, the deeper, more complex it is, thus being able to solve more intricate problems.

One of the simplest types of such networks is a Feed Forward Neural Network, a multilayer network which does not contain cycles in units' connections, as outputs from each unit are passed to units in the next higher layer, never being passed back to lower layers (Hassoun et al., 1995). An example of Feed Forward network in depicted in Figure 2.14.

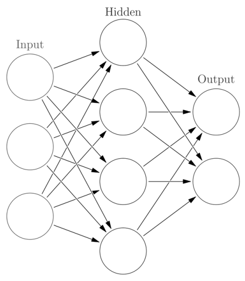


Figure 2.14: Feed Forward Network Example

Networks learn to solve problems through a number of forward and backward propagations. Forward propagation sums the values associated with each neuron and related edges, whilst backward propagation compares the values of the network's output layer with the actual answer, subsequently updating the edge values, in the order of last-to-initial layers.

2.4.1 Recurrent Neural Network

A more complex type of neural network is a Recurrent Neural Network (RNN), which can be seen as a Feed Forward Network extended over time. The output of a RNN's neural unit at a given point in time depends on both the current input and value of the hidden layer from the previous time step (Hassoun et al., 1995).

Recurrent networks deal with sequential data, input with some defined ordering, processing sequences one element at a time. Thus, Recurrent Neural Networks can be of use in NLP, as discourse can be seen a sequence of sentences, which in turn can be seen as a sequence of words. An application for RNNs are recurrent neural language models, that use the current word along with the previous hidden state to predict the next word in a sentence (Mikolov et al., 2010).

In addition, several architectures rise from RNNs, such as vector-to-sequence models, sequence-to-vector models, or sequence-to-sequence models. In sequence-to-sequence models, at each step the output is used to predict the next element in a sequence, with the size of the input and output sequences being equal.

In order for such sequences to have different sizes, sequence-to-sequence models must follow an encoder-decoder architecture, where the encoder converts the input sequence to a vector with meaning, and the decoder converts such vector to an output sequence making use of its context.

2.4.2 Transformers

Transformers, introduced by (Vaswani et al., 2017), are a recent network architecture based on attention mechanisms. These models rely solely on attention to correlate the input with the output, and have proven to be more parallelizable, when compared to recurrent networks.

Transformers follow an encoder-decoder architecture, as Figure 2.15, retrieved from (Vaswani et al., 2017), depicts. The encoder maps an input sequence to a sequence of continuous representations, whilst the decoder generates an output sequence one element at a time, making use of the sequence retrieved by the encoder. Such model, whilst generating text, considers at each step the previously generated element as additional input, thus being auto regressive.

In opposition to RNNs, such models' encoder receives all input's components at the same time and calculates the respective embeddings simultaneously, through parallelization. For instance, if the input were to be a sentence, a Recurrent Neural Network would receive a word at a time, calculating each embedding in one time-step, whilst the recently introduced model would receive the entire sentence at once and calculate all words' embeddings concurrently.

After creating the embeddings representing the input, the encoder takes into account the position of each input component, by defining new representation vectors containing that information. The new representation vectors are then passed to the encoder block, containing a Self-Attention mechanism and a Feed Forward Network.

This mechanism assigns the relevancy of each element regarding the other elements in the sequence, thus computing a representation of the sequence. The vectors resulting from the Self-Attention mechanism are then simultaneously sent to the Feed Forward Network, which retrieves a set of encoded vectors the decoder can utilize, each corresponding to an element in the sequence.

Similarly to the encoder, the decoder receives a sequence and concurrently creates the embeddings regarding its components, whilst also considering their position. The computed

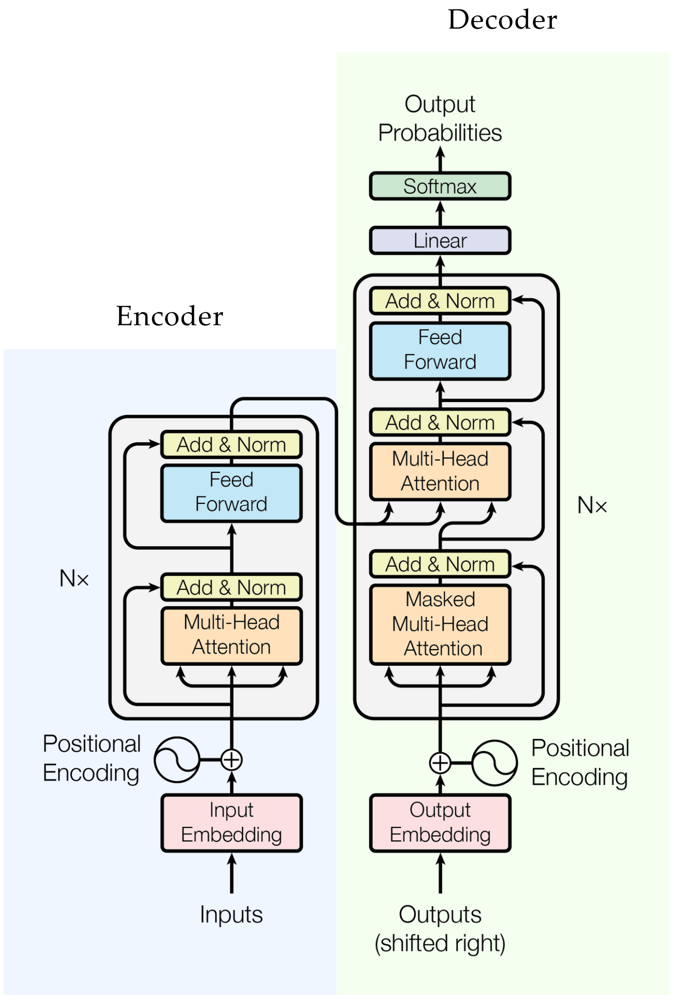


Figure 2.15: Transformer Architecture

vectors are then passed to the decoder block, that has two attention mechanisms and a Feed Forward Network.

The difference between the encoder and the decoder's first Attention block is that the latter only has knowledge about the sequence up until the last generated element. This is done in order to preserve the auto regressive property of the model, by masking the elements that appear in the positions ahead of the previously generated element. The decoder's second Attention block determines how relevant each element's vector is, with respect to remaining. It considers both the decoder's current sequence and the output of the encoder.

Finally, the attention vectors deriving from the second Attention block are sent to a Feed Forward Network, being as such transformed to a form understandable by the next layers. These vectors are then passed through the Linear and Softmax layers, which predict the next element by retrieving a probability distribution, such that the next generated element in the sequence is the one presenting the highest probability value.

Two well-known transformer-based encoder-decoder models are GPT (Radford et al., 2018, 2019; Brown et al., 2020) and BERT (Devlin et al., 2018). These models, however distinct in some architectural aspects, share the common premise of obtaining exemplary results regarding a combination of natural language processing tasks, by firstly pre-training the language model on a diverse unlabeled text corpus and subsequentially finetuning the same model, in order for it to adjust to the respective task.

GPT

GPT-2 is a transformer-based language model trained with eight million web pages, composed by 40G of English text data derived from the social media platform Reddit[1] (Radford et al., 2019). This model was trained to predict the next word given all the previous ones in a certain text span.

GPT-3 is an upgraded and much larger version of GPT-2, with its larger model having a number of architecture hyperparameters equal to 175B (Brown et al., 2020). For comparison purposes, the largest version of the GPT-2 model has 1542M hyperparameters. This third-generation language model in the GPT series was trained with a combination of distinct datasets, resulting in a total of 300B tokens derived from Common Crawl[2], an expanded version of WebText (Radford et al., 2019), two internet-based books corpora and English Wikipedia pages.

BERT

BERT is a bidirectional encoder-based transformer, originally pre-trained with the BooksCorpus (Zhu et al., 2015) and English Wikipedia text passages, resulting in a training corpora containing over 3000M words. This multi-layer model was designed with a purpose, to pre-train bidirectional representations from unlabeled text data by making each token attend to the context on both its left and right side, in all layers (Devlin et al., 2018).

2.5 Discussion

Having disclosed a brief introduction to relevant concepts related to our work including Natural Language Processing, Vector Semantics, Information Retrieval, Neural Networks and Transformers architecture, we began our research phase for related work.

The upcoming chapter contains the collected information about several dialogue systems, divided by the two architectures they can have: rule-based or corpus-based. It mentions some of the original dialogue systems and state-of-the-art approaches to the implementation of one.

[1]https://www.reddit.com/
[2]https://commoncrawl.org/the-data/

Chapter 3

Related Work

Dialogue systems can carry different purposes, such as being strictly conversational agents performing chitchat, or being task-oriented systems helping users with their requests. Such systems may adopt distinct architectures, ranging from rule-based to corpus-based systems, which include IR-based systems and encoder-decoder models.

Considering the existence of several types of dialogue systems, learning about the most relevant was deemed as important, thus having research conducted on the different types of systems, how they can be utilized and what functions they provide. The following sections present such research.

3.1 Dialogue Systems

Dialogue systems, also known as conversational agents, are computer programs capable of communicating with a human user in natural language, in the form of text, speech, or both. Usually, such programs can be partitioned into agents focusing on emulating extended human-to-human conversations or agents focusing primarily on performing actions, resorting to conversation as a method for goal achievement, or task completion (Jurafsky and Martin, 2021).

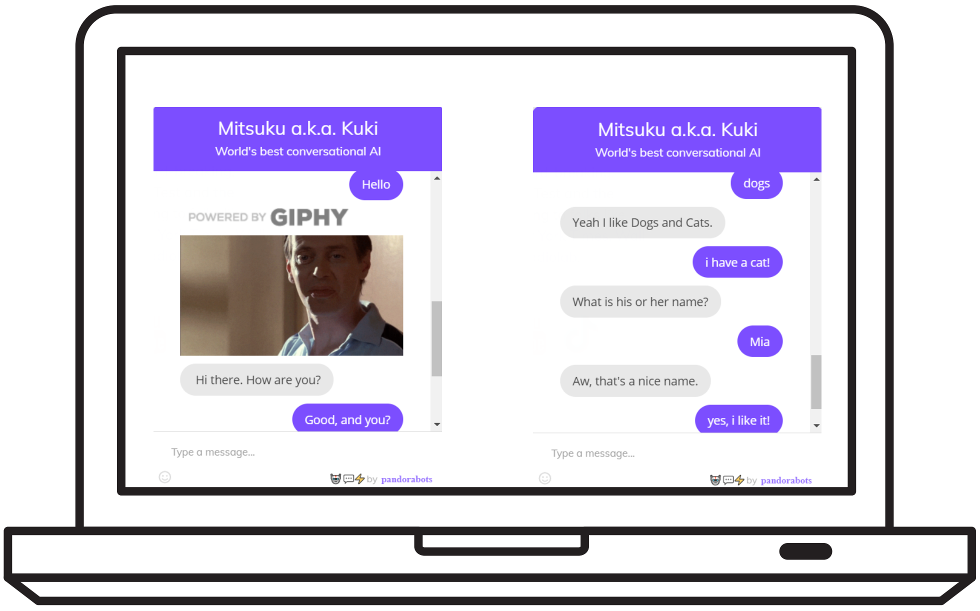
Conversational agents of the latter category, whilst capable of communicating with a human user, focus mainly on providing assistance or performing specific tasks, such as searching for restaurants (Wen et al., 2016) or assisting customers in online purchase-related tasks (Yan et al., 2017), thus not supporting extended conversations.

For instance, task-oriented systems can be part of virtual assistants such as Siri[1], Alexa[2], Google Assistant[3] or Cortana[4], and hold the purpose of increasing the user's productivity, whether in a proactive or reactive form. These systems are capable of reactively responding to user's requests, such being presented in the form of questions or actions (Sarikaya, 2017). Providing information about the weather forecast or making a restaurant reservation can be considered an example of reactive assistance. Moreover, having access to multiple data sources such as calendars, emails or personal profiles, task-oriented systems can proactively provide relevant information based on the user's profile and current time and space, without there being a specific request for such information (Sarikaya,

[1]http://www.apple.com/ios/siri/
[2]https://alexa.amazon.com/
[3]https://assistant.google.com/
[4]https://www.microsoft.com/en-us/cortana

2017). An example of proactive assistance could be a notification informing the user regarding a scheduled meeting and the estimate time to arrive to such meeting location. As well as being able to provide a vast set of services related to numerous domains, the above-mentioned systems can also rely on web search as backup to answer questions or actions not understood.

Dissimilarly, chatbots, the simplest kind of dialogue systems, are systems predominantly focusing on emulating conversations characteristic of informal interactions amidst two humans, hence sustaining protracted conversations. For instance, chatbots such as Mitsuku[5] or Rose[6] serve the exclusive purpose of sustaining a conversation in natural language. An example of a small conversation with Mitsuku can be seen in Figure 3.1.



Hello

Hi there. How are you?

Good, and you?

dogs

Yeah I like Dogs and Cats.

i have a cat!

What is his or her name?

Mia

Aw, that's a nice name.

yes, i like it!

Figure 3.1: Small Conversation with Mitsuku Chatbot

Regarding architecture, chatbots can be divided in two distinct classes, one being composed by systems relying on pre-defined rules, following a rule-based architecture, and the other by systems focusing on provided data, thus following a corpus-based architecture (Jurafsky and Martin, 2021).

3.1.1 Rule-based Systems

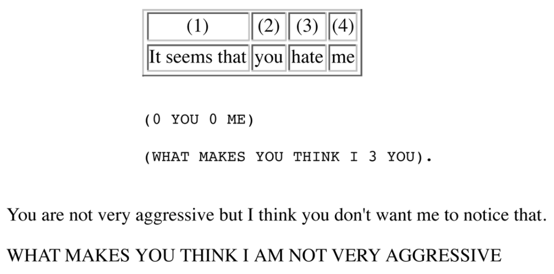
Rule-based systems, as previously mentioned, depend on a pre-defined set of rules to function. Systems with aforesaid architecture are not designed to decipher the meaning behind language, that is, are not capable of understanding the knowledge of language at, for instance, morphology and semantics levels, therefore having to rely solely on the pre-defined rules to create and provide responses to users' utterances.

ELIZA (Weizenbaum, 1966), one of the earliest chatbots to emerge, simulated a psychology consultation by analyzing the patient's statement and responding accordingly, in a similar manner as a psychologist would. This chatbot's answers were based on reflecting the input statements back at the patient, following a set of pattern-based rules defined to recognize aspects present in the input and create a suitable output. Precisely, ELIZA

[5]https://www.pandorabots.com/mitsuku

[6]http://brilligunderstanding.com/rosedemo.html

would analyze the patient's statement in search for a keyword, and subsequently transform the sentence in accordance with a rule associated with such keyword. For instance, as presented in Figure x, it would recognize the keywords you and me present in an input phrase such as “It seems that you hate me.” and conceive a response by combining keywords, the text in-between keywords and a defined phrase, leading to an answer of the sort “What makes you think I hate you?”. A similar example can be seen during a conversation with a patient, in Figure 3.2.



(1)	(2)	(3)	(4)
It seems that	you	hate	me

```
(0 YOU 0 ME)
(WHAT MAKES YOU THINK I 3 YOU).
```

You are not very aggressive but I think you don't want me to notice that.

WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE

Figure 3.2: Passage from a Conversation Between ELIZA and a Patient

Not long after ELIZA, a chatbot similarly following a rule-based architecture surfaced. PARRY (Colby et al., 1971) was created with the intention of studying schizophrenia, as it emulated a paranoid individual. Although having an identical operation to ELIZA regarding the creation of responses, this system additionally included a mental model simulating emotions, containing variables representing fear, anger, and mistrust. The level of these variables would affect PARRY's responses, as for example, it would respond with hostility if the anger level was high.

A rule-based system can be useful when dealing with problems accompanied by a predefined outcome. Systems relying on such architecture are not capable of learning by themselves, which implies that it is not possible for them to deviate from the function established by the developer, or respond outside of the defined rules.

Although not data-intensive as corpus-based, rule-based systems require the definition of rules for every possible response. This is often done manually, so rule-based chatbots can easily become a challenge to maintain, considering the diverse forms of speech and writing that vary from one individual to another.

3.1.2 Corpus-based Systems

Conversely to the aforementioned rule-based systems, corpus-based depend on corpora to communicate with the user, thus relying on a substantial amount of data to learn how to map a response from a given input.

Corpora can derive from diverse sources, such as transcripts of human-to-human conversations in natural language, for instance, textual content acquired from social networking platforms as Twitter (Ritter et al., 2010) or Reddit (Zhang et al., 2019b), dialogue retrieved from movie subtitles (Jena et al., 2017; Danescu-Niculescu-Mizil and Lee, 2011), or lists of Frequently Asked Questions (FAQs) (Ranoliya et al., 2017; Santos et al., 2020). Moreover, knowledge sources such as news articles or Wikipedia pages (Yan et al., 2016), can also constitute a non-dialogue corpora, thus allowing the possibility of, for example, a fact aware chatbot.

Considering the multitude of sources corpora can arise from, the opportunity of creating systems focused on specific topics emerges. For instance, Amaia (Santos et al., 2020) sustains the purpose of answering questions related to the provided domain, FAQs from the Portuguese Administrative Modernization Agency web portal.

With respect to the process of mapping from a user utterance to a system response, chatbots following a corpus-based architecture can be partitioned in two categories, the first being comprised by systems focusing on information retrieval methods and the second being composed by systems relying on sequence-to-sequence approaches.

Information retrieval-based systems retrieve responses by searching the corpora for a suitable answer. Concretely, these systems compare the given input with passages in the provided corpora and return as response to the user the passage deemed as the most appropriate. In order to classify each passage, different methods based on similarity are utilized, ranging from simple metrics as word overlapping to pre-trained language models or models specifically trained for the purpose, such that passages most similar to the input are attributed greater classification scores. Thus, the passage best classified is presented to the user as a response.

Information Retrieval-based Systems

Previously mentioned NLU platforms such as Google Dialogflow[7], Microsoft LUIS[8] and Altice Labs BOTSchool[9] are systems that operate with information retrieval methods to interact with the user, serving the mutual purpose of simplifying the design and integration of conversational interfaces into developers' applications. For instance, LUIS (Williams et al., 2015) is described as a service capable of conceding developers a considerably brief and straightforward deployment of a language understanding model specific to their necessity.

The aforementioned platforms share similar approaches regarding the creation of conversational agents. Intents distinguishing the action the user wishes to perform and entities specifying and retrieving details from such utterance must be defined, accordingly to the particular domain. In addition, utterances to be handled by the agent must also be designated, attributed an intent and present entities specified. For instance, given the input utterance 'I want to buy a cheap shirt', the intent would be the action of buying a shirt, whilst the entity would be the price range, as it is a relevant characteristic to the action.

In the occasion that an utterance is not recognized by the agent, the user receives a response derived from a fallback intent, regularly mentioning that the input was not understood and indirectly demanding the user to reformulate and resend the question or information. Figure 3.3 depicts an example of a chatbot created with BOTSchool, identifying the intent behind the input utterance and another example of the same chatbot not being able to do so, thus following the fallback intent.

As such systems are not capable of self-learning from previous interactions with users, in order to correct the issue of unidentified utterances, manual labor is required. The developer is required to either manually attribute the correct intent to the unrecognized input or create a new intent and posteriorly perform the attribution, which implies the necessity of constant manual labor for maintenance and improvement.

However, Google Dialogflow, recently introduced a functionality which enables the

[7]https://cloud.google.com/dialogflow
[8]https://www.luis.ai
[9]https://botschool.ai

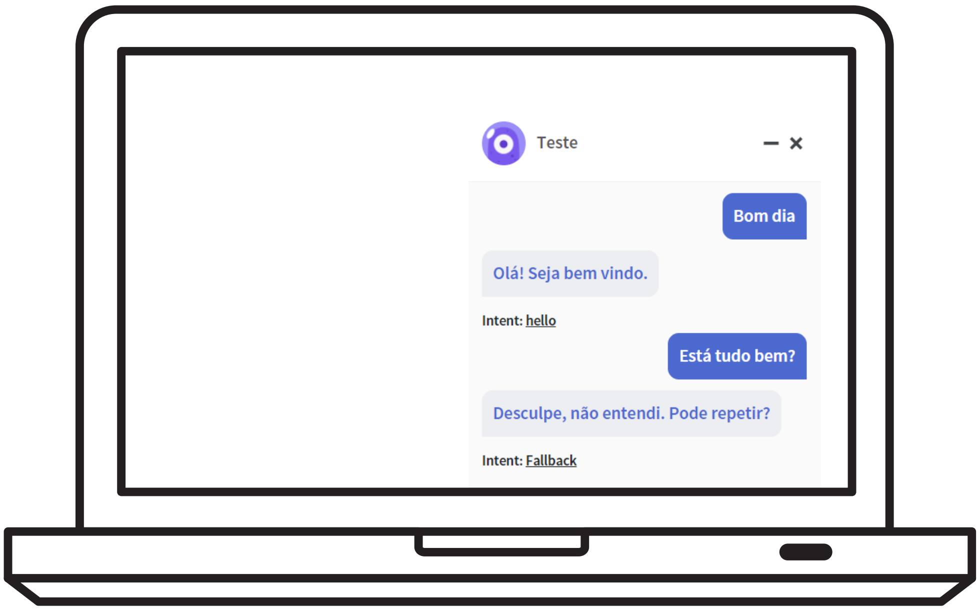


Figure 3.3: Example of Intent Identification and Non-Identification with BOTSchool

addition of knowledge bases to the system[10], thus allowing for a considerably less time-consuming approach to question answering. Instead of defining each query through intents and entities, it becomes possible to add documents of a certain format to the knowledge base. One type that can be included is a document containing question-answer pairs, FAQs, for instance.

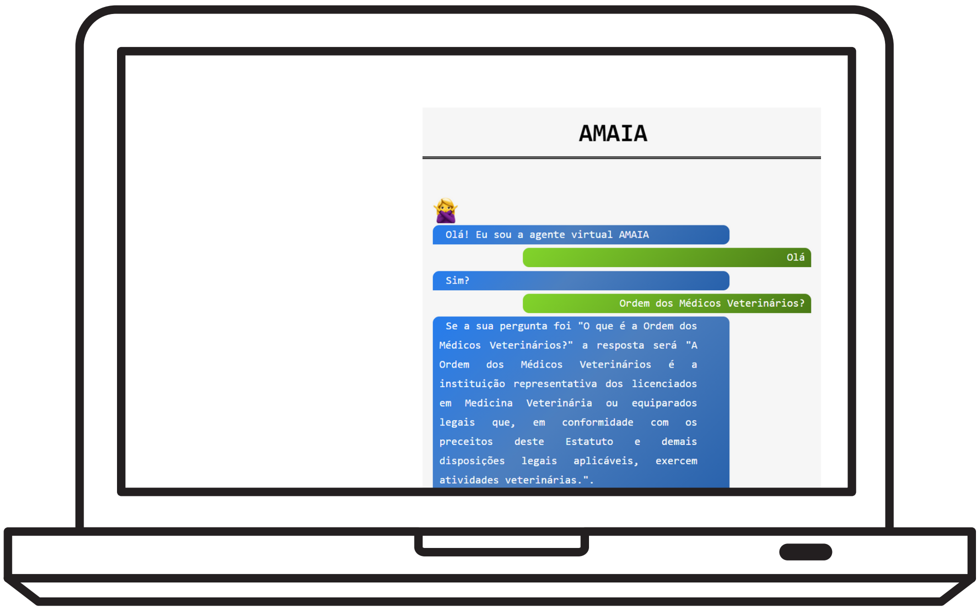
By adding documents containing information in such format, the capability of answering as many questions as necessary emerges, with an associated configuration lasting merely a few minutes. Documents in the knowledge base can also be composed by unstructured text, later to be structured and utilized for question answering. However, Dialogflow documentation states that this method is currently in experimental phase.

Notwithstanding NLU platforms, another relevant type of IR-based conversational systems are chatbots focusing in FAQs answering, a subject approached by previously mentioned Amaia (Santos et al., 2020). Such system benefits from a knowledge base composed by question-answer pairs of domain FAQs, thus utilizing IR-based methods for indexing and retrieving the thirty most significantly related questions in the corpus. Afterwards, Amaia compares the query with the collection of significant questions, through the use of a Semantic Textual Similarity model, and retrieves the answer associated with the most similar one. Figure 3.4 presents an example of a small chat with Amaia, in which the system answers a FAQ contained within its domain.

Sequence-to-sequence Models

In opposition to Information Retrieval-based systems, which retrieve information and present it to the user as a response, sequence-to-sequence models are capable of generating an output derived from the user's input. These models are generally neural networks composed by an encoder capturing the context of the input sequence and a decoder autoregressively generating the output sequence, element by element (Vinyals and Le, 2015). Transformer-based encoder-decoder models such as GPT (Radford et al., 2018, 2019; Brown

[10]https://cloud.google.com/dialogflow/es/docs/how/knowledge-bases



Olá! Eu sou a agente virtual AMAIA

Olá

Sim?

Ordem dos Médicos Veterinários?

Se a sua pergunta foi "O que é a Ordem dos Médicos Veterinários?" a resposta será "A Ordem dos Médicos Veterinários é a instituição representativa dos licenciados em Medicina Veterinária ou equiparados legais que, em conformidade com os preceitos deste Estatuto e demais disposições legais aplicáveis, exercem atividades veterinárias.".

Figure 3.4: Domain-Related Conversation with AMAIA

et al., 2020) and BERT (Devlin et al., 2018) are examples of sequence-to-sequence models.

For instance, (Budzianowski and Vulić, 2019) illustrates that the developed GPT-based generative model is capable of understanding and generating dialogue particular to a domain when finetuned with related text-only information, thus proving to be a system that allows for a seamless adaptation to diverse domains or domain-specific vocabularies.

Similarly, (Henderson et al., 2019) exposes how the respective developed transformer-based dialogue system, pretrained on large general-domain corpora from Reddit[11] social network, is able to adapt to six distinct domains through the process of finetuning the model with information in accordance with each domain.

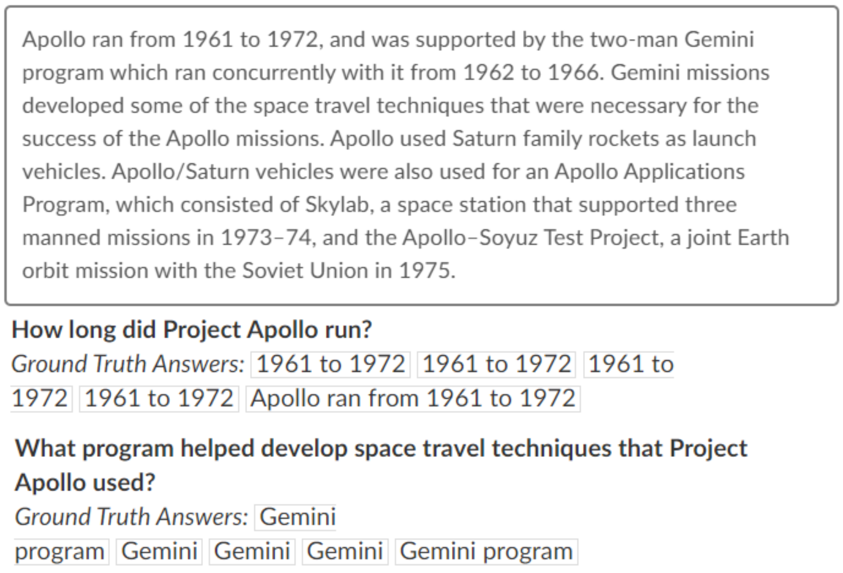
Few existing architectures (Qu et al., 2019; Gu et al., 2020) take context into consideration and, as result, are capable of maintaining multiple turns of information exchange. However, most aforementioned systems, whether being IR-based or transformer-based, focus on generating a response based on the current input, and are not able to sustain lasting conversations with the user.

The dialogue system presented by (Henderson et al., 2019) is pretrained on a large Reddit dataset, containing almost four billion comments, and finetuned with six distinct corpora, which are composed by input-response pairs. One of the datasets used for finetuning falls under the e-banking domain and is constituted by FAQs and respective answers, in question-answer pairs. The conducted experiments revealed that 94.8% of the times the top ranked response was indeed the correct one.

Likewise, BERT (Devlin et al., 2018) is pretrained on large corpora, the BooksCorpus and English Wikipedia, together containing close to three thousand million words. The system is finetuned with several different datasets, one of which the Stanford Question Answering Dataset (Rajpurkar et al., 2016), commonly known as SQuAD, a reading comprehension dataset composed by over one hundred thousand questions posed on a set of Wikipedia articles. The answer to each question is a segment of text from the corresponding reading passage of a certain Wikipedia article. Figure 3.5 depicts an example

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

of a passage with the associated questions and answers. The performed experiments with SQuAD v1.1 presented results regarding accuracy of around 80% to 90%.


Apollo ran from 1961 to 1972, and was supported by the two-man Gemini program which ran concurrently with it from 1962 to 1966. Gemini missions developed some of the space travel techniques that were necessary for the success of the Apollo missions. Apollo used Saturn family rockets as launch vehicles. Apollo/Saturn vehicles were also used for an Apollo Applications Program, which consisted of Skylab, a space station that supported three manned missions in 1973–74, and the Apollo–Soyuz Test Project, a joint Earth orbit mission with the Soviet Union in 1975.

How long did Project Apollo run?
Ground Truth Answers: 1961 to 1972 1961 to 1972 1961 to 1972 1961 to 1972 Apollo ran from 1961 to 1972

What program helped develop space travel techniques that Project Apollo used?
Ground Truth Answers: Gemini program Gemini Gemini Gemini Gemini program

Figure 3.5: Example of Passage, Questions and Respective Answers from SQUAD v1.1

Similarly, the GPT-based system introduced in (Budzianowski and Vulić, 2019) is pre-trained on large general-domain corpora and finetuned with a domain-specific corpus. As such system was built to be task-oriented, the corpus utilized for finetuning is the text-only MultiWOZ dataset (Budzianowski et al., 2018), composed by natural language conversations based on domain-specific vocabulary, such as requests for booking a restaurant or hotel. Results evaluation disclosed that around 70% of the times the system replied with an appropriate entity, such as hotel, regarding the user's request. Results also revealed that between 50% to 60% of the times the system was capable of including all the solicited attributes, the price range, for instance.

3.2 Discussion

The present work is focused on exploring several alternatives to the development of a Portuguese conversational agent capable of answering domain-related questions, and, as such, the analysis of distinct architectures performed in the current section served as a base guidance.

Although most referred systems communicate in English, the system to be developed is meant to communicate in Portuguese language and, as such, Information Retrieval-based systems such as Amaia (Santos et al., 2020) and Filipe (Ameixa et al., 2014) are considered for that purpose.

Corpus-based systems were deemed as substantially relevant, as they rely on domain to be able to interact with users. Both NLU platforms and several IR-based approaches will be further explored, as they present possible solutions to question answering, either based on FAQs or raw text. Likewise, transformer models proved to be be worth exploring, due to the considerably straightforward adaptation to new domains and the positive results obtained in the previously mentioned evaluations. Thus, the information gathered regarding such systems and architectures provides for an initial development and experimentation phase, described in detail in the next chapter.

This page is intentionally left blank.

Chapter 4

Architecture, Data and Configuration

This work explores a set of approaches for QA in Portuguese, in any given domain reflected in an available list of FAQs or collection of documents. Those include an NLU platform, Google Dialogflow, an IR-based approach, i.e., text search engine Whoosh, and two types of state-of-the-art transformer-based language models, namely BERT and answer-generating GPT2 and GPT3. Our goal is to assess the approaches regarding relevant aspects such as human labour, time consumption and answer quality.

Thus, we propose a solution where each approach is configured to receive a domain and answer questions related to it. In addition, every approach is submitted to a series of tests consisting of providing a domain, asking a set of previously created questions and registering the retrieved answers. Finally, the time consumed whilst retrieving the answers is recorded and the answers evaluated through evaluation metrics.

The upcoming subsections discuss the architecture of the proposed solution, along with a detailed description of the data used for both configuration and testing, and the relevant aspects of each approach's configuration. The final phases of experimentation and evaluation, as well as the results, are disclosed in the next chapter.

4.1 Architecture

Our solution focuses mainly on conducting the same set of experiences on the different approaches, depending on the domain type, that is, list of FAQs or collection of documents, and drawing conclusions based on the results.

To do so, we propose an architecture to follow for each approach when implementing our solution, displayed in Figure 4.1, that can be divided in several phases, namely: i) data preprocessing; ii) model configuration to question answering and adaptation to the domain; iii) experimentation, i.e., asking questions to the model; iv) retrieval and storage of answers and time consumption; v) evaluation.

The data forming the domain can prove not to be consistent, by being scattered amongst different format files, for instance. Due to this possible inconsistency, we decided to submit the data to a preprocessing phase, where all documents are converted to a plain text format. In addition, distinct identifiers, depending on the domain type, were added to the data. This phase is discussed in further detail in the upcoming subsection.

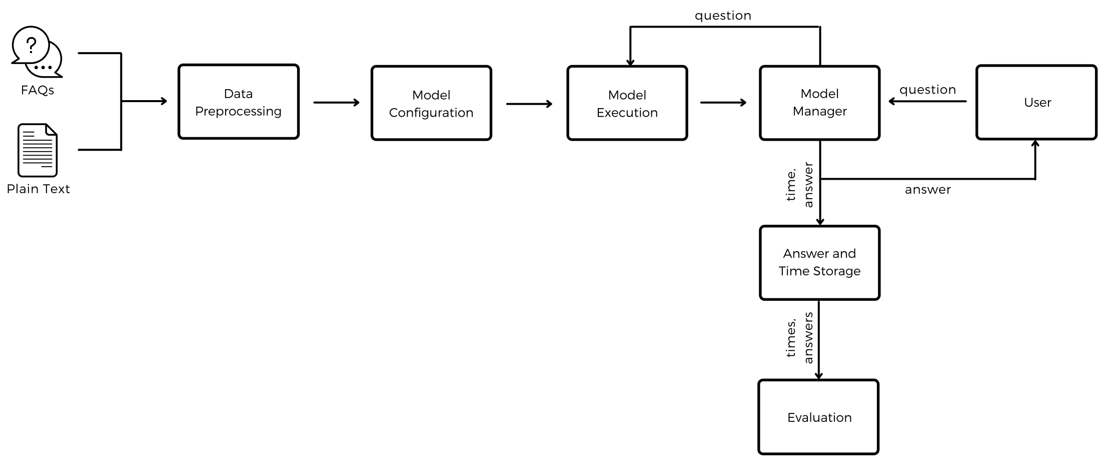
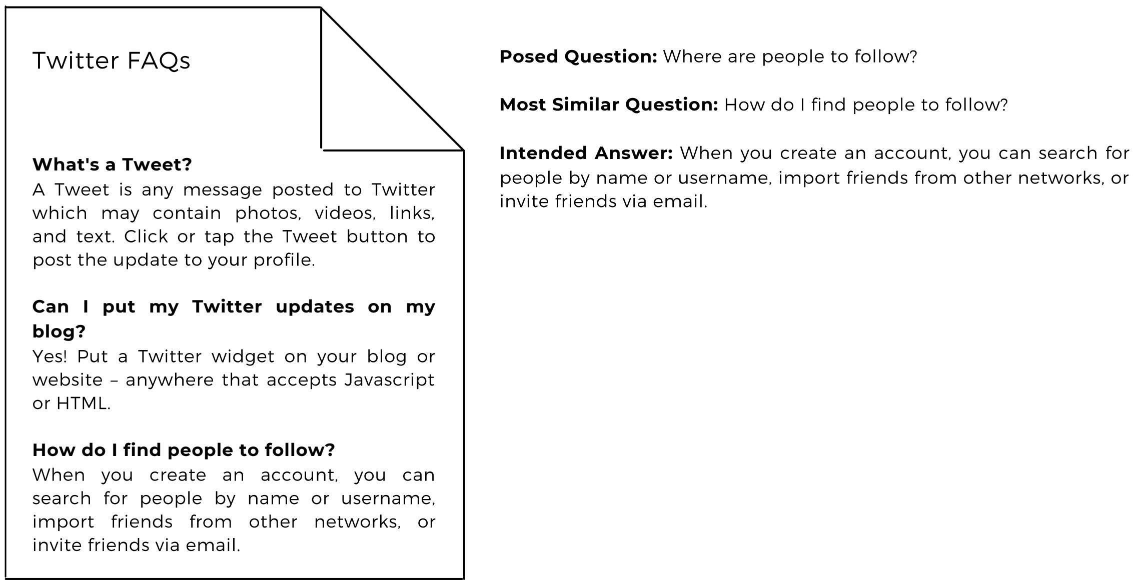


Figure 4.1: Proposed Solution Architecture

Each approach consists of one or a combination of models configured to answer questions regarding a given domain. Such configuration varies from model to model, ranging from indexation to fine-tuning.

When configured to answer questions based on a FAQs domain, the approach is meant to retrieve to the user as an answer the answer corresponding to the FAQ in the domain deemed as most similar to the posed question, whenever possible. An example, retrieved from Twitter's FAQs page[1], is present in Figure 4.2, where the intended answer would be the answer to the third FAQ, as the question posed is most similar to the same FAQ. However, as in answer-generating models this method of answer retrieval is not possible, we rely on fine-tuning the model to the domain and its generated answers.


Twitter FAQs

What's a Tweet?
A Tweet is any message posted to Twitter which may contain photos, videos, links, and text. Click or tap the Tweet button to post the update to your profile.

Can I put my Twitter updates on my blog?
Yes! Put a Twitter widget on your blog or website – anywhere that accepts Javascript or HTML.

How do I find people to follow?
When you create an account, you can search for people by name or username, import friends from other networks, or invite friends via email.

Posed Question: Where are people to follow?

Most Similar Question: How do I find people to follow?

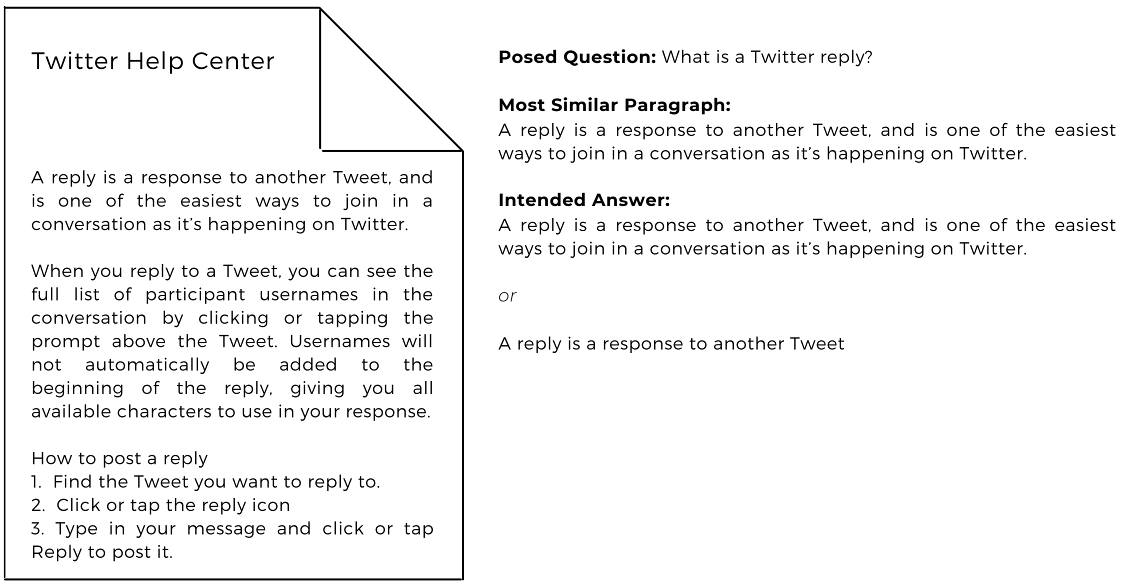
Intended Answer: When you create an account, you can search for people by name or username, import friends from other networks, or invite friends via email.

Figure 4.2: Example of Answer Retrieval in FAQs Domain

On the other hand, when configured to answer questions from a collection of documents, the approach is meant to retrieve the paragraph or an excerpt of it, out of all the paragraphs in the documents, deemed as most similar to the posed question, that is, the one deemed

[1]https://help.twitter.com/en/resources/new-user-faq

as most likely to contain the answer. An example, retrieved from Twitter's Help Center page[2], of answer retrieval in this type of domain is shown in Figure 4.3, where the first paragraph should be returned as an answer, due to it being considered as the most similar to the posed question. Similarly to FAQs, when the approach carries an answer-generating model, the method of paragraph retrieval is not feasible, and thus we once again rely on fine-tuning the model to the domain and the answers it generates.



Twitter Help Center

A reply is a response to another Tweet, and is one of the easiest ways to join in a conversation as it's happening on Twitter.

When you reply to a Tweet, you can see the full list of participant usernames in the conversation by clicking or tapping the prompt above the Tweet. Usernames will not automatically be added to the beginning of the reply, giving you all available characters to use in your response.

How to post a reply
1. Find the Tweet you want to reply to.
2. Click or tap the reply icon
3. Type in your message and click or tap Reply to post it.

Posed Question: What is a Twitter reply?

Most Similar Paragraph:
A reply is a response to another Tweet, and is one of the easiest ways to join in a conversation as it's happening on Twitter.

Intended Answer:
A reply is a response to another Tweet, and is one of the easiest ways to join in a conversation as it's happening on Twitter.

or

A reply is a response to another Tweet

Figure 4.3: Example of Answer Retrieval in Collection of Documents Domain

The configuration phase of each approach is further detailed in section 4.3.

With the configuration and adaptation to the domain completed, we can now enter the experimentation phase. This phase focuses on testing each approach on its ability to answer domain-related questions and associated time consumption, doing so by asking the approach a set of pre-defined questions and registering both the retrieved answers and measured time consumption. For each domain type there is a set of previously created questions, referred to in detail in the next subsection.

Prior to asking questions, the time consumed to adapt to the domain, e.g., fine-tuning, is measured. As questions are being made to the approach, the respective retrieved answers are stored for posterior evaluation. In addition, the time per answer retrieval is also measured. The approach is then submitted to evaluation, where the stored answers are passed through several evaluation metrics and the metrics results are compared along with the measured times.

4.2 Data

Since we have two types of domain, composed either by FAQs or a collection of documents, we need at least a Portuguese dataset per type to evaluate the selected approaches. Altice provided us with a collection of documents and a list of FAQs related to telecommunication equipment they supply, which we used in the experimentation phase of this work. In addition, to evaluate the approaches with a domain based on FAQs, we also utilized the

[2]https://help.twitter.com/en/using-twitter/twitter-conversations

most recent version of AIA-BDE corpus (Gonçalo Oliveira et al., 2020), a collection of FAQs from the Portuguese Administrative Modernization Agency web portal.

Note that, as the focus of this work lies in systems capable of adapting to any given Portuguese domain, whether a new one or an adjustment of the current, the domain used in experimentation could indeed be any set of textual documents or FAQs written in Portuguese.

4.2.1 FAQs

Upon gathering all the FAQs Altice provided us with, we were left with a total of 172 question-answer pairs. These pairs form a domain about Portuguese telecommunication equipment, containing answers to frequently asked questions related to problem solving, system features, and so on. In order to have domain consistency, all files containing FAQs and the respective answers were converted to a plain text format and put together in a single file.

With each approach receiving the full list of FAQs as the domain, we decided to create questions variations. For 103 questions, two variations were manually created, where the content of the question being asked remained the same, and therefore the answer as well, but the words differed. An example can be seen in Figure 4.4, where the highlighted question is the original and the subsequent two are the created variations. This allows for a fairer evaluation of each approach, as not all questions being made are part of the domain. Thus, we ended up with a total of 378 to test our approaches with.

```
Sistema Operativo Mac OSX: Como ligar a uma rede sem fios
Como fazer ligação a uma rede sem fios num Mac OSX?
Como ligar a uma rede sem fios com o sistema operativo mac OSX?
```

Figure 4.4: Example of Altice FAQs Question Variations

To distinguish a question and the respective answer from the others, identifiers were added to the beginning of each. A `P:` was added before each question in Portuguese and a `R:` before each answer in Portuguese, being that the corresponding answer to a question immediately follows it. The end of an answer is marked with a '\n\n' sequence of characters, corresponding to the final character of the answer joined with an empty line. An example of a question-answer pair with the identifiers is presented in Figure 4.5, where the identifiers are highlighted.

```
P: Sistema Operativo Mac OSX: Como ligar a uma rede sem fios
R: Procure as redes sem fios Clique no ícone de WiFi e procure as redes sem fios
disponíveis (...).
```

Figure 4.5: Example of Altice FAQs Question-Answer Pair with Identifiers

Besides the compilation of FAQs from Altice, we also thought it would be interesting to evaluate the selected approaches with an additional set of Portuguese FAQs. To do so, we used AIA-BDE[3], an available corpus created to be used in the evaluation of IR, QA, or task-oriented dialogue systems. This corpus is composed by 855 question-answer pairs related to the exercise of economic activity in Portugal.

In addition to the question-answer pairs retrieved from the Portuguese Administrative Modernization Agency web portal, AIA-BDE also includes close to 5,000 variations of the

[3]https://github.com/NLP-CISUC/AIA-BDE

original questions, providing an average of six variations per question. Variations can be divided into two groups: i) generated automatically with the help of Google Translate; ii) manually created by volunteers and crowdsourcers. An example of an original question and its variations can be seen in Figure 4.6, where the original question is highlighted and the following are its variations.

```
Por quanto tempo é válido o registo de um desenho ou modelo?
Quanto tempo é válido o registro de um desenho?
Quanto tempo é válido para o registro de um desenho?
Durante quanto tempo é válido o registo de um desenho ou modelo?
O registo de um desenho ou modelo permanece válido durante quanto tempo?
Qual a duração da validade do registo de um desenho ou modelo?
Qual é o período máximo de validade de um registo de desenho ou modelo?
Ao fim que quanto tempo é necessário fazer a renovação do registo de um desenho ou
modelo?
Qual é o período até a caducidade do registo de um desenho ou modelo?
Qual é o período de expiração do registo de um desenho ou modelo?
```

Figure 4.6: Example of AIA-BDE Question Variations

Similarly to Altice FAQs, AIA-BDE corpus identifies questions and answers with a `P:` before each question and a `R:` before each answer, and the ending of each answer is marked with the sequence '\n\n'. Each question is immediately followed by its variations and the respective answer, as presented in Figure 4.7, where the question and answer identifiers are highlighted.

```
P:Por quanto tempo é válido o registo de um desenho ou modelo?
VG1:Quanto tempo é válido o registro de um desenho?
VG2:Quanto tempo é válido para o registro de um desenho?
VIN:Durante quanto tempo é válido o registo de um desenho ou modelo?
VIN:O registo de um desenho ou modelo permanece válido durante quanto tempo?
VIN:Qual a duração da validade do registo de um desenho ou modelo?
VIN:Qual é o período máximo de validade de um registo de desenho ou modelo?
VIN:Ao fim que quanto tempo é necessário fazer a renovação do registo de um desenho
ou modelo?
VIN:Qual é o período até a caducidade do registo de um desenho ou modelo?
VIN:Qual é o período de expiração do registo de um desenho ou modelo?
R:A duração do registo é de cinco anos a contar da data do pedido, podendo ser
renovada, por períodos iguais, até ao limite de 25 anos.
```

Figure 4.7: Example of AIA-BDE Question-Answer Pair with Identifiers

4.2.2 Text

From Altice's data contribution, we compiled a collection of 25 Portuguese documents about telecommunication equipment. The compiled documents form a domain, where each document contains information about a particular piece of equipment such as specifications, content, user guide, or problem resolution. For the purpose of having domain consistency, all documents were converted from PDF to a plain text format.

Since there are similar pieces of equipment, some documents tend to have a similar structure. For instance, some refer to very similar pieces of equipment, with the only difference between each document being the equipment's model or minor characteristics. Thus, to minimise confusion, the filename of each document, often the name of the equipment, was automatically added to every paragraph in the document, where the end of a paragraph is represented by a '\n\n' character sequence. An example is shown in Figure 4.8, where the identifier is highlighted.

```
Placa ZTE MF65 - O utilitário (ZGPatchForEcmDriverV1.0.6.pkg)
1. Sistema Operativo: OS X Yosemite (10.10).
2. Equipamentos: ZTE MF667, ZTE MF63, ZTE MF65.
3. Sintomas: Depois de instalar o equipamento, o mesmo não é detectado.
4. Solução: Instalar o utilitário fornecido pela ZTE de acordo com os passos descritos
neste guia.

Placa ZTE MF65 - Requisitos:
1. Computador ligado à corrente, para garantir um melhor desempenho.
2. Software do equipamento ZTE (router/placa de dados) instalado no computador.
3. Equipamento ZTE (Router/Placa de dados) desligado do computador.

Placa ZTE MF65 - Detalhes de Instalação:
1. Caso o requisito 1 não seja verificado, ligue o carregador do computador à corrente.
2. Caso o requisito 2 não seja verificado, instalar o Software do equipamento ZTE
(router/placa de dados), tal como indicado no manual de utilizador do mesmo, antes de
efectuar os passos seguintes.
```

Figure 4.8: Example of Altice Document with Identifier

Then, we manually created a set of 67 questions to be answered with information in the collection, and mapped each question to the paragraph that would provide its answer. There are two or more questions per document, and each one was conceived by reading a document and forming a question to which the answer lies in a specific paragraph. Moreover, for identifying questions and their answers, a `P:` was added before each question and a `R:` to each answer, which follows its question immediately. An example is presented in Figure 4.9, where the question and respective answer identifiers are highlighted.

```
P: Quais os requisitos para montar a placa ZTE MF65?
R: Placa ZTE MF65 - Requisitos:
1. Computador ligado à corrente, para garantir um melhor desempenho.
2. Software do equipamento ZTE (router/placa de dados) instalado no computador.
3. Equipamento ZTE (Router/Placa de dados) desligado do computador.
```

Figure 4.9: Example of Question and Corresponding Answer with Identifier from Altice Document

Note that, as the creation of the questions was one of the first steps of the implementation of this work, at that time we did not possess substantial knowledge about all the approaches we would adopt and how they work. Therefore, the created questions can not be considered as favorable to obtaining better results when testing and evaluating the different approaches.

4.3 Configuration

Prior to the experimentation phase, the approaches must be prepared to receive a domain and answer questions related to that same domain. Out of the currently available approaches, we selected a few that we considered relevant, those being: the NLU platform Dialogflow, IR-based text search engine Whoosh and transformer-based language models BERT, GPT-2 and GPT-3.

Each approach, with exception to GPT-3, was configured to answer questions in the two types of domain we have, i.e., composed either by FAQs or a collection of documents, with some having more than one configuration per domain type. GPT-3 could only be configured to answer questions related to a domain composed by a collection of documents, due to the usage limitation it posed. All of the configurations were implemented in python.

4.3.1 Dialogflow

We configured Google Dialogflow to answer questions in three different ways: i) through the creation of intents based on FAQs; ii) through the creation of a FAQs knowledge base; iii) through the creation of an extractive QA knowledge base, composed by a collection of documents. Figure 4.10 depicts the multiple configurations. Regardless of the configuration type, the usage of this platform is done through API requests[4].

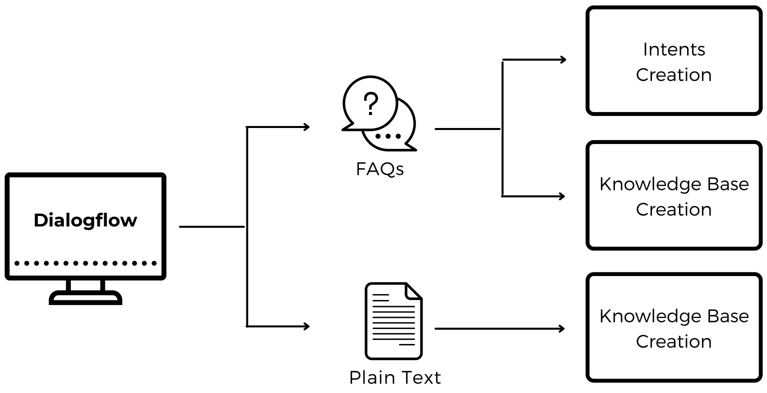


Figure 4.10: Different Configurations for NLU Platform Google Dialogflow

Prior to any configuration, this approach requires the creation of a project in Google Cloud Platform[5], followed by the creation of an agent through the Dialogflow platform. After creating a project we are given a project ID, which allows us to perform API calls and, consequently, communicate with the associated agent, as a project can only have one. When creating an agent, we associate it to the previously created project.

The same project and, therefore, the same agent, were utilized for all the different configurations, as it is possible to use one at a time, by disabling the others. It could, however, be created a project and agent per configuration, if desired.

Intents Classification

The most common way of using agents in NLU platforms is through the creation of intents, instances that categorize the intention in a user's input, for one conversation turn. When a user's input is received, the agent performs intent classification, where it matches the input to the most similar intent it contains, retrieving the answer associated to that intent.

To create a simple intent, a name, training phrases and respective answers are required. The training phrases are what is matched with the user's input during intent classification, and the answers are the reply given to the user when an intent is matched. An example of intent matching is presented in Figure 4.11, where the returned answer corresponds to the one associated with the most similar training phrase. We created one intent per FAQ, where each contained the question as the training phrase and the respective answer as the intent reply.

[4]https://dialogflow.cloud.google.com/
[5]https://cloud.google.com/

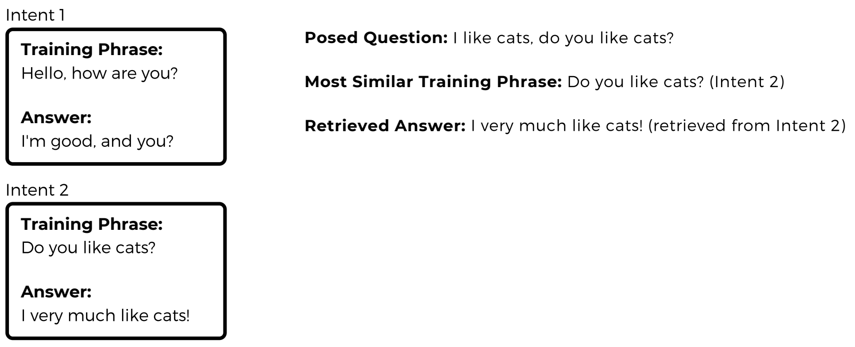

Intent 1
Training Phrase:
Hello, how are you?
Answer:
I'm good, and you?
Intent 2
Training Phrase:
Do you like cats?
Answer:
I very much like cats!
Posed Question: I like cats, do you like cats?
Most Similar Training Phrase: Do you like cats? (Intent 2)
Retrieved Answer: I very much like cats! (retrieved from Intent 2)

Figure 4.11: Google Dialogflow Intents Classification Example

Since we have a single file containing all FAQs as the domain, we divided the question-answer pairs into two lists, one with all the questions and one with all the answers. This way, we were left with the sets of training phrases and answers separated, which allowed us to create all the intents at once. Through API calls, the agent receives a question and an answer at a time, creating a new intent with the received information. Due to the number of API requests being limited per minute, we defined a waiting time of thirty seconds after every ten intents created. The API requests allow to add, list or delete specific intents, but do not allow to perform alterations on an intent, which can only be done through the Dialogflow platform.

With the intents created, the agent is now ready to answer questions. To do so, we open a session in the project we previously created and send the question through an API request, which returns the agent's answer. As the domain is composed by Portuguese FAQs, we also specified the agent's language as Portuguese, to hopefully obtain the best results in the intent classification phase. All questions asked and the respective retrieved answers are registered and stored in a text file, along with the `P:` and `R:` identifiers.

Knowledge Base

Prior to configuring a knowledge base, as it is still in beta mode, it is necessary to access once again the Dialogflow platform and select the option that allows the use of beta features. To create a knowledge base, we need a name and one or more documents to add to it. When creating and adding a document to a knowledge base, the document type defines the type of match to be made with the user's input.

For instance, if the document contains a list of question-answer pairs, then the type must be *FAQS*, and the knowledge base will perform the search for the answer based on that information. An example of answer search is presented in Figure 4.12, where the returned answer corresponds to the one associated with the most similar question in the FAQs knowledge base. Similarly, if the document is composed by plain text, then the type must be *EXTRACTIVE_ QA*, and the knowledge base will once again try to better match the user's input and reply based on that information.

We created a knowledge base per domain, through API requests to the agent we originally defined. The FAQs knowledge bases were each composed by a single document, of *FAQS* type, containing all the question-answer pairs in the domain. The collection of documents knowledge base is also composed by a single document, an aggregation of all the plain text documents in the domain, of *EXTRACTIVE_ QA* type. If desired, at any point in time a new document can be added to the knowledge base, as well as alterations

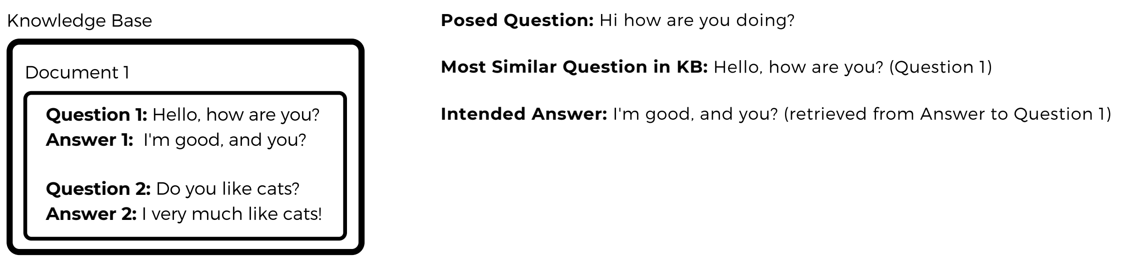

Knowledge Base
Document 1
Question 1: Hello, how are you?
Answer 1: I'm good, and you?
Question 2: Do you like cats?
Answer 2: I very much like cats!
Posed Question: Hi how are you doing?
Most Similar Question in KB: Hello, how are you? (Question 1)
Intended Answer: I'm good, and you? (retrieved from Answer to Question 1)

Figure 4.12: Google Dialogflow Knowledge Base Answer Search Example

can be made to the content of an already added document.

Likewise to the intents configuration, to retrieve answers from the agent we open a session in the previously created project and send the question through an API request, which returns the agent's answer. All questions asked and the respective retrieved answers are registered and stored in a text file, along with the `P:` and `R:` identifiers. Unfortunately, as knowledge bases are still a beta feature, the only language the agent allows to be set is English, thus complicating the input matching task.

A brief summary of the followed steps for each configuration performed for this approach is presented in Figure 4.13.

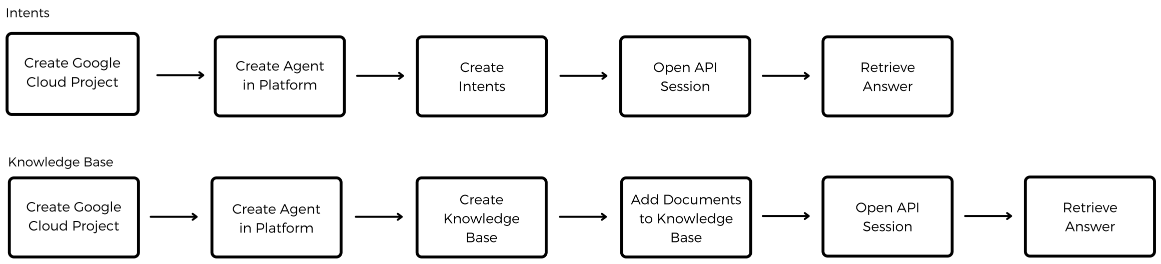


Figure 4.13: Configuration Steps Summary for NLU Platform Google Dialogflow

4.3.2 Whoosh

Following a traditional IR approach, Whoosh[6] is a python library that allows to index text and find matching documents based on a given search string. To rank each document, Whoosh uses the BM25F (Robertson and Zaragoza, 2009) ranking function. Even though it was not built for QA, if the search terms are in the form of a question, Whoosh will find the document or documents with the most similar text snippet.

When defining the search string, it is possible to adapt this approach to both domain types, i.e., composed by FAQs or by a collection of documents. For the first type, we configured Whoosh to perform two distinct searches, one based on the questions in the domain, done by comparing them to the posed question; and another based on both the questions and answers in the domain, which compares the posed question with the question-answer pairs. For the collection of documents domain, we configured this approach to search based on the documents content. Figure 4.14 depicts the multiple configurations.

Apart from defining the search string, this IR-based approach supports the definition of different options, such as selecting the token analyzer or specific filters. Thus, we decided to

[6]https://whoosh.readthedocs.io/

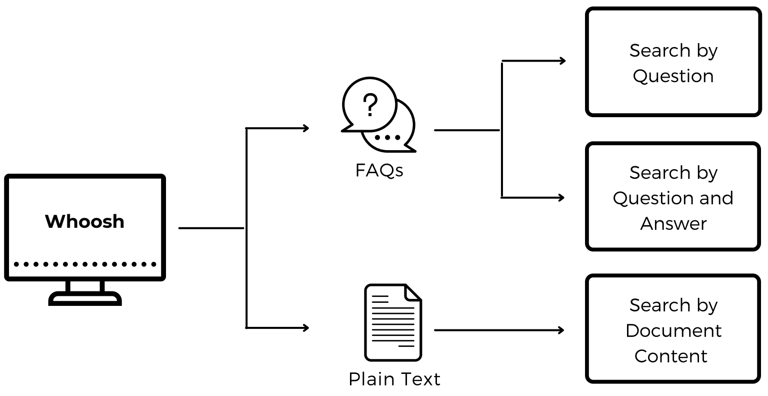


Figure 4.14: Different Configurations for Search Engine Whoosh

configure Whoosh with different combinations, namely: i) the default configuration, where no alterations were made; ii) altering the analyzer to a Portuguese language analyzer, that converts words to lower-case, removes Portuguese stop words, and converts words to their stem, following Portuguese rules; iii) adding to the analyzer 3-gram and 4-gram filters, which break individual tokens into groups of, respectively, three and four characters when searching for the most relevant document.

The first configuration step, common to all combinations, is the creation of an index object, followed by the definition of its schema. The schema specifies fields that contain information about each document in the index, such as the title or text content. These fields can then be indexed, i.e., searched, and/or stored, allowing them to later be returned with the results.

Fields can also have types that define what is being indexed and what is searchable, e.g., *ID*, that indexes the entire value of the field in a single unit; *TEXT*, used for text content as it indexes and optionally stores the text, as well as stores the terms position to allow for phrase searching; *NGRAM*, that breaks the field text into N-grams.

With the fields in the schema defined, we can create the index. This is done by creating an object that contains the index and stores it as a set of files in a given directory. Afterwords, the final step before being able to search, is to add documents to the index object. To add a document, we use the writer method provided by the index object, which receives the previously specified schema fields as arguments.

Finally, we can create a searcher object and search the index to retrieve results. The searcher object determines what fields are searched through a query object, and returns a results object containing a list of dictionaries, where each is composed by the stored fields of a document. The dictionaries position in the list determines its position in the results, i.e., the closest to the beginning, the best scored the document.

When creating the query object to parse a query string, we added the *OrGroup* query class, making terms on the search field optional by default. We believed the addition of this class to be relevant, as the terms in the posed question are highly unlikely to exactly match the terms in the domain questions. This object receives as input the fields to search, the index's schema to be searched, and, in our case, the query class. Having a searcher and query objects, we now use the search method and obtain a results object.

Considering the mentioned steps as the main Whoosh configuration to retrieve the most similar document to a posed question, we made several adjustments to the configuration depending on the type of domain.

Question and Answer Search

Since Whoosh retrieves as result a list of documents and we have as a domain a single document containing a set of FAQs, the first step when configuring this approach was to divide such domain in multiple documents, each composed by one question-answer pair. This way, by returning a list of documents, Whoosh would be returning a list of the most similar FAQs to the posed question. Thus, we could then retrieve the answer of the FAQ deemed as most similar as an answer to the posed question. An example is shown in Figure 4.15, where Whoosh retrieves the answer in the document containing the question most similar to the posed one.

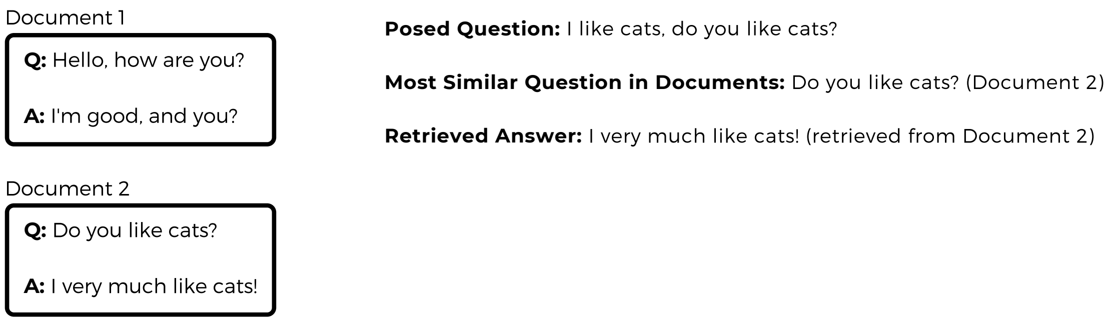

Document 1
Q: Hello, how are you?
A: I'm good, and you?
Document 2
Q: Do you like cats?
A: I very much like cats!
Posed Question: I like cats, do you like cats?
Most Similar Question in Documents: Do you like cats? (Document 2)
Retrieved Answer: I very much like cats! (retrieved from Document 2)

Figure 4.15: Whoosh FAQs Answer Retrieval Example

In addition to dividing the domain through several documents, we searched for the most similar document in two different ways: considering only the similarity between the posed question and the question field in the documents, and considering the similarity between the posed question and both the question and answer fields in the documents. To do so, we start by creating the index and its schema, by defining four fields: the name of the document, which we store and classify as *TEXT*; the path to the file, which we classify as an *ID*; the question in the document, also classified as *TEXT*; and the answer in the document, similarly classified as *TEXT*. When configuring the different combinations, the analyzer is added to the question field, and the N-gram filter is applied by changing the question field type from *TEXT* to *NGRAM*.

Subsequently to creating the schema and defining the fields, we added our collection of FAQs documents to the index and performed the two types of search, returning as a result the top scored document information. As we solely stored the documents name, we get as a result the name of the document considered to be the most similar to the posed question. Hence, we use this information to retrieve from that same document the answer it contains, returning it as the answer provided by the approach.

Once again, all questions asked and the retrieved answers are registered and stored in a text file, along with the `P:` and `R:` identifiers.

Content Search and Highlights

Having a domain composed by a collection of documents, we searched for the most similar document by considering the similarity between the posed question and the documents content. Once again, we created the index and its schema, by defining three fields: the name of the document, which we store and classify as *TEXT*; the path to the file, which we classify as an *ID*; the text content of the document, which we also store and classify as *TEXT*. This time, when configuring the different combinations, the analyzer is added to the content field, and the N-gram filter is applied by changing the same field type from *TEXT* to *NGRAM*.

After adding the documents in the domain to the index, whilst specifying the respective fields, and performing the search for the most similar document, we receive as a result the name and content of the top-scored document. Instead of returning as an answer the name of the document or its whole content, we made use of a feature Whoosh provided, called highlights.

When a document is scored based on similarity to the posed question, Whoosh highlights the text snippet most similar to the question, found in the field(s) used to search. Thus, in addition to the name and content of the top-scored document, the highlighted text span in the content field is also returned, which, according to our intuition, will often work out as an answer. An example is shown in Figure 4.16, where Whoosh retrieves the highlighted spans of text, from the document most similar to the posed question, as an answer. Considering the highlighted text as the approach's answer to posed questions, all questions and retrieved answers are stored in a text file, as well as the identifiers.

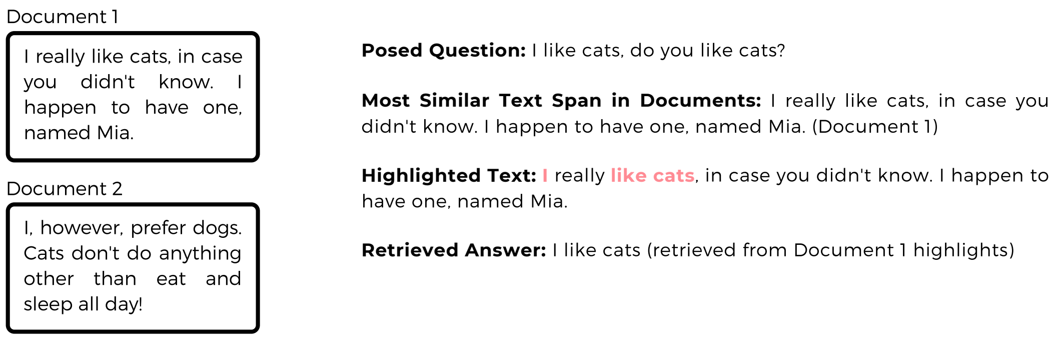

Document 1
I really like cats, in case you didn't know. I happen to have one, named Mia.
Document 2
I, however, prefer dogs. Cats don't do anything other than eat and sleep all day!
Posed Question: I like cats, do you like cats?
Most Similar Text Span in Documents: I really like cats, in case you didn't know. I happen to have one, named Mia. (Document 1)
Highlighted Text: I really like cats, in case you didn't know. I happen to have one, named Mia.
Retrieved Answer: I like cats (retrieved from Document 1 highlights)

Figure 4.16: Whoosh Highlights Answer Retrieval Example

A brief summary of the followed steps for the configurations performed for this approach is presented in Figure 4.17.

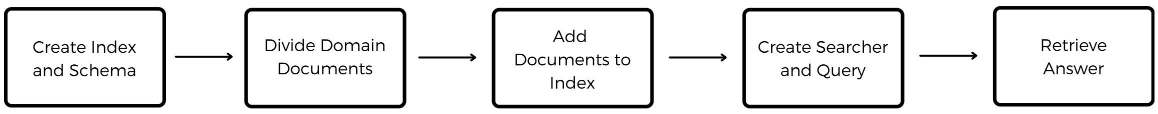


Figure 4.17: Configuration Steps Summary for Search Engine Whoosh

4.3.3 BERT

BERT's architecture allows it to be fine-tuned to perform different tasks, e.g., question answering, through the addition of an output layer, without the need of performing significant architecture modifications. To explore this approach, we used BERTimbau (Souza et al., 2020), a BERT model pre-trained on a large Portuguese Corpus, BrWaC (Wagner Filho et al., 2018).

In addition, we also used Portuguese BERT base cased QA (Guillou, 2021), a version of BERTimbau fine-tuned with the Portuguese SQUAD v.1.1 dataset[7]. BERTimbau QA,

[7]http://www.deeplearningbrasil.com.br/

given a textual context and a question, returns the span of text that better answers the question. An example is depicted in Figure 4.18.

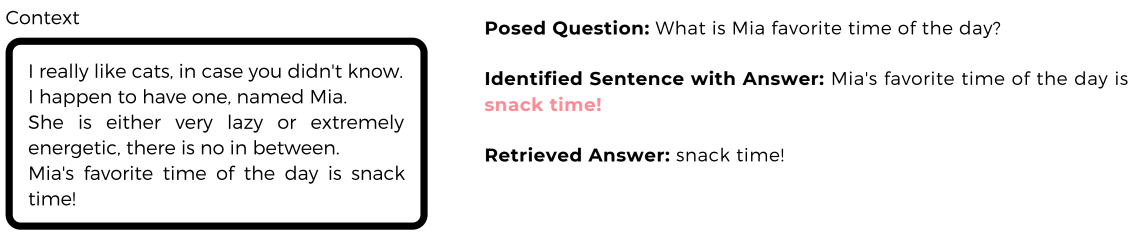

Context
I really like cats, in case you didn't know. I happen to have one, named Mia. She is either very lazy or extremely energetic, there is no in between. Mia's favorite time of the day is snack time!
Posed Question: What is Mia favorite time of the day?
Identified Sentence with Answer: Mia's favorite time of the day is snack time!
Retrieved Answer: snack time!

Figure 4.18: BERTimbau QA Answer Retrieval Example

Thus, we configured BERTimbau and BERTimbau QA to answer questions given a FAQs or collection of documents as context. In order to implement the BERT models, we used the *transformers* package and the available models from HuggingFace[8]. Figure 4.19 shows all the configurations conducted for BERT, detailed below.

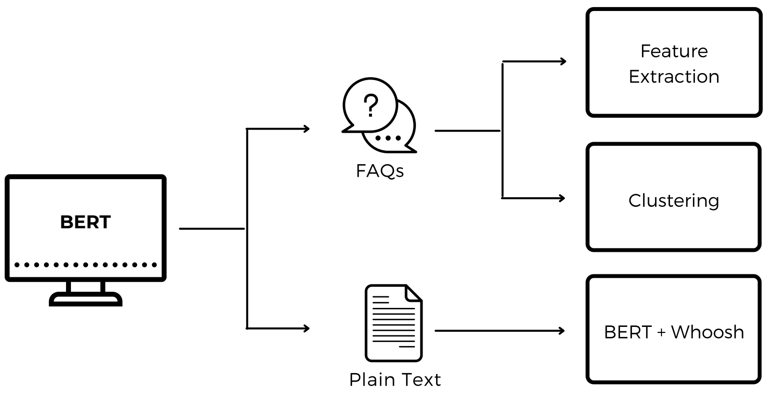


Figure 4.19: Different Configurations for BERT Model

The first configuration we made focused on calculating the similarity between the questions in the FAQs domain and the posed question. To do so, we used BERTimbau to get the vector representation of all questions, and then computed the similarity by calculating the cosine between the posed question and each question in the domain.

The larger in size the context given to BERTimbau QA, the more time required for the model to find the best text excerpt. Whilst configuring, we reached the conclusion that passing the whole domain as context to BERTimbau QA was unfeasible. Thus, we came up with two ways of making a pre-selection of the domain segment to be used as context, namely: clustering for creating groups of similar FAQs; a relevant document pre-selection made by Whoosh.

Feature Extraction

For FAQs, we used the feature-extraction pipeline of BERTimbau[9] to get the representations of all questions in the domain and of the posed question. Then, we computed the similarity between the posed question and each domain question by calculating the cosine between their vector representations.

[8]https://huggingface.co/
[9]https://huggingface.co/neuralmind/bert-base-portuguese-cased

This way, we could find the question in the domain most similar to the one posed, which lead us to the answer, as we retrieved the associated FAQ answer as the answer to the posed question.

Clustering

For the first pre-selection method, we created a k-means clustering (MacQueen et al., 1967) dividing all FAQs in the domain in a given number of clusters. The number of clusters depended on the number of question-answer pairs in the domain, where the greater the number of FAQs, the more clusters created.

Then, we used BERTimbau to get the representation of the posed question and compared it to the centroids in each cluster, based on similarity. Once again, by calculating the cosine between their vector representations. Having the most similar centroid, we used the respective cluster as context in BERTimbau QA[10]. BERT would then receive as context the given cluster and the posed question, retrieving an answer.

A brief summary of the followed steps for the feature extraction and clustering configurations performed for this approach is presented in Figure 4.20.

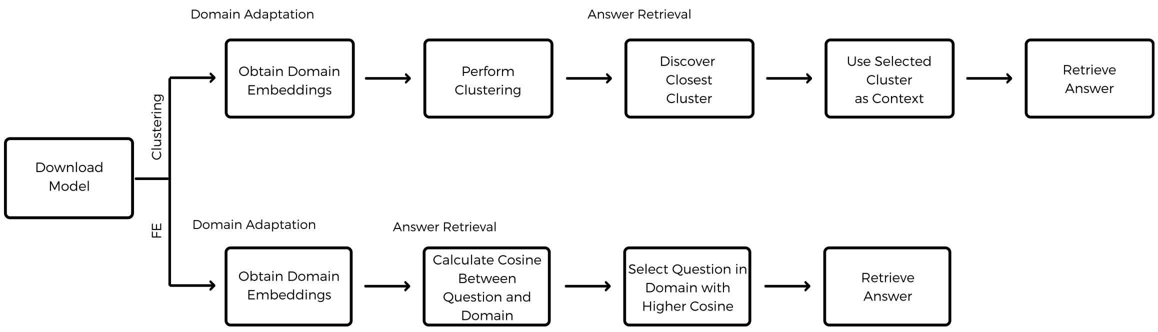


Figure 4.20: Configuration Steps Summary for Feature Extraction and Clustering with BERT Model

BERT + Whoosh

The combination of BERT and Whoosh used the previously configured Whoosh index as starting point. As in the simple Whoosh, the search was made using the question as a search string.

However, in this case, Whoosh simply retrieves the most relevant document(s), then given to BERTimbau QA as a context for extracting the answer to the question. Whoosh could be used for retrieving more than one document, thus we decided to retrieve one, three or five most relevant documents and use each as a context, for the same posed question.

Finally, BERTimbau QA would retrieve an answer to the posed question based on the documents deemed by Whoosh as most similar. For all three configurations, all the asked questions and retrieved answers are then registered and stored in a text file, along with the `P:` and `R:` identifiers.

[10]https://huggingface.co/pierreguillou/bert-base-cased-squad-v1.1-portuguese

A brief summary of the followed steps for the configuration of the BERT and Whoosh combination is shown in Figure 4.21.

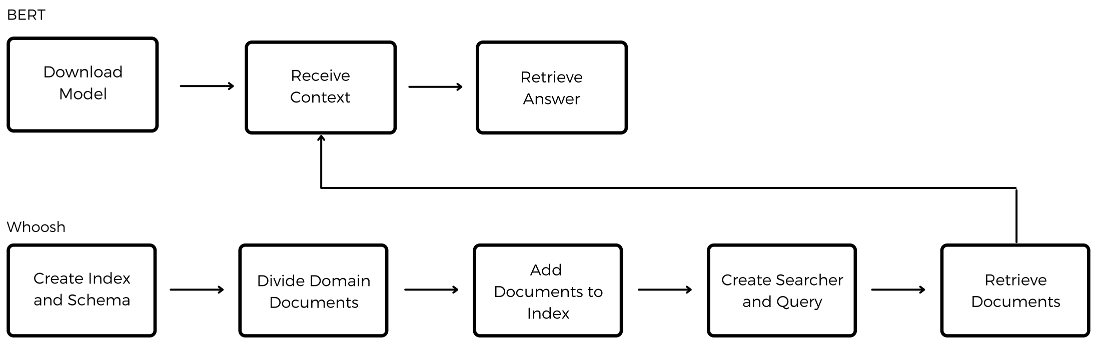


Figure 4.21: Configuration Steps Summary for Whoosh + BERT Combination

4.3.4 GPT-2

Out of the selected approaches, GPT-2 was set as a baseline for our answer generating problem. Even though not trained to do so, this transformer-based language model can be used for several NLP tasks, such as question answering, machine translation, reading comprehension or summarization. This is possible due to the model being trained on a very large dataset, which allows for it to learn tasks without any architecture modification (Radford et al., 2019).

In order to configure GPT-2 to the desired task, the model can be fine-tuned with text data and hyperparameters can be defined. Thus, as we were interested in configuring GPT-2 to the task of QA, we decided to create two configurations that allowed for the model to be fine-tuned with both domain types we explore, as well as answer questions related to the given domain. Figure 4.22 shows the model configurations.

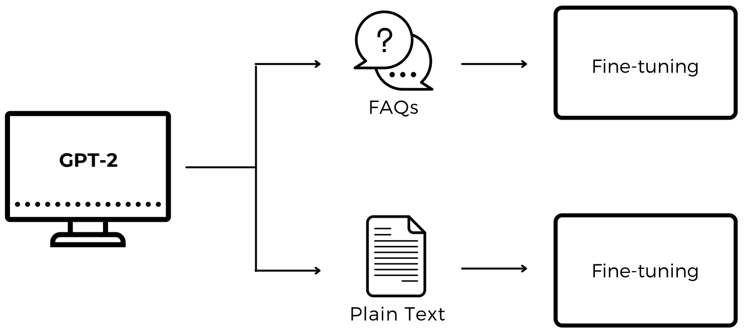


Figure 4.22: Different Configurations for GPT-2 Model

Apart from fine-tuning the model to each domain, we empirically chose a set of hyperparameters that would affect the answer generation, namely: temperature, which measures text generation randomness from 0 to 1; prefix, which defines the prefix to the generation and can be included in the answer or not; number of samples, that represents the number of answers generated and retrieved by the model; truncation token, that upon being reached stops the answer generation and the answer is retrieved.

GPT-2 has different sizes, each with a different number of architecture hyperparameters, where the larger the model, the higher the number of hyperparameters and, supposedly, the better the performance. However, along with the increase in size rises time and resources consumption. Therefore, we decided to use the medium sized model[11], containing 355M hyperparameters, as the smaller model was equivalent to the original GPT.

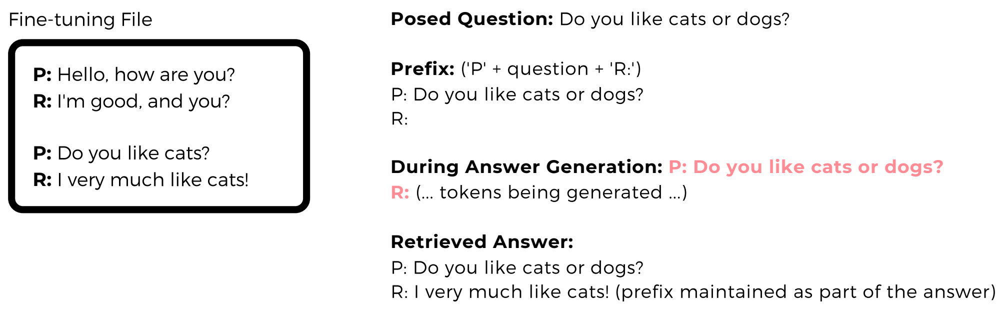
Fine-tuning

The fine-tuning configuration process was very similar for both domain types, with the only difference being the file used to fine-tune the model with. For the FAQs domain, we used a file containing all FAQs, also containing the `P:` and `R:` identifiers. As for the collection of documents domain, the fine-tuning file is a single document containing an aggregation of the content of all the domain documents.

In order to fine-tune GPT-2, we created a session where we defined the file to be used and a maximum of a thousand training steps per run. We also defined a run name for the session, which allowed us to stop fine-tuning the model and return to the process whenever desired. The session run name also allowed to perform fine-tune beyond the thousand training steps, by repeating the process whilst starting from the last step.

After fine-tuning GPT-2 to the domain, we defined the hyperparameters for the answer generation process, according to each domain type. When dealing with FAQs, our goal was to lead the model into replying to the posed question with a notion of the domain being composed by question-answer pairs, that is, understanding that it is meant to reply with an answer similar to what exists in the domain.

Hence, we started by creating a prefix composed by the identifier `P:`, the posed question and the identifier `R:`, in that order. The various steps of the prefix creation, addition and answer generation and retrieval are shown in an example in Figure 4.23. We also set the parameter of including the prefix in the answer to true, so GPT-2 can begin generating text from `R:` accordingly to the domain, instead of from scratch.



Fine-tuning File

P: Hello, how are you?
R: I'm good, and you?

P: Do you like cats?
R: I very much like cats!

Posed Question: Do you like cats or dogs?

Prefix: ('P' + question + 'R:')
P: Do you like cats or dogs?
R:

During Answer Generation: **P: Do you like cats or dogs?**
R: (... tokens being generated ...)

Retrieved Answer:
P: Do you like cats or dogs?
R: I very much like cats! (prefix maintained as part of the answer)

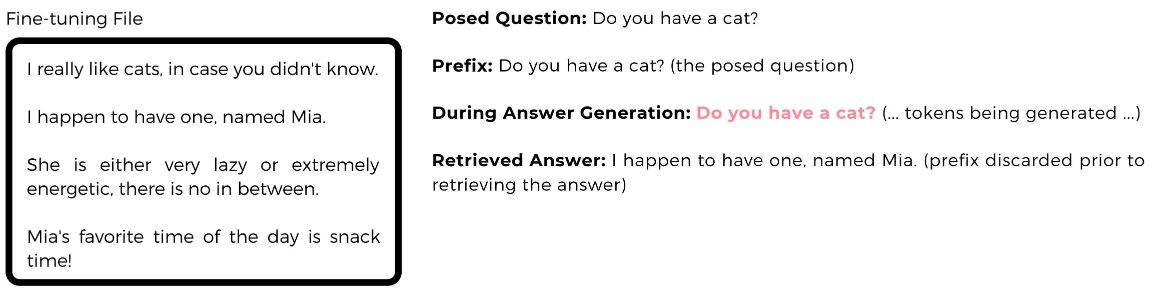
Figure 4.23: GPT-2 FAQs Domain Answer Generation Example

In addition, we define '\n\n' as the truncation token, because it marks the separation between two question-answer pairs, and set the number of samples to one, as we only want one answer per question. Finally, we set the temperature value to 0.2, thus avoiding highly random text generation. Since we add both identifiers in the prefix and include it in the answer, after having asked all questions and retrieved the answers, these are instantly registered and stored in a text file.

For a domain composed by a collection of documents, we wanted to lead the model into generating an answer similar to a paragraph in the domain, a span of text that contains the answer to the posed question.

[11]https://github.com/minimaxir/gpt-2-simple

However, as in this case there is no particular structure to the domain besides being divided by the token sequence '\n\n', we had to rely on defining the posed question as a prefix to the answer generation. Despite having the question as a prefix, this time we did not include it in the generated answer, i.e., the prefix is still used to target the beginning token sequences but is discarded while generating, thus not appearing in the retrieved answer. The various steps of the prefix creation, answer generation, prefix elimination and answer retrieval are exemplified in Figure 4.24. and followed by the prefix.



Fine-tuning File

I really like cats, in case you didn't know.

I happen to have one, named Mia.

She is either very lazy or extremely energetic, there is no in between.

Mia's favorite time of the day is snack time!

Posed Question: Do you have a cat?

Prefix: Do you have a cat? (the posed question)

During Answer Generation: **Do you have a cat?** (... tokens being generated ...)

Retrieved Answer: I happen to have one, named Mia. (prefix discarded prior to retrieving the answer)

Figure 4.24: GPT-2 Collection of Documents Domain Answer Generation Example

Similarly to the previous domain type configuration, we define '\n\n' as the truncation token, because it marks the separation between two paragraphs, and set the number of samples to one, as we only want one answer per question. Finally, we set the temperature value to 0.2, once again to avoid highly random text generation. Having retrieved all the answers, we add identifiers to all posed questions and respective answers and store them in a text file.

It is important to mention that, as GPT-2 was trained on English data, even though we are fine-tuning the model with Portuguese text, it might not be enough to fully teach it Portuguese. Thus, we consider the possibility of some generated text not making the most sense.

A brief summary of the followed steps for the configurations performed for this approach is presented in Figure 4.25.

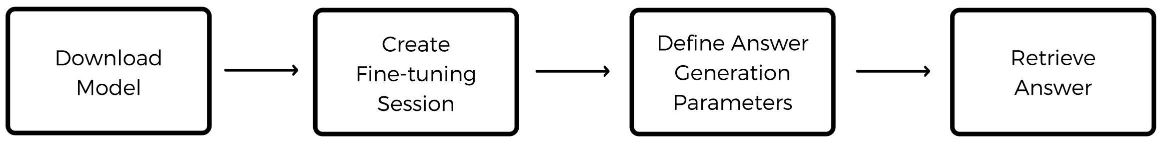

Download Model
Create Fine-tuning Session
Define Answer Generation Parameters
Retrieve Answer

Figure 4.25: Configuration Steps Summary for GPT-2 Model

4.3.5 GPT-3

GPT-3 (Brown et al., 2020) is an upgraded version of GPT-2, currently only available through OpenAI's API[12]. Instead of downloading the model to make use of it, the API manages all GPT-3 usage. The simplest way to configure this approach is to use a completion endpoint, where we give it a text input as a prompt and a set of examples, then used by the model to generate a text completion. While generating text, the model attempts to match the context or pattern given in the examples. Apart from the completion endpoint,

[12]https://openai.com/blog/openai-api/

the API offers different and more structured ones, ranging from semantic search to question answering.

In addition, OpenAI's API also allows to use different sized models. Similarly to GPT-2, the models vary in the number of architecture hyperparameters, having Ada as the smallest model with 2.7B and Davinci as the largest, with 175B hyperparameters. Once again, the larger the model, the better the performance, with time and resources consumption rising proportionally.

Due to being still in beta version, currently the API is not available to the general public, thus requiring the submission of a formal request and the acceptance by the OpenAI team, in order to have access. Although we were granted access to the API, we were given a restricted quota to perform API calls, which lead to a very limited configuration of the model, as each token generated came with an associated cost.

We were able to configure the model to answer questions related to a domain composed by a collection of documents, but, unfortunately, not to a FAQs domain, as there was no remaining quota. The sole configuration is presented in Figure 4.26.

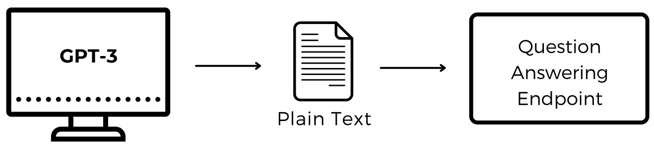


Figure 4.26: Sole Configuration for GPT-3 Model

Question Answering Endpoint

In order to configure the model to answer questions, we used the API's question answering endpoint[13], deemed as useful by the OpenAI team for applications that require high accuracy text generations based on sources, such as documentation or knowledge bases. This endpoint accepts as context a list of up to two hundred documents or a pre-uploaded file, that allows to go beyond that limit.

Figure 4.27, retrieved from OpenAI's documentation, describes the endpoint behaviour. The endpoint receives as input a question and a context, formed by either a collection of text documents or by a JSONL file, where each line contains a "text" field.

After providing the question and context, a two-step search is performed to find the best document or field in the context to answer the posed question, where: i) the number of relevant documents is narrowed to a given number, that, if not specified, defaults to 200; ii) the remaining documents are ranked according to their semantic relevance to the question. Finally, the API retrieves the answer, generated from the documents deemed as most relevant.

To configure the QA endpoint to better suit our purpose, we started by transforming the text documents in the domain to searchable documents, i.e., diving the documents in paragraphs and inserting each one into a line of a JSONL file, identified by the "text" field. This way, instead of selecting and ranking the most semantically relevant documents to the question, the API selected the most relevant paragraphs, thus further narrowing the context to answer the posed question.

After creating the context file, we query the question answering endpoint by defining

[13]https://beta.openai.com/docs/guides/answers

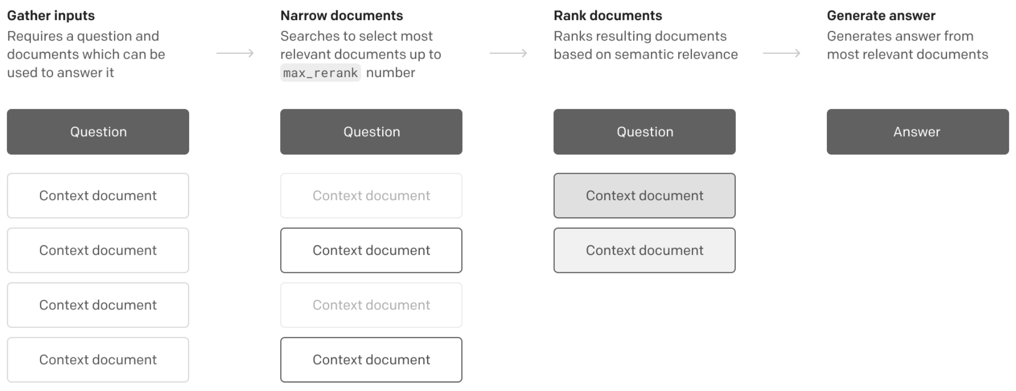
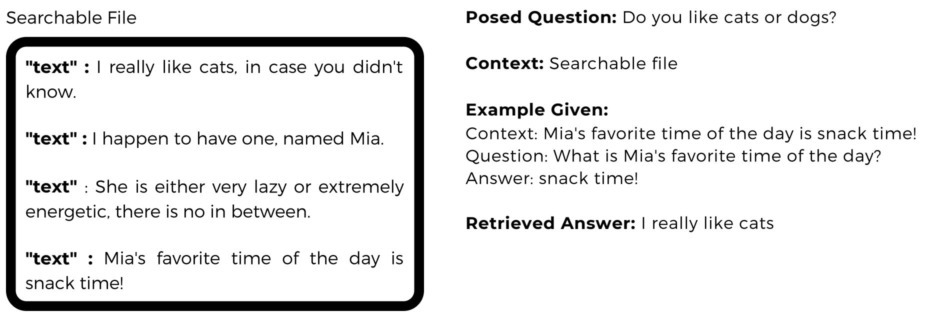

Gather inputs
Requires a question and documents which can be used to answer it
Narrow documents
Searches to select most relevant documents up to max_rerank number
Rank documents
Ranks resulting documents based on semantic relevance
Generate answer
Generates answer from most relevant documents
Question
Context document
Answer

Figure 4.27: OpenAI API Answers Endpoint Behaviour

several parameters, namely: the search model; the generating model; the question to be posed; the context file; some examples of answers to questions given a context; temperature; maximum tokens to be generated; stop sequence. This endpoint allows to select different models to perform the two-step search and the answer generation. Ada being the smallest model, is also the fastest, which was the main reason why we selected it to be our search model. When it comes to generating text, however, we would rather benefit from a better performance than from a faster approach, thus selecting Davinci model.

In addition to the question and context, we must also provide the endpoint with examples of answered questions. Thus, we took a few sets containing a question and the paragraph with the respective answer and created the examples. Figure 4.28 shows an illustration of the question, context and example to be provided for the endpoint.


Searchable File

"text" : I really like cats, in case you didn't know.

"text" : I happen to have one, named Mia.

"text" : She is either very lazy or extremely energetic, there is no in between.

"text" : Mia's favorite time of the day is snack time!

Posed Question: Do you like cats or dogs?

Context: Searchable file

Example Given:
Context: Mia's favorite time of the day is snack time!
Question: What is Mia's favorite time of the day?
Answer: snack time!

Retrieved Answer: I really like cats

Figure 4.28: GPT-3 QA Endpoint Answer Retrieval Example

Similarly to GPT-2, we then set the temperature to 0.2 and defined the stop or truncation token as '\n\n'. To conclude, we set a maximum of two hundred generation tokens, as the available quota did not allow for more. Once again, all questions asked and the retrieved answers are registered and stored in a text file, along with the `P:` and `R:` identifiers.

A brief summary of the followed steps for the sole configuration performed for this approach is presented in Figure 4.29.

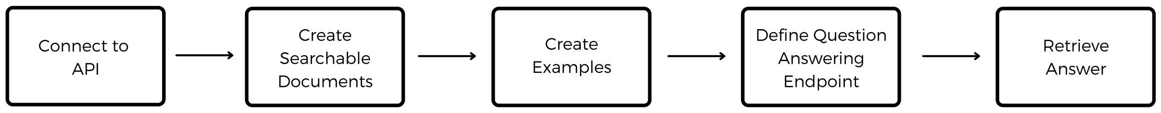

Connect to API
Create Searchable Documents
Create Examples
Define Question Answering Endpoint
Retrieve Answer

Figure 4.29: Configuration Steps Summary for GPT-3 Model

4.4 Discussion

Having all the approaches configurations ready, we could finally move on to the experimentation phase. The upcoming chapter details the process where we submit each approach to a set of experiments, store the results and perform evaluation. Furthermore, a comparison between approaches' results when using the same domain type is also conducted, with the focus being on relevant aspects such as human labor, time consumption and answer quality.

Chapter 5

Experimentation

In the previous chapter we proposed a solution to explore a set of approaches for answering questions in Portuguese related to a given domain, composed by list of FAQs or by a collection of unstructured plain text documents, also in the Portuguese language. This solution is based on experimenting with each approach whilst having the same data as the domain, i.e., same list of FAQs or collection of documents, and evaluating each approach on relevant aspects such as expended manual effort, answer quality and time consumption.

Prior to initiating experimentation, we configured all the approaches to be ready to answer questions related to the given domain, having disclosed the significant aspects of each approach's configuration in the former chapter. In addition, we also gathered three datasets for testing, two composed by FAQs and their variations and one composed by a collection of documents and questions related to them, to which we added identifiers to distinguish relevant aspects in the data.

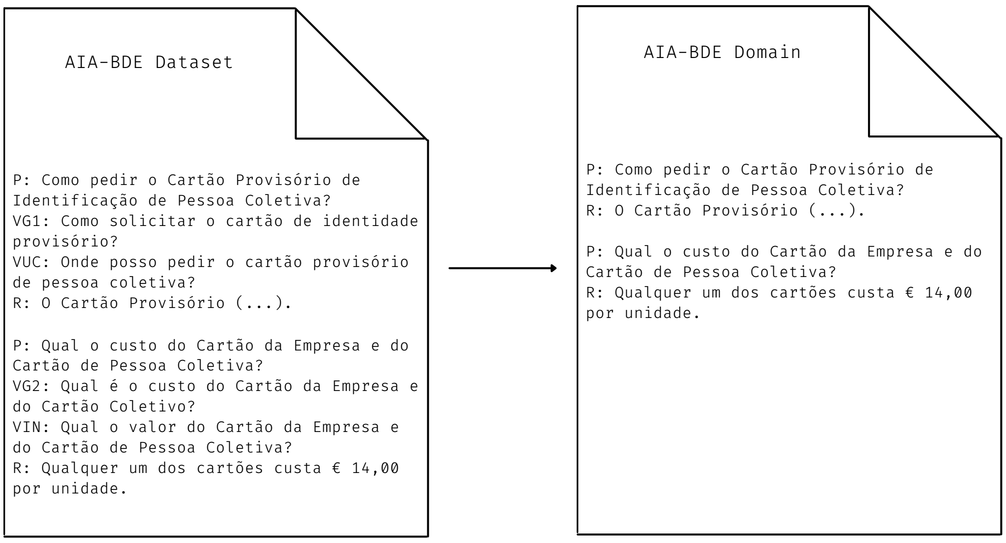
Having all approaches configured and ready to receive a domain and questions, we could then focus on the last phases of our work: experimentation and evaluation. The upcoming subsections describe in detail the setup created for the experimentation phase, including important aspects such as how the retrieved answers are stored for evaluation and how the time consumption is measured, as well as disclose the metrics used for evaluating answer quality. Finally, the obtained results are presented, along with a discussion containing the conclusions we gathered.

5.1 Setup

The FAQs datasets from Altice and AIA-BDE corpus are composed by a set of questions, their variations and the respective answers. However, as we wanted our experimentation to be conducted as similar to real use as possible, we formed the FAQs domain with only the original questions and the respective answers, leaving out the variations present in the dataset.

In a real-life scenario, the chances of a user asking a question to a QA system that is equal, i.e., the exact same words in the same order, to a question within the domain are very low. Therefore, we used the original questions and respective answers as the domain and left the variations of the questions to the question-answering step.

To do so, we created a file for each FAQs dataset, containing only the questions preceded by the identifier `P:` and the respective answers preceded by the identifier `R:`, being that the corresponding answer to a question immediately follows it. An example is shown in Figure 5.1, where only the original questions, the ones preceded by a `P:`, are used to form the domain file, along with the respective answers.

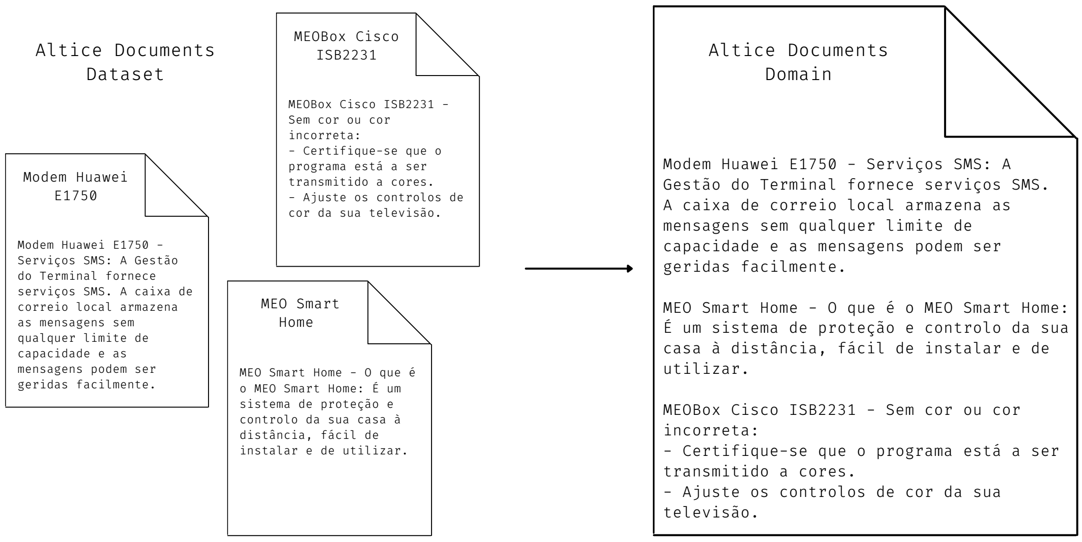


```
P: Como pedir o Cartão Provisório de
Identificação de Pessoa Coletiva?
VG1: Como solicitar o cartão de identidade
provisório?
VUC: Onde posso pedir o cartão provisório
de pessoa coletiva?
R: O Cartão Provisório (...).

P: Qual o custo do Cartão da Empresa e do
Cartão de Pessoa Coletiva?
VG2: Qual é o custo do Cartão da Empresa e
do Cartão Coletivo?
VIN: Qual o valor do Cartão da Empresa e
do Cartão de Pessoa Coletiva?
R: Qualquer um dos cartões custa € 14,00
por unidade.
```

```
P: Como pedir o Cartão Provisório de
Identificação de Pessoa Coletiva?
R: O Cartão Provisório (...).

P: Qual o custo do Cartão da Empresa e do
Cartão de Pessoa Coletiva?
R: Qualquer um dos cartões custa € 14,00
por unidade.
```

Figure 5.1: Creation FAQs Domain File Example

The remaining dataset is composed by a collection of documents, also provided by Altice, and by a set of manually created questions, each mapped to the paragraph containing its answer. To create the domain representing this dataset, we put all the documents together in the collection into a single plain text document. The documents in the collection already had the identifiers added to them, thus leaving the domain file with each paragraph identified by the filename of the document the paragraph belonged to. Figure 5.2 depicts an example where identified paragraphs from different documents are aggregated in the same domain file.



```
MEOBox Cisco ISB2231 -
Sem cor ou cor
incorreta:
- Certifique-se que o
programa está a ser
transmitido a cores.
- Ajuste os controlos de
cor da sua televisão.
```

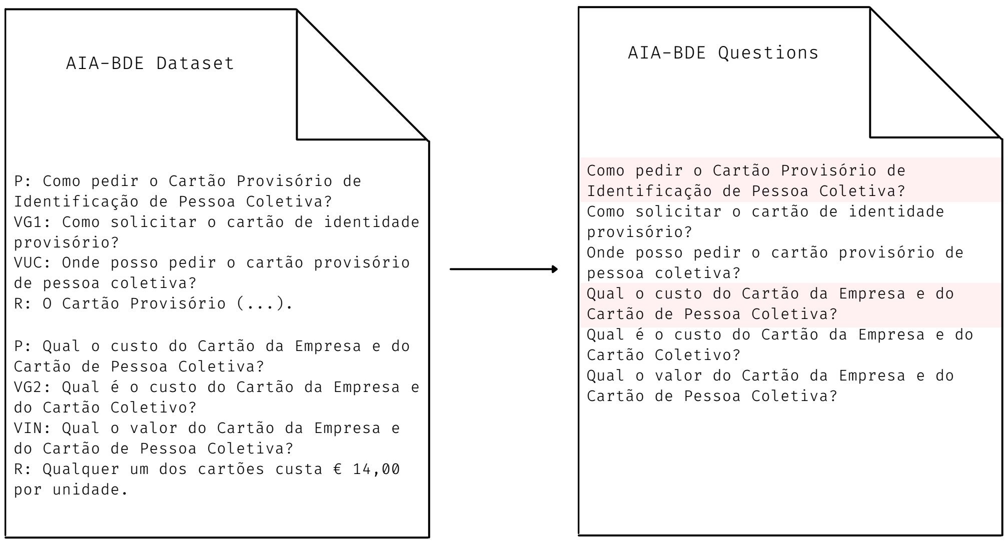
```
Modem Huawei E1750 -
Serviços SMS: A Gestão
do Terminal fornece
serviços SMS. A caixa de
correio local armazena
as mensagens sem
qualquer limite de
capacidade e as
mensagens podem ser
geridas facilmente.
```

```
MEO Smart Home - O que é
o MEO Smart Home: É um
sistema de proteção e
controlo da sua casa à
distância, fácil de
instalar e de utilizar.
```

```
Modem Huawei E1750 - Serviços SMS: A
Gestão do Terminal fornece serviços SMS.
A caixa de correio local armazena as
mensagens sem qualquer limite de
capacidade e as mensagens podem ser
geridas facilmente.

MEO Smart Home - O que é o MEO Smart Home:
É um sistema de proteção e controlo da sua
casa à distância, fácil de instalar e de
utilizar.

MEOBox Cisco ISB2231 - Sem cor ou cor
incorreta:
- Certifique-se que o programa está a ser
transmitido a cores.
- Ajuste os controlos de cor da sua
televisão.
```

Figure 5.2: Creation Collection of Documents Domain File Example

In addition to the domain files, each approach must receive the questions to be answered. As such, we created a plain text file for each domain containing a question per line, without any identifier. In both domains composed by a list of FAQs, the questions file gathers all the original questions along with their variations, whilst, on the other hand, for the collection of documents domain, this file is composed solely by the manually created questions. An example is presented in Figure 5.3, where the highlighted questions are the original and the remaining are the variations.



```
P: Como pedir o Cartão Provisório de
Identificação de Pessoa Coletiva?
VG1: Como solicitar o cartão de identidade
provisório?
VUC: Onde posso pedir o cartão provisório
de pessoa coletiva?
R: O Cartão Provisório (...).

P: Qual o custo do Cartão da Empresa e do
Cartão de Pessoa Coletiva?
VG2: Qual é o custo do Cartão da Empresa e
do Cartão Coletivo?
VIN: Qual o valor do Cartão da Empresa e
do Cartão de Pessoa Coletiva?
R: Qualquer um dos cartões custa € 14,00
por unidade.
```

```
Como pedir o Cartão Provisório de
Identificação de Pessoa Coletiva?
Como solicitar o cartão de identidade
provisório?
Onde posso pedir o cartão provisório de
pessoa coletiva?
Qual o custo do Cartão da Empresa e do
Cartão de Pessoa Coletiva?
Qual é o custo do Cartão da Empresa e do
Cartão Coletivo?
Qual o valor do Cartão da Empresa e do
Cartão de Pessoa Coletiva?
```

Figure 5.3: Creation of Questions File Example

Having created all the files, these were passed to the respective approaches, depending on the type of domain they were configured to receive. Upon receiving the files, the next steps could be divided into two phases: (i) adaptation to the domain; (ii) retrieval of answers.

When receiving a domain file, each approach had to adapt to that same domain, that is, a series of configuration steps was performed to be able to answer questions related to the information in it.

Subsequently to performing the required steps, the approaches then fetched one question at a time from the questions file and retrieved an answer for that same question, storing both the posed question and the retrieved answer in a text file, along with the respective `P:` and `R:` identifiers. This process was repeated for all questions, resulting in a file with all the posed questions and the corresponding retrieved answers.

Given the two distinct phases the approaches go through after receiving the domain and questions files, we decided to measure the time consumed in each phase separately. Regarding the first phase, time is measured between the moment when the approach receives the domain file and the moment when the last configuration step is completed.

In the answer retrieval phase, the time is measured between the moment when the last configuration step is completed and the final answer is retrieved. We then calculate an average answering time by dividing the time consumed by the latter phase for the total of posed questions. The measured times are stored in plain text files, for posterior evaluation.

Each approach performs different steps in each phase, all of which are explained in detail in the previous chapter. However, hoping to provide a better understanding of the time consumption measurements we defined, a brief description of the steps included in each phase, for each configuration, is disclosed ahead.

Google Dialogflow adapts to the domain by creating intents or a knowledge base with documents, and retrieves one or more answers by opening an API session and then posing a question and receiving an answer from the API. Thus, the times measured in Dialogflow are as depicted in Figure 5.4.

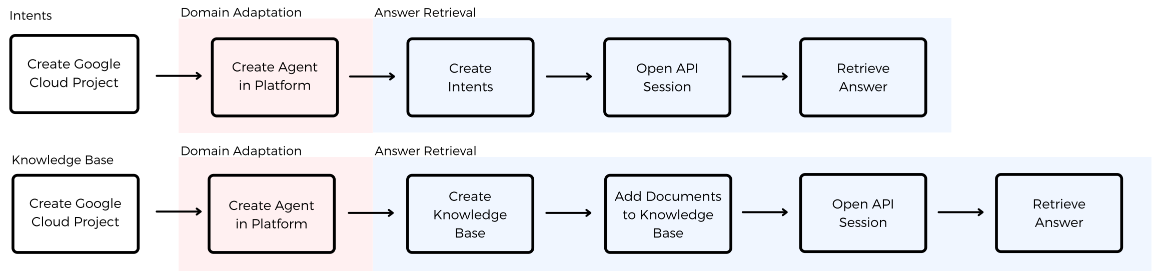


Figure 5.4: Time Consumption Measurements for Dialogflow

As for Whoosh, the domain adaptation steps include creating an index and schema objects, dividing the domain in multiple documents and indexing the created documents. The answer retrieval phase includes creating a searcher and query objects and searching for an answer in the domain, as can be seen in Figure 5.5.

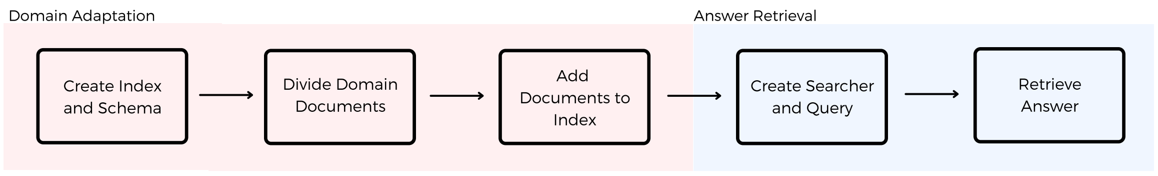


Figure 5.5: Time Consumption Measurements for Whoosh

BERT's clustering configuration performs two steps when adapting to a given domain, namely, obtaining the domains questions' vector representations and performing a k-means clustering to create several clusters of questions-answer pairs. In this configuration, BERT compares the posed question with all the created clusters and passes the most similar one to BERTimbau QA as context, with the latter retrieving an answer. Both phases of BERT clustering are shown in Figure 5.6.

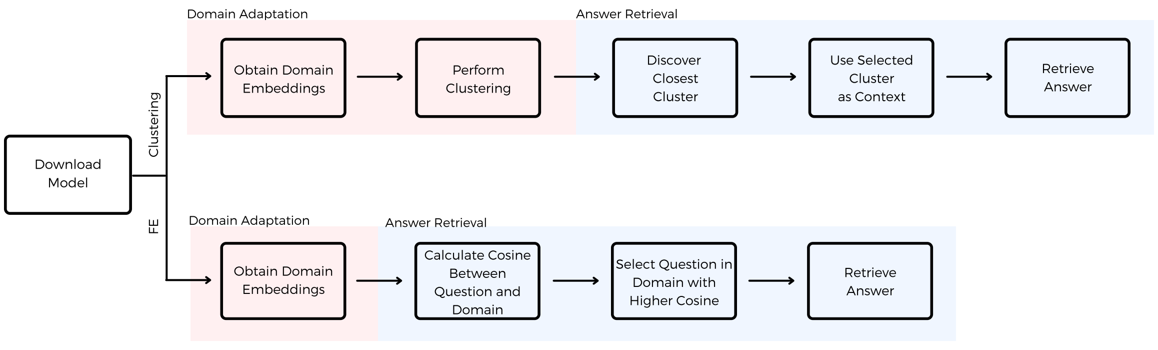


Figure 5.6: Time Consumption Measurements for BERT Clustering and Feature Extraction

BERT's feature extraction configuration adapts to the domain by obtaining the embeddings of its questions, and retrieves answers by calculating the cosine similarity between the vector representation of the posed question and of each question in the domain, thus retrieving the answer associated to the most similar domain question. The steps belonging to each phase for this configuration are also presented in Figure 5.6.

When combining BERT with Whoosh, the steps to adapt to a new domain are the same as for single Whoosh, i.e., creating an index and schema objects, dividing the domain in multiple documents and indexing the created documents. Then, an answer can be retrieved by using Whoosh to create a searcher and query objects and to search for the documents most similar to the posed question, later passing these as context to BERTimbau QA for it to retrieve an answer. Figure 5.7 depicts the several steps of BERT + Whoosh phases.

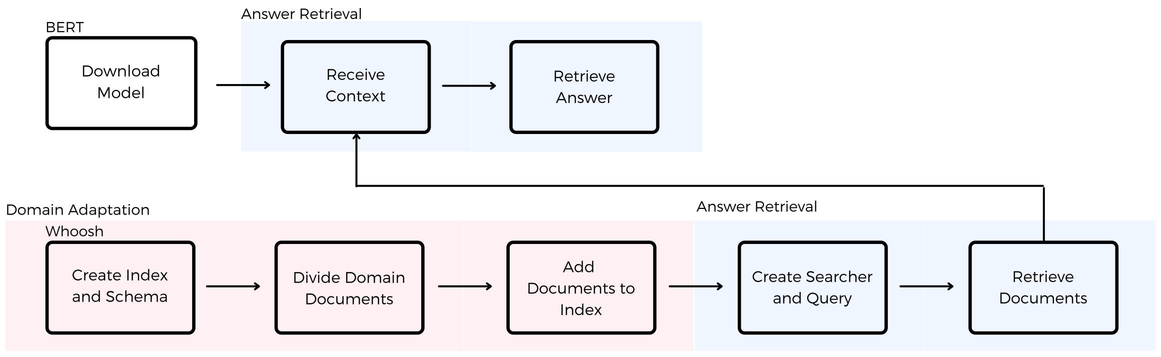


Figure 5.7: Time Consumption Measurements for BERT + Whoosh

GPT-2 adapts to a new domain with one single step: finetuning. After finetuning with the domain information, answer generation parameters are defined and the answers generated by GPT-2 are retrieved, as can be seen in 5.8.

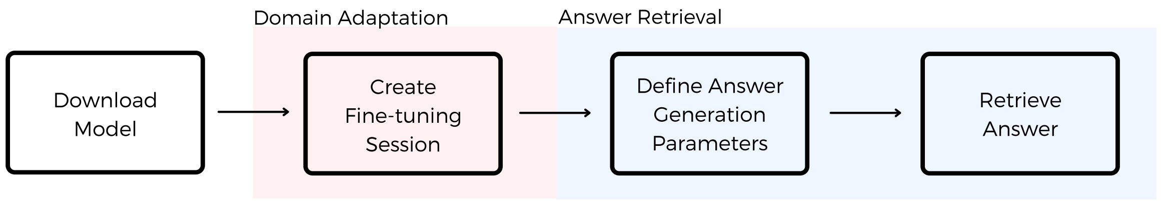


Figure 5.8: Time Consumption Measurements for GPT-2

Finally, GPT-3's time consumption is divided between the creation of the searchable documents and the query of the question answering endpoint through an API request. Thus, the times measured in GPT-3's configuration are as depicted in Figure 5.9.

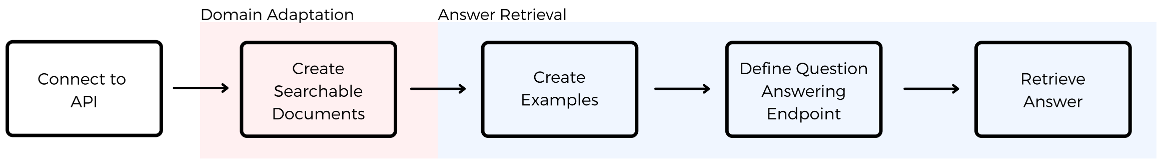


Figure 5.9: Time Consumption Measurements for GPT-3

Note that, with the exception of GPT-3, ran in OpenAI's API, our experimentation was conducted on a Google Colab notebook[1].

[1]https://colab.research.google.com/

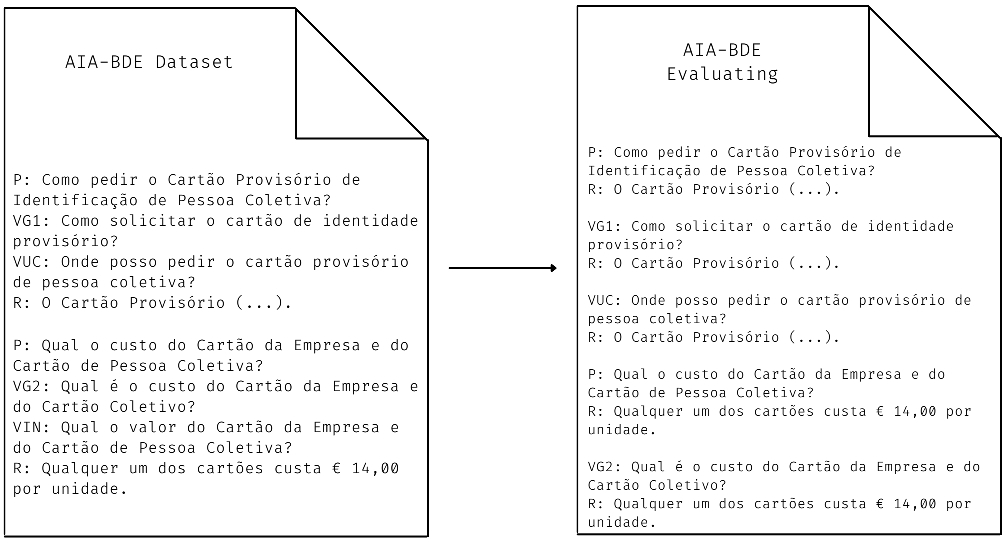
5.2 Evaluation

With all experiments conducted, we were left with a file per approach configuration of question-answer pairs, containing all the questions posed to the approach and the respective retrieved answers, preceded by the corresponding identifiers. In addition, we also collected two files per approach configuration, that stored the time of domain adaptation and the total answering time for each configuration.

With the time files, we created a table for each domain, containing the measured times for each approach configuration. Although the time of adapting to a given domain was retrieved from the files as is, we divided the total answering time by the total of questions asked, thus creating an average time per answer retrieved for each configuration. The measured times are presented in the respective tables in the upcoming subsection.

In order to evaluate answer quality, the files with all the question-answer pairs, retrieved from the experiments with the various approaches, were used to compute a set of evaluation metrics.

Prior to computing any metric, we would need to have a reference for each existing FAQ, that associated each question in the dataset, original or variation, with the corresponding answer. Therefore, we created a file per dataset that correlates each question with the respective answer, by immediately following each question with the corresponding answer and adding identifiers. An example can be seen in Figure 5.10, where an excerpt from AIA-BDE corpus is transformed to create an evaluation file.



```
P: Como pedir o Cartão Provisório de
Identificação de Pessoa Coletiva?
VG1: Como solicitar o cartão de identidade
provisório?
VUC: Onde posso pedir o cartão provisório
de pessoa coletiva?
R: O Cartão Provisório (...).

P: Qual o custo do Cartão da Empresa e do
Cartão de Pessoa Coletiva?
VG2: Qual é o custo do Cartão da Empresa e
do Cartão Coletivo?
VIN: Qual o valor do Cartão da Empresa e
do Cartão de Pessoa Coletiva?
R: Qualquer um dos cartões custa € 14,00
por unidade.
```

```
P: Como pedir o Cartão Provisório de
Identificação de Pessoa Coletiva?
R: O Cartão Provisório (...).

VG1: Como solicitar o cartão de identidade
provisório?
R: O Cartão Provisório (...).

VUC: Onde posso pedir o cartão provisório de
pessoa coletiva?
R: O Cartão Provisório (...).

P: Qual o custo do Cartão da Empresa e do
Cartão de Pessoa Coletiva?
R: Qualquer um dos cartões custa € 14,00 por
unidade.

VG2: Qual é o custo do Cartão da Empresa e do
Cartão Coletivo?
R: Qualquer um dos cartões custa € 14,00 por
unidade.
```

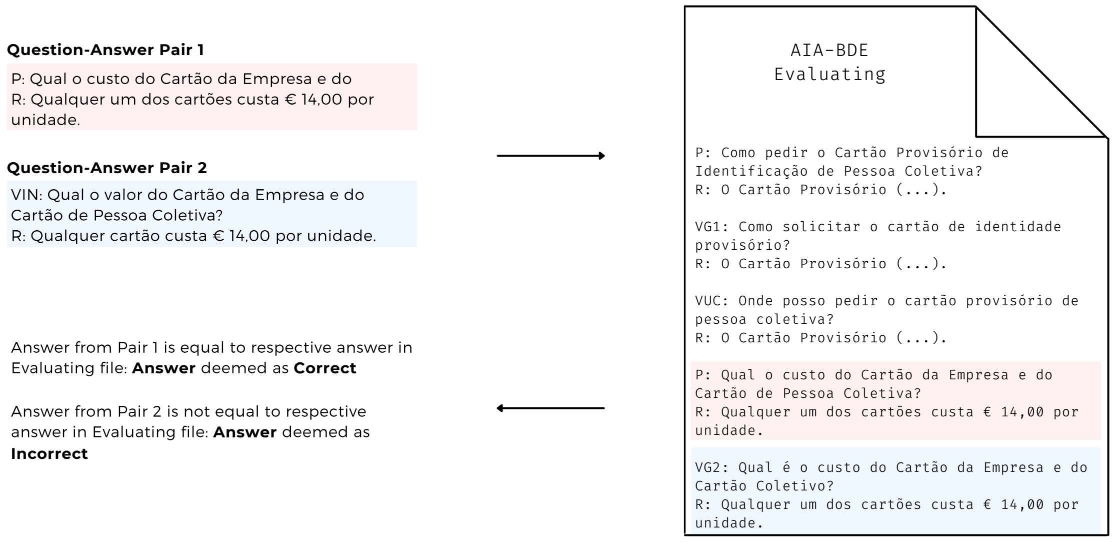
Figure 5.10: Creation of an Evaluating File Example

To compare the similarity between the retrieved answer and the corresponding answer in the evaluation file, we used two evaluation metrics, the automatic evaluation metric BERTScore and language-independent BLEU. We compute the similarity of both answers with these two metrics due to their difference in doing so, as BERTScore considers meaning representations and BLEU solely focuses on the given strings.

Thus, we used the created evaluation files along with the files composed by question-answer pairs that resulted from experimentation to compute the different evaluation metrics. For the Altice domain composed by a collection of plain text documents, we compute BERTScore and BLEU scores for each retrieved answer, and then calculate an average answering score for each metric.

As for both domains with a list of FAQs, we similarly compute BERTScore and BLEU scores for each retrieved answer, along with the average answering score for each metric. However, in this case we also determine the percentage of correct and incorrect answers, such that an answer is deemed as correct if it is completely equal to the respective answer in the evaluation file.

Since each FAQ has a well-defined and limited answer, the main challenge is thus identifying the correct question, to then retrieve the associated answer. When computing the percentage of correct and incorrect answers, we were able to determine which questions were correctly identified. Figure 5.11 presents an example of two posed questions and the corresponding question-answer pairs in the evaluation file, along with the results of the computation of this metric for both questions.



Question-Answer Pair 1

P: Qual o custo do Cartão da Empresa e do
R: Qualquer um dos cartões custa € 14,00 por unidade.

Question-Answer Pair 2

VIN: Qual o valor do Cartão da Empresa e do Cartão de Pessoa Coletiva?
R: Qualquer cartão custa € 14,00 por unidade.

Answer from Pair 1 is equal to respective answer in Evaluating file: **Answer** deemed as **Correct**

Answer from Pair 2 is not equal to respective answer in Evaluating file: **Answer** deemed as **Incorrect**

```
AIA-BDE
Evaluating

P: Como pedir o Cartão Provisório de
Identificação de Pessoa Coletiva?
R: O Cartão Provisório (...).

VG1: Como solicitar o cartão de identidade
provisório?
R: O Cartão Provisório (...).

VUC: Onde posso pedir o cartão provisório de
pessoa coletiva?
R: O Cartão Provisório (...).

P: Qual o custo do Cartão da Empresa e do
Cartão de Pessoa Coletiva?
R: Qualquer um dos cartões custa € 14,00 por
unidade.

VG2: Qual é o custo do Cartão da Empresa e do
Cartão Coletivo?
R: Qualquer um dos cartões custa € 14,00 por
unidade.
```

Figure 5.11: Correct or Incorrect Metric Example

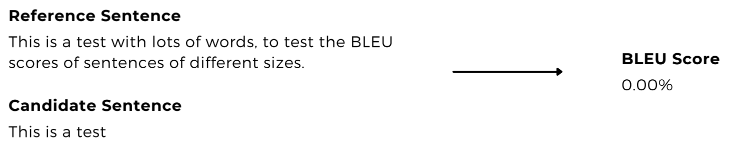
BERTScore (Zhang et al., 2019a) is an evaluation metric based on pre-trained BERT contextual embeddings, used for calculating the similarity between a candidate and a reference spans of text. This metric first obtains the BERT vector representations of each word in the candidate and reference spans, through a given BERT model.

To calculate the similarity between the two spans, BERTScore forms an alignment between the candidate and reference words, by computing pairwise cosine similarity with the embeddings retrieved from BERT. This alignment is then aggregated into a F1 score. For evaluating the answers retrieved by the different approaches during experimentation, we relied on BERTimbau (Souza et al., 2020), a BERT model pre-trained for Portuguese, for obtaining the contextual embeddings and stored the computed F1 scores.

BLEU (Papineni et al., 2002) is a precision-based metric that computes a similarity score between a candidate sentence and a reference sentence. Although typically used for evaluating machine translation systems, this metric can be used to compute the similarity between two spans of text, e.g., two sentences.

BLEU receives as arguments a candidate sentence and a reference sentence, returning the similarity score between the two. To do so, it compares the n-grams of the candidate sentence with the n-grams of the reference sentence, counting the number of found matches. The higher the number of matches, the higher the returned score.

When a candidate sentence has a different count of words than the reference sentence, the extra or missing words count as wrong words, thus significantly dropping the score for that candidate sentence. For instance, if the reference sentence were to have the reference sentence with 18 words and the candidate with 3, the BLEU score would be 0%, even if the three words from the candidate sentence were present in the reference sentence. This example can be seen in Figure 5.12.



Reference Sentence
This is a test with lots of words, to test the BLEU scores of sentences of different sizes.

Candidate Sentence
This is a test

BLEU Score
0.00%

Figure 5.12: BLEU Penalisation for Different Sized Sentences Metric Example

Multiple values of n can be used for computing BLEU, with n ranging from 1 to 4. The most commonly used BLEU configuration is BLEU-4, where the score is computed for 1 to 4-grams and the resulting scores aggregated by a geometric mean. To evaluate our experimentation results, we too use the cumulative 4-gram BLEU scores, BLEU-4.

After computing any metric, the obtained results were stored in text files, each identifying the type of domain, the name of the domain and the metrics scores. In addition, we created a table per metric and per domain, along with a graph displaying the results in each table. Both the tables and graphs are displayed in the upcoming subsection.

5.3 Results

Prior to analyzing the results provided by the different evaluation metrics and time measurements, we subjectively assess the effort related to the implementation of each approach and its configurations.

As the main focus of this work is to evaluate the selected approaches regarding their performance when answering questions related to a Portuguese domain, this section also presents the results obtained from the evaluation metrics and time consumption measurements. The final conclusions regarding the performance of the different approaches is disclosed in the next section.

Due to having two distinct types of domain, a structured list of FAQs and an unstructured collection of raw text documents, the results are divided according to the domain associated to them, and the comparative analysis is done per domain type.

5.3.1 Assessment of Implementation Effort

Table 5.1 presents the subjective levels of effort required to implement each of the distinct configurations. The levels range from 1 to 5, such that: **1** represents a very light effort; **2** represents a light amount effort; **3** a normal effort; **4** a high amount of effort; and **5** an excessive effort. Note that these results are based on a personal opinion and that, due to the implementation process being very similar for some configurations of the same approach, we decided to assess their effort simultaneously.

In the table, *Dialogflow* refers to both configurations of this approach, one where intents are created and other where a knowledge base is created. To implement both configura-

Configuration	Effort Level
Dialogflow	3
Whoosh	2
BERT Clustering	4
BERT FE	1
BERT + Whoosh	3
GPT-2	2
GPT-3	1

Table 5.1: Implementation Effort for each Configuration

tion, we had to create a Google Cloud Project, credentials for that project, and a Google Dialogflow agent, steps which were performed in two different platforms. Additionally, to create the intents or the knowledge base, we had to search for the correct commands amongst Dialogflow's documentation[2], which were not always easy to find. Finally, we noticed that, everytime there is an update, we would have to re-write most of the code. For these three reasons, we gave this approach an implementation effort level of 3.

All of Whoosh's configurations, whether it being default, with a Portuguese analyzer, with 3-gram and 4-gram analyzer filters or with the search being made by one or two fields, are represented in the table by *Whoosh*. When implementing Whoosh, even though we needed to understand how to build the different objects, i.e., index, schema, search an query objects, all of the steps to do so were seamlessly explained in its documentation[3]. Its implementation effort was assessed as a level 2 due to the amount of steps we had to implement.

BERT combined with clustering is referred to in the table as *BERT Clustering*. To implement this configuration, we first had to use BERTimbau to get the vector representations for each question. After that, we had to perform a k-means clustering with the embeddings retrieved from BERTimbau, saving both the clusters and calculated centroids. For each posed question, we had to check which was the most similar centroid to then pass the corresponding cluster as context to BERTimbau QA. When considering these steps, we gave this configuration an implementation effort level of 4.

In opposition to the previous BERT configuration, BERT used for feature extraction had a much simpler implementation process. This configuration, represented in the table by *BERT FE*, used BERTimbau to get the vector representations of each question and then computed the similarity between the posed question embedding and the embeddings of each question in the domain. Thus, its implementation effort was assessed as a level 1.

For the combination of *BERT + Whoosh*, we assessed the related implementation effort by combining the previously estimated Whoosh effort with the step of passing a context and a question to BERTimbau QA, which lead us to an effort level of 3.

The implementation of *GPT-2* relied on fine-tuning the model and retrieving answers. To do so, we had to download the model, and create a fine-tune and a generation sessions, steps explained in the model's documentation[4]. However, we also had to define hyperparameters for both sessions, which we empirically chose by reading the documentation and testing the model's results with different parameter combinations. This lead us to giving

[2]https://cloud.google.com/dialogflow/docs
[3]https://whoosh.readthedocs.io/
[4]https://github.com/minimaxir/gpt-2-simple

this approach an implementation effort level of 2.

To implement *GPT-3*, we had to make requests through OpenAI's API[5] containing the list of documents forming the domain, the posed question and a set of examples. We also had to define a set of hyperparameters, but this process was much simpler when compared to GPT-2, due to the information provided by the API[6] and the playground[7] available to experiment with different parameter combinations. We attributed this configuration an implementation effort level of 1.

5.3.2 Domain Composed by a List of FAQs

For the FAQs domain type, we used two distinct datasets, of considerably different sizes. The first, a list of FAQs provided by Altice, was composed by 172 FAQs and 106 question variations, thus leading to a domain containing 172 question-answer pairs, and 378 questions to be asked. The second, and much larger, dataset was from the AIA-BDE corpus, and was composed by 855 FAQs and 5,089 question variations, as such forming a domain with 855 question-answer pairs, and 5,944 questions to be asked.

A total of twelve configurations divided by four approaches were conducted, in order to evaluate their performance when answering questions related to a Portuguese FAQs domain. The scores computed by BERTScore and BLEU metrics and the percentage of correct and incorrect answers were measured and are presented below, for each domain.

Altice FAQs

Table 5.2 and graph 5.13 show the percentage of answers correctly answered by each configuration tested with the Altice FAQs documents, table 5.3 presents the average BERTScore, and table 5.4 the average similarity scores computed by BLEU. Finally, graph 5.14 shows the average BERTScore and BLEU scores for each configuration.

In addition to the percentage of correct answers, table 5.2 also shows what percentage of those same questions was retrieved for an original question and what was for a variation. We believed it would be interesting to know which approach performs better when presented with questions that are not written exactly like the ones in the domain. Furthermore, these results can be seen as proof that the approaches work, i.e., if they are not able to provide a correct answer when the posed question is completely equal to one in the domain, they will consequently not be able to answer a question that is different from the ones in the domain.

However, answer generating approaches such as BERTimbau QA or GPT-2 are expected to have a score of 0% on the percentage of correct answers, due to the answers not being retrieved directly from the list of FAQs. With GPT-2's generated answers and BERTimbau QA's selected span of text as answers, the chances of the retrieved answers exactly matching the ones in the evaluating file are close to none. Therefore, in these cases we focus on the scores computed by the remaining two metrics, BERTScore and BLEU.

Google Dialogflow had two configurations tested with the list of FAQs provided by Altice, one where an intent was created per FAQ and one where a knowledge base containing all FAQs was created, both represented in each table and graph by *Dialogflow Intents* and

[5]https://openai.com/blog/openai-api/
[6]https://beta.openai.com/docs/
[7]https://beta.openai.com/playground/

Configuration	Correct Answers (%)	Correct Original (%)	Correct Variation (%)
Dialogflow Intents	84.92	100.00	72.33
Dialogflow KB	56.61	73.26	42.72
Whoosh Ques Default	91.80	98.24	87.38
Whoosh Ques LangPt	91.53	96.47	88.35
Whoosh Ques NGram3	87.57	98.82	79.13
Whoosh Ques NGram4	87.57	97.65	80.10
Whoosh QuesAns Default	78.31	88.82	70.39
Whoosh QuesAns LangPt	76.19	82.94	71.36
Whoosh QuesAns NGram3	85.19	95.88	77.18
Whoosh QuesAns NGram4	83.07	96.47	72.82
BERT Clustering	0.00	0.00	0.00
BERT FE	66.40	99.42	38.83
GPT2	0.00	0.00	0.00

Table 5.2: Correct or Incorrect Results for Altice FAQs Domain

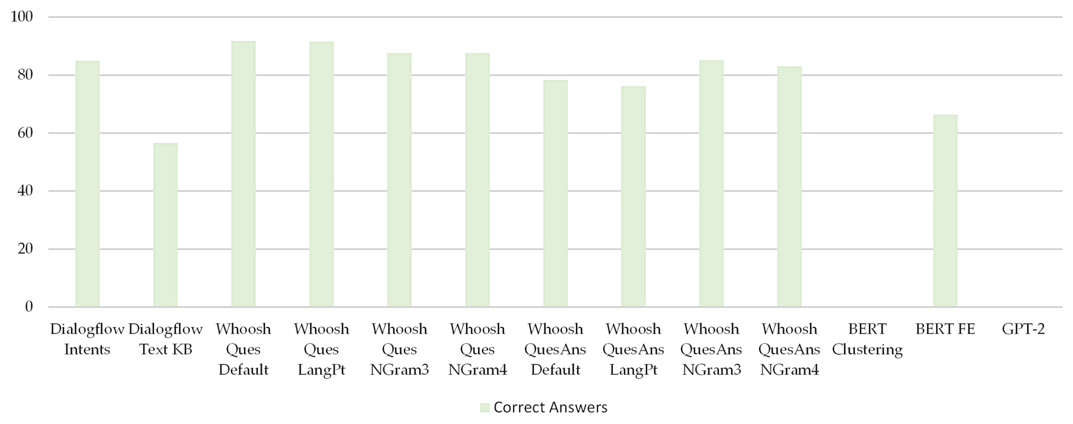


Figure 5.13: Graph with Percentage of Correct Answers for Altice FAQs Domain

Dialogflow KB, respectively.

Dialogflow's intents configuration got a high percentage of correct answers (84.92%), being able to answer all the original questions correctly. It also got a high BERTScore of 98.83% and BLEU of 89.45%. We believe this happened because, once receiving an input, Dialogflow searches for the intent containing the most similar training phrase. If the input is identical to a training phrase, it scores it very high and returns the responses associated to it. In this case, it correctly matched all the posed original questions with the training phrases formed by the questions in the domain, due to them being identical.

The knowledge base configuration however, targeting in English rather than Portuguese, scored some lower results, which we also expected. It got 56.61% of correct answers, and 73.70% and 89.45% for the metrics BERTScore and BLEU, respectively.

The configurations implemented for Whoosh can be divided into two groups: where the search for similar documents is done by the field question; where the search for similar documents is conducted whilst considering both the question and answer fields. For each group, a default analyzer, a Portuguese analyzer and the 3-gram and 4-gram analyzer filters were tested. Each Whoosh configuration is identified in the tables and graph by *Whoosh*, followed by the type of search, *Ques* or *QuesAns*, and the analyzer.

Although the results are fairly similar, the configurations where search is done by only comparing the question field presented slightly better results. Out of the four configurations

Configuration	$\overline{\text{BERTScore}}$	σ
Dialogflow Intents	98.83	± 0.03
Dialogflow KB	73.70	± 0.43
Whoosh Ques Default	98.55	± 0.09
Whoosh Ques LangPt	98.63	± 0.09
Whoosh Ques NGram3	98.34	± 0.09
Whoosh Ques NGram4	98.33	± 0.09
Whoosh QuesAns Default	97.65	± 0.09
Whoosh QuesAns LangPt	97.52	± 0.09
Whoosh QuesAns NGram3	98.19	± 0.09
Whoosh QuesAns NGram4	98.04	± 0.09
BERT Clustering	81.71	± 0.06
BERT FE	97.38	± 0.05
GPT2	90.12	± 0.04

Table 5.3: BERTScores for Altice FAQs Domain

Configuration	$\overline{\text{BLEU}}$	σ
Dialogflow Intents	89.45	± 0.25
Dialogflow KB	63.19	± 0.45
Whoosh Ques Default	94.64	± 0.19
Whoosh Ques LangPt	94.76	± 0.18
Whoosh Ques NGram3	92.82	± 0.21
Whoosh Ques NGram4	92.64	± 0.21
Whoosh QuesAns Default	87.07	± 0.26
Whoosh QuesAns LangPt	85.89	± 0.27
Whoosh QuesAns NGram3	91.25	± 0.22
Whoosh QuesAns NGram4	90.34	± 0.23
BERT Clustering	0.12	± 0.01
BERT FE	79.58	± 0.32
GPT2	22.11	± 0.21

Table 5.4: BLEU Scores for Altice FAQs Domain

belonging to that group, the one with a 3-gram filter added to the analyzer would be our choice. It presented a BERTScore of 98.34% and a BLEU score of 92.82%, as well as 87.57% of correctly answered questions.

Given a list of Altice FAQs, BERT was tested with two different configurations. The first involved clustering the FAQs, referred to as *BERT Clustering* in the presented tables and graphs, and the second relied on using BERT to perform feature extraction, referred to as *BERT FE* in the same tables and graphs.

The clustering configuration of BERT presented 0 correct answers, due to retrieving an answer corresponding to a span of text from the given context, as aforementioned. In addition, it also got a poor BLEU score of 0.12%. This is due to BLEU penalising sentences of different sizes, which happens when BERTimbau QA retrieves spans of text from the context to form an answer. With a non-penalising metric, this configuration got a 81.71% BERTScore.

BERT feature extraction on the other hand, got 66.4% of correct answers. It managed to answer correctly 99.42% of the original questions and 38.83% of the variations. This configuration also got a BERTScore of 97.38% and 79.58% in BLEU.

GPT-2's sole configuration is depicted in each table and graph as *GPT2*. Similarly to the previous BERT clustering configuration, GPT-2 also got 0 questions answered correctly and a low BLEU score of 22.11%, due to the same reasons. Even if one of the lowest scores,

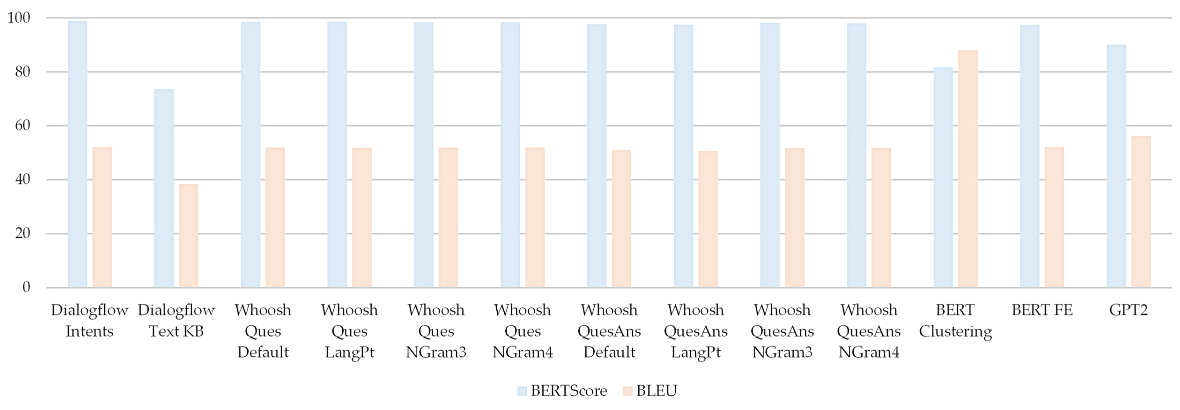


Figure 5.14: Graph with BLEU and BERTScores for Altice FAQs Domain

it still got a high BERTScore of 90.12%.

Due to the dataset provided by Altice only having a total of 172 original questions and 206 variations, we thought fit to experiment with another FAQs dataset, with a larger number of both original questions and variations. To do so, we used the dataset from AIA-BDE corpus. The experimentation results are depicted in the following section.

AIA-BDE FAQs

Table 5.5 and graph 5.15 present the percentage of correctly answered questions by each configuration tested with AIA-BDE dataset, composed by a list of FAQs. Tables 5.6 and 5.7, and graph 5.16 show the BERTScore and BLEU evaluation metrics results, respectively, for the same configurations.

Each configuration tested with a structured list of FAQs from AIA-BDE corpus is referred to in each table and graph the same way that it was presented in the tables and graphs regarding Altice FAQs dataset.

The table corresponding to the percentage of correct answers, 5.5, additionally shows the percentage of correct answers retrieved for the original questions and both types of variations, derived from Google Translate and manually created. The ones retrieved for Google Translate questions represent the variations identified by `VG1:` and `VG2:` in the original dataset, whilst the ones retrieved for manually created questions represent the ones identified by `VUC:`, `VIN:` and `VMT:`.

As for the last FAQs dataset, the answer generating approach BERTimbau QA, used in the clustering configuration of BERT, is expected to have 0 correctly answered questions, along with a low BLEU score.

However conducted with a much larger dataset, experimentation with AIA-BDE proved to have fairly similar results to experimentation with the former dataset, Altice FAQs. The time consumption was greater, although that was expected, as there is more information to process.

Google Dialogflow configured with intents got slightly lower results with this dataset, presenting 77.67% of correctly answered questions, a BERTScore of 96.81% and a BLEU score of 83.51%. On the contrary, the knowledge base configuration dropped the scores greatly, only getting 3.16% correct answers and 9.76% and 4.89% scores in BERTScore and

Configuration	Correct Answers (%)	Correct Original (%)	Correct Translator (%)	Correct Manual (%)
Dialogflow Intents	77.67	99.35	82.62	70.08
Dialogflow KB	3.16	0.00	4.53	3.38
Whoosh Ques Default	74.31	96.74	82.06	66.37
Whoosh Ques LangPt	70.22	88.90	72.96	65.59
Whoosh Ques NGram3	78.70	99.78	85.07	71.90
Whoosh Ques NGram4	77.41	99.78	82.87	70.64
Whoosh QuesAns Default	66.44	88.57	70.37	60.21
Whoosh QuesAns LangPt	60.40	80.20	60.98	56.23
Whoosh QuesAns NGram3	78.25	99.46	82.99	72.20
Whoosh QuesAns NGram4	77.05	99.24	81.04	71.06
BERT Clustering	0.00	0.00	0.00	0.00
BERT FE	75.05	99.79	74.59	68.42

Table 5.5: Correct or Incorrect Results for AIA-BDE FAQs Domain

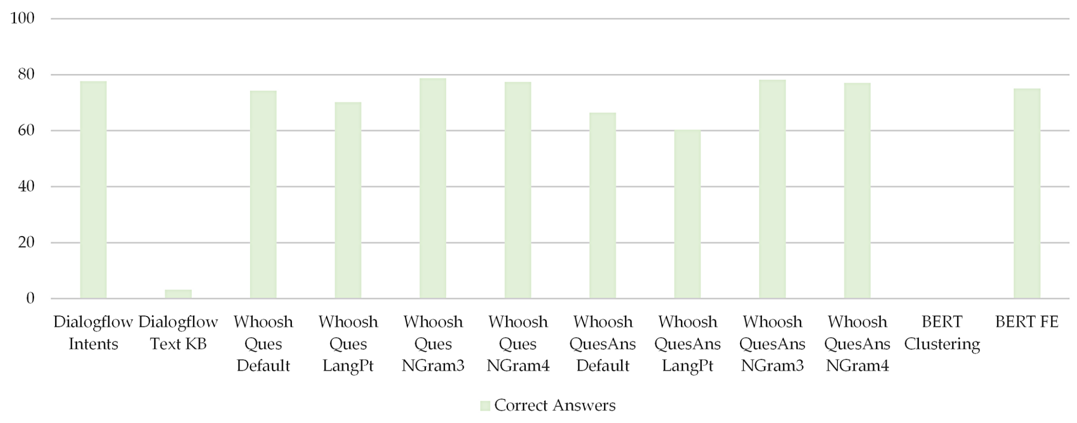


Figure 5.15: Graph with Percentage of Correct Answers for AIA-BDE FAQs Domain

BLEU respectively, due to the same aforementioned problem of performing the search for information in English.

When experimenting with a larger document of FAQs, out of the eight configurations for Whoosh, the one referring to the addition of a 3-gram analyzer outperforms the rest, even if just for a minimal difference. This configuration presented 78.70% of correct answers, whilst getting a BERTScore of 97.01% and a BLEU score of 84.25%. Out of the correctly answered questions, it got the answers correct for 99.78% of the original question, 85.07% of the Google Translate variations and 71.90% of the manually created ones.

BERT clustering did not get any answer correct, according to our correct or incorrect metric. This was expected, and does not affect our opinion on the configuration, as the answer might still be correct, just not using exactly the same terms as the answer in the dataset to which it was compared. Also expected, was the extremely low BLEU score (1%). Finally, it presented a fair BERTScore of 74.19%.

With the second configuration, feature extraction, BERT's results improved, having a 75.05% of correctly answered questions, a BERTScore of 96.48% and a BLEU score of 81.04%.

Having analyzed all the obtained results separately, we defined which configurations we believed to better suit our purpose, for each domain type. The thoughts we gathered are presented in the next section.

Configuration	BERTScore	σ
Dialogflow Intents	96.81	± 0.07
Dialogflow KB	9.76	± 0.28
Whoosh Ques Default	95.22	± 0.07
Whoosh Ques LangPt	94.34	± 0.08
Whoosh Ques NGram3	97.01	± 0.06
Whoosh Ques NGram4	96.83	± 0.06
Whoosh QuesAns Default	96.27	± 0.07
Whoosh QuesAns LangPt	95.74	± 0.07
Whoosh QuesAns NGram3	97.10	± 0.06
Whoosh QuesAns NGram4	96.86	± 0.06
BERT Clustering	74.19	± 0.08
BERT FE	96.48	± 0.07

Table 5.6: BERTScores for AIA-BDE FAQs Domain

Configuration	BLEU	σ
Dialogflow Intents	83.51	± 0.33
Dialogflow KB	4.89	± 0.19
Whoosh Ques Default	75.18	± 0.37
Whoosh Ques LangPt	70.45	± 0.39
Whoosh Ques NGram3	84.25	± 0.32
Whoosh Ques NGram4	83.29	± 0.33
Whoosh QuesAns Default	80.65	± 0.35
Whoosh QuesAns LangPt	77.77	± 0.36
Whoosh QuesAns NGram3	84.63	± 0.32
Whoosh QuesAns NGram4	83.49	± 0.33
BERT Clustering	1.00	± 0.04
BERT FE	81.04	± 0.34

Table 5.7: BLEU Scores for AIA-BDE FAQs Domain

5.3.3 Domain Composed by a Collection of Unstructured Raw Text Files

Five different approaches were configured to answer questions given a collection of plain text documents in Portuguese. Out of these five approaches, we experimented with ten configurations and collected the time consumed by each configuration whilst adapting to the domain and answering questions, along with the scores computed by BERTScore and BLEU metrics.

Table 5.8 presents the average BERTScores for each approach, table 5.9 the average BLEU scores and graph 5.17 both average scores computed by each metric.

Dialogflow's sole configuration, referred in each table and in the graph by *Dialogflow Text KB*, presented the lowest scores on both metrics, with a BERTScore of 20.83% and a BLUE score of 11.47%. This is most likely due to the knowledge base feature targeting in English and not being able to properly retrieve answers in Portuguese, as most of the API's answers were empty.

Search engine Whoosh, proved to have satisfactory results for all four of its configurations. Each configuration is represented in the tables and graph by *Whoosh Text HL*, followed by the used analyzer. The *HL* stands for highlights, as we used Whoosh's highlight function to form what we believed could pose as an answer. In addition, *Default* stands for the default analyzer, *LangPt* for the Portuguese analyzer and *NGRAM3* and *NGRAM4*

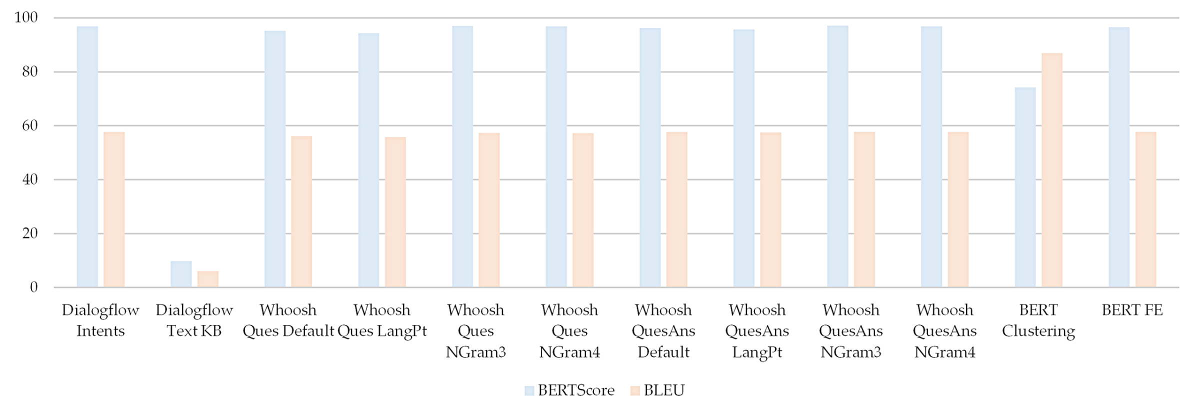


Figure 5.16: Graph with BLEU and BERTScores for AIA-BDE FAQs Domain

Configuration	**BERTScore**	σ
Dialogflow Text KB	20.83	± 0.39
Whoosh Text HL Default	90.73	± 0.02
Whoosh Text HL LangPt	90.96	± 0.02
Whoosh Text HL NGram3	86.61	± 0.04
Whoosh Text HL NGram4	86.26	± 0.05
BERT + Whoosh Text 1	81.67	± 0.05
BERT + Whoosh Text 3	81.39	± 0.06
BERT + Whoosh Text 5	81.85	± 0.05
GPT2 AlticeText	88.88	± 0.04
GPT3 AlticeText	85.66	± 0.04

Table 5.8: BERTScores for Altice Collection of Documents Domain

for the added analyzer 3-gram and 4-gram filters, respectively.

The configuration with a Portuguese analyzer got the highest BERTScore (86.26%), and the remaining three were also amongst the top BERTScore results. As for BLEU, Whoosh's configurations did not present the highest scores, but the 4-gram configuration managed a BLEU score of 74.96%, which we consider acceptable. However, even though we believe Whoosh presented a competitive performance for a baseline, the answers retrieved by the highlights function may not always make sense, as they can be composed by scattered words. Figure 5.18 shows an example of two answers retrieved by this function.

BERT + Whoosh was tested with three configurations, with the only difference being the number of documents retrieved by Whoosh during the document pre-selection phase. Each configuration is represented in the tables and graph by *BERT + Whoosh Text*, which is followed by the number of documents selected by Whoosh.

All BERT + Whoosh combinations got high scores in both evaluation metrics. The results were very similar for the three configurations, with all of them having over 81% BERTScore and around 90% BLEU, the highest BLEU scores, which is not surprising, as it answers with spans of the given context.

GPT-2 got a BLEU score of 57.69% and a BERTScore score of 88.88%. Despite having only a minor difference in BERTScore when compared to the other high scoring configurations, its BLEU scores were a few points lower. This is a consequence of a generating

Configuration	BLEU	σ
Dialogflow Text KB	6.69	± 0.16
Whoosh Text HL Default	21.97	± 0.15
Whoosh Text HL LangPt	23.11	± 0.18
Whoosh Text HL NGram3	3.10	± 0.05
Whoosh Text HL NGram4	2.94	± 0.06
BERT + Whoosh Text 1	0.77	± 0.03
BERT + Whoosh Text 3	0.43	± 0.01
BERT + Whoosh Text 5	0.43	± 0.01
GPT2 AlticeText	13.74	± 0.12
GPT3 AlticeText	2.65	± 0.06

Table 5.9: BLEU Scores for Altice Collection of Documents Domain

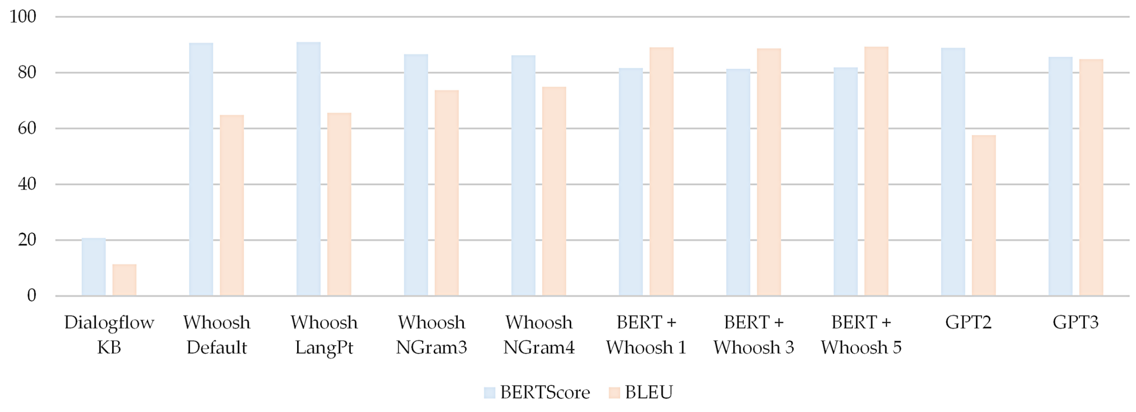


Figure 5.17: Graph with BLEU and BERTScores for Altice Collection of Documents Domain

approach, which, even if transmitting a similar information, might use different tokens than expected.

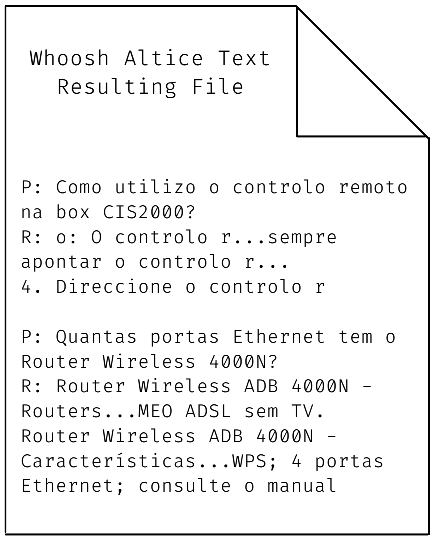
GPT3 similarly had one of the highest BERTScore and the forth highest BLEU, respectively 85.66% and 84.97%. A limitation to this approach is that it comes with a high cost associated, as each request made to the API has a cost[8]. Our experiments were limited due to only having available a small quota to perform requests.

After testing the approaches when it comes to answering questions given a collection of raw text documents, we moved on to experimenting with another set of configurations. This time, we used a domain composed by structured lists of FAQs, with two different datasets. The results are displayed in the upcoming sections.

5.3.4 Time Consumption

In addition to computing the similarity between the retrieved and the correct answers, we also measured the time consumption for each approach and its configurations, dividing this consumption in two phases: domain adaptation and answering time. The times were

[8]https://openai.com/api/pricing/



```
Whoosh Altice Text
Resulting File

P: Como utilizo o controlo remoto
na box CIS2000?
R: o: O controlo r...sempre
apontar o controlo r...
4. Direccione o controlo r

P: Quantas portas Ethernet tem o
Router Wireless 4000N?
R: Router Wireless ADB 4000N -
Routers...MEO ADSL sem TV.
Router Wireless ADB 4000N -
Características...WPS; 4 portas
Ethernet; consulte o manual
```

Figure 5.18: Answers Retrieved by Whoosh Highlights Function Example

measured on Google Colab[9], with the *time* library. For both phases, the moments marking the beginning and end of the phase were obtained, and later subtracted to compute the time spent in between.

Table 5.10 contains the times for this experimentation with the different configurations, using a domain composed by a list of FAQs provided by Altice. The average answering time stands for the average time a configuration takes to retrieve one answer.

Configuration	**Domain Adaptation (s)**	**Answering Time (s)**
Dialogflow Intents	586.87	3.15
Dialogflow KB	106.24	0.94
Whoosh Ques Default	1.32	0.03
Whoosh Ques LangPt	1.19	0.03
Whoosh Ques NGram3	1.57	0.05
Whoosh Ques NGram4	1.83	0.05
Whoosh QuesAns Default	1.21	0.05
Whoosh QuesAns LangPt	1.28	0.04
Whoosh QuesAns NGram3	1.66	0.07
Whoosh QuesAns NGram4	1.74	0.06
BERT Clustering	129.11	91.22
BERT FE	50.42	0.31
GPT2	4306.22	40.05

Table 5.10: Time Consumption Results for Altice FAQs Domain

From looking at the measured times retrieved from experimentation with a list of FAQs provided by Altice, two approaches stand out. The fine-tuning task of GPT-2 proved to be very time consuming, having lasted for more than 4,000 seconds. Furthermore, it presented an average answering time of 40 seconds, which we consider too long. All eight configurations of Whoosh, however, proved to be the fastest in both the domain adaptation and answering times.

As for the remaining configurations, both Google Dialogflow's revealed a somewhat high domain adaptation time, but fair average answering times of up to 3 seconds. BERT's feature extraction configuration, although taking 50 seconds to adapt to the domain, only

[9]https://colab.research.google.com/

took a third of a second to answer each question, in average.

In opposition, the clustering configuration of this approach, as it uses two versions of BERT, i.e., BERTimbau for getting the FAQs representations to form the clusters and BERTimbau QA, fine-tuned with Portuguese SQUAD, to retrieve the answers, proved to be very time consuming. With a domain adaptation time of 129 seconds and an average answering time of 91 seconds, we believe this configuration to be unfeasible. In the future, however, lighter approaches for encoding the FAQs can be tested.

Table 5.11 presents both the domain adaptation and average time per answer measured for each configuration when tested with a list of FAQs from AIA-BDE corpus.

Configuration	**Domain Adaptation (s)**	**Answering Time (s)**
Dialogflow Intents	3037.69	3.14
Dialogflow KB	147.21	0.83
Whoosh Ques Default	4.78	0.10
Whoosh Ques LangPt	4.64	0.08
Whoosh Ques NGram3	5.93	0.17
Whoosh Ques NGram4	7.32	0.15
Whoosh QuesAns Default	4.39	0.16
Whoosh QuesAns LangPt	4.52	0.13
Whoosh QuesAns NGram3	6.02	0.27
Whoosh QuesAns NGram4	8.07	0.23
BERT Clustering	533.31	30.49
BERT FE	289.23	0.43

Table 5.11: Time Consumption Results for AIA-BDE FAQs Domain

When comparing to the results obtained from experimenting with the previous dataset, the time consumed when adapting to a domain increased significantly. This was expected, as the size of the dataset used also increased substantially.

For instance, the domain adaptation time of Dialogflow's intents configuration increased to almost ten times more, adding up to over 3,000 seconds. On the contrary, the knowledge base configuration did not suffer from a significant increase of the time spent in both phases. Similarly, the eight Whoosh configurations did not present a great increase in time consumption.

Both BERT configurations took longer to adapt to the domain, but the clustering configuration reduced the average answering time, presenting an average time per retrieved answer of 30 seconds. As for BERT's feature extraction configuration, it slightly increased the answering time, from 0.31 to 0.43 seconds per answer, which we believe to remain reasonable.

Table 5.12 presents the time consumption of the different approaches and their configurations, when tested with a collection of unstructured text documents provided by Altice.

Once again, Whoosh's configurations managed the fastest times of domain adaptation and answering time, presenting an average time per answer below 0.11 seconds, for every configuration. GPT-3 also had a somewhat fast answering time of 3.4 seconds, but, unlike the other approaches run on Colab, it was run through OpenAI's API. Google Dialogflow proved to have a longer domain adaptation than GPT-3, but a few 0.55 seconds per retrieved answer.

As for BERT Whoosh, the time consumed when retrieving an answer is very different for

Configuration	Domain Adaptation (s)	Answering Time (s)
Dialogflow Text KB	77.18	0.55
Whoosh Text HL Default	0.69	0.03
Whoosh Text HL LangPt	0.64	0.03
Whoosh Text HL NGram3	1.88	0.09
Whoosh Text HL NGram4	3.72	0.11
BERT + Whoosh Text 1	3.13	16.47
BERT + Whoosh Text 3	3.46	52.15
BERT + Whoosh Text 5	3.59	91.42
GPT2 AlticeText	4228.10	11.54
GPT3 AlticeText	3.7	3.40

Table 5.12: Time Consumption Results for Altice Collection of Documents Domain

each configuration. Whilst BERT + Whoosh with one document had an average answering time of around 17 seconds, with three documents as context that time rose to 52 seconds, and with five documents it rose even further to 91 seconds. The domain adaptation time, however, is very similar for each configuration, being around 3 seconds. On the other hand, GPT-2's finetuning time is at least fifty times greater than any other approach when it comes to domain adaptation.

5.4 Discussion

The results obtained from the different configurations by conducting experiments with two lists of FAQs, from Altice and AIA-BDE corpora, led us to believe that one configuration outperformed the rest, namely Whoosh with a 3-gram analyzer. This configuration's results are identical to the configuration with a 4-gram filter, but it proved to consume less time on the domain adaptation phase, even if just a few milliseconds.

In fact, all eight Whoosh configurations presented better results than the remaining configurations, when it comes to the three evaluation metrics and the consumed time. In addition, given that Whoosh is an IR-based approach, the answers it retrieves are always well constructed and possible to understand, as they are copied from information in the domain.

BERT used with feature extraction was also a good candidate, especially due to the need of less effort whilst implementing than Whoosh, but had an inferior BLEU score and number of correct answers, along with a longer domain adaptation time.

After considering all the results obtained by the different approaches when tested with a collection of Altice documents, we concluded that the combination of a traditional IR approach with a fine-tuned neural language model was the most suitable to answer questions related to a domain composed by a collection of raw text documents.

Concretely, the combination of the search engine Whoosh retrieving one document and neural model BERT proved to be our choice when dealing with this type of domain. Even if similar to the other approaches, when it comes to having an high answer quality, this combination also presented a fair average answering time and a normal complexity of implementation. Furthermore, the remaining approaches revealed several limitations such as cost associated to usage, excessive time consumption when adapting to the domain, or not retrieving an intelligible response.

Despite having only experimented with a small collection of 25 documents and 67 questions, on a single domain, we believe that this combination could adapt well to much larger collections, given that BERTimbau QA answers questions in any domain.

This page is intentionally left blank.

Chapter 6

Conclusion

The use of conversational agents can escalate with the size or diversity of the domain, leading to an intensive requirement of resources, human or financial. In this work, we focused on the study of several alternatives to the creation of a conversational agent, that would easily adapt to any domain knowledge structured as a list of FAQs or a collection of plain text documents in Portuguese.

We researched the best approaches to implement an agent capable of answering questions related to a given domain and ended up with a set of relevant approaches that included NLU platform Google Dialogflow, IR-based search engine Whoosh, and fine-tuned neural language models BERT, GPT-2 and GPT-3.

After having the set of selected approaches, we implemented different configurations for each one, some for domains composed by FAQs and the rest for domains composed by collections of documents. Each approach was configured to receive a set of FAQs or documents and retrieve answers to the posed questions. The effort required to implement each configuration was subjectively assessed.

After that, experimentation was conducted with each implemented configuration, by using the same dataset as the domain and asking the same set of pre-defined questions, for each domain type. To test the domain composed by a collection of documents, the used dataset was composed by a set of documents from Altice, referring to telecommunication equipment they provide, and the set of questions was manually created. As for FAQs, we used two datasets, one also provided by Altice and the other was the AIA-BDE corpus, both containing lists of question-answer pairs.

An evaluation of the answers by each configuration was also performed, through the computation of a set of evaluation metrics, in order to compare the performance of each one. In addition, we also measured the time each configuration took to adapt to the domain, that is, the time between receiving the dataset to being ready to answer questions, and the average answering time.

After analyzing the results, we reached the conclusion that a combination of the fine-tuned neural language model BERT with IR-based Whoosh was the most suitable to answer questions given a domain composed by unstructured text documents. This approach presented a high answer quality from the metrics used to evaluate and a fair average answering time, along with a not very demanding complexity of implementation.

On the other hand, given a domain composed by a list of FAQs, we believe that the search engine Whoosh with a 3-gram analyzer is the best fit for the problem. It outperformed all of the remaining configurations, presenting very high scores of answer quality and a very short time consumption, followed by a simple implementation.

Although we experimented with several state-of-the-art approaches, using a tradition IR-based search engine proved to achieve the best results. A possible reason for this is that the variations of the questions used for testing were too similar to the original ones, contained within in the domain. Another feasible reason can be that we did not use the transformer-based models to their full extent, as we, for instance, did not attempt to fine-tune any BERT model to best fit our purpose, relying instead in the already fine-tuned models.

Adding to the information we provide regarding the configuration and performance of the different approaches, we also created a software package[1] containing all the implemented configurations, ready to receive a set of documents or a list of FAQs and answer questions related to the given domain, in Portuguese. We hope that it helps those who are trying to configure or improve a Portuguese question-answering agent, as it provides a set of already implemented options.

In the scope of the contributions, Altice's platform BOTSchool will benefit from, not only with the knowledge we gathered but also the ready-to-use approaches, some of which may be integrated in the formerly built platform in order to further improve it.

Lastly, the part of our work that focuses on domains composed by collections of documents is reported in a scientific paper, published in the proceedings of the 27th Portuguese Conference on Pattern Recognition.

For future work, we would like to further experiment with each approach with larger datasets, as the ones we had available may not represent the real use of a question-answering conversational agent relying on a considerable amount of information. We would also like to implement more configurations for each approach, including combinations of different systems, and to explore alternative models for encoding questions and answers, possibly fine-tuned in domain data.

[1]https://github.com/NLP-CISUC/PT_QA_Agents.git

References

David Ameixa, Luisa Coheur, Pedro Fialho, and Paulo Quaresma. Luke, i am your father: dealing with out-of-domain requests by using movies subtitles. In *International Conference on Intelligent Virtual Agents*, pages 13–21. Springer, 2014.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Paweł Budzianowski and Ivan Vulić. Hello, it's gpt-2–how can i help you? towards the use of pretrained language models for task-oriented dialogue systems. *arXiv preprint arXiv:1907.05774*, 2019.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*, 2018.

Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. Artificial paranoia. *Artificial Intelligence*, 2(1):1–25, 1971.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. *arXiv preprint arXiv:1106.3077*, 2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Hugo Gonçalo Oliveira, João Ferreira, José Santos, Pedro Fialho, Ricardo Rodrigues, Luísa Coheur, and Ana Alves. AIA-BDE: A corpus of faqs in portuguese and their variations. In *Proceedings of 12th International Conference on Language Resources and Evaluation*, LREC 2020, pages 5442—5449, Marseille, France, 2020. ELRA.

Jia-Chen Gu, Tianda Li, Quan Liu, Zhen-Hua Ling, Zhiming Su, Si Wei, and Xiaodan Zhu. Speaker-aware bert for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2041–2044, 2020.

Pierre Guillou. Portuguese bert base cased qa (question answering), finetuned on squad v1.1. 2021.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Mohamad H Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995.

Matthew Henderson, Ivan Vulić, Daniela Gerz, Iñigo Casanueva, Paweł Budzianowski, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrkšić, and Pei-Hao Su. Training neural response selection for task-oriented dialogue systems. *arXiv preprint arXiv:1906.01543*, 2019.

Sara Inácio, Hugo Gonçalo Oliveira, and Catarina Silva. Question answering from technical portuguese documents. In *27th Portuguese Conference on Pattern Recognition*, RECPAD 2021, pages 45–46, Évora, Portugal, October 2021. APRP.

Grishma Jena, Mansi Vashisht, Abheek Basu, Lyle Ungar, and Joao Sedoc. Enterprise to computer: Star trek chatbot. *arXiv preprint arXiv:1708.00818*, 2017.

Dan Jurafsky and James H. Martin. *Speech & Language processing.* Pearson Education India, 2021. 3rd ed. draft.

Oleksandr Kolomiyets and Marie-Francine Moens. A Survey on Question Answering Technology from an Information Retrieval Perspective. *Information Sciences*, 181(24):5412–5434, December 2011.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Anthony Valiant Phillips. *A question-answering routine.* Massachusetts Institute of Technology, 1960.

Chen Qu, Liu Yang, Minghui Qiu, W Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. Bert with history answer embedding for conversational question answering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1133–1136, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

Bhavika R Ranoliya, Nidhi Raghuwanshi, and Sanjay Singh. Chatbot for university related faqs. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1525–1530. IEEE, 2017.

Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, 2010.

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

José Santos, Luís Duarte, João Ferreira, Ana Alves, and Hugo Gonçalo Oliveira. Developing amaia: A conversational agent for helping portuguese entrepreneurs — an extensive exploration of question-matching approaches for portuguese. *Information*, 11(9):428, 2020.

Ruhi Sarikaya. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1):67–81, 2017.

Robert F Simmons, Sheldon Klein, and Keren McConlogue. Indexing and dependency logic for answering english questions. *American Documentation*, 15(3):196–204, 1964.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*, 2020.

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

Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

Jorge A Wagner Filho, Rodrigo Wilkens, Marco Idiart, and Aline Villavicencio. The brwac corpus: A new open resource for brazilian portuguese. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.

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

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

Jason D Williams, Eslam Kamal, Mokhtar Ashour, Hani Amr, Jessica Miller, and Geoffrey Zweig. Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service (luis). In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 159–161, 2015.

Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 516–525, 2016.

Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. Building task-oriented dialogue systems for online shopping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019a.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*, 2019b.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.