

NOVA

IMS

Information
Management
School

MDSAA

Master Degree Program in
Data Science and Advanced Analytics

A Chatbot for Tourism in Porto

Building a Chatbot to support daily Tourists' activities in Porto

Ricardo Francisco Montenegro Dona

Project Work

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

A Chatbot for tourism in Porto
Building a Chatbot to support daily Tourists' activities in Porto

by
Ricardo Francisco Montenegro Dona

Project Work presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Business Analytics

Supervised by
Bruno Jardim, PhD, NOVA Information Management School

November, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, 15/July/2024

Ricardo Montenegro Dona

ABSTRACT

Porto's tourism has doubled in number of visitors from 2009 to 2019 and keeps increasing, making the development of data-efficient solutions an advantage to manage these people's inflows. As a result, the present study focuses on creating a tailored chatbot that supports Porto tourists' daily activities in order to ease their engagement with the city. To reach this, the present research performs several experiments to achieve a high-quality Retrieval Augmented Generation chatbot configuration, focusing on chunk-size, retriever method, *Top-K* results, re-ranking and generative model. The chatbot was developed with *Langchain* and the final architecture is composed by a Hybrid Search Retriever with re-ranking and *Llama3.1-8B* as generative model. The information retrieval component comprises a Dense Vector and a *BM25* retriever, along with *gte-large-en-v1.5* embedding model, and *Top-K* equals to 50. On top of it, query re-ranking is applied throughout *ms-marco-MiniLM-L-6-v2* cross-encoder which retrieves the Top-8 results out of the 50. This solution is backed with 7.382 documents sourced from *VisitPorto*, *TimeOut*, *Civitalis*, *Booking*, *Tripadvisor*, *Visit-Portugal*, and *Wikipedia* that were chunked to a maximum of 3.000 characters per document. The results show that the retriever achieves a *Hit Rate* of 89% and by incorporating it with *Llama3.1-8B*, the RAG model achieves a *Bertscore* and *Semantic Answer Similarity* of 0.82 and 0.64, respectively, when testing its generation quality. Lastly, the present study also demonstrates that RAG models always outperform baseline models.

KEYWORDS

Chatbot; RAG; Tourism; Porto; Llama; Langchain; Hybrid Search Retriever

Sustainable Development Goals (SDG):

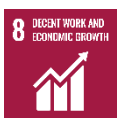


TABLE OF CONTENTS

1. Introduction.....	1
2. Background.....	3
2.1. NLP	3
2.1.1. Transformer	3
2.1.2. Text Representations.....	6
2.1.2.1. Sparse vectors	6
2.1.2.2. Dense vectors.....	7
2.2. Chatbots history.....	9
2.3. Retrieval Augmented Generation.....	10
2.3.1. Evaluation	11
3. Related Work	14
4. Data & Methodology.....	18
4.1. Data Collection and Extraction	18
4.1.1. Data Preparation	21
4.2. Chunk Creation	22
4.3. Embedding Generation	23
4.4. Construction of a Vector Database.....	24
4.5. Prompt Integration and Search Execution	24
4.6. Response Generation.....	27
4.6.1. Generative Model	27
4.6.2. Prompt Engineering.....	28
4.7. Evaluation	29
4.8. Experiments	30
5. Results and discussion	33
6. Conclusions.....	37
7. Limitations and recommendations for future works	39
8. Bibliography.....	40
Appendix A.....	46

LIST OF FIGURES

<i>Figure 1 – Transformer’s Model Architecture (Vaswani et al., 2017).</i>	5
<i>Figure 2 – Retrieval Augmented Generation process visualization.</i>	11
<i>Figure 3 – Diagram showing the methodology applied in the present study.</i>	18
<i>Figure 4 – Shows the distribution of the number of documents per source.</i>	20
<i>Figure 5 – Word Distribution per Document and its descriptive statistics.</i>	22
<i>Figure 6 – Word Cloud that shows the most common words.</i>	22
<i>Figure 7 - Token analysis to ensure the embedding models can handle the document sizes.</i>	23
<i>Figure 8 – Summary of the vector databases specifications used in the study.</i>	24
<i>Figure 9 – Differences of a Dense vector retriever and a Hybrid search retriever</i>	25
<i>Figure 10 – Token analysis to ensure the generator models can handle the prompt sizes.</i>	28
<i>Figure 11 – Number of Questions by category (Generic, Specific, Non-Related).</i>	29
<i>Figure 12 – Hit Rate evolution by Retriever configuration.</i>	33
<i>Figure 13 - Conversation simulating a situation that is related with Porto Tourism.</i>	35
<i>Figure 14 – Conversation simulating a situation that is not related to Porto Tourism.</i>	36
<i>Figure 15 - Word Documents distribution after chunking texts to 3.000 characters maximum.</i>	46
<i>Figure 16 - Word Documents distribution after chunking texts to 7.000 characters maximum.</i>	46

LIST OF TABLES

<i>Table 1 - BoW application example</i>	6
<i>Table 2 – Description of the information in each data source and allocation to each component in the six A’s Framework.</i>	19
<i>Table 3 – Summarization of the experiments to find the best retriever</i>	31
<i>Table 4 - Summarization of the experiments to find the overall model.</i>	32
<i>Table 5 – Results comparing RAG models with baseline models with Llama3.2-3B and Llama3.1-8B</i>	34

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
AIML	Artificial Intelligence Markup Language
ALICE	Artificial Linguistic Internet Computer Entity
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
BOW	Bag of Words
CPU	Central Processing Unit
GPT	Generative Pre-trained Transformers
GPU	Graphics Processing Unit
IR	Information Retrieval
LLM	Large Language Model
LSTM	Long Short-term Memory
MTEB	Massive Text Embedding Benchmark
NLP	Natural Language Processing
RAG	Retrieval-Augmented Generation
RNN	Recurrent Neural Network
RRF	Reciprocal Rank Fusion
SAS	Semantic Answer Similarity
SOTA	State-of-the-art

1. INTRODUCTION

In recent years, the global adoption of Artificial Intelligence (AI) has seen substantial expansion, and this trend is expected to continue. The projected market size for AI in 2024 is estimated to reach 180 billion dollars, with further growth to 826 billion dollars by 2030, meaning a 460% increase (Insights, 2024). One sector that is among the contributors to this situation is the tourism sector, having experienced a substantial rise in the use of artificial intelligence (Li et al., 2021). A key driver of this trend, between AI and Tourism, is AI's subfield Natural Language Processing (NLP), which pertains to the ability of machines to comprehend and communicate with human language (Holdsworth, 2024). More specifically, a visible application of this subfield in tourism is chatbots (Benaddi et al., 2024).

Chatbots are computer programs designed to be capable of having a conversation with human users (Adamopoulou & Moussiades, 2020), and they provide the necessary tools for businesses to increase productivity, as it brings higher operational efficiency (Benaddi et al., 2024). In a business perspective, it allows to reallocate resources, enabling workers to enhance guests' experiences as well as to provide support 24/7. In other words, it allows companies to either be more competitive when hiring people, or to increase their profit margins by reducing costs, or both. Regarding real-life applications of such tools, there is an example that was developed for Orlando's city. This is one of the most visited destinations in the United States of America, and it has an application to support tourists called "Visit Orlando App", which has a chatbot (Benaddi et al., 2024). Another example, Amtraks, a passenger railroad, also has a chatbot to support users to book reservations, plan a vacation, answer frequently asked questions, provide station and route information, as well as policies. Adding to this, an interesting fact that shows the rise in the awareness of the topic is the 10x increase since 2013 in published papers in Scopus Digital Database that were searched under the words "chatbot" and "tourism" (Benaddi et al., 2024).

Even though NLP and chatbots emerged several decades ago, only in recent years the adoption of such technique has been increasing. The existence of better CPUs, GPUs, more efficient algorithms, for instance, Transformers, or even the access to large amounts of data brought significant advancements. A game changer not only in NLP, but in the entire AI ecosystem was the launch of OpenAI's chatbot, ChatGPT, in November 2022. From this point onwards, companies and organizations could rely on chatbots with conversational capabilities and not on intent-based classification methods, which at best could only provide extractive answers (Maryamah et al., 2024). These early bots were fragile and limited in capability, and they were also not capable of memorizing previous conversations, leading to a chatbot that was not able to develop a conversation out of more than one prompt. Even though previous generative language models existed, their lack of accuracy, robustness, and reliability could not bring high added value. Open-AI produced, at that time, a state-of-the-art model that

could better understand user's prompts, easily synthesize information, or even generating coherent content (Akkiraju et al., 2024).

Therefore, given these recent advancements, even though there is existing literature related to domain-specific chatbots applying the most modern techniques, there is a lack of comprehensive research on specific areas, such as tourism. To address this gap, the present research focuses on studying advanced chatbot techniques used in other areas and applying them to the tourism sector, specifically to assist tourists in their general daily activities, such as finding local attractions or suggesting restaurants based on preferences. In this case, the city of Porto is chosen as a case study for two main reasons. First, due to the fact that Porto is the second most visited city in Portugal, with a steadily increasing number of tourists—doubling from 2009 to 2019 (Pavel & Romeiro, 2023). Secondly, because the existing tourist support platform for Porto, *VisitPorto*¹, only offers a basic chat feature reliant on email communication, which reinforces the potential added value of a domain-specific chatbot capable of providing 24/7 assistance without the need for human intervention.

The remaining paper will dive into detail to every aspect surrounding the creation of a chatbot. Section 2 will first provide an overview of NLP and chatbot topics, such as their history and relevant concepts. In Section 3, a comprehensive description of the best and most recent existing chatbots is going to be developed. Section 4, all the data and methodology are going to be detailed. Section 5 will present the results, section 6 the conclusions of the entire work, and finally, section 7, will describe the limitations and recommendations for future works.

¹ <https://visitporto.travel/pt-PT>

2. BACKGROUND

2.1. NLP

Nowadays, Natural Language Processing (NLP) can be defined as a branch of Artificial Intelligence and Linguistics, focused on enabling computers to comprehend words or statements written in human languages (Khurana et al., 2023), ending up being a mix of computer science and linguistics (Johri et al., 2021). Besides, it can be applied in several different ways, such as text classification, feature extraction, summarization, translation, and question-answering (Hugging Face, 2024).

Nevertheless, NLP history dates back to the mid-1930s, when there was no Artificial Intelligence (Johri et al., 2021; Nwakamma Ninduwezuor-Ehiobu et al., 2023). The earliest concepts appeared with translation machines, involving basic mapping of bilingual dictionaries from one language to another using paper tape (Johri et al., 2021). Some years later, in 1950, Alan Turing created what is thought to be the first conceptualization of a Chatbot, by questioning if a computer program could answer questions like humans without people realizing it was a computer (Adamopoulou & Moussiades, 2020). To do this, Turing created a test, also known as Turing test, in which a person needed to have a “blind” communication with a tested subject, and the goal was to identify if that subject was a real person or a chatbot (Turing, 2009). Even though this area evolved in the subsequent years, it was in the 1980’s when a major shift occurred in the field, which was the introduction of machine learning in Natural Language Processing (Adamopoulou & Moussiades, 2020). This allowed more adaptable and scalable models, thus moving beyond the current state, which was rule-based algorithms. In the early 2000’s this shift was intensified with the adoption of Deep Learning, more specifically, different types of neural networks (Khurana et al., 2023). In this century, S. Bengio & Bengio started by stating that neural networks were useful in language modelling and three years later, Y. Bengio et al., developed a Neural Probabilistic Language Model that achieved state-of-the-art results when compared to previous n-grams approaches. From this point onwards, Deep Learning started being widely adopted in the field and *Recurrent Neural Networks (RNN)*, *Long Short-Term Memory (LSTM)*, *Gated Recurrent Neural Networks (GRN)*, firmly established SOTA results in language modeling (Khurana et al., 2023). However, nowadays, the most adopted algorithms in language modeling are not based in any of the approaches mentioned before, but instead in a model created by Vaswani et al. in 2017 called Transformer, which is the one that is going to be further detailed in the next sub-section.

2.1.1. Transformer

Transformers changed the paradigm of NLP not only because it achieved state-of-the-art results in machine translation tasks but also due to its higher computational efficiency. These findings were achieved especially due to two main reasons. The first was because the new

model architecture eschews recurrence, which enables a parallelized training process, resulting in faster trainings. The second is due to the usage of self-attention, since this mechanism is better at capturing long-term dependencies across sentences, which was one of the pain points in previous recurrent models. Regarding Transformer's architecture, it is an encoder-decoder model, in which the encoder is responsible for transforming tokens² into embeddings³ and the decoder is responsible for taking these embeddings and generate an output sequence in an autoregressive way. It is autoregressive in the sense that the model uses its own predictions to predict next tokens. Below, all the Transformer's components are going to be detailed, as well as a visual representation of the entire architecture is presented in *Figure 1*.

Input & Output embeddings are static vector representations that aim to transform the input and output tokens into vectors of n dimensions. While input tokens refer to the tokens that initially enter the model, the output tokens refer to the tokens that are predicted by the model.

Positional Encoding, as the name suggests, is the component that captures information related to the position of the word in a sentence. This is needed because this model, as mentioned before, does not contain recurrence, hence, to avoid losing information related to the sequence order this method was implemented. Regarding how it is implemented, it is a vector that is added to the input and output embeddings which means that both need to have the same dimensions. This vector is created using sine and cosine functions that take into consideration the token's position and each vector's dimension. In each token's position for the even dimensions, sine function is used, for the odd ones, cosine is used. The formulas are written below:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10.000^{\frac{2i}{d_{model}}}}\right), \text{ for even dimensions}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10.000^{\frac{2i}{d_{model}}}}\right), \text{ for odd dimensions}$$

Multi-Head Attention (MHA) is a core component because it allows the network to capture the relationships between inputs disregarding their distance. A possible description of this attention function is to map a query (Q) to a set of key (K)-value (V) pairs. Q is a matrix representing the current focus of attention. K is a matrix comprising all the inputs being

² Elements that represent text, such as words, sub-words, characters, punctuation, etc.

³ Dense vectors representing tokens that are detailed in section 2.1.2.2.

compared to the current focus of attention. V is another matrix used to calculate the output for the current focus of attention. To calculate attention the following function is performed:

$$Attention(Q, K, V) = softmax\left(\frac{QxK^T}{\sqrt{d}}\right)V, d \text{ is the vectors dimension.}$$

The “Multi-Head” part of this component comes from the fact that the original Transformer architecture does not calculate one single self-attention but several in parallel in order to capture different representation subspaces. Adding to this in the decoder part of the model there is an additional note, which is the masking. This step ensures that the model’s predictions for position i can only depend on previous positions.

Position-wise Feed-Forward Networks (FFN) is a neural network that receives the output of the attention mechanism and process each input independently and identically. Mathematically it is presented by:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

Residual Connections (RC) & Layer Normalization (LN) are both techniques that are used to improve models’ performance and training process (Ba et al., 2016; He et al., 2016). These are always applied after each MHA and FFN sublayers. The Residual Connection consists of summing the input of a sublayer with its output, and the Layer Normalization aims, as the name suggests, to normalize the Residual Connections outputs. Hence, the outputs of each mentioned sublayer are always passed through a function such as:

$$LayerNorm(x + Sublayer(x))$$

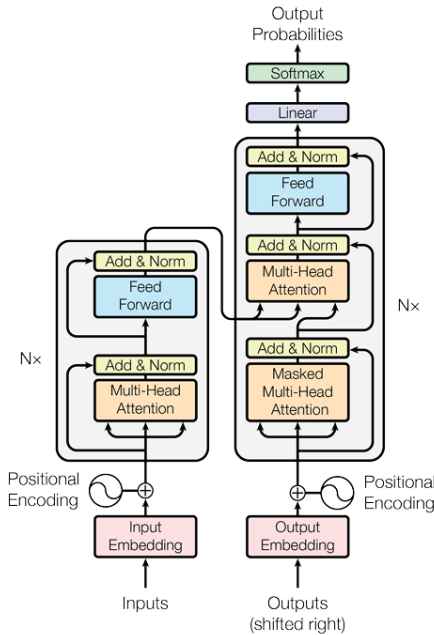


Figure 1 – Transformer’s Model Architecture (Vaswani et al., 2017).

Finally, as already mentioned, the Transformer architecture was one of the most relevant milestones in NLP, however, the first time it appeared, it achieved SOTA results specifically in machine translation and not in text generation. For text generation, the common architecture used is only the Transformer’s decoder (similar to the right block on *Figure 1*) which can be seen in current renowned Chatbots, like *GPT-3* or *Gemini* (*Brown et al., 2020; Gemini Team et al., 2023*).

2.1.2. Text Representations

Another important topic in the NLP space are text representations. Algorithms are not capable of dealing with natural text, meaning that to use them in Language Modelling, text must be transformed into numbers. This transformation has been an active research area for decades in the NLP field and is of extreme importance (*Wang et al., 2020*). For this reason, this section is going to detail the two existing types of text representations: sparse & high-dimensional vectors and dense & low-dimensional vectors.

2.1.2.1. Sparse vectors

One of the oldest and simplest forms of representing text is through *Bag of Words (BoW)* (*Harris, 1954*). This text modelling technique describes the occurrence of words within a document. Words (w_i) are represented in a binary format, where they are labelled as 1 if they are present in a document (d_i), and 0 if they are not. Upon completing this process for the entire documents collection (D) and categorizing every word in vocabulary (V), the ultimate *Bag of Words* is formed, and a document vector is created. *Table 1*, shows an example of *BoW* technique applied. On the rows, is the set of documents, D , composed by d_1 - “NLP is great”, d_2 - “Porto is amazing”, d_3 - “AI is amazing”, and d_4 - “Generative AI is the future”. On the columns, is the entire vocabulary, V , with each word categorized as 1 if present in the respective d_i , or 0, if not. In the end we have the vectorized version of the four documents. For example, d_1 vector is [1,1,1,0,0,0,0,0,0].

	NLP	is	great	Porto	amazing	Generative	AI	the	future
NLP is great	1	1	1	0	0	0	0	0	0
Porto is amazing	0	1	0	1	1	0	0	0	0
AI is amazing	0	1	0	0	1	0	1	0	0
Generative AI is the future	0	1	0	0	0	1	1	1	1

Table 1 - BoW application example.

Nevertheless, *BoW* has some drawbacks, such as, all words having the same importance, word order is discarded, the dimensionality of the vectors, and the inability to capture polysemy.

Regarding the first issue of all words having the same importance, there is a technique that can be used to solve it, which is: *Term Frequency-Inverse Document Frequency (TF-IDF)* (*Jurafsky & Martin, 2014*). This method aims to attribute a higher weight to the rarest words and a smaller weight to the most common. To do this, *Term Frequency (TF)* is multiplied by

the *Inverse Document Frequency (IDF)*, which mathematically is, $TF - IDF = \#W_i * \log\left(\frac{D}{N_t}\right)$. $\#W_i$ is the number of times a word appears in one document and D is the number of documents in the entire document collection, and N_t the number of documents that contain the term. Thus, the lower the N_t , the higher the weight attributed to the word.

The word order problem appears because in *Bag of Words* model the sentence “NLP is great” or “great is NLP” have the same vector. However, there is a solution which is, *n-grams* (Jurafsky & Martin, 2014). An *n-gram* refers to a sequence of n consecutive words/symbols arranged in a specific order. *Table 1* shows a *BoW* with an n -gram of $n = 1$. However, it is possible to use $n > 1$, instead of having single words, it would be possible to have a sequence of it. For example, if $n = 2$, column “the” and “future” could be merged to “the future”. Or if $n = 3$, instead of having “Generative”, “AI”, and “is”, it could be “Generative AI is”. Consequently, this solution yields better text representations since it considers words relations.

Regarding the high dimensionality of the vectors there are techniques to alleviate the problem, such as *Singular Value Decomposition (SVD)* and *Latent Dirichlet Allocation (LDA)* (Wang et al., 2020). However, alternative techniques that will be explained in the next section achieved better results.

Besides *BoW*, there are other solutions to represent words/documents: term-document matrix and term-term matrix (Jurafsky & Martin, 2014). The first method was first mentioned in the context of Information Retrieval in 1971. It has as rows the words of the entire vocabulary and as columns the documents of the entire document collection. Oppositely, to *BoW*, each cell of the matrix is not composed by ones and zeros, but the number of times a specific word occurs in each document. Given this, if the matrix is read on a row level has a word vector, but if read on a column level it has a document vector. The second method, term-term, is a matrix composed only by words on its rows and columns. The row word is considered the target and the column word the context. Thus, each cell counts the number of times the target word appears in the context of the column word.

2.1.2.2. Dense vectors

Another way of transforming text into numbers is throughout dense vectors, also called embeddings that have a predefined number of dimensions, d , and are composed by real-valued numbers mainly non-zero (Jurafsky & Martin, 2014). For example, a vector with four dimensions ($d=4$) could be shown as [0.2; -0.1 ; 0.08 ; 0.5]. This type of text representation has been demonstrating to be more effective than those presented in the previous section in a wide range of NLP tasks (Wang et al., 2020).

There are two types of Dense vectors: static and dynamic (Wang et al., 2020). The first type is when there is a fixed embedding for a given token. Some examples of models that produce them are, *Neural Probabilistic Language Model*, *Continuous Bag-of-Words (CBOW)* and *Skip-*

gram. The second, is related with models that attribute different embeddings to the same words when presented with different contexts, and for this reason are also called contextual embeddings. Some examples are *Cove*, and *BERT*. Below, a detailed explanation of each mentioned model will be presented.

Starting with static vectors, in 2003, Y. Bengio et al. proposed a *Neural Probabilistic Language Model (NPLM)* to tackle especially the curse of dimensionality. The authors proposed a model that simultaneously train a probability function for word sequence and trained distributed representations (embeddings) for each word. In other words, the model was trained to predict the next word given a set of previous words. However, at the same time it trained the word embeddings of 30, 60 and 100 dimensions, separately. To easily understand the impact of this proposal, if the same model were trained with vectors made up of the entire vocabulary size, the vector size was of around 17.000 dimensions. The NPLM architecture can be divided in two parts. The first is the mapping of each word, w_i , to an embedding. The second, the probabilistic function that predicts the next word given a context. Specifically, the algorithm first takes independently each embedding. Then, concatenates the chosen embeddings and pass it through a hidden layer that is further connected to an output layer of size V , which is the size of the vocabulary.

Continuous Bag-of-Words (CBOW) and *Skip-gram* are two models that were created aiming to minimize the computational complexity of the previous method (Mikolov et al., 2013, p. 4). In *NPLM*, the hidden layer adds computational complexity to the model and for this reason the authors decided to test a similar model, but without that hidden layer, being aware that could be not as accurate as the previous one, but for sure would be much more efficient. Thus, in both architectures this hidden layer is removed. Starting with *CBOW*, the goal of the model is to predict the middle word, given a number of previous and future words. The model takes the embeddings of the specified amount of previous and future words, then, oppositely to the concatenation happening in *NPLM*, it averages the embeddings. The reason to be called Continuous Bag-of-Words is because as *BoW* the word order is discarded. Regarding, *Skip-gram* the logic is different. Instead of predicting words given its context, the idea is to predict the context given a word. This is, it learns how to predict the surrounding words within a specific range given a specific text.

The first contextualized vector model, *Cove*, was proposed by McCann et al. (2017). This model uses a deep *Long Short-Term Memory (LSTM)* encoder trained in a sequence-to-sequence machine translation (MT) framework to generate contextualized embeddings. It takes a two-layer unidirectional *LSTM* encoder out of a machine translation model and uses it to encode static *GloVe* (Pennington et al., 2014) embeddings that have already been trained. Afterwards, the encoder outputs contextualized vectors that are capable of being used as input in several NLP tasks. Sentiment analysis, question classification, and question answering were all tasks that could be improved thanks to this approach.

Bidirectional Encoder Representation Transformers (BERT) was a new language representation model presented by Devlin et al. in 2019. This is, as the name suggests, a bidirectional Transformer, also known as Transformer encoder, that considers both left and right context of a token to train contextualized embeddings. The authors argue that the current Transformer-based models at the time restricted embeddings quality since they were unidirectional, which means, they only considered tokens' left context to create the representations. To solve this issue the authors created a different model architecture that achieved state-of-the-art results in several NLP categories. *BERT* was trained under two different strategies: masked language model and next sentence prediction. In the first approach, some input tokens were masked, and the goal is to predict the masked token given its left and right context. Regarding the second, two text-pairs are concatenated, and the goal of the model is to predict if the second text is the next sentence relative to the previous one or not.

2.2. CHATBOTS HISTORY

After reviewing some core concepts of NLP, such as Transformer and the different types of vector representations, chatbots' history follows. Starting with its description, chatbots are computational systems that aim to mimic human conversation capabilities. Although the interest in this topic has been rising in recent years and is expected to keep rising in the following ones (Thormundsson, 2023), its history started around 1950, when Alan Turing first conceptualized the idea of a chatbot, as mentioned in section 2.1.

Even though the concept was created in 1950, the first chatbot being developed was in 1966, and it was called *ELIZA*. Its main goal was to mimic a conversation with a psychotherapist in a therapy session and the way it worked was based on linguistic rules and pattern matching techniques. In other words, it did not understand the conversation, instead its knowledge base was organized in query-answer pairs and the algorithm searched for the query that follows the same pattern as the input and returns the messages associated with it in the knowledge base (Weizenbaum, 1966).

What is thought to be the second chatbot being developed is called *PARRY* and this happened in 1972. This had a different goal when compared with the one created in 1966 since it aimed to mimic a conversation with a paranoid person, and not a therapy session. Even though, the structure of the algorithm was similar and relied in pattern matching rules, *PARRY* is considered an upgrade to *ELIZA's* one since it could also return emotional responses (Colby et al., 1971). These better characteristics lead *PARRY* to be the first chatbot passing the Turing test (Shum et al., 2018).

In 1995 another milestone was achieved in this area, *ALICE* (Artificial Linguistic Internet Computer Entity) was developed. Its architecture were similar do *ELIZA's* one, the two main differences relied on the language they were coded and the number of categories they dealt with. More specifically, *ALICE* was coded in AIML which was a new language purposely

developed for this algorithm. Regarding the amount of question-answer combinations the knowledge base had, there was a significant difference, whereas *ALICE* had around 40.000, *ELIZA* had 200. However, even with a higher knowledge base *ALICE* was still a software that did not have the perception of an entire sentence (Adamopoulou & Moussiades, 2020; Wallace, 2009).

During 2010's decade further steps were taken in the chatbots area. The emergence of Deep Learning algorithms and creation of smartphones and smart voice assistants pushed companies such as Apple, IBM, Google, Microsoft, and Amazon to create solutions like Siri, Watson, Assistant, Cortana, and Alexa, respectively (Caldarini et al., 2022). These chatbots were activated throughout voice commands, and they aimed to support smartphone users in daily topics such as monitoring home automated devices, schedule appointments on the calendar, and send emails.

In November 2022 a significant milestone was achieved by OpenAI when launched ChatGPT (OpenAI, 2022). Even though ChatGPT model specifications are not publicly available, OpenAI's usually employs a transformer-based architecture in their models that is highly effective at understanding long-range textual dependencies, which is indispensable for language activities (Brown et al., 2020; Radford et al., 2019). Besides, this company also takes advantage of the abundant unlabeled data that exists to train the models, in contrast to the previously mentioned Deep Learning techniques that were trained with labeled datasets for a specific task which were much smaller. OpenAI first pre-train the models with trillions of words from unlabeled data and then fine-tune with a smaller labeled dataset to improve its performance on specific tasks (Brown et al., 2020). In the end, all these techniques lead to the creation of a general purpose chatbot capable of having a logical conversation.

To conclude, in the beginning chatbots were created through pattern-matching techniques and nowadays they are built with complex algorithms that can understand and generate coherent content.

2.3. RETRIEVAL AUGMENTED GENERATION

Chat-GPT's model along with other similar ones like *Llama3.1* (Dubey et al., 2024) or *Gemini* (Gemini Team et al., 2023) are undoubtedly very powerful. However, they still face limitations such as hallucinations, low level of model's interpretability, and lack of constant updated knowledge (Akkiraju et al., 2024; Lewis et al., 2020). Hallucinations refer to when the model provides undesired or inaccurate information. The low level of model's interpretability is related to the fact that LLMs are considered "black boxes", in other words, it is hard to explain the reasons for their predictions. Lastly, the lack of constant updated knowledge exists given the fact that the models' parameters are frozen after training they do not contain information that had occurred after it. Nevertheless, it is possible to mitigate all these problems by correctly applying a technique called Retrieval Augmented Generation (RAG) (Lewis et al.,

2020). For this reason, the present sub-section will dive into the fundamental concepts underlying RAG models.

RAG is a technique developed by Meta Researchers in 2020 that when applied enables LLMs to have access to external knowledge. Conceptually, it is a hybrid approach that combines two forms of memory, a parametric and non-parametric memory. The first is related to knowledge stored in the model's parameters, while the second concerns information that is accessed through external sources. To apply Retrieval Augmented Generation two components are necessary, a retriever and a generator model. The retriever is responsible for providing documents conditioned on the user's prompt (steps 1 and 2 in *Figure 2*). The generator model, as the name says, is responsible for generating an answer conditioned on the retrieved documents together with the user's input (step 4 in *Figure 2*). In other words, RAG models basically use the input sequence x to retrieve similar text documents from a vector database z . Then, both x and the additional context retrieved from z are concatenated (step 3 in *Figure 2*) and used to generate the target sequence y . Regarding the way the documents are retrieved, it is throughout embeddings. Briefly, by transforming the user input and the documents stored in the vector database into embeddings it is possible to calculate their similarity by using a distance metric. *Figure 2*, presents the process of a simple RAG model and how the input flows through the architecture.

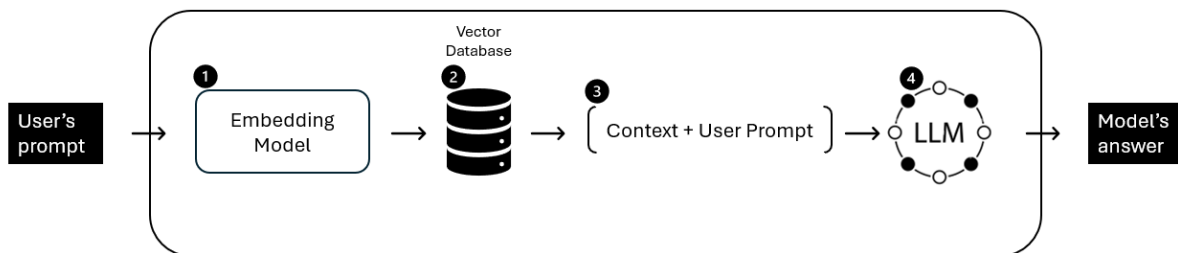


Figure 2 – Retrieval Augmented Generation process visualization.

Lastly, and worth mentioning, research has shown that RAG models can achieve better performances than baseline models alone. Lewis et al. when created this technique tested *BART* model with RAG against *BART* alone and found out that the first was more factual and specific than the second.

2.3.1. Evaluation

This section focusses on the existing methods to evaluate RAG Chatbots. The next paragraphs will detail metrics that are used to evaluate the information retrieval component, the text generation capability of the RAG model, and other that can be used for both situations.

Starting with the ones that are commonly used for information retrieval, these are recall, precision, and hit rate (Gao et al., 2023; Salemi & Zamani, 2024). Recall is a renowned metric

that is useful in this situation, because it allows to measure the retriever's capacity of correctly identifying all the relevant documents. Its formula is presented below:

$$Recall = \frac{\# \text{ of Relevant Documents Retrieved}}{\# \text{ of Relevant Documents in the Corpus}}$$

Another measure is Precision, since it compares the number of relevant documents retrieved against the total number of documents retrieved.

$$Precision = \frac{\# \text{ of Relevant Documents Retrieved}}{\text{Total Number of Documents Retrieved}}$$

Lastly, there is also a *Hit Rate* which is simple and objective to interpret. It attributes 1 when the document retrieved is the one in the ground truth and 0 when not, and in the end the average is calculated. Mathematically is represented by the formula below:

$$HR = \frac{1}{S} \sum_{i=1}^S hit(i), S \text{ is the number of documents}$$

Regarding metrics to evaluate the text generation capability of a RAG model there are *BertScore* (Yu et al., 2024), and *Semantic Answer Similarity (SAS)* (Risch et al., 2021). Starting with *BertScore* (T. Zhang et al., 2019), this is a metric that compare predicted with human-annotated sentences. Oppositely to other widely adopted metrics in the NLP field that are lexical based (ex: *BLUE*), this is semantic based. Lexical based approaches focus on checking if there are exact matches between the predicted and real answer. *BertScore*, instead, uses contextualized embeddings to measure the similarities between tokens. More specifically, each token in the predicted sentence matches with the most equivalent token in the actual sentence, and vice-versa, to compute Recall and Precision, which are subsequently combined to derive the F1 score. Adding to this, the Inverse Document Frequency (IDF) method is added to ensure the model considers token importance. Concerning *Semantic Answer Similarity*, it is based on a cross-encoder and it was initially developed with *sts-b-roberta-large* model. This approach differs from others that use bi-encoders since cross-encoders concatenate the predicted with the human-annotated sentence separated by a special token and directly output a similarity score.

Finally, one approach that evaluates RAG models holistically is called RAGAS. RAGAS framework (Es et al., 2024), has two main differences when compared with the metric presented above. First, it does not need human-annotated datasets, which is a common situation when building RAG models. Secondly, it evaluates the entire RAG architecture, meaning that not only the output's quality is assessed but also the retriever's one.

This framework's application does not need any human-annotated dataset because the first step when applying it, is to create a synthetic dataset with the help of an LLM by using the data stored in the vector database that is going to be used to feed the RAG model. Regarding

the method behind evaluating the retriever and the generator component of the model it is based in three components *Faithfulness*, *Answer relevance*, and *Context Relevance*.

- *Faithfulness (F)*: assess if the Chatbot response was based on the retrieved context. In other words, if the answer can be inferred from the context it can be considered faithful. This is performed throughout an LLM that picks each question(Q)-answer(A)-context(C) triple from the synthesized dataset and first extracts a set of statements (S) related with a given A and Q. Afterwards, an LLM is used again to classify if each created S can be inferred from C(Q). In the end, the F score is calculated by dividing the number of statements that can be inferred from the context (V) and the total number of statements (S).

$$F = \frac{V}{S}$$

- *Answer relevance (AR)*: evaluates if an answer addresses the user's question. In other words, if the generated answer aligns with the question. To this the authors developed a strategy that picks an answer of the dataset, and again, with the help of an LLM it generates n hypothetical questions related with the response. Then an embedding model is used to vectorize the generated question and the real question in order to calculate the cosine similarity between each generated question and the real question. In the end AR score is calculated by averaging all the cosine similarities calculated.

$$AR = \frac{1}{n} \sum_{i=1}^n sim(Q, Q_i)$$

- *Context Relevance (CR)*: measures if the context only has answer's related content. Thus, the score is penalized when it has redundant information. To put this in practice an LLM is used once more to extract the relevant sentences present in the context of a given question. Then, to calculate CR, the number of sentences in the context that can help to answer a specific question (H) is divided by the total amount of sentences in the context (T).

$$CR = \frac{H}{T}$$

To summarize RAGAs, this presents a holistic methodology to evaluate Retrieval Augmented Generation solutions, assessing if the context is being used correctly, if the answer responds to the question, and if there is redundant content in the context.

3. RELATED WORK

In recent years, chatbots have been widely adopted in various industries due to their potential to improve customers' interaction and enhance users' experience. This section will review existing literature on chatbot development, with a focus on RAG-based approaches, which have been implemented across diverse domains like IT, health, and education.

NVIDIA researchers recently published a paper in which it presents a framework for building effective RAG-based chatbots (Akkiraju et al., 2024). In this paper three chatbots for internal usage were presented, *NVinfo bot*, *NVhelp bot*, and *Scout bot*, as well as the guidelines to consider when developing such softwares. The first, *NVinfo bot* is designed to provide support to employees in topics pertaining to the company's internal information, for example parking information. Regarding its specific characteristics, it is fed with approximately 500M documents, which corresponds to more than seven terabytes of content, being also complemented with intranet search. Adding to this, it handles diverse data formats and enforces document access controls. The technological infrastructure is composed by *LangChain*⁴, a vector database for retrieval, a large language model, and a custom web-UI. Regarding the second, *NVhelp Bot* its primary objective is to offer IT assistance and HR benefits information. Comprising over two thousand multi-modal documents including text, tables, photos, pdfs, and html pages, this bot was constructed utilizing a comparable technology stack to *NVInfo bot*, but with a reduced data volume. The study also shows that two assessments were performed to this bot, one with *RAGAs* framework to evaluate accuracy, and the second to evaluate latency. The assessment was made with several generator models, such as *Llama2-70B*, *Llama3-70B*, *Mistral-8x7B*, among others, and the results showed that *Llama3-70B* outperforms the remaining ones in answer quality and, even though it does not have the best latency, it has acceptable levels. The last one, *Scout bot*, handles questions about financial earnings from public sources, being fed with approximately four thousand multi-modal documents containing text, tables, pdfs, and html pages. The tech stack is similar to the previous ones, including the usage of *LangChain*, a vector database, an LLM as a generator model, *RAGAs* evaluation framework, and a web-UI. An additional test performed when developing *Scout bot* was the fine-tuning of the *e5-large* embeddings model. However, the results showed that it did not produce any significant improvements.

More considerations outlined by NVIDIA pertain to aspects to be considered in order to optimally design a chatbot. NVIDIA's developers "noticed that metadata enrichment, chunking, query rephrasal and query re-ranking stages" significantly influence the Chatbot responses quality. The conducted study showed that response generation quality is highly dependent on retrieval relevancy, which is, in turn, highly dependent on document metadata enrichment, chunking, and query rephrasal. Consequently, a grid search-based approach to find the optimal parameters for chunk-sizes, prompt design, and chunk re-ranking strategies

⁴ Platform to develop LLM apps.

was adopted. Another important note mentioned in the study is related with hybrid search⁵. The authors noticed that vector databases exhibit limitations in effectively handling matching entities (ex: locations, people's names, etc) and by joining lexical search with vector search could enhance the retrieval process.

Singh et al. (2024) created a chatbot named *MindGuide* designed to serve as a mental health assistant for individuals requiring support. This application was entirely created using *LangChain* and the most relevant aspects highlighted in the paper are the generator model that was used, the implementation of conversation memory, and the deployment strategy. Regarding the first topic, GPT-4 was the only option tested. Concerning the second topic, the developers built the chatbot in a way that it remembers previous users' inputs. The memory type used was *ConversationBufferMemory*, a *Langchain's* memory variant that stores the entire conversation history and uses as context in every input query. Finally, the RAG model was deployed using a python library called *Streamlit*. This enables every developer to quickly build and share LLM apps, being Chatbots one of the specific applications this library provides.

Microsoft Researchers published a paper that assesses the quality of *Llama 2*, *GPT-3.5*, and *GPT-4* in responses to agricultural-related questions by using human examinations as reference points (Silva et al., 2023). Besides testing the baseline models, the authors also tested them together with RAG and/or Ensemble Refinement⁶ (ER). The review encompassed agricultural examinations and reference datasets sourced from Brazil, India, and the United States. The results indicate that *GPT-4* performed exceptionally well, attaining a 93% accuracy rate on examinations for renewing agronomy credentials, surpassing the 88% accuracy demonstrated by *GPT-3.5*. Besides, the results also showed that every tested model was better when using RAG and/ or ER.

Vakayil et al. (2024) designed a chatbot to provide discreet yet reliable assistance to victims of sexual harassment, fulfilling the demand for mental health care in an accessible, sensitive, and empathic way. To develop this Chatbot the authors used once more *Langchain* library and applied Retrieval Augmented Generation. The source data was gathered from PDFs containing legal information related to the topic which was chunked and consequently stored in a vector database called *ChromaDB*. Regarding the used embeddings to search for relevant data, *BERT base*, was used and the chosen distance metric was cosine similarity. In relation to the generator model, *Llama2-7b* was the selected one. Adding to this, a prompt template was used emphasizing what was the goal of that Chatbot, as well as a predefined answer in case anything in the process went wrong.

Q. Zhou et al. (2024) leverage RAG and LLMs, to develop a conversational agent dedicated to gastrointestinal diseases, *GatroBot*. To build the model the authors started by filling a vector database with 25 guidelines on gastrointestinal diseases from the Chinese Medical Journal and

⁵ Combination of lexical and semantical search in the retrieval process.

⁶ prompt strategy inspired in chain-of-thought and self-consistency (Singhal et al., 2023)

40 recent gastrointestinal literature sources from the China National Knowledge Infrastructure. When adding the gathered information to the database, the authors chose a chunk-size of 512 characters and the number of retrieved chunks was the top 3 ones. The used embedding model was a fine-tuned version of *gte-base-zh* from *Alibaba DAMO Academy*. An interesting fact from this study is the fine-tuning of the embeddings model, and especially the results that achieved. The fine-tuned model outperformed OpenAI's *text-embedding-ada-002*, and baseline *gte-base-zh*. The *Hit Rate* exceeded at least 18% when compared with the other models. Concerning the generator model that was selected, it was *GPT-3.5-turbo*. Finally, *GastroBot* was evaluated under two methodologies. The first was *RAGAS* and the second was human judgment. Besides, human judgment was not only used to evaluate the RAG model itself, but also to compare it against baseline versions of *Llama2* and *ChatGLM-6B* and *Qwen-7B*, to assess if the Retrieval Augmented Generation process was adding value. Lastly, *Llamaindex* python's library was used to design and implement the desired architecture.

Vidivelli et al. (2024) presented a chatbot that aimed to help patients with teeth-related issues. To do this, the authors presented a framework based on *LangChain* and Retrieval-Augmented Generation. Oppositely, to all the papers mentioned above, instead of using a baseline model, *TinyLlama-1.1B-Chat-v1.0* was chosen and fine-tuned with health-care related data. Regarding the selected vector database, it was *ChromaDB* and before adding information to it the data was chunked. In the framework it is also mentioned that OpenAI embeddings were used to perform the retrieval process, even though it is not explicitly mentioned which model. Finally, and distinctly from previous approaches, the authors used *ROUGE* and *BLUE* metrics to evaluate the final model. They did it by comparing the *ROUGE* and *BLUE* results of a baseline model, a fine-tuned model, and a fine-tuned model enhanced with Retrieval-Augmented Generation (RAG). In the end, the RAG model outperformed all the remaining ones. Adding to this, not only the answers of the chatbot were evaluated but also its inference time. Three metrics were used in this case, *Time to First Token (TTFT)*, *Time Per Output Token (TPOT)*, and *Latency*, which is the sum of *TTFT* and *TPOT* time the number of predicted tokens. Even though the RAG model showed a lower latency is important to notice that a limit of 200 tokens per answer was set, which is not mentioned for the fine-tuned model alone.

Alsafari et al. (2024) aimed to develop a teaching assistance chatbot. Besides developing a RAG model, it also built an intent-based version to compare both solutions. In the end, RAG demonstrated a higher capability of handling diverse queries. Therefore, given that the goal of this thesis is to build a Chatbot enhanced with RAG and in this study it was the best solution out of the two tested, the RAG version will be the only one detailed. The created teaching assistant was developed again with *Langchain* library, and it uses as generator model, OpenAI's *GPT-3.5-Turbo*. To build the entire RAG infrastructure, the author used *FAISS* vector database, it used *text-embedding-ada-002* as the embedding model, and in it uses

ConversationBufferMemory, to include chat history when answering to users' prompts. In the end, the evaluation of the final Chatbot was executed with a test set composed of 20 questions, evaluating the correctness, relevance, accuracy, clarity, depth of thought, and overall helpfulness. Two different methods were applied to assess all these metrics: an automatic one, and a human-based judgement one. The first uses *scoring evaluator* method from *Langchain*, which instructs an LLM to evaluate your model's predictions on a specified scale. The second methodology was implemented with the help of two judges that were specialized in Data Mining and Text Analytics, which was the field for which the model was being created.

Maryamah et al. (2024) developed a conversation agent to help students to access information easily, specifically in higher education levels. The authors opted for a RAG-based approach built with *Langchain* that ended being created with a *ChromaDB* vector database composed by data from 111 documents, a chunking size of 1000 tokens, OpenAI Ada embeddings, cosine similarity as a distance metric to retrieve relevant chunks, *Top-K* document retrieval equal to 2, *GPT-3.5-Turbo* as a generator model with temperature equals to 0, and *ConversationWindowBufferMemory* to consider chat history when answering follow-up questions. Nevertheless, several mentioned components were chosen after specific assessments of both the retriever and the generator model. Starting with the retriever, three different embedding models were tested OpenAI's Ada, Google Embedding-001 and *MPNet-Multilingual*, as well as two different distance metrics cosine similarity and *Maximum Marginal Relevance (MMR)*. To evaluate the retriever a test set was created with 90 query-context pairs and the precision and recall were calculated for all the possible combinations of embedding models and distance metrics. Lastly, *ROUGE* and *BLUE* were used to evaluate the generator models. For this, a test set containing 60 questions-answer pairs was used and lead to the conclusion of a much superior performance of *GPT-3.5-turbo* when compared with *Gemini-pro*.

To conclude, throughout the presented literature it was possible to highlight several recent applications of domain-specific chatbots across various fields, including healthcare, education, and IT using Retrieval Augmented Generation to enhance chatbots' capabilities. Furthermore, it was also possible to observe the different technical stack that is used, such as chunking methods, retrievers, embedding models, vector databases, and even the tools used to implement the chatbot (*Langchain* and *LLamaindex*). However, an important outcome of this review is also the lack of research related to RAG models specifically in the tourism sector, and especially, in Porto's tourism context. Consequently, as previously mentioned, the present study intends to face this gap by applying these methods to create a chatbot for the tourism sector.

4. DATA & METHODOLOGY

This section aims to present the adopted methodology to develop a chatbot to support Porto tourists (illustrated in *Figure 3*). Its structure follows the defined framework designed by Jeong (2023) to develop simple RAG models, which is based on the subsequent six steps:

1. Data Collection and Extraction: this initial phase involves gathering both structured and unstructured data.
2. Chunk Creation: the source data is divided into smaller, manageable units called chunks.
3. Embedding Generation: the text chunks are then transformed into dense vector representations.
4. Construction of a Vector Database: this step involves building a vector database from the embedded chunks.
5. Prompt Integration and Search Execution: this is the retrieval component, in other words, it is the step in which the retriever characteristics are set.
6. Response Generation: the retrieved information is added to a customized prompt template to generate an answer.

Additionally, to define the evaluation methodology and all the experiments performed the two following steps are added to the process:

7. Evaluation: test set creation, and the metrics used to evaluate the model.
8. Experiments: summary of the performed experiments.

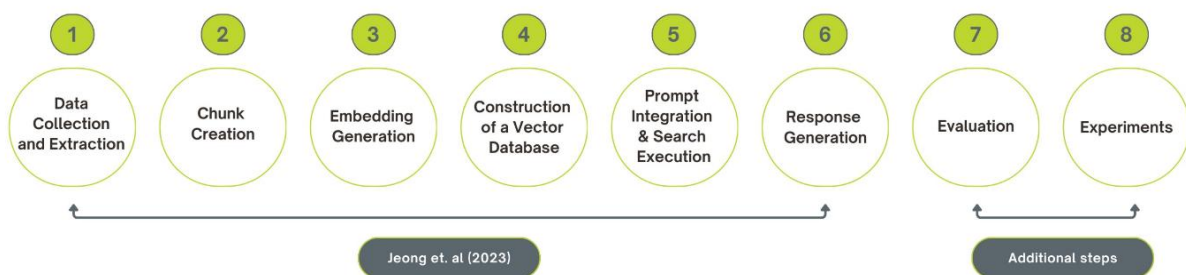


Figure 3 – Diagram showing the methodology applied in the present study.

4.1. DATA COLLECTION AND EXTRACTION

The data collected to feed the Chatbot was structured around Buhalis (2000) 6 As framework, which states that successful touristic destinations comprise six components, which are: Attractions, Accessibility, Amenities, Available Packages, Activities, and Ancillary Services (Benaddi et al., 2024).

- Attractions: points of interest that enhance the attractiveness of a destination, such as, natural parks, man-made buildings, or cultural institutions.
- Accessibility: transportations systems availability, for instance, airplane, taxi, metro, etc.
- Amenities: touristic infrastructures like restaurants, hotels, hostels, and bars.
- Activities: existence of relevant experiences and tours.
- Available Packages: pre-arranged trip packages that include hotel, transportation, and activities.
- Ancillary Services: essential daily services such as medical care and banking facilities.

These elements together form the holistic experience that tourists seek when visiting a destination, thus by collecting data according to this framework’s principles, all aspects of a tourist's journey in Porto are covered. For this reason, information from seven websites were collected, which are presented in the following table:

Website	Description	Framework Component
VisitPorto ⁷	General website to support Porto tourists.	Attractions, Accessibility
TimeOut ⁸	Magazine for discovering the best of each city.	Amenities, Activities
Civitalis ⁹	Online platform for booking tours and day trips.	Attractions, Available Packages, Activities
Booking ¹⁰	Information about Porto hotels.	Amenities
Tripadvisor ¹¹	Contains information about Porto restaurants.	Amenities
Wikipedia ¹²	Universal encyclopedia with Porto-related data.	Attractions
Visitar-Porto ¹³	Information related to events and festivals.	Activities

Table 2 – Description of the information in each data source and allocation to each component in the six A’s Framework.

Additionally, more documents were created manually based on the websites mentioned above, aiming to prepare the corpus in case of general users’ prompts, such as, “what are the best restaurants in Porto?”, “what are the main events in Porto?”, or “Do me a one day trip”. In the end, all the above websites and manual data represent a total of 7.382 documents collected. The predominant sources in terms of quantity of information are *Tripadvisor* and *Booking*, indicating a substantial amount of content pertaining to restaurants and accommodations. Nevertheless, despite being in smaller number, the other sources

⁷ www.visitporto.travel/en-GB

⁸ www.timeout.com/porto

⁹ www.civitalis.com/en/

¹⁰ www.booking.com

¹¹ www.tripadvisor.com

¹² <https://www.wikipedia.org/>

¹³ <https://www.visitar-porto.com/en/>

adequately fulfill all the outstanding data requirements pertaining to events, festivals, tours, transportation, history, attractions, safety, among others. *Figure 4* illustrates the distribution of the number of documents collected per data source.

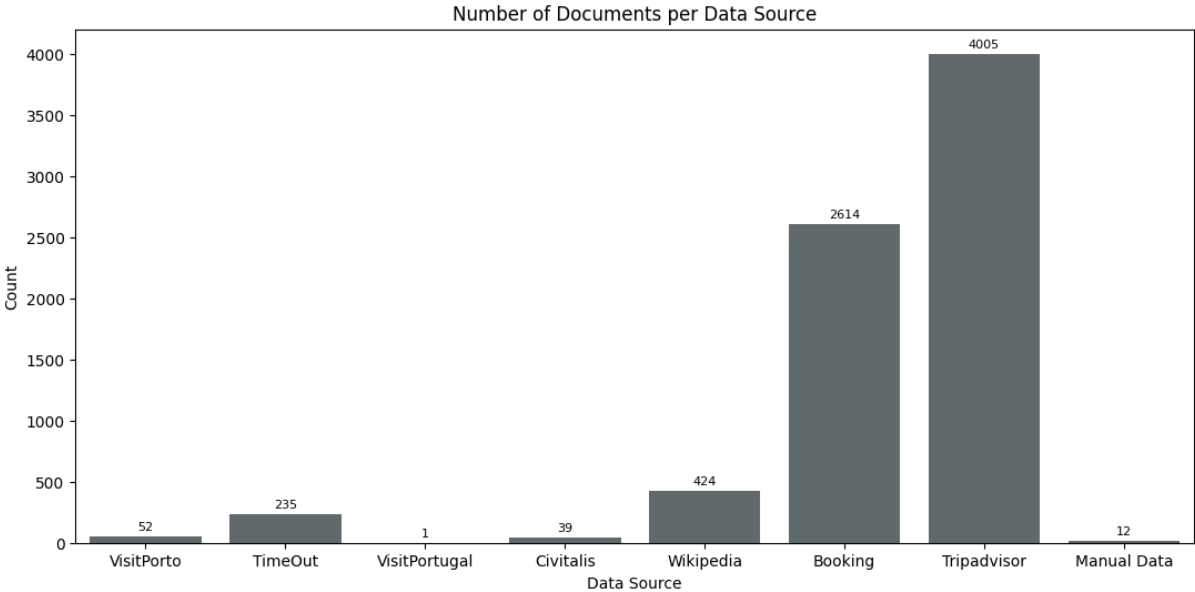


Figure 4 – Shows the distribution of the number of documents per source.

Regarding the data collection process, a variety of techniques were applied to collect data from different sources. This process is divided essentially into three groups, due to the specificities of each website. The first is composed by *VisitPorto*, *Timeout*, *Civitalis*, *Visitar-Porto*; the second by *Wikipedia*; and the third by *Booking* and *Tripadvisor*.

VisitPorto, *Timeout*, *Civitalis*, and *Visitar-Porto* data were collected applying web scrapping techniques, throughout Python’s *BeautifulSoup* library. Adding to this, *Timeout* needed an extra procedure due to the dynamic nature of the webpage. In this case, a web crawler was also used, and it was applied by using Python’s *Selenium* library.

Wikipedia data was accessed using *LangChain*. *LangChain*’s python library, more specifically, its *WikipediaLoader* method, facilitates the extraction of data from *Wikipedia* articles.

Booking and *TripAdvisor* information was extracted using a platform called *Apify*¹⁴, a cloud-based scraping platform that offers pre-built crawlers for commercial sites like the ones mentioned, thus enabling efficient extraction of structured data.

¹⁴ <https://apify.com/>

4.1.1. Data Preparation

Data preparation steps are highly associated to each data source, since these websites usually have specific characteristics that are similar among their different web pages. For this reason, all the details will be described below by data source.

From **VisitPorto** website 52 web pages scrapped and their data were collected. During its extraction process the title, sub-title, description, additional information, price, address, and url were obtained individually. Consequently, the information was merged, and a piece of text was created with all the information mentioned above. In the end, paragraph tokens (“\n”) and unnecessary spaces were removed.

Timeout enriched the corpus with 235 documents from 14 web pages. The title and the description of each place detailed in this source was extracted and merged into one piece of text per place. Additionally, given the possibility of a tourist to ask general questions, such as “tell me 5 traditional restaurants in Porto”, manual text was created summarizing several spots in one sentence.

Visit-Portugal contributed with one web page that had a general program to do in Porto. The process was similar to the ones above, first extracted the title and description, and then joined the information in a logical form.

Civitalis added information to the corpus of 42 web pages. Given that the data was already structured after scrapping, only a few regex treatments were performed.

Wikipedia augmented the corpus with 424 documents. Given that *Langchain's* method, *WikipediaLoader*, was used, the data was already organized. The only additional steps were to remove unnecessary or empty sections that were present on the page, such as *References*. This was done to reduce noise in the data and keep it focused on the main topic.

Tripadvisor and **Booking** data were extracted following a similar procedure, resulting in a similar data treatment. Initially the data was highly fragmented in columns, thus the first step was to choose the ones that provided factual and necessary knowledge about each element. Then all the selected information was merged into a template that was carefully tested and analyzed before its final version. In this way, all the text related to restaurants, cafes, and hotels are structured in a concise and objective form.

Finally, in every data source, duplicates as well as empty documents were removed. To conclude *Figure 5* was created to better understand the descriptive statistics and the distribution of words per document, and *Figure 6* to visualize the most common words in the Corpus.

Appendix A. As expected, as the chunk-size increases, the documents tend to have more words and additionally, the 7.000 character split almost does not impact the original words per document distribution seen in *Figure 5*, thus it can be almost interpreted as testing a RAG model without dividing documents.

4.3. EMBEDDING GENERATION

Concerning the embeddings model tested, it was *gte-large-en-v1.5* (X. Zhang et al., 2024). This model was chosen under three main considerations: model size, context length, and Massive Text Embedding Benchmark (MTEB¹⁶) leaderboard. The first condition had to be taken into account due to the existing hardware constraints. The second is related to a point already mentioned in the chunk creation sub-section. Since there are documents that are worth to be retrieved in its complete structure, a model that could deal with high input lengths were the most indicated one, for this reason one with a high maximum input length (8.192) was chosen. Lastly, under the two previous constraints the best possible model on MTEB leaderboard was selected. This one is in 28th place in the overall performance and in the 19th place in the retrieval task.

To ensure that the desired embedding model is aligned with its maximum input length constraints, an analysis to the original data before chunking was executed. To perform this evaluation, the tokenizer of the chosen model, *gte-large-en-v1.5*, was used. *Figure 7* demonstrates that this model can be applied given that its maximum input length is 8.192 and the maximum number of tokens in one document is 2.477.

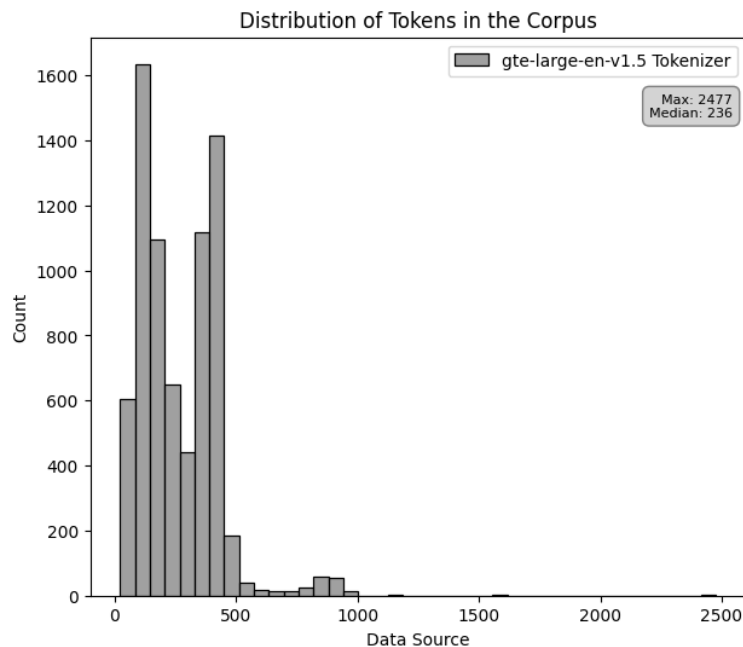


Figure 7 - Token analysis to ensure the embedding models can handle the document sizes.

¹⁶ <https://huggingface.co/spaces/mteb/leaderboard>

4.4. CONSTRUCTION OF A VECTOR DATABASE

Regarding the vector database, *ChromaDB*¹⁷, an open-source database, was used. The database is designed to store embeddings alongside the original documents and index them effectively. It enables fast search operations and similarity measurements by positioning chunks within a vector space.

In the present study, four different databases were created. The first two only differ in the chunking sizes applied to the documents in the corpus, which are defined in sub-section 4.2. The embedding model employed was the one referenced in the preceding sub-section, and in relation to the utilized distance metric, it was cosine similarity as Vakayil et al. (2024) and Maryamah et al. (2024) did in their studies. This metric is represented by:

$$COS(A, B) = \frac{A \times B}{||A|| \times ||B||}$$

Regarding the two remaining vector databases, these were created under the same conditions as the other two, the difference lies in additional data treatment steps applied to the corpus. However, these actions will be detailed in the next sub-section, as well as their reason. *Figure 8* summarizes the specifications of all the databases used in the present study.

	<i>DB₁</i>	<i>DB₂</i>	<i>DB₃</i>	<i>DB₄</i>
Corpus	Normal	Normal	Additional treatment	Additional treatment
Embedding model	<i>gte-large-en-v1.5</i>	<i>gte-large-en-v1.5</i>	<i>gte-large-en-v1.5</i>	<i>gte-large-en-v1.5</i>
Chunk-size	3 000	7 000	3 000	7 000
Distance Metric	Cosine Similarity	Cosine Similarity	Cosine Similarity	Cosine Similarity

Figure 8 – Summary of the vector databases specifications used in the study.

4.5. PROMPT INTEGRATION AND SEARCH EXECUTION

Traditional RAG approaches utilize dense vector retrievers as introduced by Lewis et al. (2020) and for this reason this was the first tested retriever type. Nevertheless, Ma et al. (2021) stated that joining a dense vector retriever with a sparse one, for instance, BM25, which is going to be further detailed below, improved information retrieval results. Additionally, Akkiraju et al. (2024) also recently suggested that enhancing traditional RAG models with other information retrieval algorithms that focus on lexical search could improve its results, especially in cases when the retrieval process needs to handle matching entities (e.g. places), which is in the scope of this study. For this reason, a second retriever is also tested, which is a combination of a dense vector retriever with a sparse retriever, usually referred to as hybrid search retriever (Akkiraju et al., 2024). *Figure 9* illustrates the differences between a vector store versus a hybrid search retriever which are going to be detailed below. Additionally, literature

¹⁷ <https://www.trychroma.com/>

also mentions another technique that is capable of enhancing the retriever’s quality, which is query re-ranking, and for this reason it is also tested in the present study (Akkiraju et al., 2024; Erak et al., 2024).

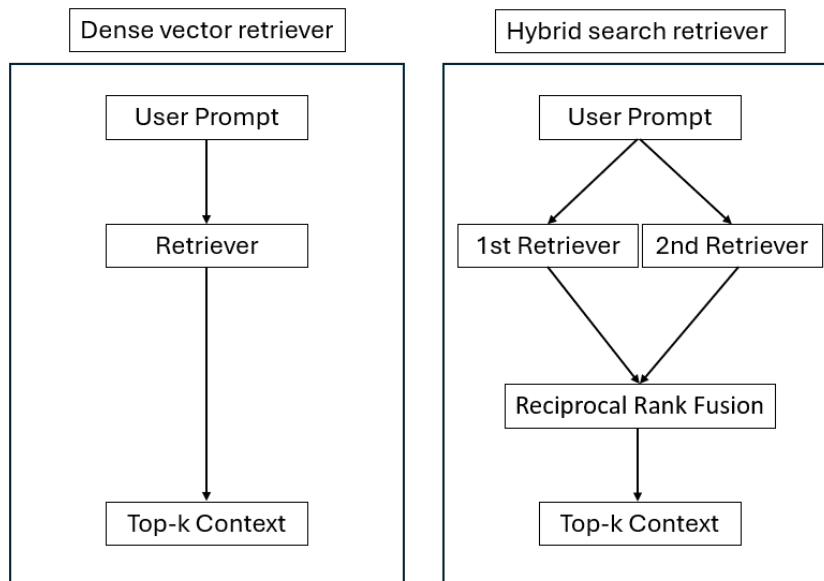


Figure 9 – Differences of a Dense vector retriever and a Hybrid search retriever

Starting with the dense vector retriever, it was implemented using the *vector store retriever*¹⁸ method from *Langchain*. This “is a retriever that uses a vector store to retrieve documents” and uses the search methods already implemented by the vector store (Langchain, 2024). This option was tested using the vector databases mentioned in 4.4, with *Top-K* ranging from 3 to 9.

In relation to the hybrid search retriever, *Langchain* library was used again, but this time the *Ensemble retriever*¹⁹ method was implemented. Briefly, this approach fetches documents from different retrievers and combines them. In this study, a *vector store retriever*, explained in the previous paragraph, was used together with a *BM25 retriever*²⁰ which relies on sparse vectors. This last method is based on BM25 ranking function which is widely employed in the information retrieval area (Svore & Burges, 2009), and is observed for instance in Ma et al. (2021) and Karpukhin et al. (2020) studies. This function integrates a combination of three primary document characteristics: term frequency, document frequency, and document length. Oppositely to semantic retrievers, BM25 retrieval function ranks a collection of documents according to the occurrence of query terms within each document while weighing their importance. Adding to this, it also considers the document length to mitigate a possible bias in the results due to the text size. Mathematically, it is presented as follows:

¹⁸ https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/vectorstore/

¹⁹ https://python.langchain.com/docs/how_to/ensemble_retriever/

²⁰ <https://python.langchain.com/docs/integrations/retrievers/bm25/>

$$BM25_{SCORE} = \sum_{t \in Q} \log \left(\frac{N - df_t + 0.5}{df_t + 0.5} \right) \times \frac{(k_1 + 1) \times tf_{td}}{k_1 \times \left(1 - b + b \times \left(\frac{L_d}{L_{avg}} \right) \right) + tf_{td}}$$

- Total Number of Documents (N): The total number of documents in the collection.
- Document Frequency (df_t): The number of documents that contain the term.
- Term Frequency (tf_{td}): The number of times the term occurs in a document.
- Document Length (L_d): The total number of terms in the document.
- Average Document Length (L_{avg}): The mean length of all documents in the collection.
- Tuning Parameters:
 - k_1 : Controls the saturation of the term frequency contribution to the score (set to 1.5).
 - b : Controls the degree of length normalization for the document (set to 0.75).

Given that in a hybrid retriever there are contexts from two retrievers to combine, the next step is to merge their results which is done by applying Reciprocal Rank Fusion (Cormack et al., 2009). This technique basically ranks each retrieved document according to its position in each retrieval algorithm. The formula is represented as:

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r_d}$$

- R: number of retrieval algorithms.
- r_d : ranking of document d in a retrieval algorithm.
- k : smoothing factor that prevents a single retriever from dominating the results (set to 60).

Considering that BM25 ranking function relies on exact matching, and the python library being used split words throughout spaces, if no data treatment is performed, the words “Porto” and “porto.” would be considered different by this algorithm. Consequently, additional preprocessing to the corpus was completed for the hybrid retriever. To start all the texts were lowercased, then, the punctuation presented in the documents, such as “.”, “!”, “,”, “:”, “-”, “”s”, “(”, and “)” were removed when attached to a letter and followed by a space. It was only removed under these circumstances because in other situations could transform data into noise, for instance, in prices, which usually have cents separated by a “.”. Adding to this, given that $RRFscore$ is also sensitive to the sentence structure, if both retrievers output similar contexts but one is lowercased and the other is not, the formula will not be correctly applied because it will consider distinct contexts. Therefore, in order to have BM25 working efficiently and to correctly join both dense and sparse retrievers’ results, vector database DB_3 and DB_4 mentioned in sub-section 4.4 were created.

Lastly, another method was implemented following the retrieval process, which was query re-ranking. This technique is based on a cross-encoder, which, unlike bi-encoders that embed each text independently and calculate their similarity afterward, processes pairs of text together by concatenating the two sentences separated by a special token. The last method is often more accurate but slower, which contrasts with the bi-encoder which are faster but less accurate. Consequently, it is a common technique used in the retriever to ensure that the most relevant contexts are ranked first. The present study follows Erak et al. (2024) in the usage of *ms-marco-MiniLM-L-6-v2* cross-encoder to implement this method.

4.6. RESPONSE GENERATION

4.6.1. Generative Model

In this research two generative models will be tested: *Llama3.1 - 8B* (Dubey et al., 2024) and *Llama3.2 - 3B* (Meta, 2024). The reason for not testing bigger models like *Llama3.1* models with 70B and 405B parameters is due to hardware and budget constraints.

The above-mentioned models are two of the most recent ones in the market and they share some similar characteristics. Both have an input context window of 128.000 tokens, both are multilingual, and both were released in 2024. Additionally, *Llama3.2 - 3B* is derived from *Llama3.1* models family. In other words, Meta pruned *Llama3.1-8B* and distilled *Llama3.1-70B* to create *Llama3.2 - 3B* (Meta, 2024). Even though the 3.2 version is state-of-the-art in its model's class, 3.1 version keeps having better results (Dubey et al., 2024; Meta, 2024; J. Zhou et al., 2023). These similarities and characteristics make both models worth comparing since the results of the study will also show to which extent a lighter model to which pruning and distillation was applied can achieve similar or better results than a bigger one.

The context window dimension was an important factor when choosing the generative models to test, since this study follows a high chunking-size strategy to ensure that bigger documents that must be retrieved in its total form to effectively contextualize a query are considered. Therefore, an analysis to the original documents before chunking was performed to ensure that their number of tokens are aligned with the maximum input length constraints of the chosen generative models. *Figure 10* shows that 50% of the documents in the corpus do not exceed 230 tokens by applying the tokenizers of the different considered models. Besides, the maximum document size is 2.508 in the worst scenario. Given that the maximum context length for both models is 128.000, *Figure 10* allows to conclude that the document sizes seem adequate for the models being used even if the retriever's *Top-K* parameter is high.

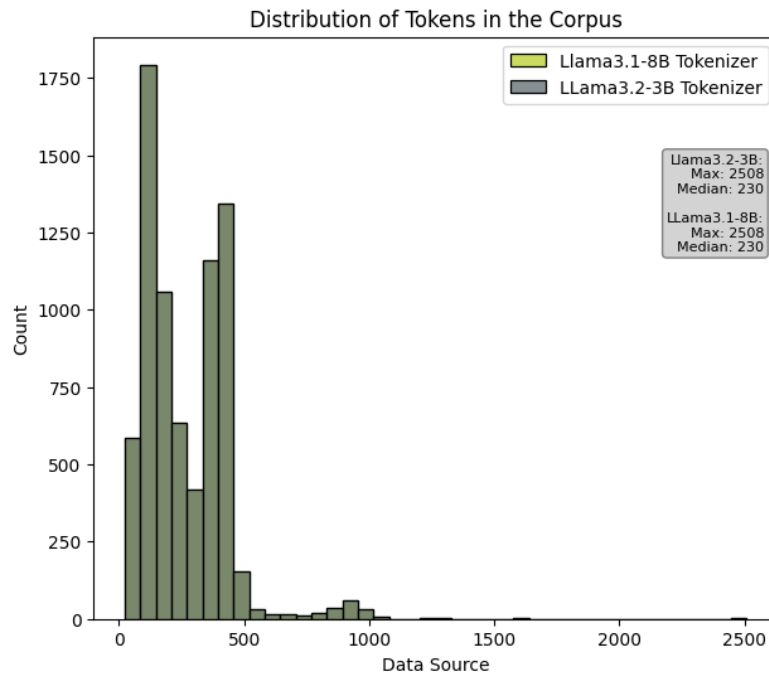


Figure 10 – Token analysis to ensure the generator models can handle the prompt sizes.

Lastly, regarding *temperature* parameter, this was set to 0. It follows what Maryamah et al. (2024) did in its study as well as the reasoning that the developed Chatbot must be the most factual and objective possible.

4.6.2. Prompt Engineering

The prompt template entering the LLM was written following the best prompt engineering guidelines presented by Chen et al. (2023): clear and precise prompts and role-prompting. Additionally, a specific answer was prepared in case the user’s question is not related to Porto tourism, similar to Vakayil et al. (2024) study. The created template is written below:

System: This is a Chatbot that only answers to questions related to Porto (Portugal) Tourism. More specifically, to topics related to attractions, accessibility, amenities, activities, available packages, and Ancillary Services. When not specified by the user assume the question is related to Porto.

If the question is not about Porto Tourism just write: "I am sorry, but my knowledge only allows me to help you with Porto Tourism topics. Can I help you with something related to Porto Tourism?"

Answer to the user's question objectively, using correct syntax and based on context written below:

{context}\n

User: {input}

4.7. EVALUATION

This sub-section aims to detail all the components used to evaluate the chatbot. It will start by describing the test set created and then elaborating about the metrics used to evaluate both the retrieval component and the RAG model responses, as in Maryamah et al. (2024) study.

To evaluate the created architecture a test set composed of question-context-answer triples were created. A total of 71 question-answer pairs and 118 context pieces are present in the dataset. The content in it is divided into three categories: generic, specific, and non-related. Generic content was suggested by ChatGPT and relates to topics such as “What are the best places to visit in Porto?” and “What to do in Porto in one day?”. Specific topics were created by the author and refer to questions like “When is São João festival?” or “What is the Check in time at Vincci Bonjardim?”. Lastly, non-related aspects refer to questions like “Tell me about Algarve history?” or “What are the best attractions in Lisbon?”, since the goal in these situations is to specifically answer “I am sorry, but my knowledge only allows me to help you with Porto Tourism topics. Can I help you with something related to Porto Tourism?”. *Figure 11* illustrates the distribution of questions by category showing 39 generic questions, 25 specific, and 7 non-related ones. The reason for the non-related questions category being only 10% is to avoid biasing the overall result, because the answers to these are probably going to have higher similarity scores when compared to the reference answer. Lastly, note that non-related questions were not used to evaluate the retrieval component since there is no correct context to be retrieved when the user’s question is not related to the topic.

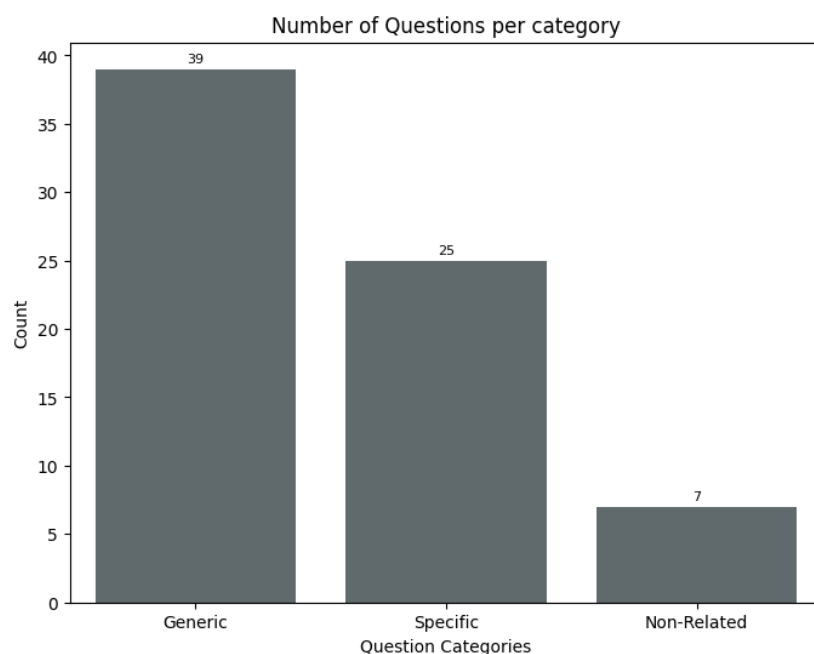


Figure 11 – Number of Questions by category (Generic, Specific, Non-Related).

Regarding the utilized metrics, it follows the measures used in Doan et al. (2024) study: *Hit Rate*, *Semantic Answer Similarity*, and additionally *BertScore* (described in section 2.3.1). The first, *Hit Rate*, was used to evaluate the retrieval component, while the second, and third were used to compare the reference and predicted answers.

4.8. EXPERIMENTS

The experiments in this study are divided in two. In the first part, a grid search is implemented to find the best retriever configurations between a dense vector, a hybrid search retriever, and a hybrid search retriever with re-ranking. Afterwards, four models are tested, a baseline *Llama3.2-3B*, *baseline Llama3.1-8B*, *Llama3.2-3B with RAG*, and *Llama3.1-8B with RAG*.

In the first phase 56 trials are performed to find the best retriever, which are summarized in *Table 3*. Note that when the retriever is applied together with re-ranking *Top-N* and *Top-K* are considered. In this case, the first relates to the initial number of contexts retrieved by the retriever, while the second is related to final chosen contexts after re-ranking. When there is no re-ranking *Top-K* is considered the final number of contexts retrieved. All these 56 configurations results are compared throughout *Hit Rate*, in the end, the one with the highest score is considered the best configuration.

id	Retriever	Chunk-size	Top-N	Top-K	Embeddings model
1	Dense Vector Retriever	3 000	-	3	gte-large-en-v1.5
2	Dense Vector Retriever	3 000	-	4	gte-large-en-v1.5
3	Dense Vector Retriever	3 000	-	5	gte-large-en-v1.5
4	Dense Vector Retriever	3 000	-	6	gte-large-en-v1.5
5	Dense Vector Retriever	3 000	-	7	gte-large-en-v1.5
6	Dense Vector Retriever	3 000	-	8	gte-large-en-v1.5
7	Dense Vector Retriever	3 000	-	9	gte-large-en-v1.5
8	Dense Vector Retriever	7 000	-	3	gte-large-en-v1.5
9	Dense Vector Retriever	7 000	-	4	gte-large-en-v1.5
10	Dense Vector Retriever	7 000	-	5	gte-large-en-v1.5
11	Dense Vector Retriever	7 000	-	6	gte-large-en-v1.5
12	Dense Vector Retriever	7 000	-	7	gte-large-en-v1.5
13	Dense Vector Retriever	7 000	-	8	gte-large-en-v1.5
14	Dense Vector Retriever	7 000	-	9	gte-large-en-v1.5
15	Hybrid Search Retriever	3 000	-	3	gte-large-en-v1.5
16	Hybrid Search Retriever	3 000	-	4	gte-large-en-v1.5
17	Hybrid Search Retriever	3 000	-	5	gte-large-en-v1.5
18	Hybrid Search Retriever	3 000	-	6	gte-large-en-v1.5
19	Hybrid Search Retriever	3 000	-	7	gte-large-en-v1.5
20	Hybrid Search Retriever	3 000	-	8	gte-large-en-v1.5
21	Hybrid Search Retriever	3 000	-	9	gte-large-en-v1.5
22	Hybrid Search Retriever	7 000	-	3	gte-large-en-v1.5
23	Hybrid Search Retriever	7 000	-	4	gte-large-en-v1.5
24	Hybrid Search Retriever	7 000	-	5	gte-large-en-v1.5
25	Hybrid Search Retriever	7 000	-	6	gte-large-en-v1.5
26	Hybrid Search Retriever	7 000	-	7	gte-large-en-v1.5
27	Hybrid Search Retriever	7 000	-	8	gte-large-en-v1.5
28	Hybrid Search Retriever	7 000	-	9	gte-large-en-v1.5
29	Hybrid Search Retriever & Reranking	3 000	50	3	gte-large-en-v1.5
30	Hybrid Search Retriever & Reranking	3 000	50	4	gte-large-en-v1.5
31	Hybrid Search Retriever & Reranking	3 000	50	5	gte-large-en-v1.5
32	Hybrid Search Retriever & Reranking	3 000	50	6	gte-large-en-v1.5
33	Hybrid Search Retriever & Reranking	3 000	50	7	gte-large-en-v1.5
34	Hybrid Search Retriever & Reranking	3 000	50	8	gte-large-en-v1.5
35	Hybrid Search Retriever & Reranking	3 000	50	9	gte-large-en-v1.5
36	Hybrid Search Retriever & Reranking	7 000	50	3	gte-large-en-v1.5
37	Hybrid Search Retriever & Reranking	7 000	50	4	gte-large-en-v1.5
38	Hybrid Search Retriever & Reranking	7 000	50	5	gte-large-en-v1.5
39	Hybrid Search Retriever & Reranking	7 000	50	6	gte-large-en-v1.5
40	Hybrid Search Retriever & Reranking	7 000	50	7	gte-large-en-v1.5
41	Hybrid Search Retriever & Reranking	7 000	50	8	gte-large-en-v1.5
42	Hybrid Search Retriever & Reranking	7 000	50	9	gte-large-en-v1.5
43	Hybrid Search Retriever & Reranking	3 000	70	3	gte-large-en-v1.5
44	Hybrid Search Retriever & Reranking	3 000	70	4	gte-large-en-v1.5
45	Hybrid Search Retriever & Reranking	3 000	70	5	gte-large-en-v1.5
46	Hybrid Search Retriever & Reranking	3 000	70	6	gte-large-en-v1.5
47	Hybrid Search Retriever & Reranking	3 000	70	7	gte-large-en-v1.5
48	Hybrid Search Retriever & Reranking	3 000	70	8	gte-large-en-v1.5
49	Hybrid Search Retriever & Reranking	3 000	70	9	gte-large-en-v1.5
50	Hybrid Search Retriever & Reranking	7 000	70	3	gte-large-en-v1.5
51	Hybrid Search Retriever & Reranking	7 000	70	4	gte-large-en-v1.5
52	Hybrid Search Retriever & Reranking	7 000	70	5	gte-large-en-v1.5
53	Hybrid Search Retriever & Reranking	7 000	70	6	gte-large-en-v1.5
54	Hybrid Search Retriever & Reranking	7 000	70	7	gte-large-en-v1.5
55	Hybrid Search Retriever & Reranking	7 000	70	8	gte-large-en-v1.5
56	Hybrid Search Retriever & Reranking	7 000	70	9	gte-large-en-v1.5

Table 3 – Summarization of the experiments to find the best retriever.

Following, the second phase is composed by four possibilities that are presented in *Table 4*. Besides the goal of achieving the best configuration possible, this study also aims to quantify the increase in quality the RAG Chatbot has when compared with the baseline model. These are going to be tested with *BertScore* and *SAS*.

id	Retriever	Generator Model
1	Best Retriever Configuration	Llama3.2-3B
2	Best Retriever Configuration	Llama3.1-8B
3	-	Llama3.2-3B
4	-	Llama3.1-8B

Table 4 - Summarization of the experiments to find the overall model.

5. RESULTS AND DISCUSSION

The goal of this section is to present and analyse the ability of the developed Chatbots to respond to questions that Porto tourists may have and choose the best one. It will present the results of the best Retrieval configuration and RAG model, as well as discuss the results.

Starting with the retrieval component, *Figure 12* summarizes the *Hit Rate* results by retriever configuration and *Top-K*. More specifically, it compares the results of the Dense Vector retrievers, Hybrid Search retrievers, and Hybrid Search retrievers with re-ranking. In the end, three retrievers achieved 89% *Hit Rate*, which was the highest accomplished. However, there is one solution that prevails among the three since it is capable of getting to that result with less contexts retrieved, thus, producing less noise. This was the Hybrid Search Retriever with re-ranking composed by Alibaba’s embeddings model *gte-large-en-v1.5*, data chunked with a maximum of 3.000 characters, *Top-N* equals to 50 and a *Top-K* equals to 8. Additionally, *Figure 12* also illustrates the fact that the Hybrid Search retriever with re-ranking have always better results than the other configurations when the *Top-K* exceeds 5, which is in line with Erak et al. (2024). Furthermore, the present study concludes the same that Ma et al. (2021) and (Akkiraju et al., 2024) stated in their studies, which is that Hybrid Retrievers improve information retrieval results when compared with Dense Vector Retrievers.

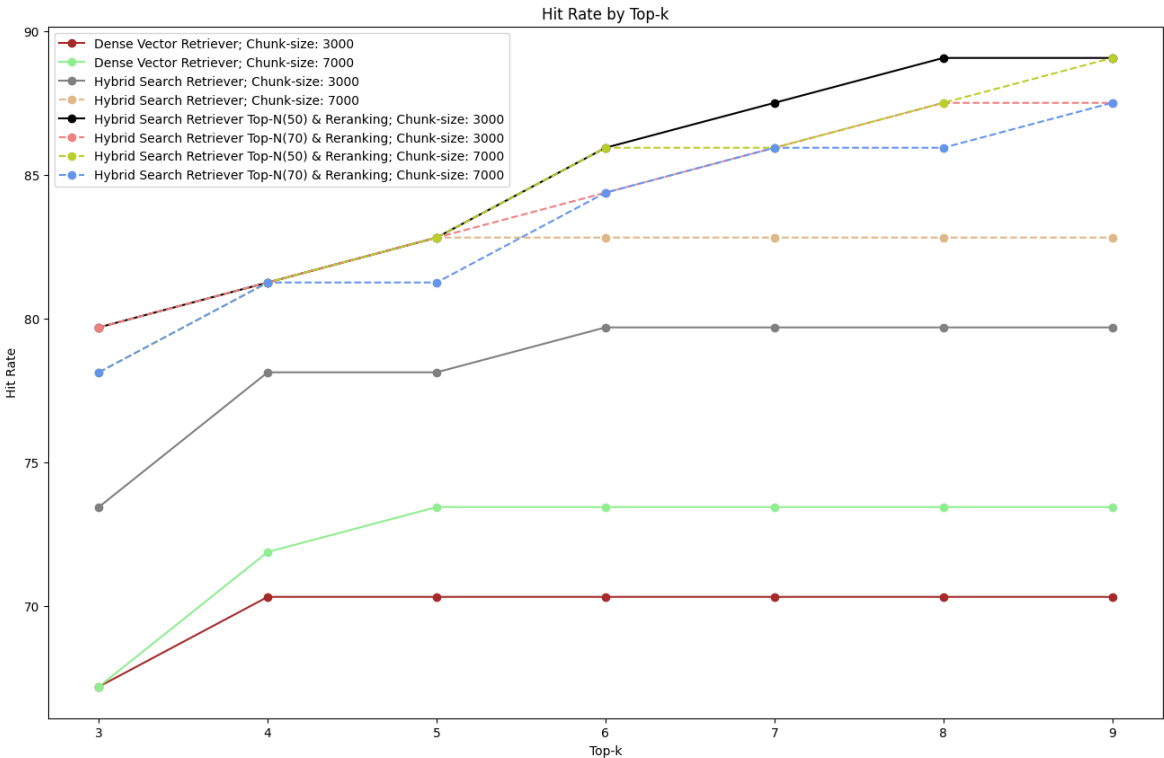


Figure 12 – Hit Rate evolution by Retriever configuration.

Following the decision of the best retriever it was used together with the two already mentioned generator models, *Llama3.2-3B* and *Llama3.1-8B*, to examine which overall

configuration achieved better results. In the end, as illustrated in *Table 5* by *BertScore* and *SAS* metrics, the best configuration uses *Llama3.1-8B* together with RAG. Furthermore, it is also possible to conclude the applying RAG always adds value and that both metrics used are always in accordance.

Generator Model	BertScore	SAS
Llama3.1-8B - baseline	0,72	0,52
Llama3.1-8B w/ RAG	0,82	0,64
Llama3.2-3B - baseline	0,72	0,54
Llama3.2-3B w/ RAG	0,80	0,61

Table 5 – Results comparing RAG models with baseline models with Llama3.2-3B and Llama3.1-8B.

These findings contribute to face the gap in the literature consisting in the lack of advanced chatbots applied in the tourism sector. By exploring these types of applications in various economic sectors such as technology (Akkiraju et al., 2024), telecommunications (Erak et al., 2024), medicine (Singh et al., 2024; Q. Zhou et al., 2024), and education (Alsafari et al., 2024; Maryamah et al., 2024), it was possible to bring to tourism, insights about RAG, hybrid retrievers, re-ranking, evaluation methods, among others. With this, the created solution, besides helping to ease the engagement of Porto with its tourists, it can be also used to increase the effectiveness of the tourism offer in Porto, since the entities responsible for managing this destination could leverage the information flows that arise from tourists’ questions, to customize the offer for the visitors (Liberato et al., 2018). The solution created in the present study can specifically handle this situation, given that by monitoring the users’ conversation with tools like *Langsmith*²¹, it enables the city’s management to constantly have updated information regarding the activities which tourists have more interest. Another aspect that can be explored with the creation of this chatbot is its potential commercial use. In case this solution is taken into production, this chatbot can be used to create an “advertisement fee” for every business that want to be mentioned in its answers. For instance, if a user asks about tours related to wine cellars, the wine cellars that wanted to appear in the chatbot’s answer would have to pay for it. Therefore, the chatbot can be used not only as a tool that eases the engagement of tourists and Porto city, but also as a data collection tool that allows the tourism agencies to take more informed decisions, and even in a business perspective. Concluding, this work contributes to Porto’s tourism industry by elevating consumer engagement, personalizing services, and improving operational efficiencies (Benaddi et al., 2024; Gu, 2024) by using the created solution.

In the end, a front-end to test the RAG model was created with *Streamlit*²². Below a simulation of a conversation between the user and the Chatbot is illustrated in *Figure 13*, as well as a

²¹ Platform to monitor every step of the LLM-powered application lifecycle.

²² <https://streamlit.io/>

situation in which the Chatbot refuses to answer since it is not a question related to Porto tourism, which is represented in *Figure 14*. Note that when taking the model to production every user query goes through the same data treatment the corpus had when creating the hybrid search retriever (see section 4.5 for more information). Adding to this, besides the expected context from the retrieval process, it also considers chat history to ensure that a user can have follow-up answers. To implement this, *create_history_aware_retriever*²³ method from *Langchain* was used.

Porto Tourism Assistant



what are the best restaurants to eat francesinhas in Porto?



Based on the information provided, here are some of the top-rated restaurants to try a traditional Porto-style Francesinha in Porto:

1. **Barrigas:** A familiar and welcoming space that serves the best Francesinhas since 1996. (Rating: 3.5/5)
2. **Café Santiago:** Try your first Francesinha at this popular spot, which is considered one of the best places to eat it in Porto. (No rating provided)
3. **Taberna das Fontainhas:** Enjoy a traditional Porto-style sandwich while listening to Portuguese Fado music. (Rating: 0/5 - no reviews available)
4. **Cufra:** A mix of brewery and seafood restaurant that serves a special Francesinha with a seafood-infused sauce. (No rating provided)
5. **Vaccarum Francesinhas:** A new and exciting restaurant located in the heart of Porto, offering a warm atmosphere and delicious tapas boards. (Rating: 4.5/5)
6. **Taberna Londrina S. Joao:** Home to the best Francesinha, a special type of sandwich typical of Portugal. (Rating: 3.5/5)
7. **O Afonso Restaurante:** A restaurant with many years of experience serving traditional Porto-style dishes, including the famous Francesinha. (Rating: 4.0/5)

Please note that ratings may change over time, and it's always a good idea to check current reviews and make reservations in advance, especially for popular restaurants like these.



what is the adress of the second restaurant?




The address of the second restaurant, "Cafe Santiago", is:


Rua de Passos Manuel 226 Porto 4000-382 Portugal

Figure 13 - Conversation simulating a situation that is related with Porto Tourism.

²³ https://python.langchain.com/docs/how_to/qa_chat_history_how_to/

Porto Tourism Assistant

 what is the best hotel in Lisbon?

 I am sorry, but my knowledge only allows me to help you with Porto Tourism topics. Can I help you with something related to Porto Tourism?

How can I help you?



Figure 14 – Conversation simulating a situation that is not related to Porto Tourism.

6. CONCLUSIONS

This thesis presents a robust methodology for developing a chatbot tailored to support tourists visiting Porto. The methodology besides aligning with Jeong's (2023) simple RAG model framework, progressing through data collection, data processing, embedding generation, vector database construction, prompt integration & search results, and response generation, it also enriches it by adding more advanced components such as hybrid search retriever and re-ranking.

Another important point is the fact that the data feeding the chatbot was meticulously structured around Buhalis's (2000) "6 A's" framework, encompassing attractions, accessibility, amenities, activities, available packages, and ancillary services. This was sourced from diverse websites like *VisitPorto*, *TimeOut*, *Civitalis*, *Tripadvisor*, *Visitar-Porto* and *Booking*, covering a comprehensive range of tourist needs. Additionally, to improve retrieval efficiency and response generation, data preparation techniques were included, ensuring that the information was coherent and pertinent.

Following the corpus creation, throughout several experiments, various configurations for chunk-sizes, *Top-K*, *Top-N*, and retrieval mechanisms were explored. Findings showed that a hybrid search retriever with re-ranking, which combines a dense vector search with a sparse retriever (BM25) and a cross-encoder, improved retrieval accuracy significantly when compared with simple Hybrid Search retrievers (except when *Top-K* is lower or equal to 5), and dense vectors retrievers alone. In the end, the optimal configuration, which achieved the highest *Hit Rate*, is constituted by *gte-large-en-v1.5* embeddings model, chunking-size of 3.000 characters, a *Top-N* equal to 50 and a *Top-K* parameter set to 8. When combined with *Llama3.1-8B* as the generative model, this setup outperformed *Llama3.2-3B* under the same conditions in key performance metrics, such as *BertScore* and *Semantic Answer Similarity*, demonstrating a more contextually accurate approach. However, given that the results do not differ significantly, from a business perspective it is worth comparing both prices when deciding which one to use. Finally, another output from the present study is that both generative models are better when coupled with RAG.

In the end, the chatbot was implemented and deployed using *Streamlit*, providing a user-friendly interface for testing. It effectively handled both Porto-related inquiries and non-relevant prompts by directing users appropriately. This dual-response mechanism ensures that users receive relevant and precise information about Porto tourism, while also aligning with the chatbot's intended scope.

To summarize, this research highlights the effectiveness of a RAG-based chatbot framework, leveraging advanced retrieval and generative models for a domain-specific use case. The model configurations and optimizations introduced here offer significant insights for future

RAG chatbot development aiming to enhance user engagement in tourism and other sector-specific applications.

7. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

This study encountered five limitations that impacted the development and assessment of the chatbot and three possible future works.

Regarding the limitations, the two primary ones are interrelated, being the first the selection of English embeddings only and the second the Chatbot's evaluation being solely in English. Regarding the first, it might impact the retrieval accuracy and limit the chatbot's effectiveness for non-English-speaking users, which is not ideal given that multilingual embeddings would better address the diverse linguistic needs of all international visitors. In case there is budget or hardware, *NV-Embed-v2* embedding model would be a good solution. The second limitation is connected with the first in the sense that by using English embeddings if the test set had multiple languages the Chatbot's performance would be poor. Another limitation was the absence of an external judge in the evaluation process. While automated metrics like *Semantic Answer Similarity* and *BertScore* provided objective insights, the lack of human independent evaluators may limit a full understanding of the Chatbot's quality. For this reason, to take the final model to production would be advisable to have independent judges providing qualitative feedback. Additionally, chat history was not tested in this study and an effective conversational chatbot for tourism ideally should retain some memory of past interactions to maintain context in extended conversations. Finally, the lack of query rewriting technique implies the possibility of improving the current results.

Concerning future research, the three topics that could be focused on are: fine-tuning the generator model, fine-tuning the embeddings model, and develop a survey providing insights about the best vector databases. It would be interesting to compare results of RAG models against RAG models with fine-tuning or simply pre-trained models fine-tuned. In relation to the vector databases' survey, since these are emerging with the increased usability of RAG models and Agents it would add value to summarize the main advantages and disadvantages of each one.

8. BIBLIOGRAPHY

- Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2. <https://doi.org/10.1016/j.mlwa.2020.100006>
- Akkiraju, R., Xu, A., Bora, D., Yu, T., An, L., Seth, V., Shukla, A., Gundecha, P., Mehta, H., Jha, A., Raj, P., Balasubramanian, A., Maram, M., Muthusamy, G., Annepally, S. R., Knowles, S., Du, M., Burnett, N., Javiya, S., ... Boitano, J. (2024). *FACTS About Building Retrieval Augmented Generation-based Chatbots*. <https://arxiv.org/html/2407.07858v1>
- Alsafari, B., Atwell, E., Walker, A., & Callaghan, M. (2024). Towards effective teaching assistants: From intent-based chatbots to LLM-powered teaching assistants. *Natural Language Processing Journal*, 8, 100101. <https://doi.org/10.1016/j.nlp.2024.100101>
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). *Layer Normalization*. <https://arxiv.org/pdf/1607.06450>
- Benaddi, L., Ouaddi, C., Jakimi, A., & Ouchao, B. (2024). A Systematic Review of Chatbots: Classification, Development, and Their Impact on Tourism. *IEEE Access*, 12, 78799–78810. <https://doi.org/10.1109/ACCESS.2024.3408108>
- Bengio, S., & Bengio, Y. (2000). Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks*, 11(3). <https://doi.org/10.1109/72.846725>
- Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3(6). <https://doi.org/10.1162/153244303322533223>
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020-December, 8. <https://arxiv.org/abs/2005.14165>
- Buhalis, D. (2000). Marketing the competitive destination of the future. *Tourism Management*, 21(1), 97–116. [https://doi.org/10.1016/S0261-5177\(99\)00095-3](https://doi.org/10.1016/S0261-5177(99)00095-3)
- Caldarini, G., Jaf, S., & McGarry, K. (2022). A Literature Survey of Recent Advances in Chatbots. *Information*, 13(1), 41. <https://doi.org/10.3390/info13010041>
- Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2023). *Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review*. <https://arxiv.org/abs/2310.14735>

- Colby, K. M., Weber, S., & Hilf, F. D. (1971). Artificial Paranoia. *Artificial Intelligence*, 2(1). [https://doi.org/10.1016/0004-3702\(71\)90002-6](https://doi.org/10.1016/0004-3702(71)90002-6)
- Cormack, G. V., Clarke, C. L. A., & Buettcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 758–759. <https://doi.org/10.1145/1571941.1572114>
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1. <https://arxiv.org/abs/1810.04805>
- Doan, N. N., Aki Härmä, Remzi Celebi, & Valeria Gottardo. (2024). A Hybrid Retrieval Approach for Advancing Retrieval-Augmented Generation Systems. *Association for Computational Linguistics*, 397–409. <https://aclanthology.org/2024.icnlp-1.41.pdf>
- Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., ... Zhao, Z. (2024). *The Llama 3 Herd of Models*. <https://arxiv.org/abs/2407.21783>
- Erak, O., Alabbasi, N., Alhussein, O., Lotfi, I., Hussein, A., Muhaidat, S., & Debbah, M. (2024). *Leveraging Fine-Tuned Retrieval-Augmented Generation with Long-Context Support: For 3GPP Standards*. <https://arxiv.org/pdf/2408.11775>
- Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2024). RAGAS: Automated Evaluation of Retrieval Augmented Generation. *Association for Computational Linguistics*, 150–158. <https://aclanthology.org/2024.eacl-demo.16>
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023). *Retrieval-Augmented Generation for Large Language Models: A Survey*. arXiv:2312.10997
- Gemini Team, Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., ... Vinyals, O. (2023). *Gemini: A Family of Highly Capable Multimodal Models*. 3. <https://arxiv.org/abs/2312.11805>
- Gu, S. (2024). Reimagining Tourist Engagement: Integrating ChatGPT into the Tourism Industry's Service Ecosystem. *Qeios*. <https://doi.org/10.32388/8ARNZ8>
- Harris, Z. S. (1954). Distributional Structure. *WORD*, 10(2–3), 146–162. <https://doi.org/10.1080/00437956.1954.11659520>

- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- Hugging Face. (2024). *Tasks*. <https://huggingface.co/tasks>
- Jeong, C. (2023). A Study on the Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture. *Advances in Artificial Intelligence and Machine Learning*, 03(04), 1588–1618. <https://doi.org/10.54364/AAIML.2023.1191>
- Johri, P., Khatri, S. K., Al-Taani, A. T., Sabharwal, M., Suvanov, S., & Kumar, A. (2021). Natural Language Processing: History, Evolution, Application, and Future Work. *Lecture Notes in Networks and Systems*, 167. https://doi.org/10.1007/978-981-15-9712-1_31
- Jurafsky, D., & Martin, J. (2014). Speech and Language Processing. In *Speech and Language Processing*. (Vol. 3). <https://web.stanford.edu/~jurafsky/slp3/6.pdf>
- Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. <https://arxiv.org/abs/2004.04906>
- Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, 82(3). <https://doi.org/10.1007/s11042-022-13428-4>
- Langchain. (2024, October). *Vector store-backed retriever*. https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/vectorstore/
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems, 2020-December*. <https://arxiv.org/abs/2005.11401>
- Li, M., Yin, D., Qiu, H., & Bai, B. (2021). A systematic review of AI technology-based service encounters: Implications for hospitality and tourism operations. *International Journal of Hospitality Management*, 95. <https://doi.org/10.1016/j.ijhm.2021.102930>
- Liberato, P., Alen, E., & Liberato, D. (2018). Smart tourism destination triggers consumer experience: the case of Porto. *European Journal of Management and Business Economics*, 27(1), 6–25. <https://doi.org/10.1108/EJMBE-11-2017-0051>
- Ma, X., Sun, K., Pradeep, R., & Lin, J. (2021). *A Replication Study of Dense Passage Retriever*.
- Maryamah, M., Irfani, M. M., Tri Raharjo, E. B., Rahmi, N. A., Ghani, M., & Raharjana, I. K. (2024). Chatbots in Academia: A Retrieval-Augmented Generation Approach for

- Improved Efficient Information Access. *2024 16th International Conference on Knowledge and Smart Technology (KST)*, 259–264. <https://doi.org/10.1109/KST61284.2024.10499652>
- McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. *Advances in Neural Information Processing Systems, 2017-December*. <https://arxiv.org/pdf/1708.00107.pdf>
- Meta. (2024, September 25). *Llama 3.2: Revolutionizing edge AI and vision with open, customizable models*. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. <https://arxiv.org/pdf/1301.3781.pdf>
- Nwakamma Ninduwezuor-Ehiobu, Olawe Alaba Tula, Chibuike Daraojimba, Kelechi Anthony Ofonagoro, Oluwaseun Ayo Ogunjobi, Joachim Osheyor Gidiagba, Blessed Afeyokalo Egbokhaebho, & Adeyinka Alex Banso. (2023). TRACING THE EVOLUTION OF AI AND MACHINE LEARNING APPLICATIONS IN ADVANCING MATERIALS DISCOVERY AND PRODUCTION PROCESSES. *Engineering Science & Technology Journal*, 4(3), 66–83. <https://doi.org/10.51594/estj.v4i3.552>
- OpenAI. (2022, November 30). *Introducing ChatGPT*. <https://openai.com/index/chatgpt/>
- Pavel, F., & Romeiro, P. (2023). Challenges of Public Policies for Tourism in Medium-Sized Cities in Europe. The Cases of Lisbon and Porto (Portugal). *GOT - Journal of Geography and Spatial Planning*, 25, 129–150. <https://doi.org/10.17127/got/2023.25.007>
- Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. <https://doi.org/10.3115/v1/d14-1162>
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutkever, I. (2019). *Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners*. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
- Risch, J., Möller, T., Gutsch, J., & Pietsch, M. (2021). *Semantic Answer Similarity for Evaluating Question Answering Models*. <https://arxiv.org/abs/2108.06130>
- Salemi, A., & Zamani, H. (2024). *Evaluating Retrieval Quality in Retrieval-Augmented Generation*. <https://arxiv.org/abs/2404.13781>

- Shum, H. yeung, He, X. dong, & Li, D. (2018). From Eliza to Xiaolce: challenges and opportunities with social chatbots. In *Frontiers of Information Technology and Electronic Engineering* (Vol. 19, Issue 1). <https://doi.org/10.1631/FITEE.1700826>
- Silva, B., Nunes, L., Estevão, R., Aski, V., & Chandra, R. (2023). *GPT-4 as an Agronomist Assistant? Answering Agriculture Exams Using Large Language Models*. <https://arxiv.org/abs/2310.06225>
- Singh, A., Ehtesham, A., Mahmud, S., & Kim, J. H. (2024). Revolutionizing Mental Health Care through LangChain: A Journey with a Large Language Model. *2024 IEEE 14th Annual Computing and Communication Workshop and Conference, CCWC 2024*. <https://doi.org/10.1109/CCWC60891.2024.10427865>
- Singhal, K., Tu, T., Gottweis, J., Sayres, R., Wulczyn, E., Hou, L., Clark, K., Pfohl, S., Cole-Lewis, H., Neal, D., Schaekermann, M., Wang, A., Amin, M., Lachgar, S., Mansfield, P., Prakash, S., Green, B., Dominowska, E., Arcas, B. A. y, ... Natarajan, V. (2023). *Towards Expert-Level Medical Question Answering with Large Language Models*. <https://arxiv.org/abs/2305.09617>
- Svore, K. M., & Burges, C. J. C. (2009). A machine learning approach for improved BM25 retrieval. *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 1811–1814. <https://doi.org/10.1145/1645953.1646237>
- Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*. https://doi.org/10.1007/978-1-4020-6710-5_3
- Vakayil, S., Juliet, D. S., J, Anitha., & Vakayil, S. (2024). RAG-Based LLM Chatbot Using Llama-2. *2024 7th International Conference on Devices, Circuits and Systems (ICDCS)*, 1–5. <https://doi.org/10.1109/ICDCS59278.2024.10561020>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 2017-December*. <https://arxiv.org/pdf/1706.03762.pdf>
- Vidivelli, S., Ramachandran, M., & Dharunbalaji, A. (2024). Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG, and Performance-Optimized LLM Fusion. *Computers, Materials & Continua*, 80(2), 2423–2442. <https://doi.org/10.32604/cmc.2024.054360>
- Wallace, R. S. (2009). The anatomy of A.L.I.C.E. In *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*. https://doi.org/10.1007/978-1-4020-6710-5_13

- Wang, Y., Hou, Y., Che, W., & Liu, T. (2020). From static to dynamic word representations: a survey. *International Journal of Machine Learning and Cybernetics*, 11(7). <https://doi.org/10.1007/s13042-020-01069-8>
- Weizenbaum, J. (1966). ELIZA-A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1). <https://doi.org/10.1145/365153.365168>
- Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., & Liu, Z. (2024). *Evaluation of Retrieval-Augmented Generation: A Survey*. <https://arxiv.org/abs/2405.07437>
- Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). *BERTScore: Evaluating Text Generation with BERT*. <https://arxiv.org/abs/1904.09675>
- Zhang, X., Zhang, Y., Long, D., Xie, W., Dai, Z., Tang, J., Lin, H., Yang, B., Xie, P., Huang, F., Zhang, M., Li, W., & Zhang, M. (2024). *mGTE: Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval*. arXiv:2407.19669v2
- Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., & Hou, L. (2023). *Instruction-Following Evaluation for Large Language Models*. <https://arxiv.org/pdf/2311.07911>
- Zhou, Q., Liu, C., Duan, Y., Sun, K., Li, Y., Kan, H., Gu, Z., Shu, J., & Hu, J. (2024). GastroBot: a Chinese gastrointestinal disease chatbot based on the retrieval-augmented generation. *Frontiers in Medicine*, 11. <https://doi.org/10.3389/fmed.2024.1392555>

APPENDIX A

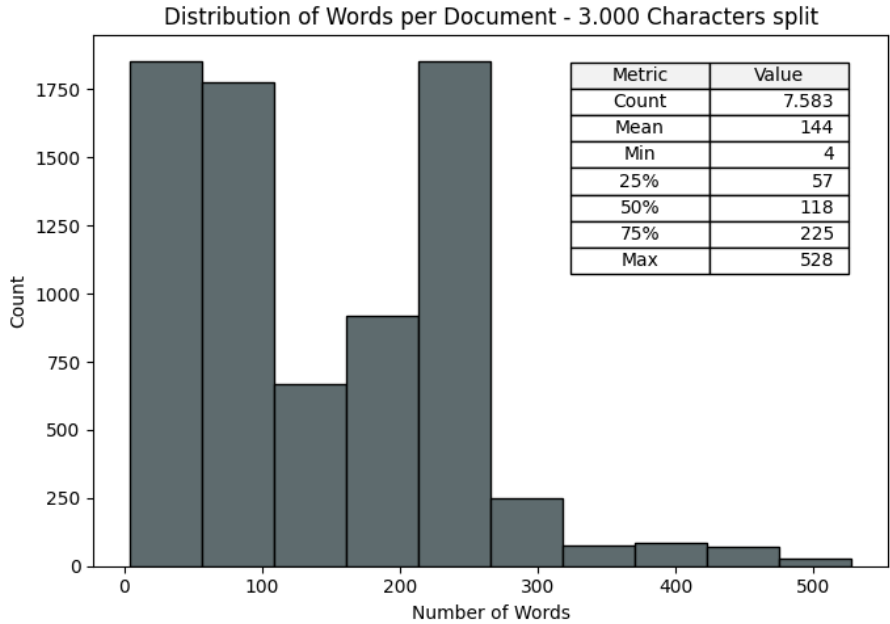


Figure 15 - Word Documents distribution after chunking texts to 3.000 characters maximum.

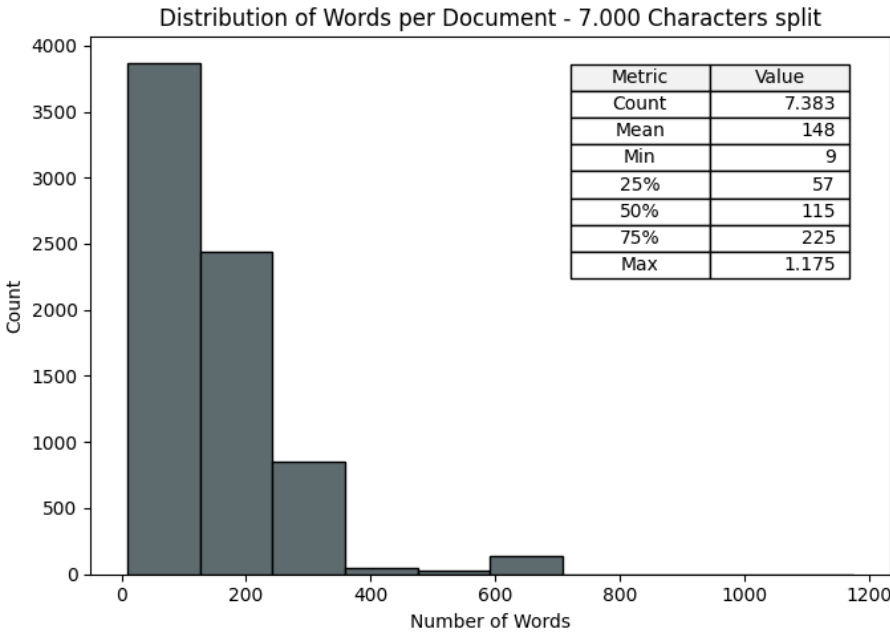


Figure 16 - Word Documents distribution after chunking texts to 7.000 characters maximum.



NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa