NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

JOÃO MANUEL COELHO BARROSO VARANDAS ARVANA

BSc in Computer Science and Engineering

TIME-AWARE QUESTION-ANSWERING FOR THE PORTUGUESE WEB ARCHIVE

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
December, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

TIME-AWARE QUESTION-ANSWERING FOR THE PORTUGUESE WEB ARCHIVE

JOÃO MANUEL COELHO BARROSO VARANDAS ARVANA

BSc in Computer Science and Engineering

Adviser: David Semedo
Assistant Professor, NOVA School of Science and Technology

Examination Committee

Chair: Carmen Morgado
Assistant Professor, NOVA School of Science and Technology

Rapporteur: Ricardo Campos
Assistant Professor, Universidade da Beira Interior

Member: David Semedo
Assistant Professor, NOVA School of Science and Technology

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
December, 2023

Time-aware Question-Answering for the Portuguese Web Archive

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.10.10) [29].

Acknowledgements

First and foremost, I would like to express my gratitude to my adviser, Prof. David Semedo, whose guidance, support, encouragement, and availability were integral to the development of this thesis. I would also like to express my gratefulness to the VisionBox Project (CC 04040101) and thank everyone involved for granting me a research scholarship for 9 months.

Secondly, I would like to thank my colleagues Ricardo Lopes and Daniel Castanho, with whom I collaborated on the development of the ArquiWIZ.pt prototype referenced in this work. I would also like to extend my gratitude to the team behind Arquivo.PT, not only for their availability and support in the early stages of this work, but also for their day-to-day work preserving the web, which made this thesis possible. Additionally, I want to thank the NOVA Vision & Language Research group for providing me with the infrastructure required to support this work, as well as for their support and assistance throughout this thesis.

I am also very thankful for my girlfriend, for being my guiding light through my academic experience, without whose support and patience I wouldn't have made it this far. Furthermore, I also want to thank my friends for being with me through thick and thin, for all the fun times we shared, but also for all the times we suffered together and supported each other.

Finally, I would like to thank my family for all their support and encouragement throughout my life, without which I wouldn't be on the position where I am today.

Abstract

The rise of the internet has dramatically increased the rate at which we produce digital information. However, due to the ephemeral nature of the web, much of this information is lost because it goes offline as quickly as it comes online. As a result, web archives were established to preserve and make information accessible in perpetuity. However the traditional approach of accessing information within these, were one uses the URL of the archived resource, is both prone to loss of knowledge due to forgotten URLs, as well as no longer sufficient to meet modern expectations for ease of access to information. Expectations like searchability and question-answering, which modern users have acquired through daily use of search engines, due to the way these features provide a more natural and easier way to express and satisfy an information need. Thus Web Archives increasingly exercise efforts to provide users with new access methods to it's archived content, one such example is the Portuguese Web Archive which offers full-text search.

In this work we propose a Time-aware Multi-document Extractive Question-Answering approach over the Portuguese web archive, capable of independently answering questions without the need for user specified reference documents, selecting those instead by searching through the millions of archived documents. This approach is achieved by combining two components: 1) a time-aware semantic search component which retrieves candidate reference documents; 2) an ensemble of transformer-based extractive QA models, which collaborate on both the selection of the best reference document, and on the selection of the answer from within it.

We also present ArquiWIZ, a prototype implementing our proposed approach over a subset of the Portuguese web archive's data.

Finally, to evaluate our proposed solution we introduce Arquivo.pt-QA, an extractive Single and Multi document QA evaluation set, focused on Portuguese archived news articles originating from the Portuguese Web Archive.

Keywords: Web Archive, Search System, Information Retrieval, Question Answering

Resumo

O crescimento da internet aumentou dramaticamente a velocidade a que produzimos informação. No entanto, devido à natureza efêmera da web, muitas desta informação é perdida porque fica offline tão rapidamente quanto é posta online. Por esta razão, para preservar para sempre o acesso a esta informação, foram criados arquivos web. No entanto, a abordagem tradicional de acesso à informação nestes arquivos, que envolve o uso do URL do recurso arquivado, é tanto suscetível à perda de conhecimento por URLs esquecidas, como já não é suficiente para as expectativas currentes de facilidade de acesso à informação. Expectativas estas de funcionalidades como pesquisa e perguntas-respostas, algo que os usuários atuais tem vindo a adquirir do uso diário de motores de busca, devido à forma como estas oferecem maneiras mais naturais e fáceis de expressar e satisfazer necessidades de informação. Portanto, os Arquivos da Web estão cada vez mais a dedicar tempo e esforços ao desenvolvimento de novos métodos de acesso ao seu conteúdo arquivado, sendo um exemplo disso o Arquivo da Web Português, que oferece pesquisa por texto. Neste trabalho propomos, para o Arquivo da Web Portuguesa, uma abordagem de Perguntas e Respostas (QA) Extrativas Multi-documentos com consideração pela componente temporal, capaz de responder a perguntas de forma independente sem necessitar de documentos de referencia identificados pelo utilizador, selecionando estes atravez de pesquisa entre os milhões de documentos arquivados. Esta abordagem resulta da combinação de dois componentes: 1) um componente de pesquisa semântica com consideração pela componente temporal, que pesquisa por candidatos a documentos de referência; 2) um conjunto de modelos de QA extrativos baseados em transformers, que colaboram tanto na seleção do melhor documento de referência quanto na seleção da resposta a partir deste. Apresentamos também o ArquiWIZ, um protótipo que implementa nossa abordagem proposta sobre um subconjunto dos dados do Arquivo da Web Portuguesa. Por fim, para avaliar a solução proposta, introduzimos o Arquivo.pt-QA, um evaluation set para QA extrativo tanto para modalidade de documento singular como para Multi-documentos. Este é focado em artigos de notícias portugueses arquivados no Arquivo da Web Portuguesa.

Palavras-chave: Arquivos da Web, Sistema de Pesquisa, Recuperação de Informação, Perguntas Respostas

Contents

List of Figures

1 Introduction

1.1 Context and Motivation

The rise of the internet changed forever the rate at which we produce media. This huge network of connected devices empowers its users to create and distribute terabytes upon terabytes of data daily, which unfortunately goes offline as quickly as it comes online. Specifically, 80% of web pages created are not available after one year [31]. This is a problem that leads to an unfortunate loss of knowledge that once existed and might forever be lost due to the ephemeral nature of the internet.

To combat this problem, web archives, like the Portuguese Web Archive[1], were created with the mission of permanent knowledge, thus becoming hosts to ever growing collections of multimodal documents that they must store and make accessible to the public in perpetuity forever. Traditionally, these archives provided users access to archived versions of websites given the users knew the address they were trying to access. This however, is deemed unsatisfactory by today's standards, on two fronts: First, this access method betrays the archive's purpose of keeping information accessible forever due to it becoming lost to forgotten URLs. One such example would be the news publication "Público" which has had 139 different domains and subdomains between 1996-2019[2], some of which were active alongside each other while others replaced previous ones, thus making it hard to know which one to use to access a specific archived article at a specific date. Second, with the evolution of the internet, the way people consume media has changed. Due to daily use of search engines, users expect the internet to be searchable and to be able to ask questions and have them answered right away without going through a web resource's whole content.

The existence of one such archival system that made these features available is as such very desirable, thus Web Archives increasingly exercise efforts to provide users with new access methods to its archived content, one such example is the Portuguese Web Archive which offers full-text search, efforts which we intend to aid by bringing Question-Answering to this very same archive.

[1] https://arquivo.pt/

[2] https://dados.gov.pt/pt/datasets/lista-de-dominios-do-jornal-publico-no-arquivo-pt-1996-2019/

1.2 Problem Formulation and Objective

This thesis objective is to bring to the Portuguese Web Archive the capabilities of natural and intuitive knowledge acquisition that are unlocked through the power of questions, thus enabling users to explore archived content, or more closely analyze something, through the simple method of asking questions. This means integrating Question-Answering, and also Question-Generation, language models with archives with millions of documents, while taking into consideration the temporal aspect inherently present in an archive's documents.

A system capable of providing the user with such experience as previously described must have the following features and capabilities:

- **Document specific or Single Document Question-Answering -** users must be able to, after identifying an interesting document, more closely explore it by asking questions about it's content;
- **Semantic search -** the system must enable the user to search amongst it's millions of documents by taking a user's query or question and retrieving candidate relevant documents;
- **Time-awareness -** the system must take into consideration the inherent temporal properties of the documents in the archive, as well as possible temporal references in a user's query or question, and must accurately relate the two;
- **General or Multi Document Question-Answering -** the system must also be able to answer user questions when no specific document is under analysis, by autonomously searching for relevant documents to use as reference, and analyzing each of those in search for the answer;
- **Contextualized Question-Generation -** to fully unlock the power of questions, the system must suggest questions contextualized in the current research topic and documents, as means of guiding the user in it's acquisition of knowledge.

This system should be capable of receiving and analyzing each of the users' questions, and taking into consideration the context upon it was asked, in a timely manner provide the user with an answer. In this work we research how to efficiently combine and orchestrate the required models and technologies to deliver Multi-Document Question Answering to the Portuguese Web Archive.

1.3 Challenges and Research Hypothesis

Our Hypothesis is that by combining existing language models into and ensemble, and using it alongside large-scale databases and information-seeking systems, one may build

a system capable of efficient time-aware semantic search and question answering over archived data composed of millions of documents.

During this thesis we made use of data coming from the Portuguese Web Archive[3], which provided us with access to millions of archived web pages. Web pages are inherently multimodal data that is very noisy, having it's main content always surrounded of side-content and navigation interfaces full of text and images that aren't meaningful to that page's purpose. The noisy nature of the data raises the challenge of how to identify and capture the page's meaning for use in semantic search. As such we focused on news since each of these pages have well defined main content and purpose that we can extract in a robust manner and work with.

Even then, there are further challenges:

RQ. 1 - News articles are composed of various **excerpts of text**, like title, summary, and body. Which of these better represents the document? Should all be used?

RQ. 2 - Given the inherently **temporality** of document's in an archive, how to accurately represent the document in time and relate it to the user's queries? Should the publishing date, the crawl date, or temporal references within the document's body be used?

Furthermore, besides the challenges pertaining to the data, selecting and orchestrating the various different technologies required to implement our system raises it's own set of challenges:

RQ. 3 - Multi-Document Question-Answering for Portuguese Web Archives - to the best of our knowledge no such Portuguese Multi-Document Question-Answering model exists, specially not one targeted at the Portuguese Web Archive. How to build such system? Can regular Portuguese extractive QA models with no specialized training be used instead? Which of said models to use?

RQ. 4 - Portuguese Question-Generation model - no such model exists for the Portuguese language. Can we somehow use existing models from other languages to help user navigation and exploration by providing suggested questions?

RQ. 5 - System processing must take place in a practical time-frame - the systems efficiency and quality of results proves to be a challenge, as these present as opposing variables. Should the best performing models be used even if they take longer to process? How long is considered too long for a user to wait? What's the best combination of technologies to provide the best possible results in the least amount of time?

[3] https://arquivo.pt/

Finally, challenges also arise on how to evaluate our proposed Question-Answering approach, since, to the best of our knowledge, no Time-Aware Multi-Doc Portuguese QA benchmark currently exists.

1.4 Contributions and Achievements

This work's main contributions are as follows:

- A novel approach to **Time-aware Multi-document QA for the Portuguese Web Archive**, achieved by an ensemble of transformer-based extractive QA models with no specialized training;
- **ArquivoPT-QA**[4], a new Question-Answering benchmark, focused on Portuguese archived news articles originating from the Portuguese Web Archive, for both the Single and Multi document modalities, and which takes special attention to questions containing temporal references, as well as to the categorization of questions into the 5W1H groups;
- **ArquiWIZ**[5,6], a prototype that demonstrates how our proposed QA solution could be applied to the Portuguese Web Archive by integrating it with work from two other thesis, with whom the prototype was jointly developed, into a complete prototype over a subset of the Portuguese Web Archive's data. The prototype was submitted for consideration in the "Prémio Arquivo.pt 2023" competition where it achieved a place among the top 8 contenders;
- A **paper** that has been submitted to **ECIR 2024** with the previous contributions.

1.5 Document Organization

The remainder of this document is organized in the following chapters:

Chapter 2 - Background and Related Work: Presents an overview of current multimodal understanding models and current ways of orchestrating their use, as well as an analysis of existing archive systems and techniques used.

Chapter 3 - Dataset Pre-Processing and Acquisition: Describes the initial acquisition and processing methods used to compile the subset of the Portuguese web archive's data that was used in this work. It also presents statistics describing the resulting subset of data.

[4] https://github.com/newsArchivePT/ArquivoPT-QA
[5] http://ArquiWIZ.pt/
[6] https://github.com/newsArchivePT/ArquiWIZ

Chapter 4 - Temporal Multi-document Question-Answering and Search: Introduces our proposed Time-aware Multi-document Extractive Question-Answering approach and it's various components.

Chapter 5 - ArquiWIZ: Prototype for the Portuguese Web Archive: Describes ArquiWIZ, a prototype making use of the proposed QA approach from the previous chapter to answer questions over the subset of the Portuguese web archive's data used in this work.

Chapter 6 - Evaluation: Describes the process behind the creation of our ArquivoPT-QA evaluation set, and analyses it's resulting composition. Furthermore, it presents and discusses the performed experiments on the various components of our proposed approach.

Chapter 7 - Conclusions: Summarizes the conclusions of this work and outlines possible future works.

2

Background and Related Work

In this chapter we discuss previous works and existing systems, in order to better understand concepts and existing methodologies that might be used throughout the development of our solution. First we introduce different techniques of achieving text, image, and multimodal embeddings, and the uses of their respective originating models, followed by an exploration of information retrieval and the role of embeddings there. Next we explore Question-Answering, before taking a look into the orchestration of ML models and information retrieval systems. Finally we analyze current web archive systems.

2.1 Natural language representations

Natural language processing is a very active and relevant research topic for its uses in representing and analyzing human language computationally. Some of it's many uses are translation, summarization and question answering and information extraction.

As a result of research in this area various different techniques, models and architectures, have emerged and been used, some of which will be discussed in this section.

The embedding, the representation used by these models, is a low dimensional numeric vectorial representation of a piece of information. Being of vectorial nature, embeddings may be spatially represented in the so called embedding space, in which these numerical representations of information provide us with the ability to correlate them by calculating their distance, which will be smaller the more similar, or meaningfully related, the original pieces of information are to each other. A simple practical use of these representations and the vectorial properties of them would be to compare the similarity between two pieces of text by comparing their distances in the embedding space.

2.1.1 Contextual representations

The same word may have multiple meanings depending on the context it is used. This dependency on context to truly convey the meaning of a word led to the creation of contextual embeddings. Contextual embeddings are a type of word embedding that take into account the context surrounding the word in order to achieve a better representation.

One that truly captures it's meaning in the given context, unlike regular word embeddings that would map said word to a fixed generic vector representation regardless of different meanings.[30][33] One example of said contextual embeddings would be ELMo[34] (Embeddings from Language Models) representations, which are generated by making use of a bi-directional Recurrent Neural Network (RNN).

Recently, transformer-based models such as BERT[15] (Bidirectional Encoder Representations from Transformers) have become popular for generating contextual embeddings, replacing RNNs approaches which had until then been firmly established as state of the art approaches in sequence modeling problems such as language modeling.

2.1.1.1 Transformers

The Transformer [45], represented in the Figure 2.1, is based solely on the attention mechanism to draw the dependencies between the input and output. It follows an encoder-decoder structure, where the encoder takes an input sequence, for example a string of text, and maps it to a sequence of continuous representations. These are then used by the decoder to generate an output sequence, one symbol at a time, process that for every new symbol it outputs it also takes every other previously previously generated symbol as input.

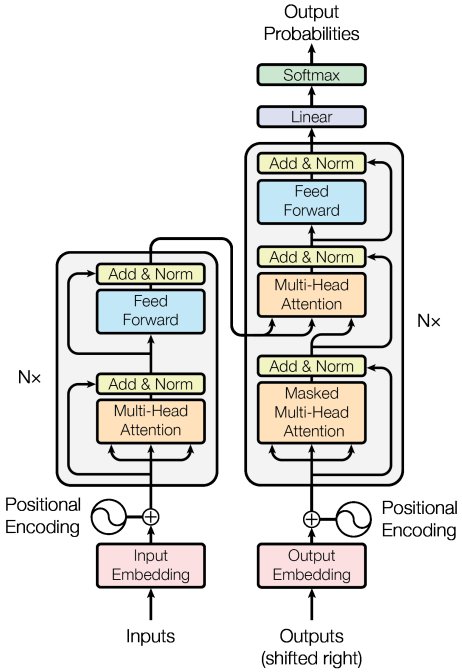


Figure 2.1: Transformer architecture - encoder on the left, decoder on the right [45].

At the start of both the encoder and the decoder we have an initial word embedding layer where each word is mapped into a continuous vector. Since the same word can have different meanings according to where in the sentence it is present, the model needs to be aware of the position of the word in the input. Thus a positional embedding that encodes

positions across the input sequence is generated and combined with the word embedding created in the previous step.

After the creation of the initial embedding, this is sent into the encoder. The encoder is composed of a stack of various identical layers, each composed of a multi-head self attention layer and a feed forward network, each with a residual connection around them. The stacking of transformer blocks gives the model the ability to learn different and higher level representations by focusing on different combinations of attention, which can lead the model to improve. The embeddings start by entering the self-attention mechanism, which allow the model to relate each of the words with each other. The output of this layer is the added to the original input and normalized before being feed into the feed forward network, after which it will be added with it's own original input and normalized again. The output of this full encoder layer in the stack is then used as input into the next layer on the stack until the final one is reached. The final output will be one of the inputs into the decoder.

The residual connections around the layers help some information that otherwise would be lost in the processing, to be passed on to the next layer, and the normalization prepares the output to be used as input in the next layer.

The decoder follows a similar structure to the encoder, the initial input is processed similarly, and in the same way decoders can be stacked for the same reasons. Each decoder is also composed of various layers with residual connections around them, that while being 3 sub-layers unlike the encoder's 2, the final layer is still a feed forward network too. The differences are on the first two layers, and after the entire decoder stack. The first layer of the decoder is an attention mechanism slightly different from the one in the encoder. This is because the decoder processes a sequence word by word, this self attention mechanism is different to prevent it from calculating attention for future tokens. This is achieved by implementing a look ahead mask, and only calculating the self-attention of tokens that were already processed. The second layer of the decoder is another attention mechanism, which differs from the one in the encoder because instead of taking an input and relating the word with each other, it takes one input from the previous layer in the decoder and the output of the encoder stack as another, and relates them. At the end of the decoder stack it's output goes thought linear layer and a softmax that act as a classifier identifying the next token.

2.1.1.2 BERT - Bidirectional Encoder Representations from Transformers

Unidirectional models are sub-optimal for sentence-level tasks, and can specially harm token-level tasks, such as question answering, where it is crucial to incorporate and comprehend context from both directions. As such it's desireable for models to be Bidirectional, meaning they can process their input both from left-to-right and right-to-left.

Devlin et al. [15] introduced the BERT (Bidirectional Encoder Representations from

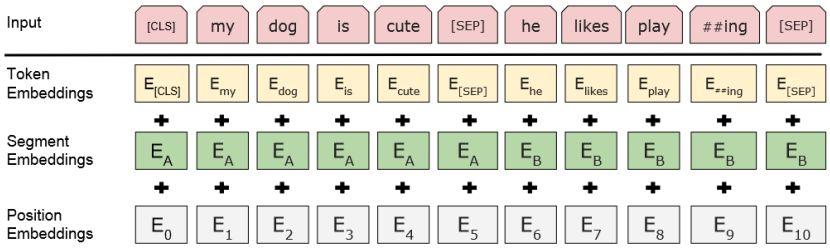


Figure 2.2: BERT input representation [15]

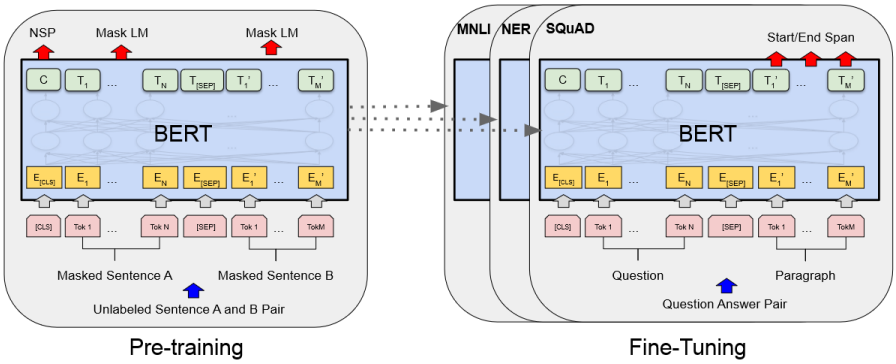


Figure 2.3: BERT's training architectures [15]

Transformers) language representation model, a multi-layer bidirectional Transformer encoder based on the transformer architecture in Vaswani et al. [45], designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers, unlike ELMo[34] which process them separately and then concatenates the results to form the final representation.

To make BERT support a variety of down-stream tasks the input representation is able to unambiguously represent both single sentence or a pair of sentences, like for example a question and answer. To create said input, both sentences are packed into the same sequence with a separation token ([SEP]) at the end of each of them, a special classification token ([CLS]) is also put at the beginning of the sequence. Following this initial preparations the embedding for each token is calculated, and segment and position embeddings are added. Segment embeddings are used to indicate whether the token corresponds to the first or second sentence, and the positional embeddings encode the position within the sequence. An illustration of this construction can be seen in Figure 2.2.

BERT's trainning is split into two phases: pre-training and fine-tuning, as seen in Figure 2.3. During the first phase, BERT is pre-trained in two unsupervised tasks:

- **Masked Language Model** - in this task 15% of the input tokens are masked and then predicted by using the final hidden vectors of the masked token in a softmax over the vocabulary. The masking of said tokens corresponds to replacing the token

with the [MASK] token 80% of the times, with a random token 10% of the times, or not replacing it at all 10% of the times;

- **Next Sentence Prediction** - in this task two sentences, A and B, are given to the model which then predicts whether sentence B follows sentence A. This task is specially important because downstream tasks such as Question Answering relly on the understanding of the relationship between two sentences. When choosing the sentence pair, 50% of the time sentence B is the actual sentence that follows A, and 50% of the time is a random sentence from the corpus.

Fine-tuning BERT is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks without relying on task specific architectures. Compared to pre-training, fine-tuning is relatively inexpensive. To fine-tune BERT for a particular task, we simply input task-specific inputs and outputs and train all the parameters end-to-end, with the addition of an output layer specific to the task. For token-level tasks, the token representations are fed into the output layer, while for classification tasks, the [CLS] token is used.

2.1.2 Sentence representation

BERT's [15] success extends to sentence-pair regression tasks like semantic textual similarity where it set state-of-the-art results. However due to it's cross-encoder (Figure 2.4) architecture it requires both sentences to enter the network together to compute it's similarity. This limitation leads to a high computational cost, and demonstrates a clear unsuitability for use in semantic similarity search due to the model having to re-process the same sentence repeatedly when searching for it's similar pair.

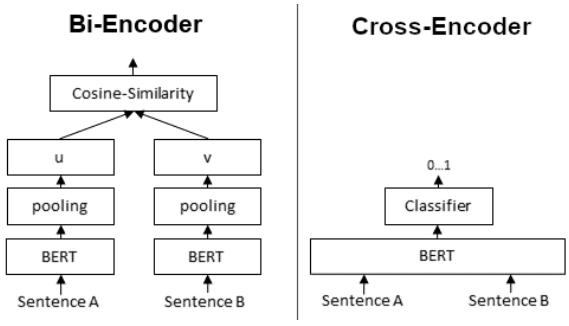


Figure 2.4: SBERT Bi-encoder vs BERT Cross-encoder [14]

A solution for this problem is the use of sentence embeddings. Sentence embedding is the mapping of a sentence to vectorial space such that sentences with simillar meanings are positioned close to each other in embedding space. Making use of such embeddings semantic similarity would be reduced to a simple and inexpensive Euclidean distance or cosine-similarity computation. Different techniques using BERT have been attempted to achieve sentence embeddings, like averaging BERT's output layer, or using the output of the [CLS] token, none of which have yielded good sentence embeddings [39].

2.1.2.1 Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

Reimers et al. [39] introduced Sentence-BERT (SBERT), a modification of BERT using siamese networks, with the goal of deriving semantically meaningfull sentence embeddings. A siamese network is a type of network architecture composed of two sub networks of indentical structure, where the weights of said sub networks are shared.

SBERT takes the existing BERT model and adds a pooling layer at it's output to derive a fixed sized sentence embedding. This pooling layer can use one of three pooling strategies: 1) Using the output of the [CLS] token; 2) the mean of all output vectors (the default strategy); 3) the max of the output vectors. Given this modification to the BERT model, a siamese network is created and trained such that the output after each of the pooling layers gives semantically meaningful comparable sentence embedding. The final result is a bi-encoder model that given a sentence generates a sentence embedding close to similar sentences in embedding space. In this final result lies the advantage of SBERT over BERT for semantic search, as with a bi-encoder architecture each sentence can be processed separately, need only to be processed once, and can have it's resulting sentence embeddings stored for future comparisons.

2.2 Image Embeddings

Computer vision is another active topic of research for its uses in understanding, representing, and retrieving meaningful information from visual media. Some of it's many uses are image or video classification, object detection, and image segmentation. As a result of research in this area, various different techniques, models and architectures, have emerged and been used, some of which will be discussed in this section.

Visual data can be represented in an embedding just the same way text is, keeping the very same properties textual embeddings have, like similarity being represented by distance in the embedding space.

2.2.1 Image Classification and Global Embeddings

Image classification is an image understanding task in which models capture the main features and content of the original image and attribute a label to it. These models create a global representation of the image data, a global embedding, such that neighboring vectors in embedding space are of images with similar content.

Two examples of such models are ResNet [20] which is based in a Convolutional Neural Network architecture, and ViT [16], based on a Transformer Architecture, which is currently state-of-the-art.

2.2.1.1 Convolutional Neural Network-based approach

Convolutional neural networks (CNNs) [3] have shown to yeld excellent results in image tasks [20] owing their success to it's ability to recognize patterns in it's inputs by applying convolution.

CNNs consist of two main types of layers:

The convolutional layer is where the convolution calculation takes place, which extracts patterns in the input image. Each convolutional layer has a learned filter or kernel that is applied to the input image, moving left to right and top to bottom, to generate an output feature map. By stacking multiple convolutional layers, CNNs are able to capture increasingly complex features in the image, for example, such as edges in the first layer, and shapes like circles in the second layer.

Pooling layers are typically placed between convolutional layers and are used to reduce the computational requirements of the model by decreasing the dimensionality of the data. This reduction also helps to extract the dominant features in the image, as these will be the features that are strong enough to survive the reduction.

Adding more layers to a CNN, although making training harder, tends to be beneficial up to a certain point where results start to degrade [20], a common problem with deep CNNs.

ResNet, proposed by He et al. [20], is one example of a CNN, it's distinguishing architectural change is the introduction of residual connections. The existence of said residual connections allow some information that otherwise would be lost in the processing, to be passed on to the next layer, helping with the degradation problem. This small improvement has since become widespread and can be seen in most models.

2.2.1.2 Transformer-based approach

With the introduction of the transformer architecture by Vaswani et al. [45], originally used for Natural Language tasks, there has been a move from CNNs to this new architecture for image tasks too. ViT (Dosovitskiy et al. [16]) is one such example of an image understanding model, making use of the transformer architecture, which achieved state-of-the-art by outperforming CNNs, like ResNet, when it comes to computational efficiency and accuracy.

One initial naíve attempt at using the transformer's self-attention on images would be to have every pixel attend every other pixel. Due to the size of realistic inputs, and the quadratic cost associated with the number of pixels, such a solution isn't scalable.

Being originally designed for NLP, and consequently much smaller inputs, to enable the use of the Transformer for image task some changes had to be made. There was however efforts to stay as similar as possible to it's NLP counterpart so that further improvements originating from NLP architectures to transformers could easily be applied.

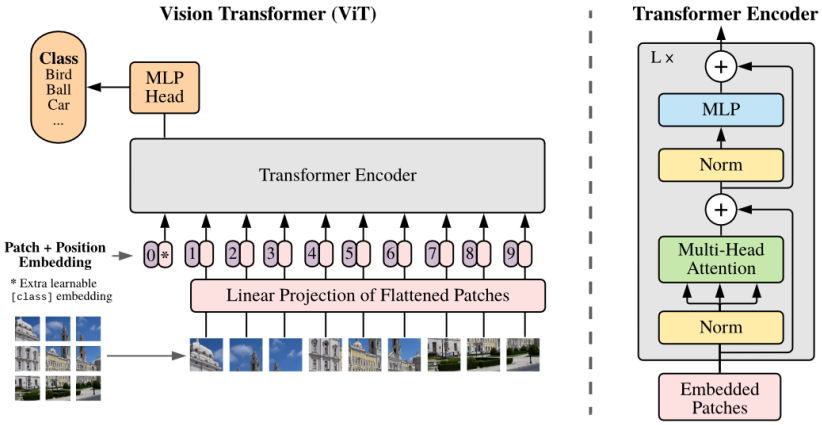


Figure 2.5: ViT Model overview [16]

Following the model overview of ViT presented in Figure 2.5, the adaptations to the input were as follows: images are split into fixed sized patches, flattened into a single vector and linearly projected to the desired input dimension. These are refered to as patch embeddings and treated the same way as word embeddings in NLP tasks; A [class] token is added to the begging of sequence, to be used by the classification task since the model is trained on classification. As a last step, position embeddings are added to each of the patch embeddings to retain positional information. Following this preparations, the resulting input embeddings are used as input to a regular Transformer encoder.

2.2.2 Object detection and Region-based Embedding

Object detection is another image understanding task, differing from image classification as it is applied at a region level instead of the complete image. These models detect objects in images and classify them. The resulting representations, for each region, created by these models, their embeddings, represents the object in the corresponding region in much the same way as any other embedding, meaning similar objects would be represented by regional embeddings close to each other in embedding space.

Object detection techniques consist of a three step pipeline: region proposal, feature extraction, region classification. The process starts with the generation of regions which might possibly contain an object, each of these region is then processed and has a feature map encoding it's contents generated, which follows to the next and last step. This feature map is then used as an input for a classification network which will decide whether or not there is an object in the region and if so what object it is.

Traditionally each of the steps in the object detection pipline would be handled by individual models and algorithms, this is the case with R-CNN, a CNN model proposed by Girshick et al. [18], but there have since been proposed other models that iterate on this initial work like Fast R-CNN [17], and Faster R-CNN [40], each coupling more tightly the 3 parts of the object detection pipeline. The move towards a more coupled approach

as opposed to using different models for each part, is due to both performance as well as quality reasons. By coupling together all 3 steps into one, Faster R-CNN is not only capable of avoiding reprocessing the same part of the image, thus becoming more efficient, but also able to be trained end to end which produces better results as opposed to using multiple independently trained parts.

2.3 Multimodal embeddings

The advancements in natural language representations research enabled us to perform a variety of tasks dependant on the understanding of language. Taking these advancements and adapting them to their field, vision representations closely followed. These were unfortunately unimodal representations of data in a world inherently multimodal where language and vision are so closely linked. This meant it was time to bridge the gap between these modalities and unite their understanding, and consequently their representations in one common embedding space. A multimodal embedding. Some of the tasks enabled by the joining of multiple modalities, like image and text, are cross-modal retrieval, visual commonsense reasoning, and visual question answering.

Multimodal embeddings are representations of multiple types of data, like text and image, which get represented in the same embedding space. They have the same properties other types of embeddings have, like similarity being represented by distance in the embedding space, with the added benefit of being cross-modal. This cross-modality means that for example the vectorial representation of the word "circle" in the embedding space would be close to the representation for the image of a "circle".

As a result of research in this area, various different techniques, models and architectures, have emerged and been used, some of which will be discussed in this section.

2.3.1 Cross-encoder approaches

Li et al. [27] proposed VisualBERT, a model for learning joint contextualized representations of vision and language. This model is based on the BERT language model, and proposes to achieve such representations by reusing the self-attention mechanism within the Transformer to implicitly align elements of the input text and regions in the input image. Being based on the BERT language model, a model intended for use with only text, VisualBERT makes some adaptations to BERT in order to jointly support the processing of visual and language media, as seen on Figure 2.6. Such adaptations target both the model input as well as it's training tasks.

At the input level, VisualBERT follows the BERT model when generating it's initial input, but instead of using two sentences as input, it replaces one of the sentences' text embeddings by what it calls visual embeddings, which originate from a process much like what the sentences go through when being prepared to be used as input into BERT. These visual embeddings are created by summing a visual feature representation, extracted from

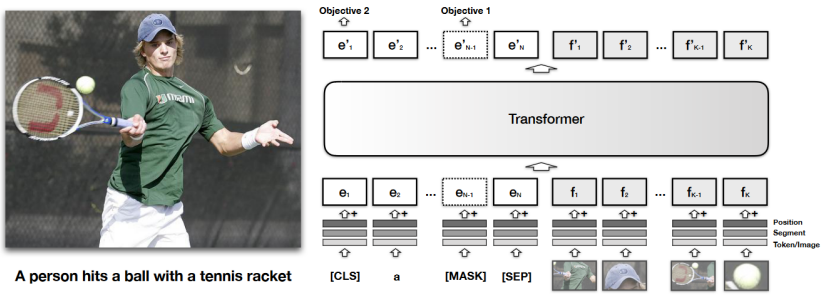


Figure 2.6: VisualBERT's architecture [27]

an object proposal system such as Faster-RCNN, with a segment embedding, indicating it is an image embedding as opposed to a text embedding. It also includes a position embedding, which are used to indicate alignments between words and images when provided. After the creation of said input it is then passed to the Transformer, allowing the model to relate the inputs from both modalities.

When it comes to training, is as straightforward as it is on BERT with the process being the same. On the other hand, while pre-training is similar to BERT's, the tasks have some changes. VisualBERT is pre-trained on the following two visually-grounded language model objectives:

- **Masked language modeling with the image (analogous to BERT's MLM task)** - in this task, while the visual input is kept intact, some of the text input is masked and must be predicted;
- **Sentence-image prediction (analogous to BERT's NSP task)** - in this task, alongside the visual input the model receives a text input composed of two captions. While one of the captions always corresponds to the image, the other one only does 50% of the time. As the tasks' objective the model must predict whether only one, or both, of the captions correspond to the image.

The UNITER (UNiversal Image-TExt Representation Learning) model proposed by Chen et al. [9], which can be seen in Figure 2.7, takes a similar approach to VisualBERT while proposing different pre-training tasks which proved to enable it to surpass VisualBERT.

UNITER's pre-training tasks are as follows:

- **Masked Language Modeling conditioned on image regions (MLM)** - in this task input words are randomly masked (replaced by the special token [MASK]) with a probability of 15%. The goal of the task is to predict the masked words based on the other words and image regions in the input;
- **Masked Region Modeling conditioned on input text (MRM)** - in this task image regions are masked at random with a probability of 15%. The tasks objective is to reconstruct the masked regions given the remaining regions and words.

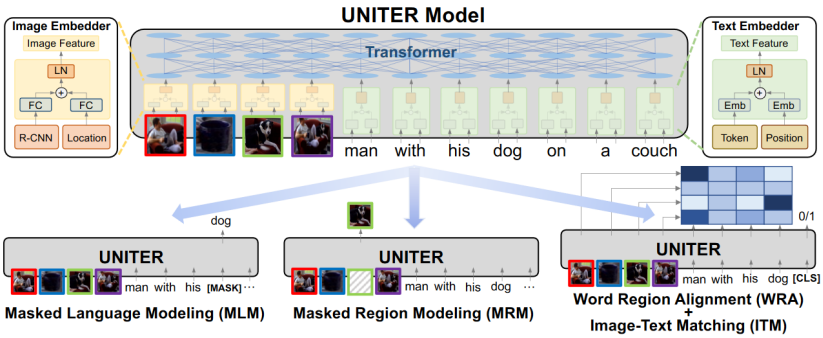


Figure 2.7: UNITER's architecture and pre-training tasks [9]

- **Image-Text Matching (ITM)** - this task's objective is predicting whether the image regions and text in the input match. For this task a special token [CLS] is fed into the model along with the input. This special token will later be used as a fused representation of the input containing both modalities, and will be feed int a fully-connected layer and a sigmoid function to compute the final result, whether its a match or not.

- **Word-Region Alignment (WRA)** - this task's objective is to find correlations between image regions and words by optimizing the alignment (minimizing cosine distance) between word tokens and image regions denoting the same entity.

2.3.1.1 Critical Discussion

Both of these models follow a cross-encoder architecture, that while has been shown to achieve better results in most tasks, including retrieval which is of upmost importance in this work, they are however less computationally efficient in said retrieval task due to the inability of processing inputs separately, leading to the re-processing of the same input over and over.

2.3.2 Bi-encoder approach - CLIP: Contrastive Language–Image Pre-training

Another approach to multimodal models is CLIP (Contrastive Language–Image Pre-training), proposed by Radfor et al. [35], which is a bi-encoder model trained by a simple new proposed contrastive pre-training task, that can be used for image-text similarity and for zero-shot image classification. CLIP's proposed pre-training task takes advantage of a large image-text dataset collected from the internet, composed of images and their natural language captions. It's objective is predicting which caption goes with each image. This approach proved successful, reaching state-of-the-art results, and has show to enable zero-shot transfer of the model to downstream tasks.

The CLIP model, Figure 2.8, being a bi-encoder model, is composed of two encoders, a Transformer for text and ViT for images. Each encoder is followed by a learned projection

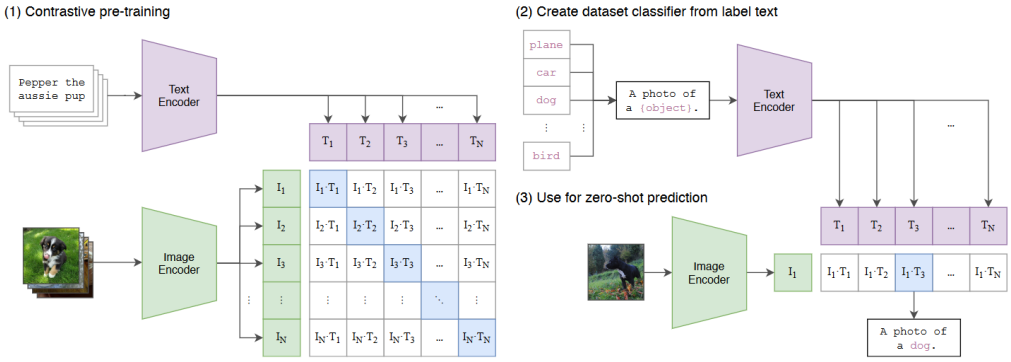


Figure 2.8: CLIP model overview of pre-training and zero-shot use [35]

layer, that is learned during the pre-training task, that projects their output into the multi-modal embedding space. It's this learned projection layer that enables us to related the output of each of the encoders of the different modalities in one common vector space that places the same entities close to each other, independently of the original modality.

CLIP's pre-training task consists of pairing each of the images with their correct caption. Given a batch of (image, caption) pairs, the components of each of these pairs are processed by their respective encoders, then normalized and projected to the common embedding space. CLIP is then to predict which image goes with each caption by calculating the cosine similarity between each image and caption, the values in the projection layer at each of the encoders are tuned until the cosine similarity between the correct pairings is maximized and minimized for the incorrect ones.

Applying CLIP to downstream task like for example image classification is non-trivial and requires clever tricks with natural language that turn the desired task into the image-text similarity task that CLIP is pre-trained on. On the right in Figure 2.8 we can see an example of this, as to classify the image provided to CLIP, a set of potential image captions is created from the potential image labels, and similarity between these and the image is computed in order to find the caption with the classification.

2.3.2.1 Critical Discussion

By following a bi-encoder architecture, CLIP is able to process inputs separately unlike the previous cross-encoder models, which make this model much more computationally efficient at retrieval tasks, as one can store the embeddings of already encoded inputs and reuse them. This makes it a strong contender for use in this work, since efficient multimodal retrieval is of the upmost importance.

2.3.2.2 Multilingual CLIP

While the original CLIP model is only trained over English captions, there have been further works that extend it to support other languages, like clip-ViT-B-32-multilingual-v1[1] by the sentence-transformers team, which created Sentence-BERT [39].This model is a multilingual version of the CLIP model, with support to 50+ languages, amongst which Portuguese is included.

It was created by using Multilingual Knowledge Distillation [38] for the text encoder while leaving the image encoder unchanged. As teacher model the original clip-ViT-B-32 was used for training a multilingual DistilBERT model as a student model. Using this technique, the student model learns to align it's vector space to the teacher's. By aligning the multilingual model text embeddings to the embedding space of the CLIP encoder one gets a multilingual text encoder compatible with CLIP's embedding space that works for 50+ languages.

2.3.3 Hybrid approach - FLAVA: A Foundational Language And Vision Alignment Model

Tipically, multimodal models tend to follow one of two aproaches: they use dual encoders where each of the modalities is encoded separately into the same vector space and put through a small interaction layer for downstream tasks, which is is successfull for unimodal or cross-modal retrieval tasks; or they use fusion-encoders (multimodal cross-encoders) which directly relate each of the modalities to each other with the use of self-attention, which makes it successful on tasks involving both modalities at the same time, like question answering and visual commonsense reasoning.

FLAVA, a Foundational Language And Vision Alignment Model proposed by Singh et al. [42], attempts to join both these approaches in one complete model that targets vision, language, and their multimodal combination, by supporting unimodal and retrieval tasks, as well as cross-modal and multimodal vision-and-language tasks. It attempts this by learning representations through pre-training on both unimodal and multimodal data with cross-modal "alignment" objectives and multimodal "fusion" objectives.

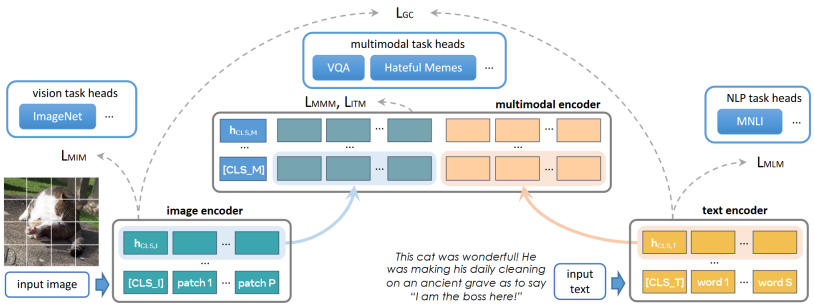


Figure 2.9: FLAVA's architecture [42]

[1] https://huggingface.co/sentence-transformers/clip-ViT-B-32-multilingual-v1

The FLAVA model, show on Figure 2.9, is composed of an image encoder to extract unimodal image representations, a text encoder to extract unimodal text representations, and a multimodal encoder to fuse and align the previous two for multimodal reasoning.

Both the image and text encoder follow the ViT architecture, with the text input being pre-processed like BERT's, while the image input follows the normal ViT pre-processing. Each of the encoders, image and text, returns a list of hidden state vectors (h_I & h_T respectively), including one hidden state vector ($h_{CLS,I}$ & $h_{CLS,T}$) for the special classification token ([CLS_I] & [CLS_T]) prepended at the beginning of each input. For the multimodal encoder, another transformer based on the ViT architecture is used. To create it's input a learned projection specific to each of the previous encoders is applied to their outputs. The result is concatenated into a single list and prepended with ([CLS_M]), a special classification token. This list is then fed into the multimodal encoder allowing cross-atention over both of the projected unimodal representations, to relate and fuse them. The output from this encoder is a list of hidden states vectors (h_M), including one hidden state vector ($h_{CLS,M}$) for the special classification token ([CLS_M]).

This model architecture enables FLAVA to be applied to both unimodal and multimodal tasks in a straight-forward manner, by adding an output layer specific to the task on top of the encoder. This encoder is responsible by the modalities required by the task. For example, in image recognition tasks, a classifier head is placed on top of $h_{CLS,I}$ from the image encoder, while in multimodal reasoning tasks, a classifier head is placed on top of $h_{CLS,M}$ from the multimodal encoder.

FLAVA's training starts by pre-training each of the unimodal encoders, after which the model continues to pre-train in a round-robin fashion between multimodal and unimodal tasks. The unimodal tasks used in FLAVA's training are Masked Image Modeling (MIM) for the image encoder and Masked Language Modeling (MLM) for the text encoder. In both tasks, part of the input is masked and the model must predict the masked input based on the rest of the input. For multimodal tasks, FLAVA uses: a) Masked Multimodal Modeling (MMM) - an extension to the tasks used in the unimodal encoders, in this task both image patches and text tokens are masked and predicted at the same time; b) Image-Text Matching (ITM) - this task's objective is predicting whether the image regions and text in the input match; c) Global contrastive (GC) loss - similar to CLIP's, given image and text pairs, this tasks objective is to maximize the cosine similarities of matching pairs, and minimize them for unmatched pairs.

2.4 Preprocessing, Indexing and Retrieval

Traversing large document collections in order to retrieve relevant documents to a given query is fairly computationally expensive, becoming even more expensive the bigger the collection. As a solution to this problem data indexing is needed. These systems employ various efficient data-structures and techniques that allow for a fast initial retrieval of potential relevant documents to the given query. The results obtained by this initial

retrieval can then be more closely inspected, and re-ranked, due to their reduced quantity comparatively to the whole collection.

Examples of such systems are Apache Lucene[2], Apache Solr[3], or NutchWAX[4] which is specific to web archive workloads. More recently, large-scale indexing systems for big-data have been released such as ElasticSearch[5] or OpenSearch[6], a fork of the former, which we will be using.

2.4.1 Document pre-processing - Newspaper3k

Documents may contain information that is of no importance to it's main content. The including of such irrelevant information would pollute the index and affect matches and rankings, as such initial pre-processing to index only the important parts of documents must be executed.

In this thesis we will be making use of web publications of news. The resulting document of scraping such web pages contains not only the article but also the surrounding HTML which is of no importance to us. To solve this problem we will be making user of newspaper3k[7], a library made specifically for the scraping of newspapers web publications. It provides utilities to extract all of the articles main components, like title, body, author, publication date, and any images included within.

2.4.2 Full-text search

Document retrieving based on full-text search typically uses simple techniques based around matching the query terms with the terms in the document, giving more relevance, and thus a higher ranking, to documents that contain a higher count of occurrences of each term in the query. Even though these simple retrieval models, given their term frequency approach, have no understating of either the query's or the document's meaning, they have shown satisfactory results.

BM25[41], is one such technique used in retrieval making use of term frequency. It implements a ranking function that estimates the relevance of documents to the given search query by relating the amount of times the queries terms appear in the document with the document's length, and overall popularity of said terms in the document collection. This essentially results in higher relevance of documents in accordance with higher density of matching terms. BM25F further extends BM25 to consider each document as a set of fields, with the possibility of each carrying a different weight depending on their attributing importance, meaning term matches in different fields affect relevance differently.

[2] https://lucene.apache.org/
[3] https://solr.apache.org/
[4] https://archive-access.sourceforge.net/projects/nutchwax/
[5] https://www.elastic.co/
[6] https://opensearch.org/
[7] https://newspaper.readthedocs.io/

2.4.3 Embedding-based search

Indexes may also contain embedding of each of their documents, enabling the possibility of executing semantic search over the documents, provided that the embeddings acuratly capture the meaning of the document. In embedding based search the query is converted to an embedding and through similarity functions compared to the embeddings of indexed documents. The result of these similarity functions determine relevance and ranking of each document to the given query. A document is more relevant to a query the closer in embedding space to the query's embedding.

K-nearest neighbors (k-NN), is an algorithm that works over a index composed of vector points, in this case embeddings, and finds an embedding's "nearest neighbors", meaning it finds other embeddings that are closest to it's location in vector space.

2.4.4 OpenSearch

OpenSearch is a distributed scalable indexing system and search engine. When interacting with OpenSearch interacts with an OpenSearch cluster. Each OpenSearch cluster may support various Indexes which are split across multiple Shards (Apache Lucene instances), that are each ran on one of the cluster's Nodes (machines).

After adding data into OpenSearch one may perform full-text search, or embedding based search, over the indexed data's fields, being also possible to apply filters to each search.

2.5 Question answering

Question answering is an area of research in the fields of natural language processing and information retrieval, that focuses on creating systems that provide answers to questions written in natural language. One such system requires deep understanding of natural language to interpret the questions, deep understating of the context modality in order to understand context over which the question is asked, and information retrieval capabilities to acquire context withing a corpus in situations where context isn't given.

Question answering problems may be formulated in various different ways, each with it's own requirements and solutions, be it different ways to present the questions, different modalities to present context, or different ways to present the answers. Each of these properties defines the system as whole. Examples of question format include:

- presenting the question in a natural language way, as if speaking with the system;
- presenting the question as structured language queries;
- presenting the question in cloze format, meaning presenting the question as text with slots to be filled.

As for response formats these can be classified in:

- generative, where the model completely generates the response;
- classification, where the result is selected from pre-determined options;
- extractive, where the response to the query is extracted from the given context.

Concerning the modality of context provided to theses systems, typically it's text, keeping QA a single modality problem, but there are systems supporting other modalities like images, that turn QA into a multimodal problem.

2.5.1 Span-based QA

Span-based QA are extractive question answering systems for textual QA. Span refers to a contiguous sequence of text in a document, typically represented by their start and end indexes within the document. The goal of these kind of QA systems is to identify and extract from the context document the specific text span that contains the answer to the question.

In Chapter 2.1.1.2 we previously referenced BERT, one of the most popular transformer-based models for natural language understanding, and how it easily fine-tuned into many different downstream tasks without much changes.

Thanks to it's pre-training and architecture, BERT easily fine-tunes into the Span-based QA task, just like any other down stream task, without many changes. To fine-tune BERT one simply uses tasks specific inputs and outputs, applies a small task specific layer at BERT's output, and trains everything end-to-end, has seen on Figure 2.3 which presents BERT's training architectures. To fine-tune for span-based question answering one might use the Stanford Question Answering Dataset (SQuAD)[37] as the task specific dataset. By making use of this dataset during fine-tuning one simply replaces BERT's normal input sentence pair by the question and context paragraph pairs from the dataset and trains with the objective of predicting the span's boundaries at the output of the task specific layer.

2.5.1.1 SQUAD - Stanford Question Answering Dataset

The previously mentioned SQUAD, or Stanford Question Answering Dataset, was presented by Rajpurkar et al.[37], and is a large-scale reading comprehension dataset, composed of 100,000+ crowdsourced questions over a set of Wikipedia articles, and corresponding answers. Each answer in the dataset is a span from the corresponding Wikipedia article used as context. Some of the questions in the dataset are unanswerable given the provided context, situation which is denoted by the answer being an empty span. SQuAD is widely used to train and evaluate the performance of natural language processing and machine learning models, particularly with regards to question answering tasks.

2.5.2 Visual Question Answering

As previously mentioned QA tasks may be multimodal, one such example of this is Visual Question Answering (VQA), first proposed by Agrawal et al.[2]. In VQA the system takes a natural language text and image pair, as question and context respectively, and must return a natural language based on the image's visual information. In order to effectively answer questions about visual data, which encompasses various aspects of an image including its context, a strong underlying understanding of both vision and language is crucial. In order to achieve this, these model require pre-training in joint language and visual tasks.

One example of such a model with joint understanding of vision and language, that can fine-tune into VQA is VisualBERT, previously referenced in Chapter 2.3.1. This model approaches the VQA task as a classification problem, where it must choose and answer amongst a pre-determined set of answers. This proves disadvantageous as it limits the possible answers the model can predict, not allowing for new answers.

This approach approach comes in stark contrast to the Span-based QA for language discussed in the previous chapter, which is much more flexible in the answers it provides.

2.5.3 Retrieving context for QA

Question answering systems are typically paired with large collections of documents from which to gather their answer. It is however infeasible to transverse and analyze every document within a collection to find the answer. For this reason, document retrieval methods are used initially to restrict the collection to a subset of relevant documents, where at least one of them might contain the answer, before processing them with the QA model. This step is of the upmost importance to get right, after all, a Question Answering model is only as good as the context it is provided. It doesn't matter how well it performs at finding the right answer, it can't find the right answer if it was never there to begin with.

Wang et al.[46] proposed QANA (Question Answering in News Archives), which is am example of a QA system as previously described, that integrates document retrieval with question answering models to arrive at the desired answer. QANA's aproach is different from other QA retrieval systems as it opperates specifically over the news domain. Such specialization enables it to take advantage of the inherently temporal characteristics of these documents by implementing an additional component, the "Time-Aware Reranking Module". This module sits between other two modules common to other large scale question answering systems, these being the "Document Retriever Module", responsible for selecting candidate articles from the the full news collection, and the "Document Reader Module", which extracts the answers from the top ranked documents and returns the most common answer. The existence of the "Time-Aware Reranking Module" is what sets this system apart, as it is responsible for inferring the time-scope from the question, and re-ranking the candidate documents selected by the first module in accordance to closeness to the query time-scope.

With this approach QANA was capable of outperforming the existing QA systems, over a subset of the New York Time archive as the document source, and a carefully constructed test set with questions associated with past events selected from existing datasets and history quiz websites.

2.6 Machine Learning Models Orchestration Strategies

The development of end-to-end systems making use of multiple Machine Learning models raises challenges on how to coordinate them. The orchestration of these models has the goal of leveraging the strengths of different models to achieve the desired results. To achieve this, it's important to carefully select the models used, and carefully coordinate the flow of data over the system.

One of two approaches may be taken when creating system made of various components: 1) build a monolithic system that tightly integrates with every component and moves data between them; 2) build each of the components into micro-services and create an orchestrator service to control the data flow pipeline between them.

More recent systems tend to select the second approach due to multiple advantages like:

- the easy scalability by increasing the number of each micro-service;
- the modularity achieved by separating each component which allows for component replacement without taking down the entire service;
- and the interoperability provided by isolating each component which enables them to be implemented in different technologies.

Chakravarti et al.[8] proposed the Computation Flow Orchestrator (CFO), an orchestration framework for building end-to-end interactive Natural Language Processing and Information Retrieval systems. The main idea behind this work is that data in these systems flow from service to service, forming a "computation flow graph", with each service, or node, executing their part of the data processing until the final result is produced and shown to the user. CFO takes this idea and making use of two specification languages defines the "computation flow graph" of the overall service. First using the Google's Protocol Buffer Interface Definition Language, each node, which must be an already existing containerized gRPC3 microservices, defines it's service name, input and output fields. Then the orchestrator is described using a custom specification language which allows to describe the set of previously defined nodes to be used, as well as the connections and data flow between them. Given the specification files, CFO generates the orchestrator node that implements the computation flow graph logic, and simple REST and GUI interfaces for interaction with the system.

2.7 Web Archives

Web archives, like the Internet Archive[8] (IA) and the Portuguese Web Archive[9] (PWA), are digital libraries created with the purpose of storing and preserving various kinds of digital media, namely web pages, and maintaining said contents freely accessible by the public in perpetuity forever. Current web archives enable users to browse through their library of content by providing the original URL of the webpage they wish to visit and selecting the date of one of the available snapshots of said page, or by so called time-travel queries [4]. This corresponds to a full-text search over the archive's content on a user-specified time frame, being this the preferred and most used method of accessing the archives content whenever it is supported [12].

As one might think, storing and serving an ever increasing amount of content to users comes with its technological challenges, not only about how to store said data but also how to index it and retrieve it efficiently, and how to scale the service indefinitely. In this chapter we shall discuss solutions currently implemented by web archives like the PWA.

2.7.1 Document storage

Very early into the creation of web archives arose the necessity for a different way to store the files created by the web crawlers that were scraping the web. This necessity results from the fact that the web, and consequently the data created by the crawlers, consisted of a very large quantity of small files, which in current file systems is very difficult to manage. And thus the ARC file format was created. [5]

This new ARC file format is a self describing concatenation of various files. Each ARC file is then commonly referred to as a collection. This format provides the file with the property of being self contained, enabling it to be easily moved and backed up, as well as accessed, without the need for a companion index file of its contents. It also provides it with the ability to have stream-like access, since it is only a concatenation of various files, multiple readers can read different parts of the collection file to access different files within. Each entry in an ARC file consists of a block of metadata, including the URL where content was retrieved from as well as its size, followed by a data block with the content retrieved. These entries are concatenated each following another forming the ARC file.

As web archives evolved and their needs changed, the need to support more digital media types other than web pages arose, and as such the ARC file format was extended to include this other type of digital media. The resulting file format is called WARC [23] [1] and is the running standard among web archives. [11]

[8] https://archive.org/web/
[9] https://arquivo.pt/

2.7.2 Document access and indexing

Web archives' libraries have become massive, and serving and managing all of that content requires careful data management and indexing to ensure no data gets misplaced or lost, and any resource can be easily accessed in a timely manner. In order to fulfill these requirements, especially the last one, each archive has one or more document indexes, each optimized for one of the offered resource access modalities, like by URL or full-text search.

2.7.2.1 Indexing for URL access

URL access is the most basic way to make the resources in a web archive available, making it also the most common with 89% of web archives supporting it[19]. The type of index required for this type of access is as simple as the functionality it enables, as this functionality can be supported by a simple flat file sorted alphabetically by URL, where each line maps an URL to the corresponding ARC file, and where the system should find the documents archived. [13] These files can be CDX (Capture inDeX) [25] [22] files, or more recently CDXJ (Capture inDeX JSON) [32], and each URL is in the SURT format.

SURT, or Sort-friendly URI Reordering Transform [26], is a transformation applied to URIs that makes them better represent the natural hierarchy of domain names, and thus enables them to be alphabetically sorted. This acquired property, of alphabetical order, is true if the desired outcome of the sorting is to place related URIs, as in subdomains of the same domain, together. SURTs achieve this by converting "domain.tld/path?query" to "tld,domain)/path?query". For example if we had 3 URLs "a.example1.com", "a.example2.com", "b.example1.com", and tried to alphabetically sort them, the desired outcome would be for "a.example1.com" and "b.example1.com" to be together since they are both subdomains of "example1.com", unfortunately by using the URLs normal format "a.example2.com" would endup between them. This ordering problem is solved by the SURT transform which converts them into "com,example1,a)/", "com,example2,a)/", "com,example1,b)/" respectively, which when sorted alphabetically returns our desired order.

The CDX index file format is a CSV-like file where each line corresponds to one entry in the index, and maps one URL to which arc file the documents is archived, amongst other information. CDX files make use of SURTs to represent the URL, these being the first field in each line, this provides the CDX file with the desired property of being alphabetically sortable to optimize the index searches. The default first line of a CDX file is "CDX A b e a m s c k r V v D d g M n" [24], where each of the letters after "CDX" represents a field that each of the following lines will have. Notable fields include "A" for the URL, "b" for date of crawl, "m" for mime type, and "s" for response code.

The CDX file is very restrictive on the fields and information able to be stored in the index, so recently there has been a move away from CDX to the more flexible CDXJ. These CDXJ files provide the possibility to store not only the same information the CDX files did,

but more, since instead of a SURT and various other fields in each line, CDXJ files store a SURT and a JSON object, which is flattened into a single line, in which one can include whatever fields they may deem necessary.

2.7.2.2 Indexing for full-text search access

Full-text search has become the default way people expect to search and access information, and because of this, even with it's higher complexity and processing requirements, about 67% of web archives support this kind of access[19]. Due to the inherent complexity related to the implementation and scalability of this feature, web archives make use of specialized tools to manage their indexes associated with this feature. Tools like Solr, Apache Lucene, or the most prominent NutchWAX. [13] NutchWAX is a fork of Nutch and was developed by the International Preservation Consortium (IIPC), it is both a full-text and URL search engine with built in support for Web Archive workloads, features which make obvious the reason for it's widespread use. Whichever tool used, all of them have on thing in common, the techniques implemented to index and enable full-text search, techniques like "Inverted Files", also know as Inverted Index, with Document or Term partitions. Web archives also make use of indexing strategies as "Temporal Inverted Files" to partition by time, to better support time-travel queries.

Inverted files [47], or inverted index, is a way to index documents for efficient retrieval in full-text search. This technique reduces the documents to the count of each word, or term, within it and then uses that information to index the document. The index created by this technique takes on the format of a matrix (Figure 2.10) where on one axis is the set of all terms present across the documents (the lexicon), and on the other the ID of each document, being the value at each position of the matrix the count of said term in the document.

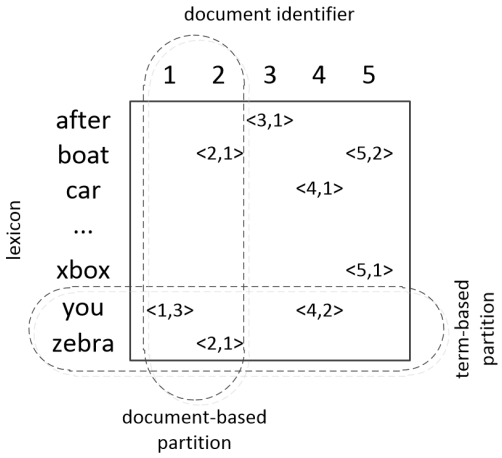


Figure 2.10: Representation of the inverted file matrix, and it's possible partitions. [13]

As web archive's collections grow their storage and respective indexes must distributed across different machines. In the case of the Inverted Files index, it's partitioning can be

based on Documents or on Terms. Document-based partitioning means each system has a set of documents and the count for each term in the lexicon for it's documents. On the other hand term-based partitioning means each system has a set of terms from the lexicon and the count for that term in every document. Each of these approaches comes with it's own advantages and disadvantages, but document-based partitions tend to be the ones used because of one property: the fact that they don't require the rebuilding of the indexes every time new documents are added. This is a desirable and necessary property to have since these collections and indexes are forever growing.

Temporal inverted files refers to a partitioning technique that enables efficient time-travel queries. In temporal inverted files, one may take existing inverted files partitions and further partition each of them by time intervals. Instead, one may partition by time first and then partition using inverted files. (Figure 2.11) The latter technique tends to be the preferred one since it more close reflects the scaling of a web archive and it's ever growing collection, and consequently infrastructure, since adding newer, more recent, collections and entries becomes as easy as creating a new time partition and building it's inverted file index. [13]

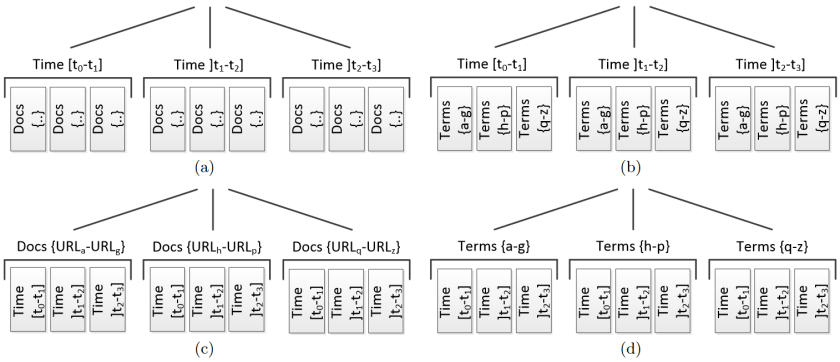


Figure 2.11: Index partitions using three dimensions: time, documents and terms. [13]

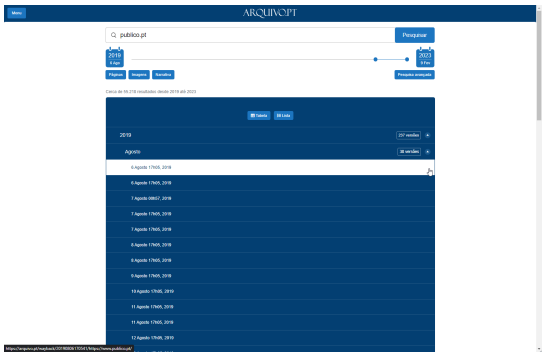
2.7.3 Web Archive Interfaces

In this section some user interfaces for interacting with web archives will be discussed, as inspiration for ideas to implement in our system. First we discuss official PWA's interface, followed by some prototypes resulting of a contest PWA held for projects using it's data.

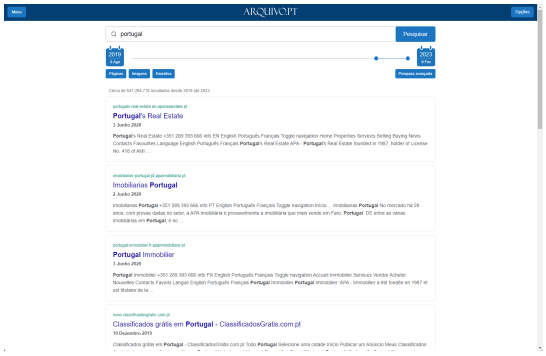
2.7.3.1 Portuguese Web Archive's Interface

PWA's main functionality is consulting the archived version (Figure 2.12(a)) of a web page in it's archive collection. As ways of accessing said archived pages it's interface provides URL access (Figure 2.12(b)) to it's data, as well as full search queries (Figure 2.12(c)), which can be filtered in the advanced search (Figure 2.12(d)).

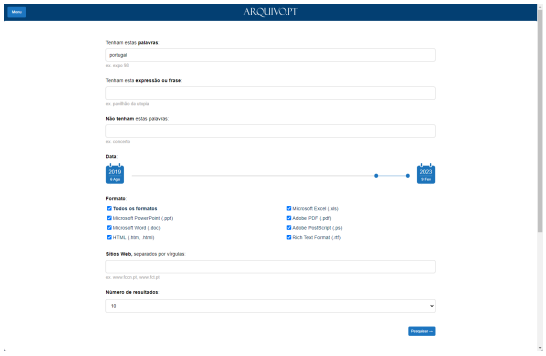
(a) PWA's archived page view



(b) PWA's url search



(c) PWA's text search



(d) PWA's adavanced search

Figure 2.12: Portuguese Web Archive's interface

2.7.4 Conta-me Histórias

"Conta-me Histórias"[10] is a website resulting of the PWA contest. This website compiles news timelines based on a user query, with the purpose of outlining a narrative following the topic of said query. It does so by leveraging archived web pages of news articles in the PWA. It crawled and indexed these, and then on the occurrence of a user's query returns a chronological list of the most relevant news, as seen on Fig 2.13.

2.7.4.1 Desarquivo

"Desarquivo"[11] is a prototype resulting of the PWA contest. This prototype accesses news publication web pages archived by the PWA, extracts common entities across articles and relates said entities in a graph structure, as seen in Figure 2.14. This visualization enables the user to find connections by at first seemingly unrelated entities.

[10] https://contamehistorias.pt/
[11] https://msramalho.github.io/desarquivo

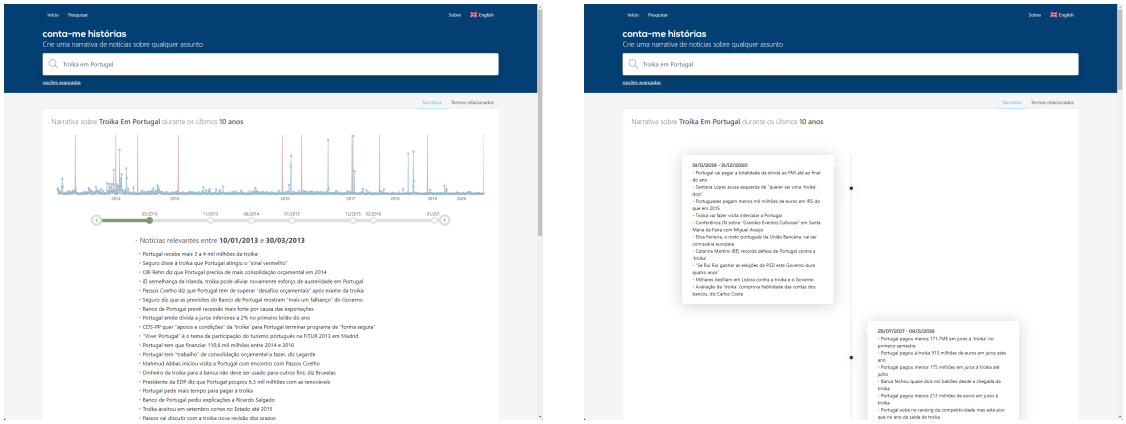


Figure 2.13: "Conta-me Histórias"' narrative view

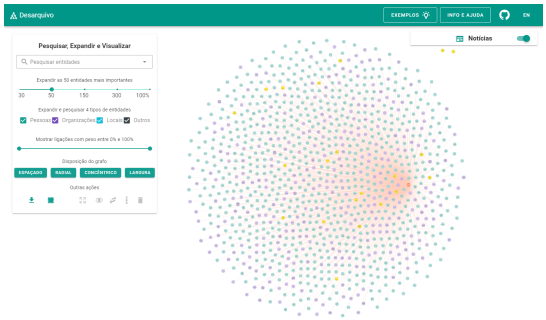


Figure 2.14: "Desarquivo"'s graph view relating entities related to "Operação Marques"

2.7.4.2 Politiquices

"Politiquices"[12] is a prototype resulting of the PWA contest. This prototype also accesses news publications archived by PWA, but instead performs sentiment analysis over the article's titles to identify relationships of support/opposition between politicians. It's main view is also a graph like (Figure 2.15) where one can visualize said relationships, with green arrows between nodes denoting support, and red arrows denoting opposition. Other functionalities include seeing statistics on the amount of articles written on each politician, statistics on the amount of articles written on each pair of politicians, and a index of political party filliations.

2.7.4.3 MajorMinors

"MajorMinors"[13] is a prototype resulting of the PWA contest. This prototype also accesses news publications archived by PWA, and retrieves articles, opinion articles, and images, related to minorities. It tags each document with keywords, and performs sentiment analysis over it, and stores them in a graph based databased where it links related articles. It present's users with various statistics over the dataset (Figure 2.16), but also a categorized

[12] https://www.politiquices.pt/
[13] http://minors.ilch.uminho.pt/

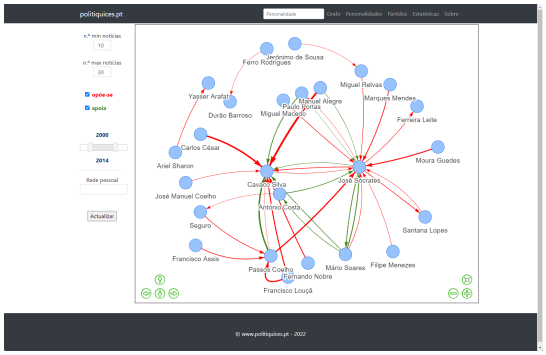


Figure 2.15: "Politiquices"' graph view denoting support/opposition relationships between politicians

list view of all the documents where one can open and view each of them (Figure 2.17).

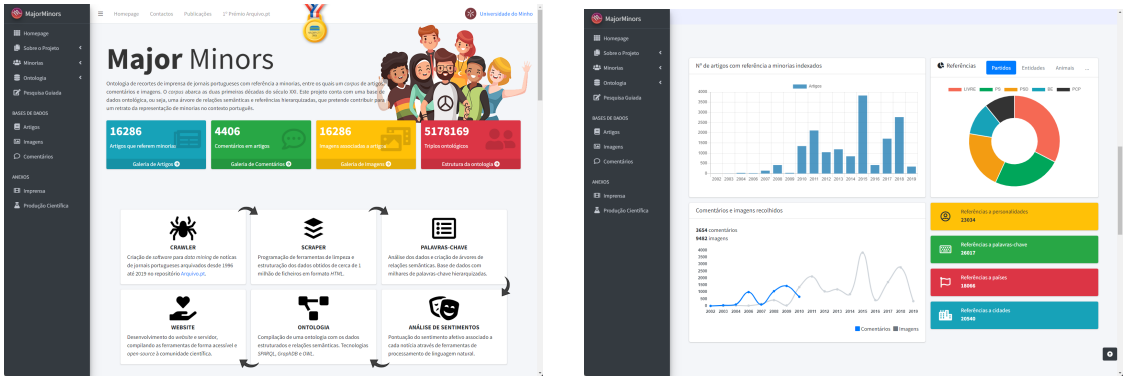
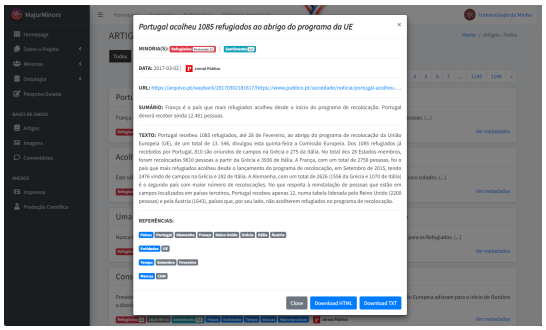
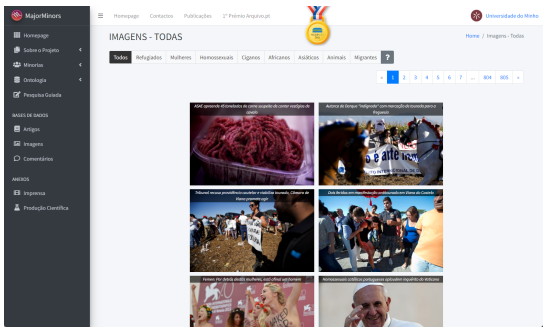


Figure 2.16: "MajorMinors"' statistics page

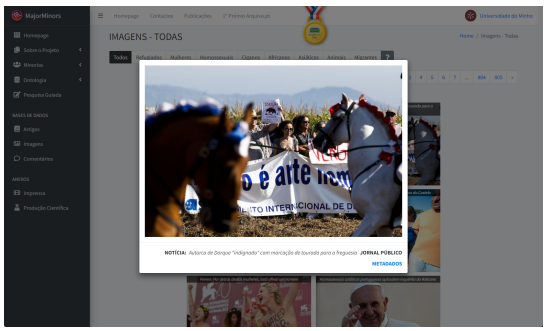
(a) Article List



(b) Article View



(c) Image List



(d) Image View

Figure 2.17: MajorMinor's categorized list view of documents

3

Dataset Pre-Processing and Acquisition

The Portuguese Web Archive, as every web archive, is host to an expansive dataset of archived content, from the obvious web pages, to every other kind of digital media. This archived content is very diverse not only in its format but as well as its domain and time period, as it relates to the most various topics, and spans decades across time. In the special case of archived web pages the temporal aspect is very important, having multiple versions of each archived page across time, to register the page's evolution. Such a large and diverse dataset would be unmanageable in the context of this work. Thus, we decided to restrict our corpus to online news articles of specific news outlets.

We contacted the Portuguese Web Archive on how to efficiently crawl and scrape their content without affecting their service, which resulted in a collaboration were we jointly developed the API and methodology for Bulk[1] download of archived resources. They also provided us with access to the CDXJ index files describing their archived resources, and pointed us towards the collections they considered to be of the most interest to us.

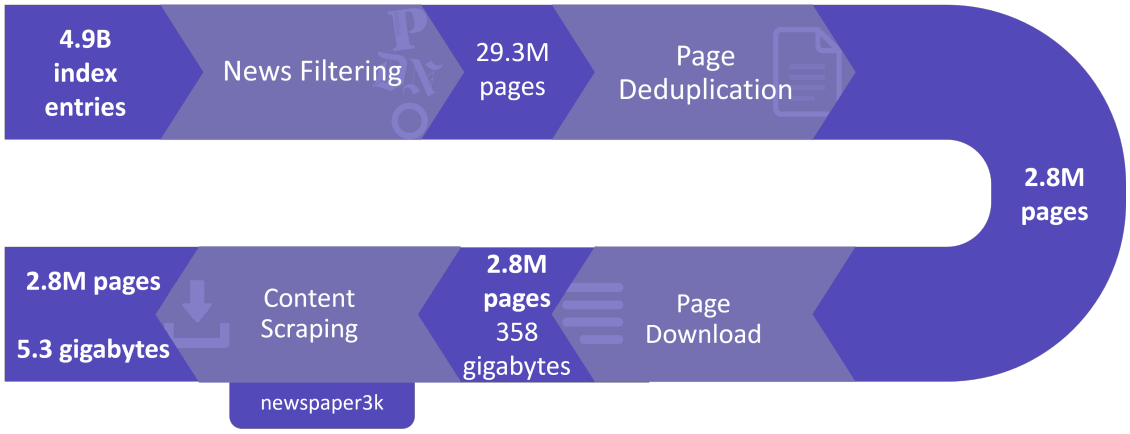


Figure 3.1: Corpus acquisition pipeline.

[1] https://arquivo.pt/api#bulk-download-of-web-archived-resources

The Portuguese Web Archive splits their crawls by collections, each with it's own CDXJ index, which contain an entry for every time a page is archived. These collections are labeled into different categories, one of which is the FAWP category, which identifies collections that contain frequently crawled pages like web news publications. We were given access to the indexes of the 42 FAWP collections available at the time, which contain crawls ranging from 2010 up to 2021.

We selected some of the most prominent Portuguese news publications, these being "Diário de Notícias", "Observador", "Expresso", and "Público", and started filtering the index to only keep URLs from these news publications. Special attention was given to "Público" due to the fact it has had 139 different domains and subdomains between 1996-2019 and thus, by making use of a list compiled by the team at the Portuguese Web Archive of all these URLs from "Público"[2], we made sure to include all of its relevant domains across time.

The initial index had **4 911 080 349** entries which got reduced to **29 323 851** after filtering. The distribution of these entries by domain can be seen in Figure 3.2.

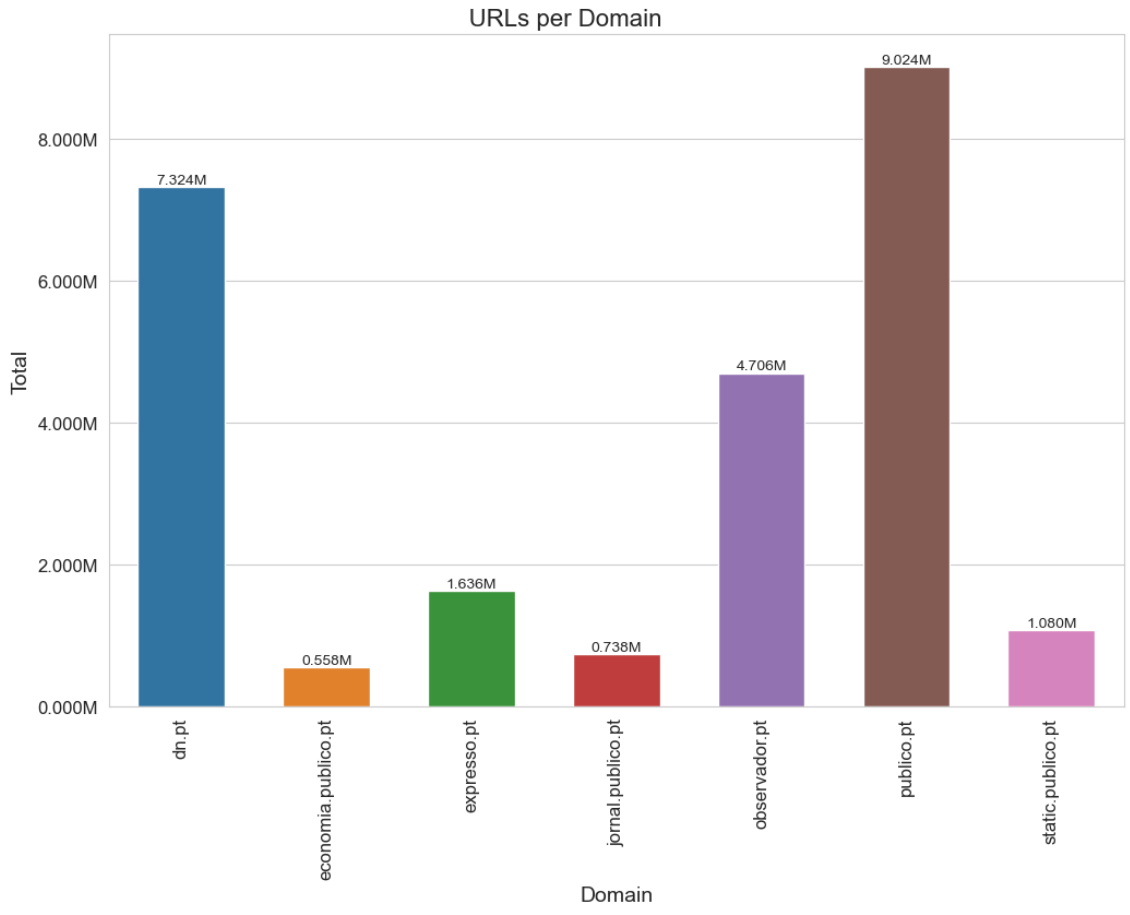


Figure 3.2: URLs in the filtered PWA's indexes, with duplicate entries from different timestamps.

[2] https://dados.gov.pt/pt/datasets/lista-de-dominios-do-jornal-publico-no-arquivo-pt-1996-2019/

These collections contain duplicate entries of the same URL to record the evolution of pages. Since we aren't interested in the page's evolution but instead on the original content of the article, further filtering is required to deduplicate. The result of the deduplication effort, where we choose to keep the earliest crawl of each page, was a reduction from **29 323 851** entries to **2 816 341**, distributed amongst the domains as seen in Figure 3.3. The filtering process took about **6 hours** and the final index file[3] sized at **1.2 gigabytes** of just CDXJ entries.

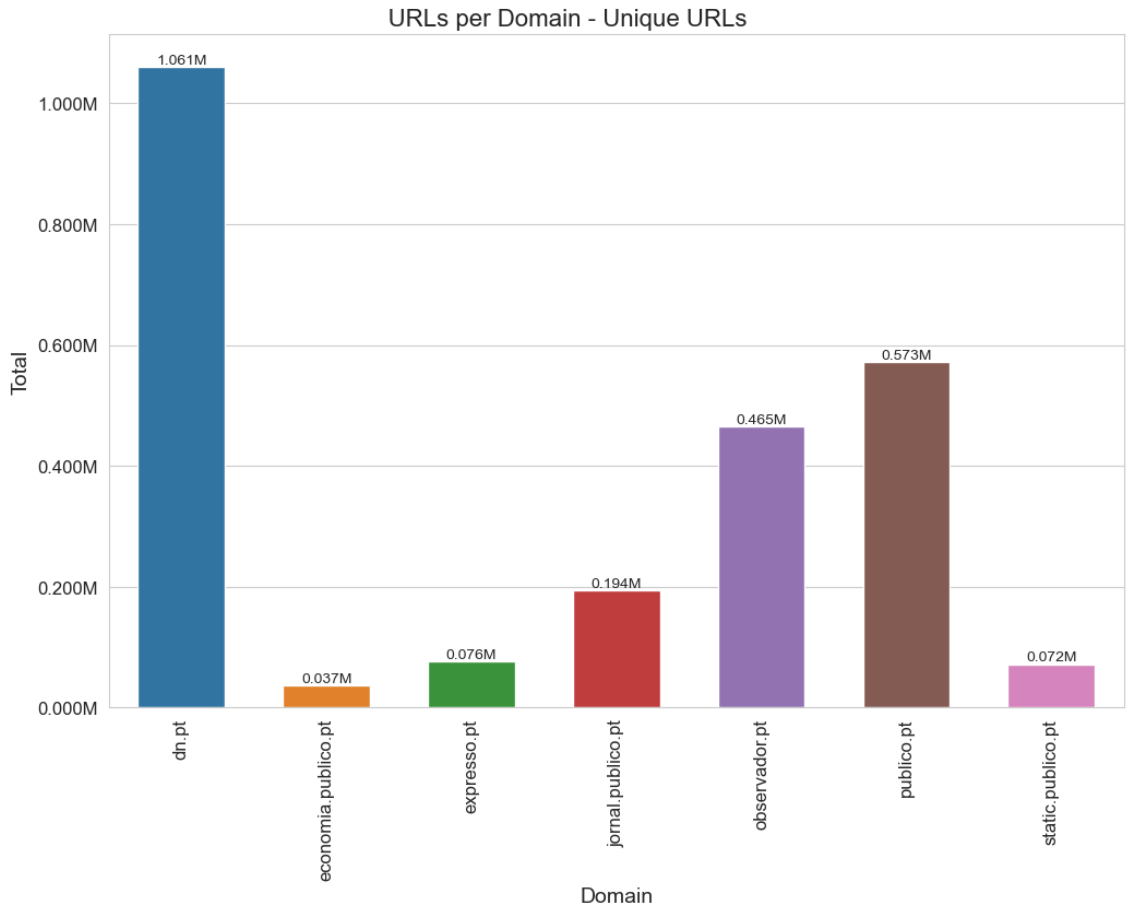


Figure 3.3: Only the earliest timestamp of each URL in the filtered PWA's indexes.

After completing the filtering, the scraping process to create the dataset corpus started. Each of the remaining **2 816 341** web pages in the filtered index were downloaded, at the selected timestamps, through the PWA's archived page direct view (referenced in the Arquivo.pt API as NoFrame view), process which took about **4 days** and resulted in **358 gigabytes** of HTML files.

Next, making use of newspaper3k[4], each of the downloaded web pages was scraped for it's relevant content - the article's title, body, and publish date - which were saved in

[3] https://github.com/newsArchivePT/ArquivoPT-QA/blob/main/filtered_ALL_SORTED_OnlyFirstTimeStamp_FAWP1_43.cdxj

[4] https://newspaper.readthedocs.io/

JSONs. Sometimes the scrape of the publishing date failed and we used the crawl's date as a good approximation because, given the regularity of these crawls, the first ever crawl of said article is likely to have been close to it's publishing date. We also kept the PWA's archived page view URL of each article, for any further scrapping required, for example to gather the article's images. The resulting corpus consists of **5.3 gigabytes** of news articles encoded as JSON data.

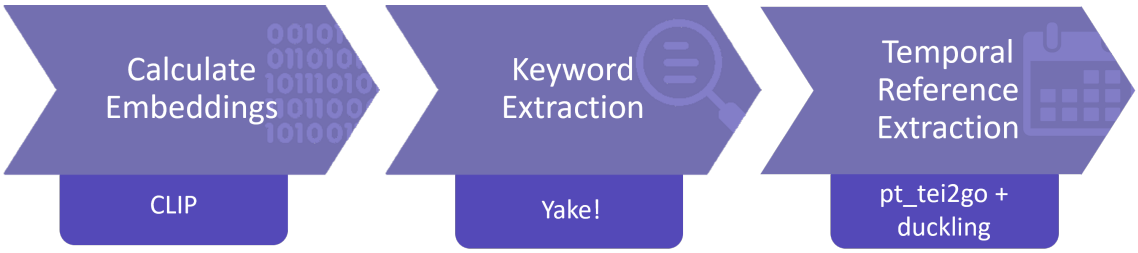


Figure 3.4: Corpus processing pipeline.

This corpus was then later subjected to further processing. For each news article CLIP[5] embeddings were calculated, as well as keywords extracted using Yake! [6], and finally through a combination of pt_tei2go [43] and duckling[6] every temporal reference present was extracted.

[5] https://huggingface.co/sentence-transformers/clip-ViT-B-32-multilingual-v1
[6] https://github.com/facebook/duckling

4

Temporal Multi-document Question-Answering and Search

With the belief that implementing Question-Answering over Web Archives would provide users with a more natural and easier method of acquiring information, and thus enhance their experience with such archives, we set out to build one such system capable of browsing a catalog of documents within the millions, find the right document, and retrieve an answer to the user's query from the document. This system, which we will now describe in this chapter, was thus built by employing various document search and indexing strategies, and assembling existing language models, with no specialized training, into a voting ensemble.

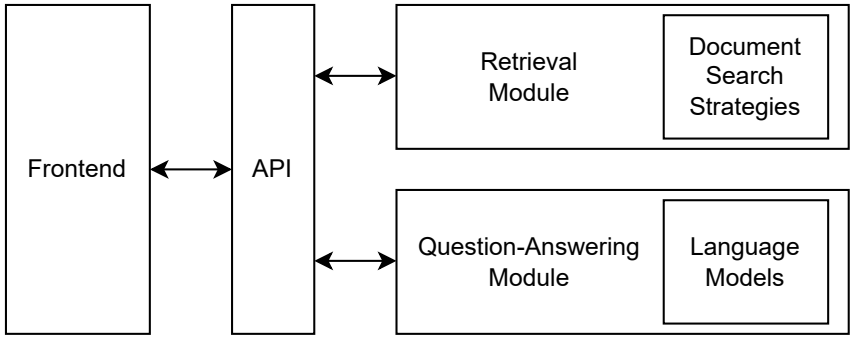


Figure 4.1: QA system overview.

4.1 Semantic Search for Extractive QA

When answering a question, one is always referencing a source of knowledge previously acquired. In a Question Answering system, it is no different and as such, in order to answer a question, one must first find source material to reference when answering. We seek for a system capable of searching through millions of documents, not only through their content, but also through their meaning, as finding a meaningfully related document to the question is the top priority, as a question's content might not be directly comparable to the document's content, in the sense that the question might refer to the same topic as

the document but not have compatible wording with it for it to be returned by a simple full-text search. Also given the inherent temporality of documents within a web archive, special attention must be given to temporal references both within the document and the question.

As such, we propose an hybrid document ranking approach comprised of four parts, full-text (BM25) search, embedding-based search, keyword search, and time filtering. We choose to use OpenSearch[1] as our document index and search engine, as it supports the required search capabilities to support these.

At document ingestion time, we indexed both the title and body of each document, along with CLIP [35] embeddings[2] for the title, the first paragraph of the body, and the full body. The top 20 keywords from each document were also extracted using Yake! [6] and indexed along side the document. Having these enables us to boost documents whose main topic matches that of the question. Furthermore, by using pt_tei2go [43], we extracted every natural language temporal reference within the document, which was then processed, using duckling[3] with the document's creation date as base reference for relative time references, into standard date strings[4] that where indexed alongside it's corresponding document. Having these enables us to, when in presence of a temporal reference within a question, refine the search by filtering out the documents that don't contain dates surrounding the referenced date.

In the proposed system, searches are performed over the index with a query comprised of 4 parts:

- The question text as a query for the full text search over both the documents' title and the body;
- An embedding generated from the question text for the embedding search on the 3 fields (documents' title, body, and first paragraph);
- The top 5 keywords extracted from the question for the keyword search;
- Any temporal reference expressed in the question originates a time range for the time filtering; these time ranges are based on the granularity of the extracted temporal reference from the question (I.e. if the extracted time is a year the resulting time range will contain every timestamp within that year); multiple time ranges originating from the same question are combined by means of disjunction.

Matches with any of the first 3 parts of the query will result in the document getting its relevancy boosted the more matches it accumulates. However, when it comes to the temporal references, these are used as filter, meaning, that when present, at least one

[1] https://opensearch.org/

[2] these would be embeddings from the sentence-transformers/clip-ViT-B-32-multilingual-v1 model found on: https://huggingface.co/sentence-transformers/clip-ViT-B-32-multilingual-v1

[3] https://github.com/facebook/duckling

[4] https://www.iso.org/iso-8601-date-and-time-format.html

match is required for the document to be considered, and consecutive matches won't affect the relevancy score.

4.2 Single-document QA Strategies

Given a question and the existence of a relevant document containing the answer, how does one extract the answer from the document? Here we turn to existing language models trained on exactly this task, so called Extractive QA, where encoder type language models, like the ones in the BERT [15] family, are fine-tuned to read a context and return a start and end index to the span, within the context, that contains the answer.

Sticking to the idea of using only existing models, and due to the in-existence of a comprehensive Portuguese extractive QA dataset, we leveraged existing extractive QA models on Hugging Face[5]:

- Model 1 - timpal0l/mdeberta-v3-base-squad2 [6]
- Model 2 - pierreguillou/bert-base-cased-squad-v1.1-portuguese [7]
- Model 3 - mrm8488/bert-base-portuguese-cased-finetuned-squad-v1-pt [8]
- Model 4 - pucpr/bioBERTpt-squad-v1.1-portuguese [9]
- Model 5 - eraldoluis/faquad-bert-base-portuguese-cased [10]
- Model 6 - pierreguillou/bert-large-cased-squad-v1.1-portuguese [11]

Model	Base Model	Parameter Count	Finetuned on
1	mdeberta-v3-base [12]	86M	SQuADv2 [13]
2	bert-base-portuguese-cased [14]	110M	SQuADv1.1-PT [15]
3	bert-base-portuguese-cased [14]	110M	SQuADv1.1-PT [15]
4	bert-base-portuguese-cased [14]	110M	SQuADv1.1-PT [15]
5	bert-base-portuguese-cased [14]	110M	FaQuAD [16]
6	bert-large-portuguese-cased [17]	335M	SQuADv1.1-PT [15]

Table 4.1: Base models and datasets used to finetune to achieve the models used in our approaches.

In Table 4.1 we present further details about each of the available models. In this table we can see the base models that the models we use were based upon, their sizes

[5] https://huggingface.co/models?pipeline_tag=question-answering&language=pt&sort=downloads

[6] https://huggingface.co/timpal0l/mdeberta-v3-base-squad2

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

[8] https://huggingface.co/mrm8488/bert-base-portuguese-cased-finetuned-squad-v1-pt

[9] https://huggingface.co/pucpr/bioBERTpt-squad-v1.1-portuguese

[10] https://huggingface.co/eraldoluis/faquad-bert-base-portuguese-cased

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

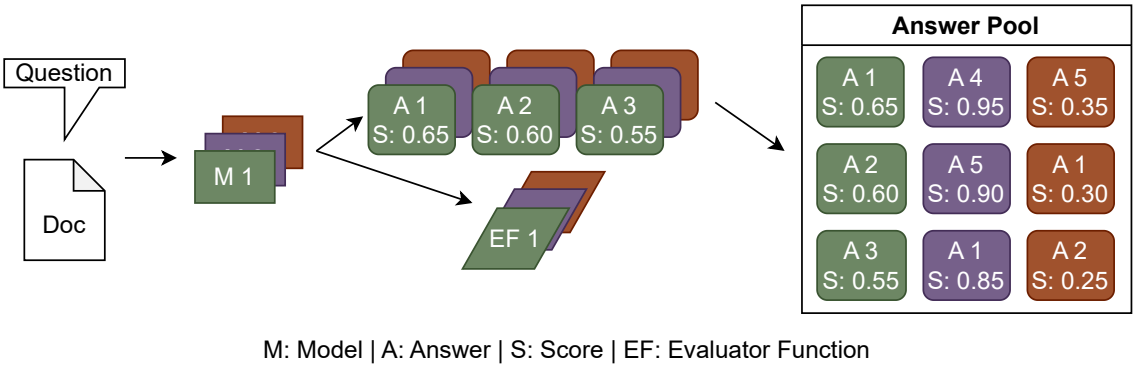


Figure 4.2: Single-document QA's 1st step: per model answer propositions and logits encapsulated in a evaluator function.

in terms of parameters, and the dataset used to finetune the base model into the models we use. As can be seen, all but one of the models are based on the same base model, BERTimbau[44], with models 2 to 5 being based on BERTimbau-base, and model 6 being based on BERTimbau-large, being model 6 the largest model here present with its 335M parameters, while every other model is close to 100M. The last model, model 1, is based on the base model mdeberta-v3-base, which is a multilingual version of the DeBERTa model[21]. Lastly, with the exception of model 1 which was finetuned on the SQuADv2 dataset, and of model 5 which was finetuned on the FaQuAD dataset, all other models were finetuned on the Portuguese translation of SQuAD, SQuADv1.1-PT.

The available options left us however without a clear choice as thus, even within our target language of Portuguese, there were still various options all with similar performance, as can be seen in Table 6.2. Although the similar reported performance, the choice isn't as easy as picking any of them, specially when generalizing to a different domain, like Portuguese web archived news. Due to the nature of training such models, each might've become better at answering different types of questions.

Thus we set forth, such that given a set of models, we pick the best answer amongst the ones given by these. We start by forwarding the document (as context) and question through each model and joining all the answers in an answer pool, as seen on Fig.4.2, and propose different strategies to select the best one.

4.2.1 Document-level - Highest Scoring Answer

To achieve such goal we initially attempted the naïve approach of simply picking the one with the highest score, as seen on Fig.4.3. This approach quickly revealed to be sub-optimal,

[12] https://huggingface.co/microsoft/mdeberta-v3-base
[13] https://huggingface.co/datasets/squad_v2
[14] https://huggingface.co/neuralmind/bert-base-portuguese-cased
[15] https://huggingface.co/datasets/squad_v1_pt
[16] https://huggingface.co/datasets/eraldoluis/faquad
[17] https://huggingface.co/neuralmind/bert-large-portuguese-cased

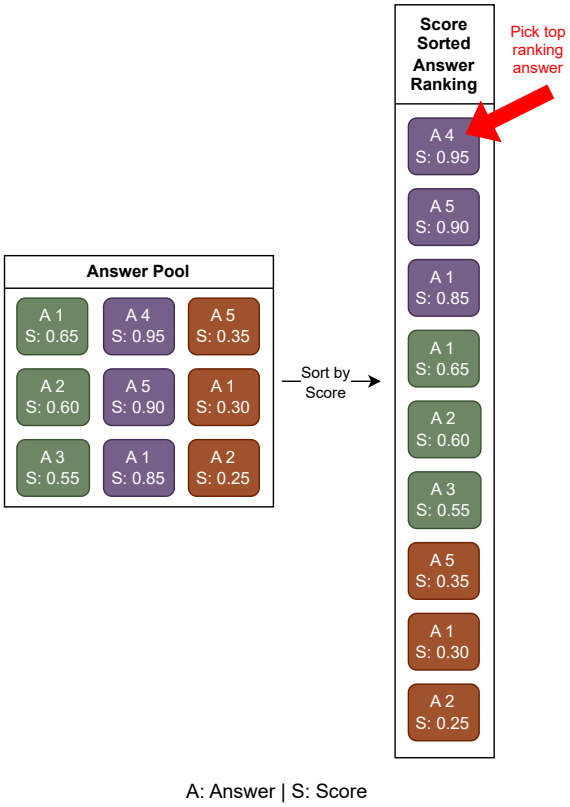


Figure 4.3: Single-document QA's - Top Score.

as the scores were given to each answer by the model which presented them, and as such not only was each model fairly certain of it's answer, but also, due to differences in training, each model's score scale differed greatly making them not comparable. For example some models always answered with a score close to 100% while others never went over 50%.

To overcome this, instead of each answer only being evaluated by their origin model, we propose a strategy in which each model evaluates *all* the candidate answers from other models. Then, individual model assessments can be combined, such that the final answer is a result of the models' joint effort.

But first we must introduce two concepts, Reciprocal Rank Fusion, and Voting Ensemble.

4.2.1.1 Reciprocal Rank Fusion

Reciprocal Rank Fusion or RRF, Cormack et al. [10], is a simple approach to combine different rankings of a group of documents, without taking into consideration each of the ranking's relevance indicators and document scores, thus enabling the combination of rankings with unrelated scoring criteria.

$$RRFscore\,(d \in D) = \sum_{r \in R} \left(\frac{1}{k + r\,(d)} \right) \tag{4.1}$$

RRF is a very simple method that sorts the documents in the rankings by using only their positions in these. It's scoring, as seen on Equation 4.1 where D is a set of documents, R a set of rankings, and k=60, is the sum of the inverse of the document's position in each rank, thus favoring documents at the "top" of the rank and penalizing them the "lower" they are.

4.2.1.2 Voting Ensemble

A voting ensemble is an algorithm that combines predictions from multiple models. It may be composed of any existing machine learning models with no specialized training or awareness they are used in an ensemble. The models used however should be of similar performance and mostly already agree on their predictions, because every model's opinion will be weighed the same. The ensemble's purpose is achieving better performance than any single model used, meaning that if any single model surpasses the ensemble it should be used instead.

A voting ensemble can be used for both regression or classification tasks, the former by averaging the models' predictions, and the latter by summing the predictions for each label and predicting the majority vote. Classification tasks may be further divided between hard voting and soft voting. Hard voting, used for models that choose one of the class labels, is when the amount of predictions for each class label are counted and the class label with the most predictions by the models is the one chosen. Soft voting on the other hand may be used when the models attribute probability, or probability-like scores, to each of the class labels, it works by summing each of the label's scores and selecting the highest one.

4.2.2 Document-level - Best Answer Voting by Rank Fusion

With the goal of having every model evaluating each of the answers in the answer pool and contribute to it's score, we made it so that, as seen on Fig. 4.2, along with it's suggested answers, running each model also returns an evaluator function that encapsulates the models logits. This evaluator function attributes a score to any span within the previously given context. Making use of this evaluator function we make every model evaluate each of the answers in the answer pool, effectively creating for each model, a ranking of it's confidence in each of the possible answers. We then, as seen on Fig. 4.4, combine each of the model's ranks into one single ranking by means of Reciprocal Rank Fusion 4.2.1.1, where the score in the new rank is given by the Equation 4.1, with D being the answer pool, and R the answer ranking by model. This approach constitutes a soft voting ensemble 4.2.1.2, as each model through it's answer ranking, during RRF, "votes" on each answer with a probability-like score relating to the position it attributed to it on it's rank. Answers that rank higher in each of the model's ranks, get more higher votes and thus end higher in the final ranking. With this approach, we reach the desired contribution of every model

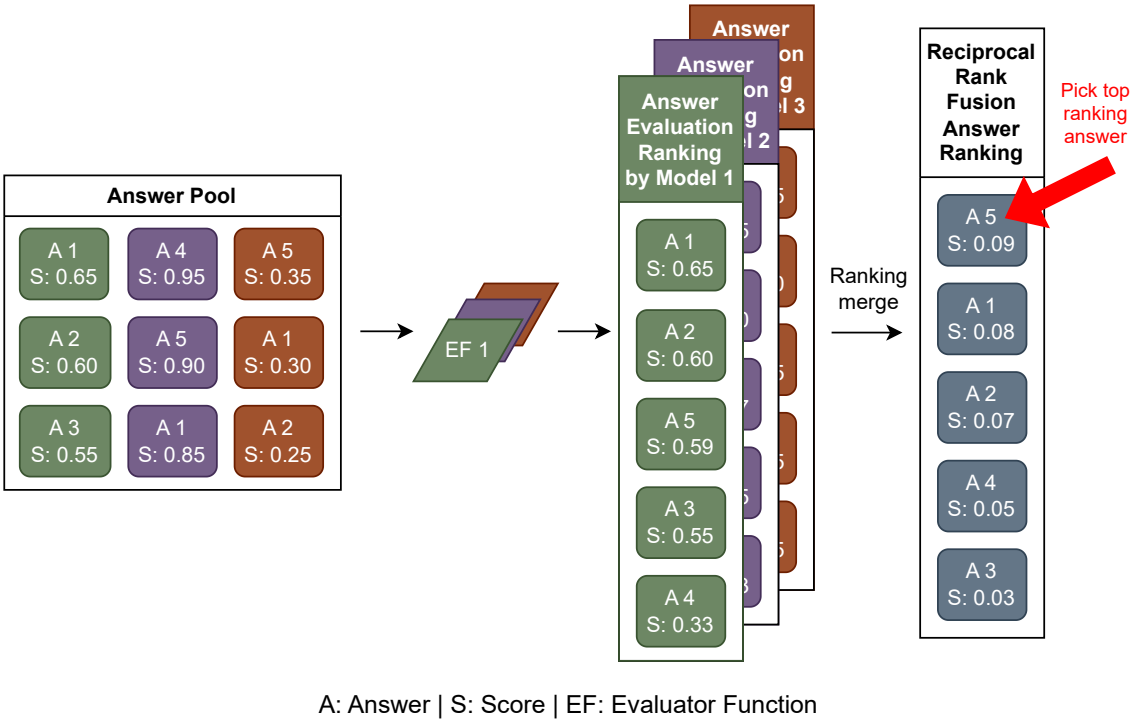


Figure 4.4: Single-document QA's - Answer Voting.

to the chosen answer, being the chosen final answer the most "voted" on by the models, and consequently the one at the top of the merged ranking.

4.2.3 Evaluation Function with Temporal Modifier

Following the belief that temporal references are important within the context of QA over web archives, and continuing the efforts that were already started at the search level in section 4.1 with the time filtering, we set out to also take these into consideration at a document level.

We started with the simple assumption that when an question contains a temporal reference also present within the document text, candidate answers positioned closer in the text to the temporal reference might be of greater relevancy than those further away. This assumption comes from the intuition that the closer two parts of the document text are, the bigger the probability they pertain to the same event, this is particularly the case for news, given the way they are structured.

Thus we set out to build a new evaluation function with consideration for these temporal aspects, that could take the place of the evaluation function in the solutions of the previous sections, making them temporally aware.

Our approach consists of calculating a distance-based "temporal importance" percentage, that would be higher the closer within the document the candidate answer being evaluated and a target temporal reference are, and then apply it as a modifier to the result

of the original scoring function by multiplication.

We propose 3 different ways of calculating this modifier, all following the formula $1 - \frac{d}{t}$. Where $\frac{d}{t}$ represents the percentage of how separated the answer and the temporal reference are, with d being the distance between the two and t the maximum distance, 1 would mean they are as far as possible and 0 as close as possible. Given we require instead a value that represents how close the answer and the temporal reference are, we employ the complement of this percentage instead. The proposed modifier options are as follows:

- **Character distance** - in this method d is the minimum distance, in number of characters, between the span containing the target temporal reference and the span of the candidate answer, and t is the document's character count;
- **Sentence distance** - in this method d is how many sentences away from each other the target temporal reference and the candidate answer are, and t is the document's sentence count;
- **Context distance** - this method takes into consideration the inner workings of the Question-Answering pipeline of the Hugging Face library upon which we are building our solutions. This pipeline, when the context provided overflows the token limit, splits the context into smaller chunks by means of a sliding window. This method takes into consideration that when the document gets split into various contexts for the QA, both the target temporal reference and the candidate answer might end up in various different contexts, and thus for d finds the minimum distant contexts, and for t uses the context split count;

This function is given the candidate answer to evaluate, but how does it select the target temporal reference to score it against, and consequently find the modifier? Between a document and a question, both with temporal references, there might be multiple temporal references that match by means of a similar time-range method to the one presented in Section 4.1. Here for each temporal reference in the question, a time-range based on it's ganularity is created and any reference in the document that falls within it is a match, and thus a potential target temporal reference for the modifier calculation. Assuming that as long as it is a match, every temporal reference is equally relevant to the question, and thus any answer will be more relevant the closer to any temporal reference, so we proceed to compute the modifier with every match as a target temporal reference and choose the one which originates the highest relevance modifier.

4.3 Multi-document QA Strategies

Previously we saw how to perform QA over a single document, however, in a real use-case scenario, it is likely that the user does not know the reference document, and is instead in possession of a collection of documents, possibly, the whole web archive. Therefore, it is crucial to generalize to a multiple document setting, where one considers more than

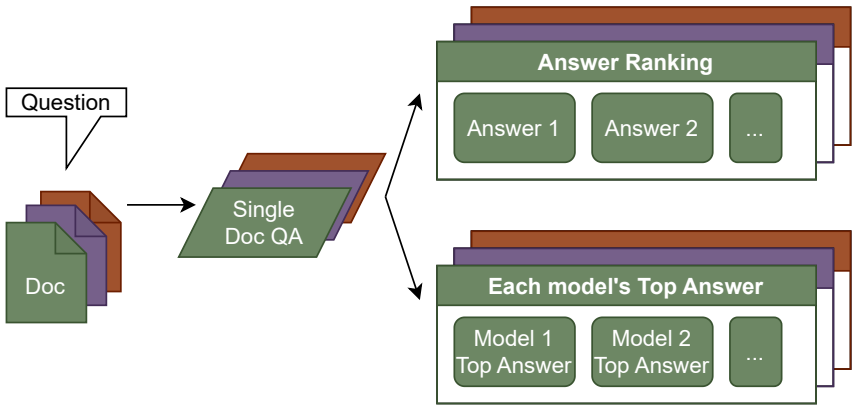


Figure 4.5: Multi-document QA's 1st step: per document single-doc QA resulting in answer ranking and top answer per model.

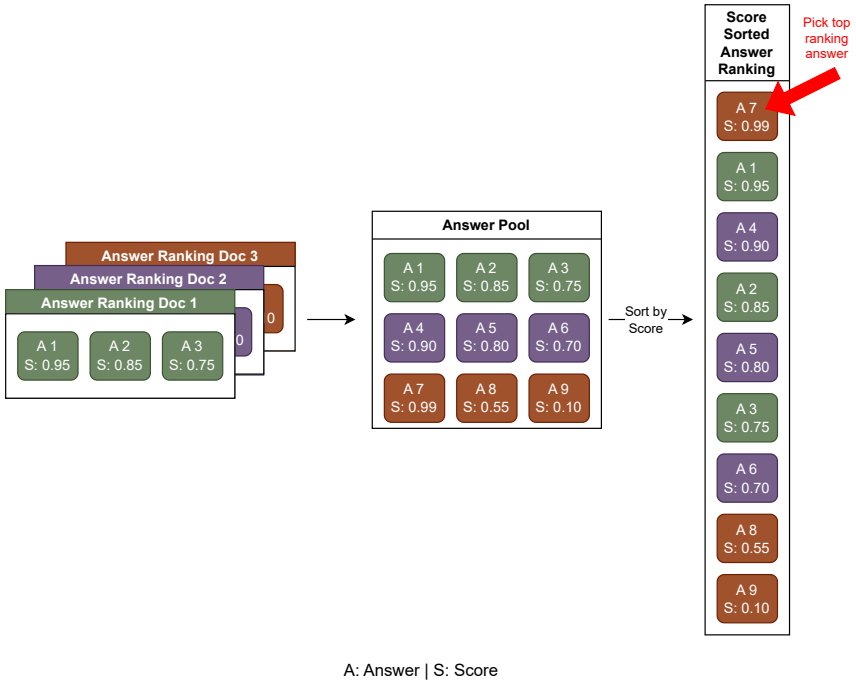


Figure 4.6: Multi-document QA's - Top Score.

one document as an answer reference candidate. Given this need, the problem of how to select the best document, and consequently the best answer, needs to be addressed.

For our proposed solution, given a query, we start by using our semantic search component to reduce the collection of documents to a pool of promising candidates. Then, we forward the question and each candidate document through our single document QA approach, and collect the top answers from each document, as seen on Fig.4.5. For the next step, similarly to the single-document QA strategy, we propose two distinct strategies.

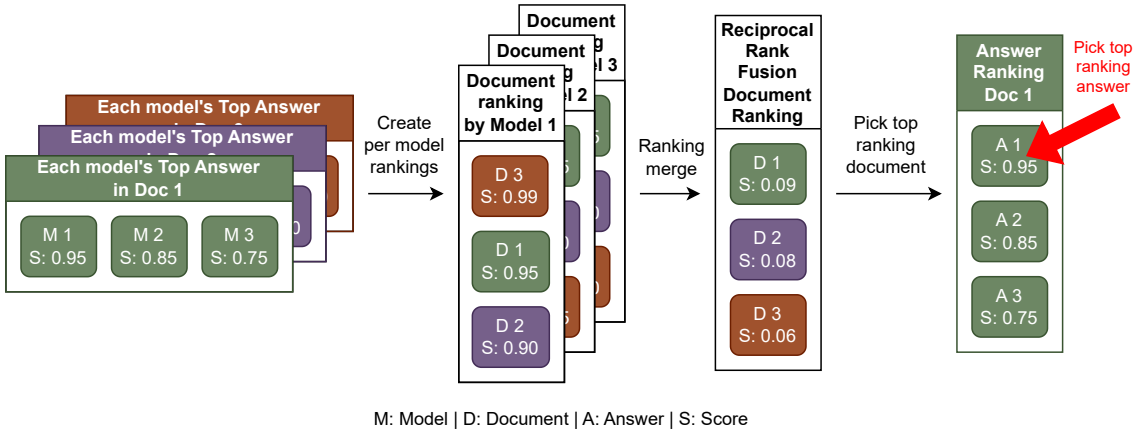


Figure 4.7: Multi-document QA's - Document Voting.

4.3.1 Multi-document level - Highest Scoring Answer

First we attempted the same naïve approach we did in the single document QA, and got the very same problems, as seen on Fig.4.6. We joined all the top answers from the multiple documents in a global answer pool, and tried selecting the highest scoring one as our final answer. Much like in the single document QA, this approach is sub-optimal since the scores aren't comparable. If we use the this strategy with the single doc approach from 4.2.1 we run into exactly the same problems of score incompatibility between models, leading to some models dominating the answers. Furthermore, if instead the single document approach chosen were to be the one in 4.2.2, then the incompatibility would be between documents, since the RRF scores in the top answers of each document wouldn't have taken the answers from other documents into consideration, which makes them of no significance in relation to each other.

4.3.2 Multi-document level - Best Document Voting by Rank Fusion

We took a step back, and realized we were once again trying to decide which answer was the best amongst a pool of answers, but this time removed from their contexts, making answers from different contexts incomparable and their scores incompatible, as it was this shared context that allowed the models used in single doc QA to score an answer over another. Thus we shifted our focus to selecting the best document to answer the given question.

We started by expanding the Single Doc QA, as seen on Fig. 4.5, to not only to return the Document's top answers ranking, but also specifically return each model's top answer within the Document. The score of each model's best answer within a document will be used as the "confidence" score that said model attributes to the document being the one that contains the answer to the question. Thus, when running Multi-Document QA, after forwarding the question and Documents through Single-Document QA, each

model's best answer from each document is collected and compiled into a document "confidence" ranking for said model. These document rankings are then, as seen on Fig. 4.7, merged into one single rank by means of Reciprocal Rank Fusion 4.2.1.1, with D being the document pool, and R the document confidence ranking by model. Similarly to the answer voting 4.2.2, this approach constitutes a soft voting ensemble 4.2.1.2, as each model through it's document "confidence" ranking, during RRF, "votes" on each document with a probability-like score relating to the position it attributed to it on it's rank. Thus documents that place higher in each of the model's ranks, get more higher votes and thus end higher in the final ranking. From this merged ranking the top document is chosen and it's top answers ranking accessed, selecting it's top answer as the final answer.

5

ArquiWIZ: Prototype for the Portuguese Web Archive

With the purpose of demonstrating our proposed solutions and how these could be applied to an existing web archive like the Portuguese Web Archive, we created ArquiWIZ[1], a prototype which we will be discussing in this chapter. First we will introduce the prototype's interface and features, and then dive deep into it's inner workings to explore how it was built.

This prototype was jointly development between this and two other works [7] [28], and was submitted for consideration to the "Prémio Arquivo.pt 2023"[2] competition where it earned a place among the top 8 contenders. This is an annual competition organized by the Portuguese Web Archive with the purpose of awarding innovative works that make use of the data within the archive, as well as demonstrate and bring awareness to the utility of the archival of web data. From the data in the Portuguese Web Archive, this prototype's main focus is on archived web articles from four of the primary Portuguese news outlets: Público, Diário de Notícias, Expresso and Observador. And thus, we positioned ourselves in the competition as a tool for journalism investigation following the 5W1H methodology (Why?,What?,Where?,When?,Who?,How?), were we provided users with open and easy access to information, and most importantly, answers to these questions.

5.1 User Interface

With the goal of providing users with easy access to information, we set out to build a simple interface that let users naturally interact with the system and easily browse the content within it, which we will now present in this section.

The User Interface is split into 3 main pages:

1. Questions and Search - here users can perform a traditional search through the documents in the system, or ask questions and get answers extracted from said

[1] http://arquiwiz.pt/

[2] https://sobre.arquivo.pt/pt/colabore/premios-arquivo-pt/premio-arquivo-pt-2023/

documents;

2. Entities in Time - this page provides users with a bubble cluster view of how an entity (People, Organizations, Locations, etc.) relates to others;

3. Explaining Connections - in this last page the user can visualize how two entities are connected;

An important part of our interface, present in every page, is the contextual interactive agent we introduced to aid our users. This agent is represented by our mascot, the wizard, and provides the user with guidance and contextualized suggestions on how to further explore and research the topic at hand.

As previously mentioned this prototype was a joint effort between this and two other works, and as such not only it has features that integrate with our proposed solutions, as well as features that fall outside of the scope of this work, specifically the "Entities in Time" and "Explaining Connections" pages [7], and as such we will only quickly overview these.

5.1.1 Questions and Search

The first page is "Questions and Search", a page were users can interact with a search box, and perform either traditional keyword searches, or input a question for the system to answer.

In both cases, the system provides the user with a list of relevant documents for it's query, as can be seen on the left of Fig.5.1, as well as contextualized follow up questions, that were generated based on the relevant documents, to aid the user in it's research. They can be seen on Fig. 5.1 right under the search bar at the top. The relevant documents list starts with the top 10 relevant documents, but can be further expanded by scrolling down. When the user's query is a question, it gets detected by the system and, making use of the proposed solution in Section 4.3.2, it tries to find an answer amongst the relevant documents it previously presented. The answer is then provided to the user by the wizard on the right side of the page, and the document it originated from gets highlighted, as seen on Fig.5.1 on the second result.

At any point during search, a user might want to read a document further. For this, he can then click on any of the documents in the list to open it. In this view the user is provided with two ways to visualize the document:

- Archived Page Mode (Fig. 5.2) - in this mode the user will see a live reproduction of the original page as it were in the moment it was archived by the Portuguese Web Archive. This mode is fully backed by the PWA's "noFrame" view API endpoint;

- Extracted Content Mode (Fig. 5.3) - in this second mode the user will instead see a parsed version of the webpage's main content, where we extracted the news article's title, body, date, and images, as well as highlight any entities present (clicking on these will take the user to it's "Entities in Time" 5.1.2 page)

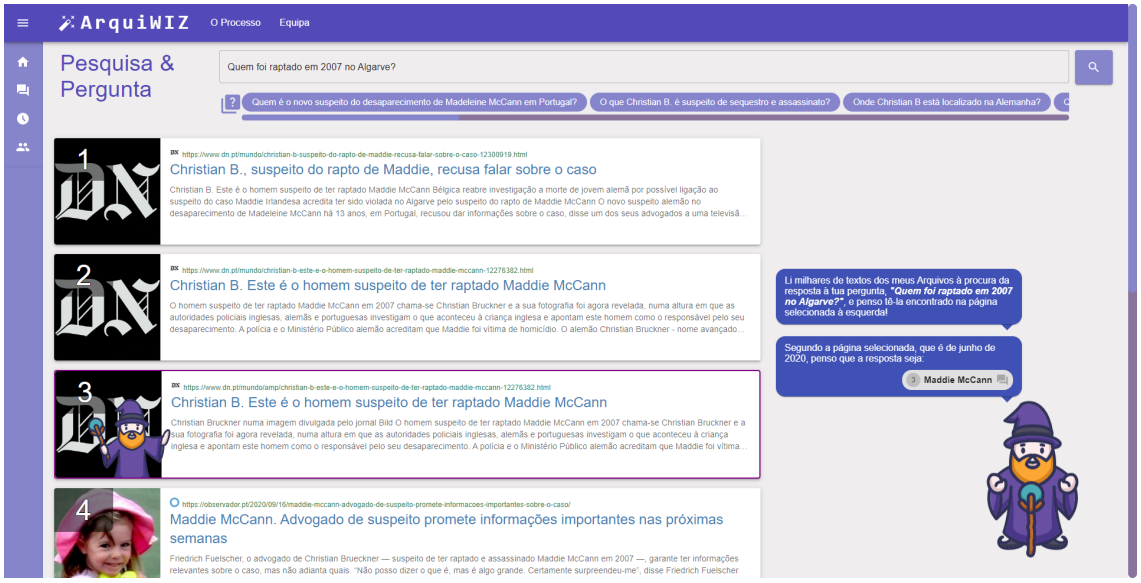


Figure 5.1: "Questions and Search" page showing example of question with temporal reference.

Figure 5.2: "Questions and Search" page document view in Archived Page Mode.

Figure 5.3: "Questions and Search" page document view in Extracted Content Mode, with question being answered.

Also on this view, on the right, there is a text box were we provide the user with the opportunity to ask more focused questions, now only within this document, this makes use of the proposed solution in Section 4.2.2. Above are again contextualized follow up question suggestions, generated based on this document's content, to aid the user exploring it. Answers to these questions, and the ones made by the user, will once again be provided by our wizard, but he will also, as seen in Fig.5.3, draw the user's attention to the "Extracted Content Mode" where the answer will be highlighted in red, and the context used by our Question Answering system to determine the answer will be highlighted in green. These highlights enable the user to more easily pinpoint the origin of the answer and easily gather any additional related knowledge that might be useful to it's question.

5.1.2 Entities in Time

In "Entities in Time", as seen on Fig. 5.4 we provide a bubble cluster view of how a selected entity relates to others. In this view, the user is presented with color coded bubbles representing any entity that the selected one relates to. These bubbles' size will vary across time depending on the variation across time of the strength in relationship between the two entities. The user may also select bubbles to be separated from the cluster to more easily see it's entity evolution, and, on the right of the cluster view, there is further information about the selected bubble and the news articles that relate the bubble's entity with the selected one.

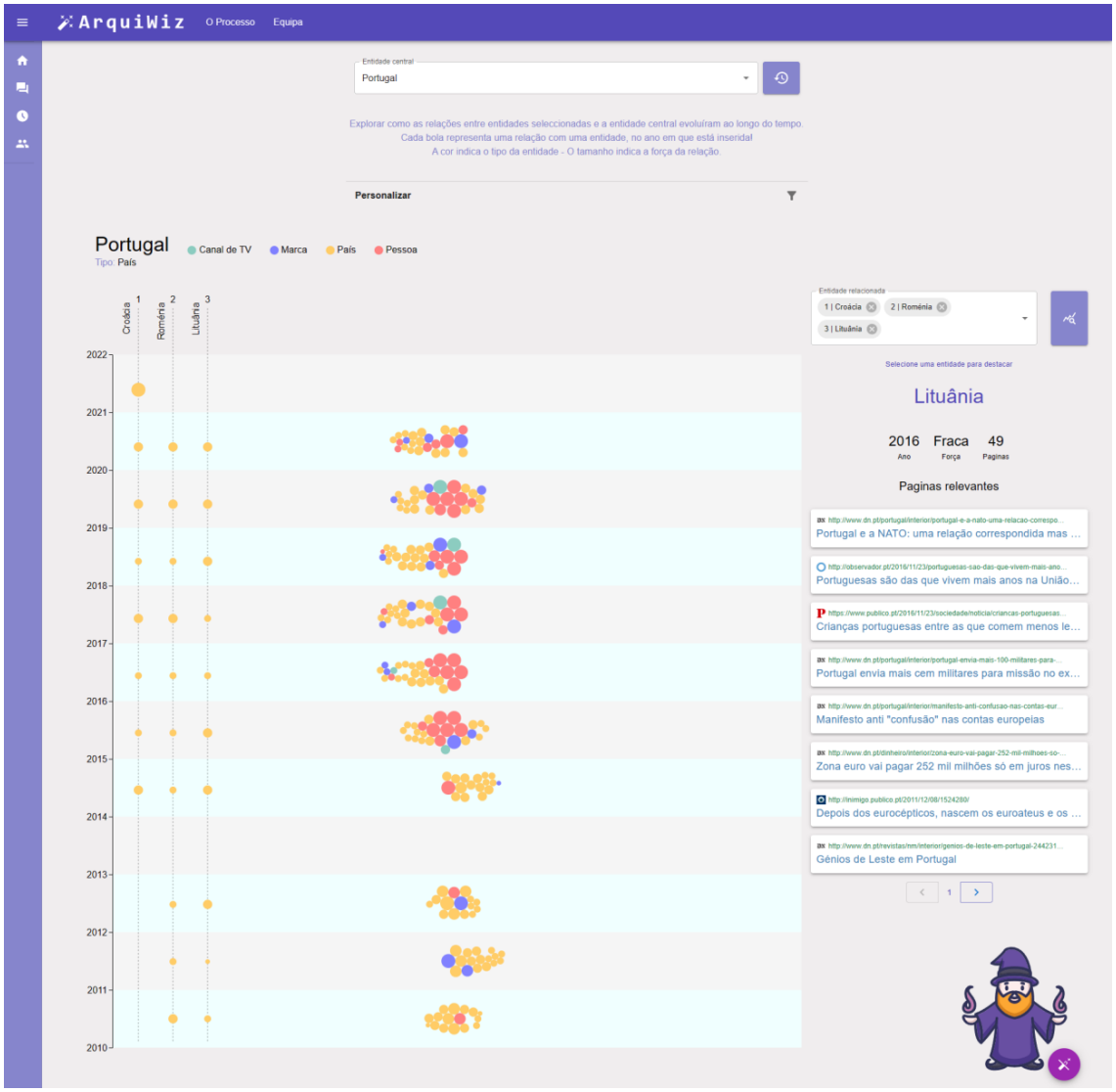


Figure 5.4: "Entities in Time" page showing example of relationships with Portugal.

5.1.3 Explaining Connections

The final page is "Explaining Connections", a very simple but powerfull visualization. In this page we allow the user to select any two from the entities available and find the shortest path, in news articles, that relates the two entities. This allows users to find unexpected connections between entities, that would be hard relate otherwise. As seen in Fig 5.5, relationship paths are presented top to bottom, with every connection arrow being accompanied with the news articles that relate the two entities in that step.

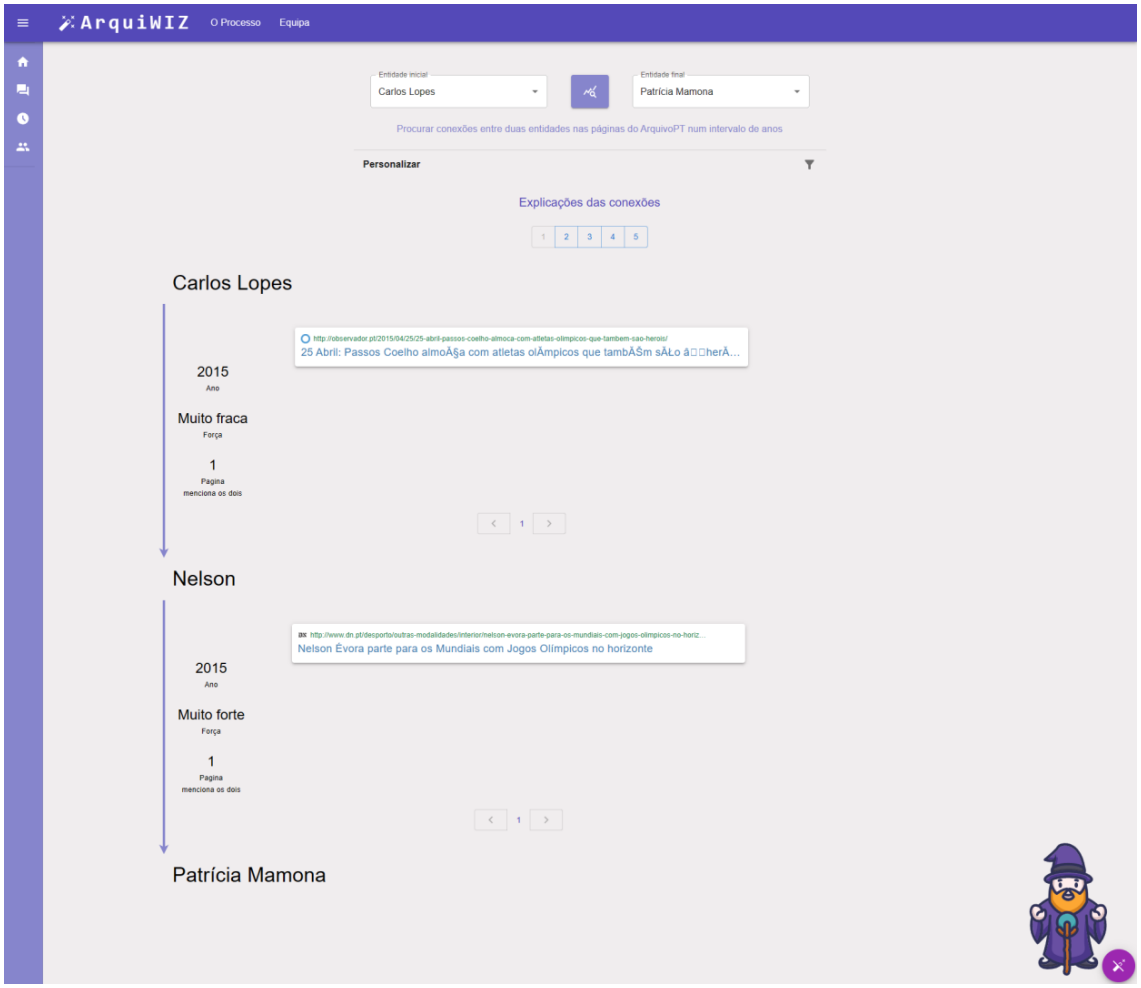


Figure 5.5: "Explaining Connections" page showing example of path between entities with one intermediary step.

5.2 System Architecture

A system capable of providing users with these features is a system that must articulate many different parts. One such system requires careful consideration in it's implementation and coordination of it's pieces, as well as in the distribution of computation across the available resources. For reference, ArquiWIZ was deployed on a system with 4 cores from an Intel Core i7-3930K (3.2GHz), 16GB of RAM, and two NVIDIA Geforce GTX 1080 TI (11GB) GPUs. The management of GPU resources required special attention, due to memory constraints and it being the deciding factor in system response speed, since the language models, which run on the GPU, are by far the most processing intensive part of the system. In this section we shall discuss the pieces chosen to build this system, illustrated by the diagram seen in Fig.5.6, and how they are connected.

Starting front to back, the system's UI is built using React and communicates with the backend through a Flask API. When the API's endpoint are hit, calls to the Orchestrator are made to perform the required operations. This Ochestrator oversees everything, it

controls and coordinates the interaction between each of the other 3 modules:

- **Graph Module** - it oversees the parts of the system related to Entity Relationships, it is responsible for both the "Entities in Time"5.1.2 and "Explaining Connections"5.1.3 pages, and it's out of the scope of this work;
- **Retrieval Module** - this module controls the retrieval of documents, assuring that relevant documents to the user query are returned by making use of various filtering and search techniques;
- **Question-Answering Module** - lastly this is the biggest module, it is in charge not only of answering the user's questions, but as well of identifying if the user's query is a question, and finally it is also in charge of generating contextualized follow up questions;

The backend follows a stateless pattern, so there are no user sessions. Any session state, like currently selected filters for example, is sent up to the server on every new request. This approach is possible as there is no sensitive state, and all endpoint and functionalities can technically be used independent from one another.

Endpoints may however complement one another and thus the UI make multiple calls to the server when providing certain functionalities. One such example would be when asking a question in "Questions and Search" 5.1.1. For this feature the search for relevant documents and the answering of the question is split into two different API requests, where the first request provides the UI with the list of relevant documents to show the user, but also identifies if the user's query is a question, in which case the UI triggers the second request using the previously acquired list of relevant documents as the context for the question-answering. While both parts of this feature could have been bundled into a single request, since they correspond to a single interaction from the user, due to the processing time of QA this would have left the user with no feedback for its entire duration, which is about 3-10 seconds. Splitting the feature across two requests allows us to keep the user entertained during the small duration of the QA, by quickly providing feedback to the user by displaying the relevant documents, and by masking the processing time by having the interactive agent stall for time by praising the user on it's interesting questions, and claiming it needs to think before answering. This approach to mitigate the processing time of these models might also be seen when it comes to the contextualized suggested follow up questions, as these are always in a separate request due to it's long processing time, of about 15-35 seconds, unless previously cached.

5.2.1 Retrieval Module

Delving deeper into each of the modules, we shall start with the Retrieval Module. As previously mentioned this module is responsible for searching and retrieving relevant documents to the user query, which it does by following the methodology previously

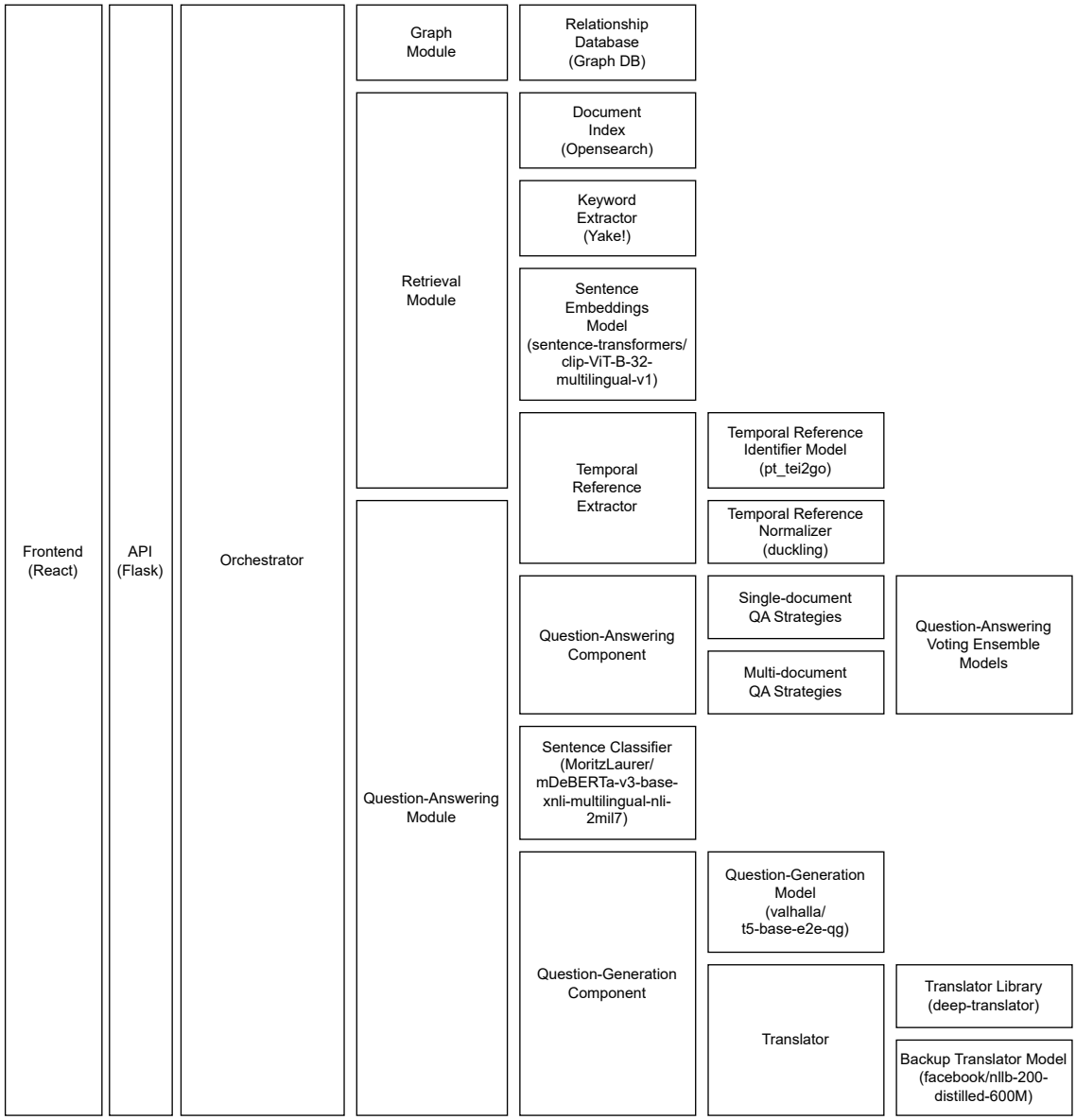


Figure 5.6: ArquiWIZ's architecture from frontend to backend and its components.

described in 4.1. It's main responsibility is the search feature in the "Questions and Search" page.

It makes use of 4 other components to perform its function, as seen on Fig.5.6:

- Sentence Embedding Model - this component's purpose is to encode the query's meaning into an embedding. For this we chose "sentence-transformers/clip-ViT-B-32-multilingual-v1"[3];
- Keyword Extractor - this component is responsible for identifying the query's top 5 keywords. For this purpose we chose "Yake!"[6];
- Temporal Reference Extractor - this component, also shared with the Question-Answering Module, is used to extract temporal references from the user's query, and is composed of two parts:
  - Temporal Reference Identifier Model - which has the purpose of identifying any temporal reference in natural language in the query text. For this we use "pt_tei2go"[43];
  - Temporal Reference Normalizer - which takes the temporal references in natural language originating from the previous component and normalizes them to the standard timedate string format. For this we make use of "duckling"[4];
- Document Index - where the documents are stored and over which the module performs the search after processing the query with the previous components. Here we make use of "Opensearch"[5];

At search time, by making use of the previously mentioned parts, this module compiles an OpenSearch DSL Query[6] that satisfies the query described in 4.1. At the root of this query is a boolean[7] query, which allows for the logical combination of various DSL queries. Firstly the queries that are included in the "Should" field, which acts as logical "Or" in the boolean query, are created. Consecutive matches in any of the queries in this field further boost the document higher in the result ranking, with only one match required for it to be included.

The first of these queries is a simple multi-match[8] query which performs full-text (BM25) search over multiple text fields, which in this case are the title and the body of the document. In this query, for scoring purposes, double the importance of the title is given to the body, as we believe the document body should be more important because while the title might be a good indicator of what the document is about, the body is where the content is, and where answers to questions will be present.

[3] https://huggingface.co/sentence-transformers/clip-ViT-B-32-multilingual-v1
[4] https://github.com/facebook/duckling
[5] https://opensearch.org/
[6] https://opensearch.org/docs/latest/query-dsl/index/
[7] https://opensearch.org/docs/latest/query-dsl/compound/bool/
[8] https://opensearch.org/docs/latest/search-plugins/sql/full-text/#multi-match

Next, an embedding of the user query is computed using CLIP, the chosen Sentence Embedding Model, and from it originates 3 k-NN[9] queries, one for each embedding field in the index. These are, the title, the body, and lastly the body's first paragraph. CLIP was chosen due to it's multimodal nature, which enables the possible future development of introducing image search. Regarding the existence of a embedding field both for the full body as well for only the body's first paragraph, this decision was made due to the innate limitations that language models have on their maximum input sizes. Ideally one would have an embedding that fully represents the meaning of the body. However this sometimes isn't possible due to it's size, so the decision to use the first paragraph as an approximation was made.

The last of the queries in the "Should" field is one that originates from the use of the Keyword Extractor Component. By making use of Yake!, the top 5 keywords within the user's query are extracted and used in a Term[10] query, that will exactly match each of these against the ones in the keywords field of every document. We expect these keywords to identify the main topic of both the query and the documents, and thus matching the top 5 keywords within the user's query against the top 20 in the documents will further boost relevant documents.

Finally we have the last of the queries described in Section 4.1, which pertains to temporal references and thus makes use of the Temporal Reference Extractor component. This component works in two steps, first it makes use of pt_tei2go, the Temporal Reference Identifier Model, to extract natural language references to time in the user's query. The resulting natural language temporal references are then ran through duckling, the Temporal Reference Normalizer, with the current date as a base reference for relative time references, which normalizes them into standard date strings[11], enabling us to process them. Each of the resulting temporal references from the Temporal Reference Extractor component are then converted into a time range depending on it's granularity. For example if the natural language that originated the date string only reference a year, then the time range will include the full year. If it instead made reference to a specific month of said year then only that month in that year will be included in the range, and so on. These ranges are then converted into range[12] queries to be matched against the ones in the scrapped dates field of every document. This queries are not included in the "Should" field of the boolean query, but instead in the "Filter" field, excluding immediately any document that doesn't satisfy it, before even ever being scored by the queries in "Should", and never affecting score themselves.

The "Filter" field however works by combining every query within it by means of conjunction, meaning only documents who satisfied every single date range would be accepted, which is opposite to the desired mode function, in which a document should

[9] https://opensearch.org/docs/latest/search-plugins/knn/approximate-knn/
[10] https://opensearch.org/docs/latest/query-dsl/term/term/
[11] https://www.iso.org/iso-8601-date-and-time-format.html
[12] https://opensearch.org/docs/latest/query-dsl/term/range/

only have to match one filter, meaning they should works by disjunction. Thus to solve this problem instead of inserting the time range queries directly into the "Filter" field, all of these are encapsulated into a singular boolean query in it's "Should" field, meaning when one time range matches, the inner boolean query immediately matches, and thus the document passes the filter of the outward boolean query.

5.2.2 Question-Answering Module

Following the Retrieval Module we have the Question-Answering Module which is the biggest one. This module has 3 responsibilities: 1) identifying if the user's query is a question, 2) answer said question, and 3) suggest follow-up and related questions. In order to perform these, this module combines 4 components which can be seen in Fig.5.6, and one of which, the Temporal Reference Extractor, we already discussed.

5.2.2.1 Sentence Classifier Component - Identifying Questions

When it comes to identifying whether a user's query is a question or not this is the work of the Sentence Classifier component. For this component we make use of "MoritzLaurer/ mDeBERTa-v3-base-xnli-multilingual-nli-2mil7"[13], a zero-shot sentence classification model. A zero-shot sentence classification model is a sentence classification model with no specific training on the classification categories, meaning that for any given sentence and given set of potential labels, it will attribute one of the labels in the given set to the sentence.

For our purpose we make use of the model by providing it with the user's query and a set of labels that represent the various types of sentences, with these labels being, "declarative sentence", "imperative sentence", "exclamatory sentence", and finally "interrogative sentence". The use of this model to detect questions, instead of for example a method based in regular expressions, has the benefits of the language models' versatility to understand language, meaning that it understands the sentence intent even if it presents itself in somewhat broken language and with no punctuation. This versatility that is also shared by the question-answering language models, means that even with a somewhat syntactically malformed questions these should be able to answer it, and thus makes this approach viable. This component is used by the "Questions and Search" page at search time to decide weather or not the user's query is a question and weather or not it should trigger the Question-Answering.

5.2.2.2 Question-Answering Component

On the topic of Question-Answering, answering questions is a joint effort, coordinated by the Orchestrator, between the Retrieval Module and the Question-Answering Module, the first gathering relevant documents for context and the latter using these to answer. We

[13] https://huggingface.co/MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7

shall next discuss the specific part of the Question-Answering Module that's involved in this partnership, the Question-Answering Component.

The Question-Answering Component can be regarded as this module's main component, and it's responsibility is answering the user's questions. It can be sub-divided in two very closely related subsystems, the "Single-document QA strategies" which implements the proposed solutions in Section 4.2, and the "Multi-document QA strategies" which in turn implements the proposed solutions in Section 4.3. These subsystems purpose are, respectively, as the name implies, answer the user's questions when there is only one document to consider, or, in the case of the latter, answer the user's questions when there are more than one document.

The acquisition of said documents over which to answer the user's questions is the extent of this modules collaboration with the Retrieval module, which by request of the orchestrator finds and retrieves the relevant documents which are then used by this module. In this prototype the single-document QA is used when a document is already selected (the user is in the "Questions and Search" page document view), and thus known, and as such the Orchestrator simply uses the Retrieval module to get the desired document and pass it onto the "Single-document QA strategies" in the Question-Answering module's QA Component. When it comes to multi-document QA it is used when a specific document isn't known (the user is currently searching in the "Questions and Search"), and thus the Orchestrator requests the Retrieval module to search for the top 10 relevant documents about the question, which are then passed onto the "Multi-document QA strategies" in the Question-Answering module's QA Component.

As previously stated, these subsystems can be described as very closely related. This is for two reasons, the first being that during it's processing the latter makes use of the former, and the second that internally both make use of the very same extractive question answering models. These models in the Question-Answering Component, are combined into an ensemble as per the solutions in Chapter 4 and collaborate on picking the answer to the user's question.

5.2.2.3 Question-Generation Component

Lastly, we have the Question-Generation Component, whose responsibility is to generate contextualized follow-up and related questions, which happens to be the most computationally expensive, and time-consuming, part of this System. Also worth noting is this component's memory requirements, which lead to the need to introduce a second GPU to support it, while every model before this component was running in a single GPU.

The first of two parts which make this component is the Question-Generation Model, for which we choose "valhalla/t5-base-e2e-qg"[14]. This model takes the body of a document as context and generates 1 to 5 questions that it believes one may answer by reading the document. Unfortunately this model only supports the English language, which is a

[14] https://huggingface.co/valhalla/t5-base-e2e-qg

problem since all our documents are in Portuguese. This leads us to the second part of this component, which is used to mitigate this shortcoming, the Translator. The Translator makes use of the "deep-translator"[15] library and as a backup the "facebook/nllb-200-distilled-600M"[16] model, and is used to translate the documents and questions back and forth between English and Portuguese.

A typical use of the Question-Generation Component is then a 4 step process that would proceed as follows: first the document's body gets translated into English, this translated version is then used as input for the Question-Generation model, the resulting generated questions are then translated back into Portuguese, and finally as a quick check these are ran through the Question-Answering Component to ensure we aren't sugesting impossible questions.

When the user is in the "Questions and Search" page document view, meaning the document is already selected, the question generation is an easy process occurring as just previously described, however when the user is in the "Questions and Search" page search view there are further considerations to make. When the user is in the "Questions and Search" page search view it means there isn't currently a selected document from which to generate the questions, but instead we are in the presence of a list of document's relevant to the user's query. In this situation we must instead select which from these to use as context to generate the questions. A simple solution would be to use the top 10 as in the Question-Answering Component, and this is what we first tried, but due to the expensive nature of the Question-Generation Component this would be too time-consuming, and thus return the suggestions in a non-practical time-frame. Alas we settled on the top 3 relevant documents, whose resulting suggestions would be compiled and shown in the search page.

We also implemented suggested questions cache at a document level, meaning we wouldn't have to recompute suggested questions for commonly hit documents, further reducing this component's latency.

[15] https://github.com/nidhaloff/deep-translator
[16] https://huggingface.co/facebook/nllb-200-distilled-600M

6

Evaluation

In this chapter, we will evaluate the approaches proposed in Chapter 4 and present and discuss the results obtained.

These will be evaluated both individually and in combination as part of our ensemble approach, as described in Chapter 4. We will thus assess their performance in both the described Single-Document Question-Answering and Multi-Document Question-Answering tasks, by evaluating them on our new evaluation set, ArquivoPT-QA[1], which we will introduce in Section 6.1.

6.1 Creation of the ArquivoPT-QA Evaluation Set

Given the specialized domain of our document collection, comprised of archived pages of news articles, and the focus of existing Portuguese Extractive Question-Answering datasets on the Single-Document task, we set out to create an evaluation set based on these documents, that enabled us to also evaluate the Multi-Document Question Answering task, which is not present in other datasets. The resulting dataset is composed of two main splits: a) handmade questions, and b) generated questions, each comprised of 150 question-answer pairs.

For the handmade questions the volunteers, a group of 9 computer science students, all performed a search in ArquiWIZ, our prototype described in Chapter 5, for interesting topics and selected any appealing documents in the results. Then, for the selected documents, volunteers crafted extractive questions pertaining to the content within said documents, and selected the text span that corresponded to the answer. Due to how our system works, in which the search query is also used as the question in Multi-Document Question Answering, in these handmade questions further care was put into it's creation so they would also be viable search queries.

The extra care placed in the Handmade questions can easily be seen in Fig. 6.1, where the question from the Handmade split is independent from it's context passage, meaning it contains all the necessary information for one to understand it, while on the other hand

[1] https://github.com/newsArchivePT/ArquivoPT-QA/blob/main/perguntas.json

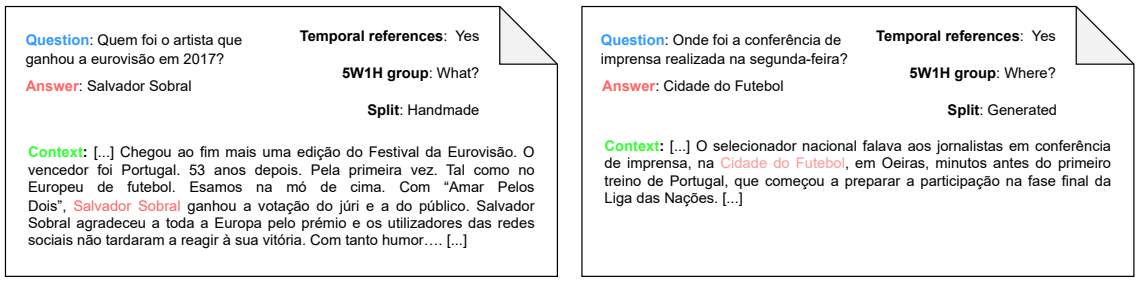


Figure 6.1: Two examples from the ArquivoPT-QA evaluation set, one from each split, with the corresponding context passage, question and answer, as well as categorized accordingly with their 5W1H group and weather or not the question contains a temporal reference.

the question from the Generated split is more closely tied with it's passage, such that one must read the passage for context in order to understand the question. This independence is one of the factors that make a question also a viable search query, as being dependent on your context passage might mean the question doesn't provide enough information to find relevant documents, or it might even only be answerable in that specific context passage.

When it comes to the generated questions, these are the result of the component described in 5.2.2.3. This component, as previously mentioned, would generate questions in realtime based on the documents returned by the search. These generated questions would then be cached to mitigate their expensive computation and make the system faster. Here we make use of these cached questions by having volunteers meticulously curate them. This curation involves sifting through the generated questions, identifying and excluding those that are either broken or unanswerable by simply selecting a span, and reading the source document to pinpoint and select the precise answer text span contained within.

Question on each of these splits are then classified by the volunteers on whether or not they contain temporal references, and categorized into the 5W1H's groups ("What?", "Where?", "When?", "Who?", "Why?", "How?"), with the possibility of being "hard-categorized" into one of these groups if the volunteer is certain, or "soft-categorized" into multiple of these groups if they are unsure. On the table 6.1 we can see an overview of this categorization. As one may see, the categorization is dominated by the first four categories, leaving the last two with barely any questions. We believe that this is due to the fact that the first 4 categories pertain to types of questions we believe really easily lend themselves to extractive question-answering, as they have more direct answers, unlike the last two categories which tend to require more extensive and complex answers, this makes them less suitable for extractive QA.

When it comes to the presence of temporal references, only 27 of the handmade questions have temporal references, while the generated questions where selected such that all 150 of them have temporal references.

	What?		Where?		When?		Who?		Why?		How?	
	Hard	Soft	Hard	Soft	Hard	Soft	Hard	Soft	Hard	Soft	Hard	Soft
Handmade	7	63	7	6	26	1	40	9	2	7	2	1
	70		13		27		49		9		3	
Generated	20	67	14	3	5	0	41	7	0	0	0	0
	87		17		5		48		0		0	
Total	27	130	21	9	31	1	81	16	2	7	2	1
	157		30		32		97		9		3	

Table 6.1: Evaluation set categorization overview over 5W1H categories.

6.2 Methodology and Metrics

Following the literature on Extractive Question-Answering [37, 36], we will assess both the previously presented models and our approach, using the Exact Match and F1 score metrics. These metrics are computed by averaging the results across individual question-answer pairs.

The Exact Match metric is rather simple: in a given question-answer pair, if the predicted answer exactly matches the true answer, it is given a score of 1, otherwise it gets 0. When averaging this metric, the resulting value is the percentage of answers that the approach undergoing evaluation got exactly right.

The F1 score, on the other hand, is a bit more complex. It's formula is $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$, the harmonic mean of the precision and recall, and is used when both precision and recall are important. Here precision and recall are based on the amount of common words between the predicted answer and the true answer. Precision is calculated by dividing the number of common words by the length of the predicted answer, or by other words, precision represents how much of the predicted answer is correct. Recall on the other hand is calculated by dividing the number of commons words by the length of the true answer, or by other words, recall represents how much of the true answer was predicted. The final F1 score for the approach will also be an average of the individual question-answer pair results.

6.3 Single-document QA Results

For the evaluation of our Single-Document Question Answering approach, we will, as previously stated, be evaluating it on our new ArquivoPT-QA evaluation set. But first we will also be presenting early assements and evaluations made during early development of our approach, which were made on the Portuguese translation of SQuAD, due to the inexistance of our evaluation set at the time.

6.3.1 Assesements

During early research and development of our approaches, a way to test them was needed, not only to assess their performance but also to test their viability and make decisions on

which solutions to further develop. Due the inexistence of our evaluation set at the time, the Portuguese translation of SQuAD, SQuaDv1.1-PT, was used for this purpose. Here we present our results and early assessemnts made on this dataset, as the conclusions allowed by these proved to be relevant to establish a research direction.

The Portuguese translation of SQuAD, SQuaDv1.1-PT, composed of about 100,000 QA pairs split between a train set (about 90k QA pairs) and a validation set (about 10k QA pairs), had its validation set selected for an initial evaluation for two mains reasons: being a large general domain extractive question-answering dataset, and being the dataset that was used to train the models that we make use of in the ensemble. The first reason ensures, that due to a large and varied set of question-answer pairs, results are accurate and not due to luck with selecting the pairs when creating the evaluation set, as may happen with a smaller evaluation set like ours. The second reason on the other hand enables us to accurately gauge the improvement in performance originating from our approach, as thus since it was the dataset used to train these models their performance is already known and assured, and as such every improvement must be due to our approach.

Answer Method	**Exact Match**	**F1 Score**	**Secs. per Ans.**
Model 1	68.67	81.40	0.02
Model 2	70.37	82.36	0.01
Model 3	60.29	75.61	0.01
Model 4	67.23	79.72	0.01
Model 5	60.49	74.50	0.01
Highest Scoring Answer (4.2.1)	66.33	79.77	0.05
Answer Voting by RRF (4.2.2) tkm=1	**71.08**	**83.04**	**0.05**
Answer Voting by RRF (4.2.2) tkm=5	69.90	82.61	0.07
Answer Voting by RRF (4.2.2) tkm=10	68.52	82.13	0.09
Model 6	72.46	84.26	0.02
Highest Scoring Answer (4.2.1)	67.31	80.57	0.07
Answer Voting by RRF (4.2.2) tkm=1	**72.54**	**84.15**	**0.07**
Answer Voting by RRF (4.2.2) tkm=5	71.65	83.80	0.09
Answer Voting by RRF (4.2.2) tkm=10	70.39	83.39	0.12

Table 6.2: QA results over SQuaDv1.1-PT - **Single-document** setting.

The Table 6.2 shows, with the previously discussed metrics, the results of evaluation over this dataset, as well as the time spent per question-answer pair (Sec. per Ans.) of each model.

Although in the ensemble we make use of all six models, here we decided to start only with the first five and later introduced the sixth one. The reason for this decision is that while the first 5 models are all similar in performance, and present themselves with a number of parameters equivalent to a BERT-base, the last one has a higher performance, and a number of parameters of a BERT-large. This is relevant because as discussed in

4.2.1.2, an ensemble should only be used with models of similar performance as a means of achieving better performance by cooperation, and that when in the presence of a higher performant model, then it should be used alone instead of in the ensemble. Since every model in the ensemble has the same weight, the worse models would only be hampering the better one's performance.

As seen in Table 6.2, above the results for Model 6, results of either of our strategies (Highest Scoring Answer 4.2.1 and Answer Voting by RRF 4.2.2) have shown to be promising, both surpassing individual models, with Answer Voting by RRF being the top-performing approach overall. This thus shows that our proposed approach has a positive impact in Question-Answering performance. (relevant for RQ 3)

Next, we introduced the sixth model with the intent of asserting whether the use of better performing models in the ensemble would further improve it's performance. The introduction of this model, as expected, proved an increase in the ensembles' performance, with every ensemble approach improving as can be seen in Table 6.2 bellow the results for Model 6. As a test of whether replacing/introducing better models into the ensemble would increase it's performance, it had succeeded in supporting this hypothesis. (relevant for RQ 3) Furthermore, due to the nature of this work, and it proposing ensemble based approaches, we decided to continue including the sixth model in further testing.

In regards to the tkm parameter, or top_k_model parameter, it will be discussed with the ArquivoPT-QA Evaluation Set.

6.3.2 Evaluation on ArquivoPT-QA Evaluation Set

Answer Method	Exact Match	F1 Score	Secs. per Ans.
Model 1	68.33	77.33	0.05
Model 2	69.67	79.88	0.02
Model 3	63.67	74.70	0.03
Model 4	65.00	74.51	0.03
Model 5	60.00	71.75	0.02
Model 6	73.33	82.59	0.06
Highest Scoring Answer (4.2.1)	66.33	76.62	0.20
Answer Voting by RRF (4.2.2) tkm=1	74.00	83.30	0.22
Answer Voting by RRF (4.2.2) tkm=5	75.00	84.37	0.28
Answer Voting by RRF (4.2.2) tkm=10	**75.33**	**84.81**	**0.34**
Answer Voting by RRF (4.2.2) tkm=15	74.67	84.65	0.40
Answer Voting by RRF (4.2.2) tkm=30	73.67	84.65	0.57

Table 6.3: QA results over ArquivoPT-QA - **Single-document** setting.

Next we evaluated our proposed solution on our ArquivoPT-QA evaluation set. Thus in this evaluation each approach will be provided with question-context pairs from our set. In these pairs the context is the known correct document that contains the answer.

On this evaluation set we reached much of the very same conclusions we did in the previous section, when evaluating on the Portuguese translation of SQuAD, with exception to the behavior of the tkm parameter. The tkm parameter, or top_k_model parameter, defines how many of their best possible answers each model returns. In our proposed solution, specifically in the Answer Voting by RRF, these "best possible answers" will be the available options for voting, meaning this parameter sets how many answer alternatives each model contributes to the voting option pool.

In our evaluation with the ArquivoPT-QA evaluation set, as seen in Table 6.3, we observed performance increases of our top performing approach, Answer Voting by RRF, when increasing the tkm parameter. This results seemed rather intuitive to us, as, although, providing the voting ensemble with more options to choose from won't affect how it performs it's choice, it would however mean less probability of the correct answer being left out of the voting options, thus increasing the ensemble performance simply because the answer, which no single model thought was relevant enough, was available as an option, and through voting ended up being selected as the best answer. However, reality isn't as simple as increasing tkm for better performance, as for higher values it started to decline. This behavior we attribute to saturation and degradation of the voting option pool when options start being too many and of declining quality. As such, due to it being the best result we have achieved, for future testing we set $tkm = 10$.

When analyzing the results of the SQuADv1.1-PT evaluation, we see that in this dataset this behavior doesn't appear to be present, as performance immediately drops as soon as tkm increases. We believe this is due to this being the dataset upon which the models in the ensemble were trained upon, and thus they are strongly more tuned to this dataset. This means their top answer is likely to be of quality. Thus, including any other option in the voting option pool will degradate it's quality and hinder the vote, thus lowering the ensemble performance.

6.3.2.1 Time-aware Evaluation Function

Lastly, we evaluated the Single-Document Question-Answering approach, in the ArquivoPT-QA evalaution set, while replacing the evaluation function with the one from Subsection 4.2.3 that has a Temporal Modifier. Results from this evaluation, which can be seen in Table 6.4, unfortunately proved this replacement evaluation function to be of no benefit by mantaining the performance of the approach with the original evaluation function.

6.4 Multi-document QA Results

When it comes to Multi-Document Question-Answering, it's performance is highly dependent on the context documents available. As such, for the evaluation of our Multi-Document Question-Answering approach, evaluation must be done in three parts: the

Evaluation Function Method	Exact Match	F1 Score
Original	75.334	84.806
Character Distance Based	75.334	84.806
Sentence Distance Based	75.334	84.806
Context Distance Based	75.334	84.806

Table 6.4: QA results over ArquivoPT-QA with Timeaware Evaluation Functions - **Single-document** setting.

evaluation of the Multi-Document QA approach on itself, the evaluation of the Semantic Search approach it is dependent upon for context aquisition, and the combination of both.

All these evaluations will be conducted on our ArquivoPT-QA evaluation set.

6.4.1 Evaluating the Multi-document QA approaches

Answer Method	EM	F1	Secs. per Ans.
Model 1	0.00	0.00	0.52
Model 2	2.34	2.34	0.26
Model 3	12.67	12.91	0.29
Model 4	2.67	2.67	0.29
Model 5	0.00	0.22	0.26
Model 6	57.67	64.00	0.59
SD: Score Sort (4.2.1) & MD: Score Sort (4.3.1)	12.00	12.25	2.15
SD: RRF Voting (4.2.2) & MD: Score Sort (4.3.1)	15.34	16.07	3.97
SD: Score Sort (4.2.1) & MD: RRF Voting (4.3.2)	66.34	76.66	2.15
SD: RRF Voting (4.2.2) & MD: RRF Voting (4.3.2)	**75.34**	**84.94**	**3.97**

Table 6.5: QA results over ArquivoPT-QA - **Multi-document** setting - Correct+Random context docs.

Let us first start by evaluating our Multi-Document Question-Answering approaches. The performance of these is determined by two factors, the first being the performance of the Single-Document QA approaches which the Multi-document QA makes use of internally to select the answers, and the second being the performance of the approach used to select the documents. Given that we already know the performance of the Single-document QA, this evaluation's purpose is instead to assert whether or not our approach performs when selecting the right document.

For the purpose of evaluating the context documents selection strategy, each of the Questions in our evaluation set is answered by our approaches while having as context documents the correct document and 10 random documents unrelated to the question. Given the composition of the context, the probability of there being another good context besides the correct one is very low, and as such if our selection strategy performs well it should select said correct document, and thus the reported performance should be similar

to the one of it's Single-Document QA counterpart. As seen in Table 6.4.1, this is indeed the case when making use of the "RFF Voting" (4.3.2) multi-doc approach. However both the "Score Sort" (4.3.1) and singular model multi-doc approaches greatly under-perform. We thus conclude the combination of Answer voting by RRF with Document voting by RRF to be our best performant approach. (relevant for RQ 3)

6.4.2 Evaluating the Semantic Search approaches

Embeddings	Keyword	Temporal Ref.	Top 10 %	Top 100 %	Absent %
None	None	None	0.627	0.827	0.173
None	Yake!	None	0.620	0.840	0.160
None	None	Duckling	0.613	0.807	0.193
None	None	Duckling+Tei2Go	0.640	0.827	0.173
None	Yake!	Duckling	0.607	0.827	0.173
None	Yake!	Duckling+Tei2Go	0.633	0.840	0.160
Clip	None	None	0.620	0.847	0.153
Clip	Yake!	None	0.613	0.853	0.147
Clip	None	Duckling	0.607	0.827	0.173
Clip	None	Duckling+Tei2Go	0.640	0.840	0.160
Clip	Yake!	Duckling	0.600	0.833	0.167
Clip	**Yake!**	**Duckling+Tei2Go**	**0.633**	**0.853**	**0.147**

Table 6.6: Semantic Search Queries - Original context doc retrieval rank position test results.

Next we evaluate our Semantic Search component. In this section however, we won't be making use of the full evaluation set, but only of the split containing the handmade questions. This decision stems from the fact that, as previously mentioned in Section 6.1, in the handmade questions, further care was put into their creation so they would also be viable search queries, which is important since we are testing the Semantic Search.

For Semantic Search we intend to know whether our proposed solution provides the best possible context to the Multi-Document QA. As such, we experiment multiple combinations of our components. First, we tested every combination's retrieval capability when searching for the correct document that accompanies the question in the evaluation set. The test objective was to understand which method would perform better in placing these correct context documents in the top 10 of it's ranking, as only the top 10 documents are considered in Multi-Document QA. Thus, the metric is the percentage of these documents in the Top 10.

As seen in Table 6.6, our proposed solution, "Clip+Yake!+Duckling+Tei2go", didn't achieve the top performance by this metric, however not only it scored fairly high it also achieved the lowest percentage of missing correct documents in it's ranking, as denoted by "Absent %", resulting in the best combination between the presented metrics. Given this combination, we believe that our proposed solution is indeed retrieving documents

relevant to the question, and even though the right document might sometimes not be in the Top 10, some other document with the answer might be. (relevant for RQ 1 and RQ 2)

6.4.3 Evaluating the Multi-document QA + Semantic Search integration

At last we evaluate the integration of both of the previous parts. Following the previous section, we use the same split of the evaluation set, the handmade split. This is thus, since as previously stated, further care was put into these questions so they would also be viable search queries, which is important since the questions will be used by search queries by the Semantic Search which is part of the combined system. Furthermore, due to the nature of how these questions were created and collected, this being by interacting with our prototype, as described in 6.1, they also more closely resemble how a user would interact with this combo system. This is relevant because the type of questions a user makes with no context are very different from the ones where the user is inquiring about a specific document. The first type of questions may be answered by various documents and provide more context for finding a suitable one, while the second type, where most of the generated questions fit, is more direct and document specific.

Following the hypothesis presented at the end of the previous section, where we stated that although the correct document wasn't in the top 10 documents retrieved by the semantic search, some other relevant document must be, we shall start by using the full system to test this belief. To do so, we put to test every permutation of search settings by having them retrieve their top 10 documents, which were then provided to our best performant Multi-Document Question-Answering approach as established in 6.4.1. From the result presented in the Table 6.7, we can conclude that our intuition was correct, and, from all the available queries, ours, "Clip+Yake!+Duckling+Tei2go", is providing the Multi-Document QA approach with the best documents. (relevant for RQ 1 and RQ 2)

Embeddings	Keyword	Temporal Ref.	Exact Match	F1 Score
None	None	None	45.334	55.807
None	Yake!	None	46.000	56.092
None	None	Duckling	45.334	55.756
None	None	Duckling+Tei2Go	46.000	56.807
None	Yake!	Duckling	45.334	55.374
None	Yake!	Duckling+Tei2Go	46.000	56.425
Clip	None	None	46.667	55.732
Clip	Yake!	None	46.667	56.168
Clip	None	Duckling	46.667	55.681
Clip	None	Duckling+Tei2Go	47.334	56.732
Clip	Yake!	Duckling	46.667	56.117
Clip	**Yake!**	**Duckling+Tei2Go**	**47.334**	**57.168**

Table 6.7: Multi-Document QA with different Semantic Search Queries.

Let us finally then, making use of the semantic search approach we just concluded to

be best, use it in combination with every possible Multi-Document QA approach, and evaluate their performance. From the results presented in Table 6.8, we can immediately confirm that the approach using RRF in both steps as being the best approach, as previously stated in section 6.4.1, but we can also reach two conclusions. First, when it comes to Multi-Document Question-Answering, the usage of a single QA model greatly under-performs when compared to it's capabilities in the single-document task, and second, our proposed approach of answering questions by combining all models in ensembles, to choose both the answer and the document, largely improves the performance. (relevant for RQ 3)

Answer Method	**EM**	**F1**	**Secs. per Ans.**
Model 1	20.00	23.31	0.55
Model 2	29.33	34.92	0.27
Model 3	30.00	38.76	0.30
Model 4	27.33	34.36	0.30
Model 5	18.00	22.47	0.27
Model 6	36.00	45.77	0.62
SD: Score Sort (4.2.1) & MD: Score Sort (4.3.1)	29.33	38.76	2.30
SD: RRF Voting (4.2.2) & MD: Score Sort (4.3.1)	34.67	38.88	3.85
SD: Score Sort (4.2.1) & MD: RRF Voting (4.3.2)	44.00	53.45	2.31
SD: RRF Voting (4.2.2) & MD: RRF Voting (4.3.2)	**47.33**	**57.17**	**3.89**

Table 6.8: QA results over ArquivoPT-QA Handmade Split - **Multi-document** setting - integrated with Semantic Search.

Following the Table 6.8 evaluation over the Handmade split, we also present Table 6.9 were we evaluate over the full evaluation set. From these results we can see a decrease in performance in the Full split when in comparison with the evaluation over the Handmade split, we believe this to be due to the presence of generated questions on the Full split. We believe generated questions to impact performance like this due to, unlike the handmade questions, not taking into consideration the need to be both viable questions and viable search queries, thus impacting the performance of the system by being poor search queries to find relevant documents containing an answer. (relevant for RQ 4)

6.5 Question-Answering Results by Question Type (5W1H Categories)

Further analysis of QA performance may be made by breaking down the results by the question categorization. Table 6.10 and 6.11 shows the results in both settings per question type. One may see that the question categories in the dataset are dominated in number of questions by the first four categories, leaving the last two with barely any questions. As previously stated, we think such happened when assembling the dataset because the first 4 categories pertain to types of questions we believe really easily lend themselves to extractive question-answering, as they have more direct answers, unlike the last two

Answer Method	**EM**	**F1**	**Secs. per Ans.**
Model 1	15.00	17.046	0.55
Model 2	23.34	27.82	0.27
Model 3	25.00	31.89	0.31
Model 4	25.34	31.15	0.30
Model 5	17.34	21.11	0.27
Model 6	36.00	43.95	0.62
SD: Score Sort (4.2.1) & MD: Score Sort (4.3.1)	24.34	31.56	2.26
SD: RRF Voting (4.2.2) & MD: Score Sort (4.3.1)	31.34	35.31	3.84
SD: Score Sort (4.2.1) & MD: RRF Voting (4.3.2)	43.00	51.53	2.26
SD: RRF Voting (4.2.2) & MD: RRF Voting (4.3.2)	47.00	55.02	3.85

Table 6.9: QA results over ArquivoPT-QA Full Split - **Multi-document** setting - integrated with Semantic Search.

categories which tend to require more extensive and complex answers. This intuition is further supported by the performance being higher in these categories, however the last two categories don't have nearly enough questions for us to be able to draw any definitive conclusions.

	Who?	**What?**	**When?**	**Where?**	**Why?**	**How?**
Exact Match	84.54	70.25	87.50	79.41	55.56	0.0
F1 Score	90.47	81.93	89.06	91.25	55.56	28.57
No. of Questions	97	158	32	34	9	3

Table 6.10: QA results over ArquivoPT-QA **by Question type** using Answer Voting by RRF (4.2.2) - **Single-document** setting.

	Who?	**What?**	**When?**	**Where?**	**Why?**	**How?**
Exact Match	46.94	45.71	59.26	41.18	33.33	0.00
F1 Score	55.93	57.68	60.91	50.94	41.67	0.00
No. of Questions	49	70	27	17	9	3

Table 6.11: QA results over ArquivoPT-QA Handmade Split **by Question type** using Answer Voting by RRF (4.2.2) & Document Voting by RRF (4.3.2) - **Multi-document** setting.

7

Conclusions

In this thesis, we introduced Question-Answering to a subset of the Portuguese Web Archive comprised of Portuguese archived news.We did so by proposing Time-aware Extractive Question-Answering strategies for the Portuguese Web Archive, capable of performing in two Question-Answering modalities, Single-Document and Multi-Document. Due to both a lack of existing models focused on this particular setting, as well as a lack of datasets for fine-tuning exiting models for it, our proposed approach instead leverages multiple sub-optimal transformer-based extractive QA modeles trained on the Portuguese translation of SQuAD.

We proposed assorting these models into a voting ensemble which is used to perform different choices in each of the QA modalities. In Single-Document QA each of the models in the ensemble would suggest possible answers and then the ensemble would vote on the best. On Multi-Document QA the ensemble would vote to choose which of the documents is most likely to have the answer, and then use the Single-Document approach to choose an answer. When it comes to the choice of documents for the Multi-Document QA we proposed a semantic search component, which has been enhanced with time-awareness to attend to the inherent temporal characteristics of document from a web archive and it's relevancy to any temporal reference in a user's query when retrieving relevant documents.

To evaluate the performance of our proposed approach we introduced ArquivoPT-QA, an extractive Single and Multi document QA evaluation set, focused on Portuguese archived news articles originating from the Portuguese Web Archive, which contains questions categorized both by the 5W1H framework (Who, What, When, Where, Why, How), as well into temporal and non-temporal classifications. The results of our experiments conducted on this evaluation set, demonstrated the effectiveness of our approach in both modalities. It proved our approach capable of, in both modalities, improving upon the performance of the underlying models, with largely more noticeable improvements in the Multi-Doc modality as the underlying models largely under-perform here.

With the purpose of demonstrating in practice how our proposed QA solution could be applied to the Portuguese Web Archive, we created the ArquiWIZ prototype, which was submitted for consideration in the "Prémio Arquivo.pt 2023" competition where it

achieved a place among the top 8 contenders.

Finally, we also wrote a paper to be submitted to ECIR 2024, detailing all of the previously mentioned contributions. We believe our work will help web archives better accomplish their goal of keeping information accessible forever by positively contributing to the efforts in increasing the ease of access to information in web archives.

7.1 Future Work

Despite the positive results in integrating QA with the Portuguese Web Archive, as well as the successful creation of the ArquivoPT-QA evaluation set, we believe there to be still room for improvement of our system as well as expansion of our evaluation set.

When it comes to QA system, one clear topic which requires further work is the introduction of time-awareness at the level of a document's content, as currently temporal considerations are only taken at the search level. Further work is thus required as the attempt at integrating temporal considerations with the QA system by using a "distance to temporal reference" score modifier failed. We also would like to explore integration of multimodal models, both at the level of semantic search as well as Question-Answering, thus allowing users not only to explore the news articles content through questions but as well the images which illustrate these.

As for the evaluation set, we would like to further expand it with both more "handmade" questions as well as "generated" questions which originate from organic use of our prototype. Such expansion would not only allow for it to more accurately represent a system's performance when used in testing, but would also allow us to take further conclusions on the categorization of the dataset's questions.

Bibliography

[1] 14:00-17:00. *ISO 28500:2009*. en. URL: https://www.iso.org/standard/44717.html (visited on 2023-01-05) (cit. on p. 25).

[2] A. Agrawal et al. *VQA: Visual Question Answering*. arXiv:1505.00468 [cs]. 2016-10. DOI: 10.48550/arXiv.1505.00468. URL: http://arxiv.org/abs/1505.00468 (visited on 2023-02-09) (cit. on p. 23).

[3] S. Albawi, T. A. Mohammed, and S. Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. 2017-08, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186 (cit. on p. 12).

[4] K. Berberich et al. "A Time Machine for Text Search". In: *Clarke, Charlie; Fuhr, Norbert; Kando, Noriko; Kraaij, Wessel; de Vries, Arjen P.: SIGIR'07 : 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 519-526 (2007)* (2007-01). DOI: 10.1145/1277741.1277831 (cit. on p. 25).

[5] M. Burner and B. Kahle. *Internet Archive: ARC File Format Reference*. URL: https://archive.org/web/researcher/ArcFileFormat.php (visited on 2023-01-03) (cit. on p. 25).

[6] R. Campos et al. "A Text Feature Based Automatic Keyword Extraction Method for Single Documents". en. In: *Advances in Information Retrieval*. Ed. by G. Pasi et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 684–691. ISBN: 978-3-319-76941-7. DOI: 10.1007/978-3-319-76941-7_63 (cit. on pp. 36, 38, 56).

[7] D. Castanho. "Structuring and Organizing Large Scale Graph Temporal information". MSc diss. NOVA School of Science and Technology, 2023-10 (cit. on pp. 48, 49).

[8] R. Chakravarti et al. "CFO: A Framework for Building Production NLP Systems". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*. arXiv:1908.06121 [cs]. 2019, pp. 31–36. DOI: 10

.18653/v1/D19-3006. URL: http://arxiv.org/abs/1908.06121 (visited on 2023-02-09) (cit. on p. 24).

[9] Y.-C. Chen et al. *UNITER: UNiversal Image-TExt Representation Learning*. arXiv:1909.11740 [cs]. 2020-07. DOI: 10.48550/arXiv.1909.11740. URL: http://arxiv.org/abs/1909.11740 (visited on 2023-01-25) (cit. on pp. 15, 16).

[10] G. V. Cormack, C. L. A. Clarke, and S. Buettcher. "Reciprocal rank fusion outperforms condorcet and individual rank learning methods". en. In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. Boston MA USA: ACM, 2009-07, pp. 758–759. ISBN: 978-1-60558-483-6. DOI: 10.1145/1571941.1572114. URL: https://dl.acm.org/doi/10.1145/1571941.1572114 (visited on 2023-09-09) (cit. on p. 41).

[11] M. Costa, D. Gomes, and M. J. Silva. "The evolution of web archiving". en. In: *International Journal on Digital Libraries* 18.3 (2017-09), pp. 191–205. ISSN: 1432-5012, 1432-1300. DOI: 10.1007/s00799-016-0171-9. URL: http://link.springer.com/10.1007/s00799-016-0171-9 (visited on 2023-01-05) (cit. on p. 25).

[12] M. Costa and M. J. Silva. "Characterizing Search Behavior in Web Archives". en. In: (2011) (cit. on p. 25).

[13] M. Costa et al. "A survey of web archive search architectures". en. In: *Proceedings of the 22nd International Conference on World Wide Web*. Rio de Janeiro Brazil: ACM, 2013-05, pp. 1045–1050. ISBN: 978-1-4503-2038-2. DOI: 10.1145/2487788.2488116. URL: https://dl.acm.org/doi/10.1145/2487788.2488116 (visited on 2023-01-01) (cit. on pp. 26–28).

[14] *Cross-Encoders — Sentence-Transformers documentation*. URL: https://www.sbert.net/examples/applications/cross-encoder/README.html (visited on 2023-01-23) (cit. on p. 10).

[15] J. Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805 [cs]. 2019-05. URL: http://arxiv.org/abs/1810.04805 (visited on 2023-01-09) (cit. on pp. 7–10, 39).

[16] A. Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv:2010.11929 [cs]. 2021-06. URL: http://arxiv.org/abs/2010.11929 (visited on 2023-01-24) (cit. on pp. 11–13).

[17] R. Girshick. *Fast R-CNN*. arXiv:1504.08083 [cs]. 2015-09. DOI: 10.48550/arXiv.1504.08083. URL: http://arxiv.org/abs/1504.08083 (visited on 2023-01-25) (cit. on p. 13).

[18] R. Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. arXiv:1311.2524 [cs]. 2014-10. DOI: 10.48550/arXiv.1311.2524. URL: http://arxiv.org/abs/1311.2524 (visited on 2023-01-25) (cit. on p. 13).

[19] D. Gomes, J. Miranda, and M. Costa. "A Survey on Web Archiving Initiatives". en. In: *Research and Advanced Technology for Digital Libraries*. Ed. by S. Gradmann et al. Vol. 6966. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 408–420. ISBN: 978-3-642-24468-1 978-3-642-24469-8. DOI: 10.1007/978-3-642-24469-8_41. URL: http://link.springer.com/10.1007/978-3-642-24469-8_41 (visited on 2023-01-05) (cit. on pp. 26, 27).

[20] K. He et al. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385 [cs]. 2015-12. URL: http://arxiv.org/abs/1512.03385 (visited on 2023-01-24) (cit. on pp. 11, 12).

[21] P. He, J. Gao, and W. Chen. *DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing*. 2021. arXiv: 2111.09543 [cs.CL] (cit. on p. 40).

[22] IIPC. *The CDX File Format*. URL: https://iipc.github.io/warc-specifications/specifications/cdx-format/cdx-2006/ (visited on 2023-01-06) (cit. on p. 26).

[23] IIPC. *The WARC Format*. URL: https://iipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/ (visited on 2023-01-03) (cit. on p. 25).

[24] *Internet Archive: About IA*. URL: https://archive.org/about/ (visited on 2023-01-01) (cit. on p. 26).

[25] *Internet Archive: CDX File Format Reference*. URL: https://archive.org/web/researcher/cdx_file_format.php (visited on 2023-01-05) (cit. on p. 26).

[26] *Internet Archive: CDX Legend Reference*. URL: https://archive.org/web/researcher/cdx_legend.php (visited on 2023-01-06) (cit. on p. 26).

[27] L. H. Li et al. *VisualBERT: A Simple and Performant Baseline for Vision and Language*. arXiv:1908.03557 [cs]. 2019-08. DOI: 10.48550/arXiv.1908.03557. URL: http://arxiv.org/abs/1908.03557 (visited on 2023-01-25) (cit. on pp. 14, 15).

[28] R. Lopes. "Rich Large-Scale Portuguese Language Models from Large Portuguese Corpora". MSc diss. NOVA School of Science and Technology, 2023-10 (cit. on p. 48).

[29] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[30] T. Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. arXiv:1301.3781 [cs]. 2013-09. DOI: 10.48550/arXiv.1301.3781. URL: http://arxiv.org/abs/1301.3781 (visited on 2023-02-10) (cit. on p. 7).

[31] A. Ntoulas, J. Cho, and C. Olston. "What's new on the web?: the evolution of the web from a search engine perspective". en. In: *Proceedings of the 13th conference on World Wide Web - WWW '04*. New York, NY, USA: ACM Press, 2004, p. 1. ISBN: 978-1-58113-844-3. DOI: 10.1145/988672.988674. URL: http://portal.acm.org/citation.cfm?doid=988672.988674 (visited on 2023-01-05) (cit. on p. 1).

[32] *ORS/CDXJ Drafts*. original-date: 2015-10-06T23:30:16Z. 2022-08. URL: https://github.com/oduwsdl/ORS (visited on 2023-01-05) (cit. on p. 26).

[33] J. Pennington, R. Socher, and C. Manning. "Glove: Global Vectors for Word Representation". en. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: http://aclweb.org/anthology/D14-1162 (visited on 2023-02-10) (cit. on p. 7).

[34] M. Peters et al. "Deep Contextualized Word Representations". en. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: http://aclweb.org/anthology/N18-1202 (visited on 2023-01-22) (cit. on pp. 7, 9).

[35] A. Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. arXiv:2103.00020 [cs]. 2021-02. DOI: 10.48550/arXiv.2103.00020. URL: http://arxiv.org/abs/2103.00020 (visited on 2023-01-25) (cit. on pp. 16, 17, 38).

[36] P. Rajpurkar, R. Jia, and P. Liang. *Know What You Don't Know: Unanswerable Questions for SQuAD*. arXiv:1806.03822 [cs]. 2018-06. URL: http://arxiv.org/abs/1806.03822 (visited on 2023-09-25) (cit. on p. 63).

[37] P. Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. arXiv:1606.05250 [cs]. 2016-10. DOI: 10.48550/arXiv.1606.05250. URL: http://arxiv.org/abs/1606.05250 (visited on 2023-02-08) (cit. on pp. 22, 63).

[38] N. Reimers and I. Gurevych. *Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation*. arXiv:2004.09813 [cs]. 2020-10. DOI: 10.48550/arXiv.2004.09813. URL: http://arxiv.org/abs/2004.09813 (visited on 2023-02-03) (cit. on p. 18).

[39] N. Reimers and I. Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. arXiv:1908.10084 [cs]. 2019-08. URL: http://arxiv.org/abs/1908.10084 (visited on 2023-01-01) (cit. on pp. 10, 11, 18).

[40] S. Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv:1506.01497 [cs]. 2016-01. URL: http://arxiv.org/abs/1506.01497 (visited on 2023-01-24) (cit. on p. 13).

[41] S. Robertson and H. Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389. ISSN: 1554-0669. DOI: 10.1561/1500000019. URL: https://doi.org/10.1561/1500000019 (visited on 2023-02-08) (cit. on p. 20).

[42] A. Singh et al. *FLAVA: A Foundational Language And Vision Alignment Model*. arXiv:2112.04482 [cs]. 2022-03. DOI: 10.48550/arXiv.2112.04482. URL: http://arxiv.org/abs/2112.04482 (visited on 2023-01-25) (cit. on p. 18).

[43] H. Sousa, R. Campos, and A. Jorge. "TEI2GO: A Multilingual Approach for Fast Temporal Expression Identification". In: (cit. on pp. 36, 38, 56).

[44] F. Souza, R. Nogueira, and R. Lotufo. "BERTimbau: Pretrained BERT Models for Brazilian Portuguese". In: *Intelligent Systems*. Ed. by R. Cerri and R. C. Prati. Cham: Springer International Publishing, 2020, pp. 403–417. ISBN: 978-3-030-61377-8 (cit. on p. 40).

[45] A. Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. 2017-12. DOI: 10.48550/arXiv.1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 2023-01-11) (cit. on pp. 7, 9, 12).

[46] J. Wang et al. "Answering Event-Related Questions over Long-Term News Article Archives". In: 2020-04, pp. 774–789. ISBN: 978-3-030-45438-8. DOI: 10.1007/978-3-030-45439-5_51 (cit. on p. 23).

[47] J. Zobel and A. Moffat. "Inverted files for text search engines". en. In: *ACM Computing Surveys* 38.2 (2006-07), p. 6. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/1132956.1132959. URL: https://dl.acm.org/doi/10.1145/1132956.1132959 (visited on 2023-01-10) (cit. on p. 27).