uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática

2021-2022

Trabajo Fin de Grado

Open Domain Question Answering System

Andrés Langoyo Martín

Tutor: Raquel Fuentetaja Pizán

Leganés, 2022

ABSTRACT

This project develops an Open Domain Question Answering system called Elise. It is a comparative study between different Machine Learning models that can order and read documents. The training and evaluation processes are described. A two-stage retriever-reader approach is followed. In the retriever, whose goal is to rank potential documents, methods like TF-IDF, BM25, word2vec embeddings, and BERT embeddings are compared using metrics Mean Average Precision and Mean Reciprocal Rank. The reader extracts an answer span from a document given a question. Here is an LSTM based architecture with attention and a fine-tuned BERT model for Question Answering. These were compared using EM and F1-score. Then, a comparative analysis was performed using the TriviaQA dataset which showed that a combination of BM25 ranker with a fine-tuned BERT achieved the best results.

The system, which supports English and Spanish languages, classifies input queries using a multilayer perceptron. Also, it identifies keywords using grammatical information by performing a POS tagging task. This is done with a word and character level LSTM.

The selected system is deployed for user interaction through a web application developed in React.js. The web interface interacts through HTTP requests with a REST API that hosts the developed models to respond to user queries.

A user evaluation was performed with real users who helped identify errors and suggested improvements for the application.

Keywords: NLP, Question Answering, Open Domain Question Answering, Reader, Retriever, POS Tagger, BERT, TF-IDF, BM25, word2vec, LSTM, API REST, Docker, Deployment, React.js, AWS, Firebase, AWS Lambda.

ACKNOWLEDGEMENTS

I want to express my gratitude to all the people that have walked by my side during this program and the development of this work. Classmates, friends, professors and family members were all relevant during these last four years. They made the highs more enjoyable and the lows more bearable. These remarkable people with whom I got to share my way acted as mentors in some way and I managed to always learn something from everyone. For no one is truly self-made, this work is also yours in some manner.

TABLE OF CONTENTS

FIGURES

TABLES

1. INTRODUCTION

This section aims to describe the organization followed in the document as well as the motivation behind the project and the objectives established for its completion.

1.1 Organization

The document is structured following the following chapters:

1 **Introduction**: In this chapter the motivations and the objectives of this project are presented as well as the outline of the contents of the following sections.

2 **State of the Art:** In the chapter the necessary material to understand and develop the machine learning models to obtain the answer from the query is described. Also, the concepts and technology of the model deployment and the web interface are described.

3 **System Analysis**: This section outlines how the implemented system must behave by specifying requirements, user personas, use cases and classes and method contracts.

4 **Design and Training Process:** This chapter describes the Machine Learning models developed with the data processing process, training and validation.

5 **Implementation and Deployment:** This chapter presents the implementation selected based on the technologies available, the requirements of the project and costs associated.

6 **Testing and Validation:** The testing process of the inference pipeline and the user validation is described in this section.

7 **Project Organization:** This chapter describes the time plan and the costs associated with the development of the project. The socioeconomic environment and the regulatory framework related to the project are also described.

8 **Conclusions:** This section summarizes the results of the work developed as well as some comments for future directions and improvements.

1.2 Motivation

Question Answering (QA) is a long-standing problem in the Natural Language Processing (NLP) field and a very active research direction. A Question Answering system gives appropriate answers to questions or statements expressed in natural languages such as English, Spanish or Chinese.

The interest of Question Answering research can be traced back as soon as the ‘60s. Here the first approaches were developed based on questions classification, propositional logic. A parsing and grammatical analysis was made in order to place the information into data

structures to be processed with transformations and logical inference. This is the case of the ACM, ALA and LOGOS systems. In the early days systems were thought of as interfaces to databases [1].

Eventually, more complex models appeared, like DeepQA [2]. It was developed by IBM as a challenge to participate with humans at Jeopardy. This was an example of a successful model as it was able to surpass Jeopardy's champions in a live contest.

In the beginning, research was based more on machine comprehension, to be able to give a text and let a program extract the answer to a question, knowing that the answer is in that text. During the last decade, an already good performance was obtained using Machine Learning architectures in this task. The focus shifted towards Open Domain Question Answering. Here, the systems try to complement the reading capabilities of ML models with the ability to search documents and identify which of them are more likely to contain the answers. The aim of current research is to build models like DrQA, proposed in 2017, where a two-stage retriever-reader approach is used to solve this task. It used a ranker that searches and orders documents and a reader which extracts answers from documents. [3].

But more complex approaches have been attempted like end-to-end training or Retriever-free approaches, which rely heavily on using big Machine Learning models trained on huge datasets [4].

Question answering is interesting because it has lots of immediate applications like in search engines or dialogue systems. It is currently a test task to see how well computer systems are able to understand human language.

Question Answering can be embedded into conversational agents and can be useful in a wide variety of applications valuable in the enterprise world. Customer service is a clear example. Systems can answer questions about products and services which might be helpful for the company without the need to allocate human resources. For instance, resolving internal desk questions as an internal tool for the company. It is also possible to guide customers through a website, help the customer complete a payment and solve technical support issues [5].

As seen, Question Answering has applications in the real world and it is an active research problem in the NLP community. A comparative study between different architectures and methods is appealing to start learning about the NLP field and its practices. It is also a motivation to implement a generic tool, without narrowing down to any specific domain, that is flexible enough to be adapted in the future to different problems.

1.3 Objectives

This work aims to study, train and implement a system that is able to answer questions that users want to formulate from an unrestricted domain. The objective of this work is divided into two parts.

First and foremost, it is necessary to create a system based on NLP tools and Machine Learning architectures, which can be powerful enough to fulfill this task. For that, several sub-objectives can be identified:

1. The goal is to make it available in two languages: English and Spanish. For that, the system shall be able to distinguish between Spanish and English sentences.
2. In terms of content it must be open domain. This means that the sources from where the answers are extracted must not be restricted to any topic. Also, the training process of the models must be done with diverse documents as well. The system must be modular enough to change the source of documents to be adaptable to more concrete applications.
3. In order to search, it shall be able to make an analysis of the input sentences in a way that is able to extract useful information and rule out unnecessary words.
4. The system will be based on a reader-retriever architecture, which makes the problem approachable. The retriever must be able to order documents based on their relevance in relation to a query. The reader must be able to select a fragment from a document.
5. It is of interest to make a comparative analysis of each of the modules. It is proposed to train and implement at least two different architectures for each of them. An analysis must be made at different levels to see which one performs better. The two readers implemented must be compared against the same dataset. And the two rankers must be compared in a similar way. Then, a global comparison using different combinations of rankers and readers must be made which could be done with a challenge dataset or with real users if there was no other option.

Secondly, it is also interesting to be able to make the system developed accessible to users. A research of different deployment techniques can be done with the aim of building a web application that wraps the models developed. The idea is to give access to users through a chat-like interface which is lightway and adaptable to multiple devices. As optional objectives, text to speech and speech recognition could be included as well. Moreover, it will include all the models created and will allow the user to select which ones will participate in the answer.

2. STATE OF THE ART

The purpose of this chapter is to describe the research process and materials looked in order to understand and construct an open domain Question Answering system. The research is focused on the areas of natural language processing and machine learning, and software development technologies as well as deployment techniques.

2.1 Question Answering System

Open Domain Question Answering is the field of study in which automatic systems that generate answers to questions so that the way of generating the response and the sources of information are left open and flexible with the possibility of using diversified topics [6].

There have been several approaches to implement this kind of system. They can be classified into three [4]. First, there were proposed **Two-stage Reader-Retriever approaches** which contain two modules, a reader and a retriever. The retriever performs an information retrieval task and searches in a corpus of documents fragments that might contain the answer to the question. The retriever utilizes machine comprehension techniques in order to extract the answer from the fragments selected by the ranker. More recently, Dense **retriever and end-to-end training approaches** appeared. These architectures utilize dense document representations in the retriever stage and train together both retriever and reader. It is also worth mentioning **Retriever-free approaches**, that use large scale pretrained Machine Learning models trained on a massive corpus of text. During inference they don't need to search any text source base and they are able to generate the answer directly. This is the case of large language models like GPT-3 [7] or PaLM [8] [3].

Some Open Domain Question Answering systems have been implemented:

- **Elicit [9]:** It is a research assistant built by the research lab Ought [10]. Its goal is to help automating researchers' tasks. The one most developed is the Question Answering tool, used for literature review.

  The way it works is when the user asks a question, Elicit tokenizes it and removes stopwords in order to search the title and abstract of relevant papers in an academic API. Then, it ranks the papers using GPT-3 [7] and a finetuned T5 model [11]. For the top 8 papers, it computes the answer to the question that the abstract might contain using GPT-3 finetuned with 2000 examples of questions, abstracts, and answers. It identifies if the study was a controlled trial using a SVM based on bag-of-words. Also it uses GPT-3 to extract other information like number of participants or number of studies. [12]. Finally, the results are displayed on a table on a web or native application interface.

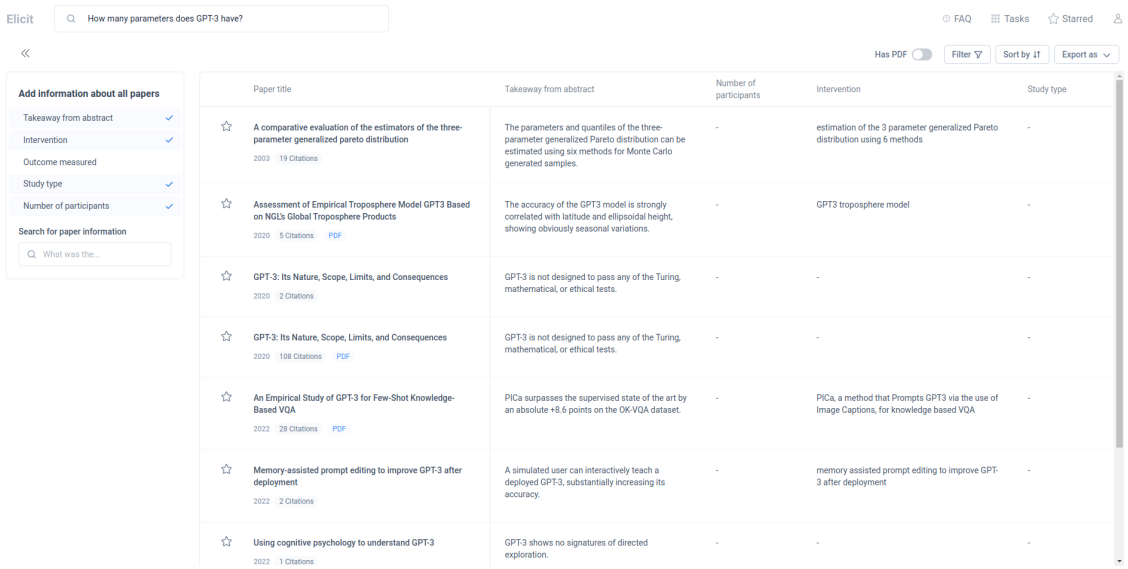


Figure 1: Elicit interface screenshot displaying the results of a query [9].

- **DeepQA [2]:** The IBM research team built DeepQA as a challenge to create a system able to compete at jeopardy in real time against humans. This is a game quite similar to the question answering problem. The contestants are given a fact and they have to find the person, place or concept that the clue describes.

  The process they followed to solve this problem was first to analyze the question asked by performing parses, logical analysis, semantic role labels, name entity recognition, and question classification. With that analysis a primary search is made to retrieve potential documents from multiple text search engines. From them, the system generates some candidate answers and with them, it searches for passages that contain the keywords for the answer that support the evidence that those answers are indeed the real answer. Finally, the answers are ranked and a confidence estimation is made.

Additionally, virtual assistants like Google's Assistant [13], Apple's Siri [14] or Amazon's Alexa [15] already contain embedded question and answering functionalities that users can use with a voice interface.

2.2 Machine Learning and Natural Language Processing

Here the concepts related to Machine Learning and Natural Language Processing necessary to develop the project are described.

2.2.1 Machine Learning

Related to Question Answering, several pre-neural approaches were followed in the past but their performance is nowhere near to the newest Machine Learning models. **Machine Learning** is the field of study which attempts to build and understand methods that are

able to learn patterns from data and improve the future performance of specific tasks. This is an alternative to classical programming where the behavior of the program is coded. Here the behavior is learned from data. [16].

Inside this field, there are several types of learning like supervised or unsupervised learning. In **Supervised Learning**, the goal is to predict from an input the correct output based on training data which comes in input-output format. The input is normally a vector of features and the output is a class or a number. In **Unsupervised Learning**, the input data doesn't contain any output. The goal is to obtain new patterns and relationships from the features. [16]. This project will make use of supervised learning mostly. Inside, supervised learning, one of the most relevant tasks is **classification**. Here the goal is to predict the class or label given some input instance.

One of the methods used in machine learning is **Neural Networks.** These are networks of smaller components called neurons. They take an input vector and output another one. A unit follows the following expression, where given an input, the output is calculated using weights and a bias term [17]:

$$z = w \cdot x + b$$

The output of a unit can be passed through an **activation function**:

$$y = a = f(z)$$

Their purpose is to further modify the output of a neuron to achieve different effects. The **Sigmoid** function, for instance, transforms the output of a unit to a range from 0 to 1. It can be used in binary classification tasks, where 0 would represent one class and 1 the other[18].

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

Another relevant activation function is the **Softmax,** useful for multi-class classification. Returns a vector of probabilities with the size of the dimension of classes. The addition of all the values sum 1. [19]

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

It is also worth mentioning the **ReLU** activation function**.** This one is used to avoid saturation and 0 derivatives which help to avoid vanishing gradients. [17]

$$ReLU(z) = max(0, z)$$

Neural networks can be organized into several architectures. One of the most famous is the **Multilayer Perceptron**: This is an architecture in which the processing units are organized into layers and the information is passed from lower to higher layers with no cycles.

Normally, they have an input layer which recieves the features from the input vector, an output layer, which gives the result and a number of hidden layers, which process the data. When neural networks contain several layers we start talking about **Deep Learning.**

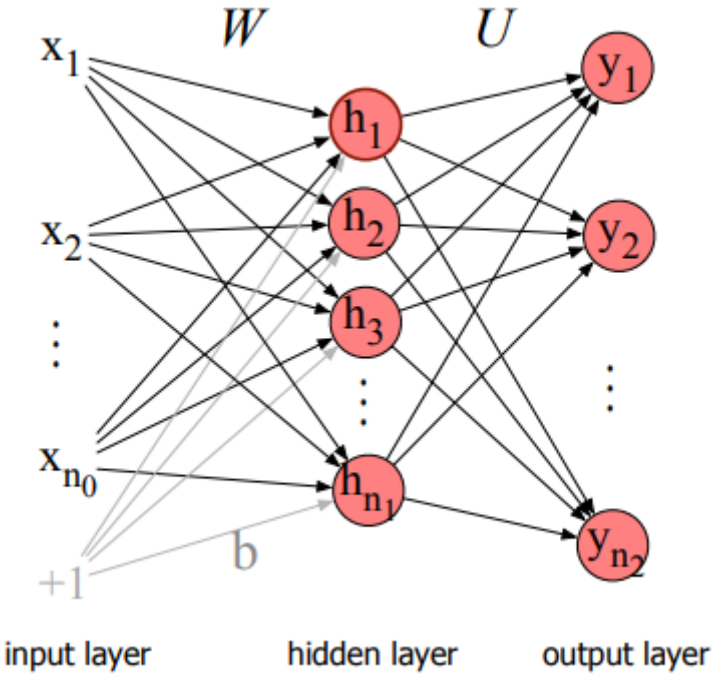


Figure 2: Multilayer perceptron [17].

To train neural network architectures the weights of the different layers must be modified. To know how to do it is necessary a loss function, which measures the difference between the model output and the expected output. An example of loss function is the **Cross Entropy Loss**, useful for function for multiclass classification [17].

$$L_{CE}(\hat{y}, y) = - log(\frac{e^{z_c}}{\sum_{j=1}^{k} e^{z_j}})$$

Gradient Descent is the method used to guide the modification of weights using the loss function. It finds the minimum of the loss function and calculates the direction in which the parameters must be modified to get to that minimum. [19] Applied to a loss function would be to find the neural network parameters that minimize the value of the loss function:

$$w^{t-1} = w^{t} - \eta \frac{d}{dw} L(f(x: w), y)$$

The parameter η Is the learning rate, which modifies how much change is produced to the weights in every modification.

There is a probabilistic approach to the **Gradient Descent** algorithm, **Stochastic Gradient Descent**. Instead of calculating the gradient for all the training instances, it randomly selects one of them each time and applies the same modification to all the instances. This

avoids computational overhead. [21]

Sometimes it is necessary to include cyclic connections inside neural networks. This is the case of Recurrent neural networks. Cycles makes the output of its cells depend on the values of previous outputs. They contain a hidden layer, situated between input and output. The output of the hidden layer depends on the value of the input and the previous value of the hidden layer. That makes the output dependent on a sequence of inputs. The hidden layer provides some kind of memory and the elements of an input sequence are presented one by one. These kind of architectures are useful for text input due to their sequential nature. [22]

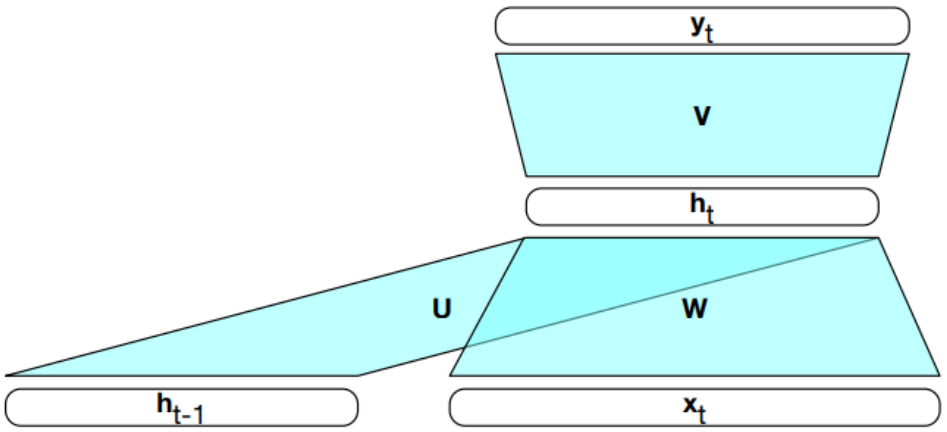


Figure 3: Recurrent Neural Networks [22]

The inference process is performed according to the following expressions:

$$h_t = g(Uh_{t-1} + Wx_t)$$
$$y_t = f(Vh_{t-1})$$

A **bidirectional RNN** utilizes two RNN one from left to right and another from right to left. Then, both hidden states are concatenated. **Stacked RNN** are formed by using the output of one RNN to the input of another. [22]

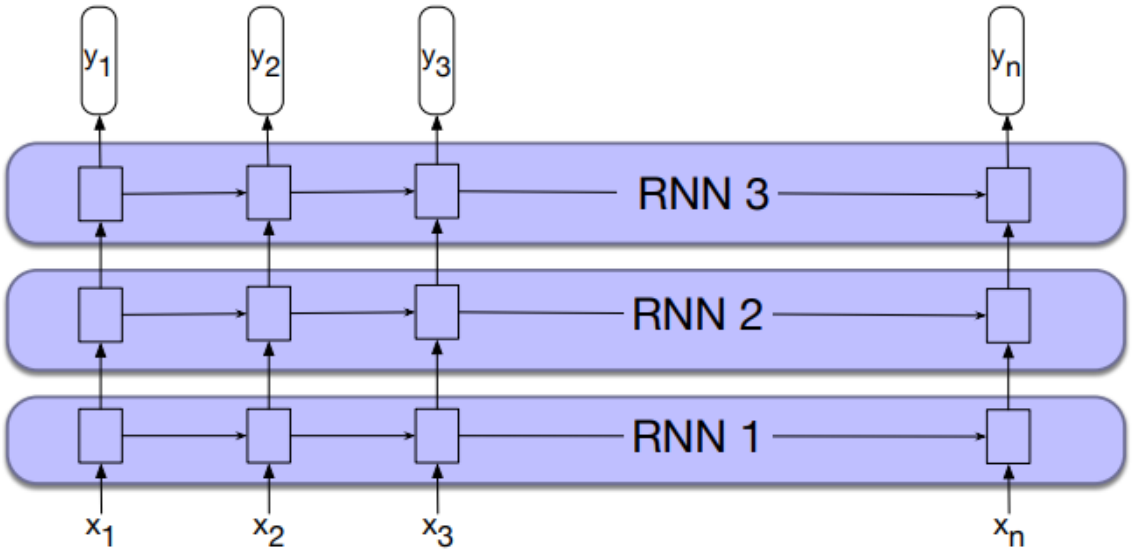


Figure 4: Stacked RNN [22]

However, recurrent neural networks come with several problems. They contain very local context which can induce errors and are prone to vanishing gradients as the hidden states are subjected to many repeated multiplications. **LSTM (Long short-term memory)** networks come to solve the problems of normal RNN. LSTMs remove the context not needed and keep the ones more likely to be used. They use for that purpose a context vector added to the already existing hidden vector. Also, gates are used to manage the flow of information. [22]

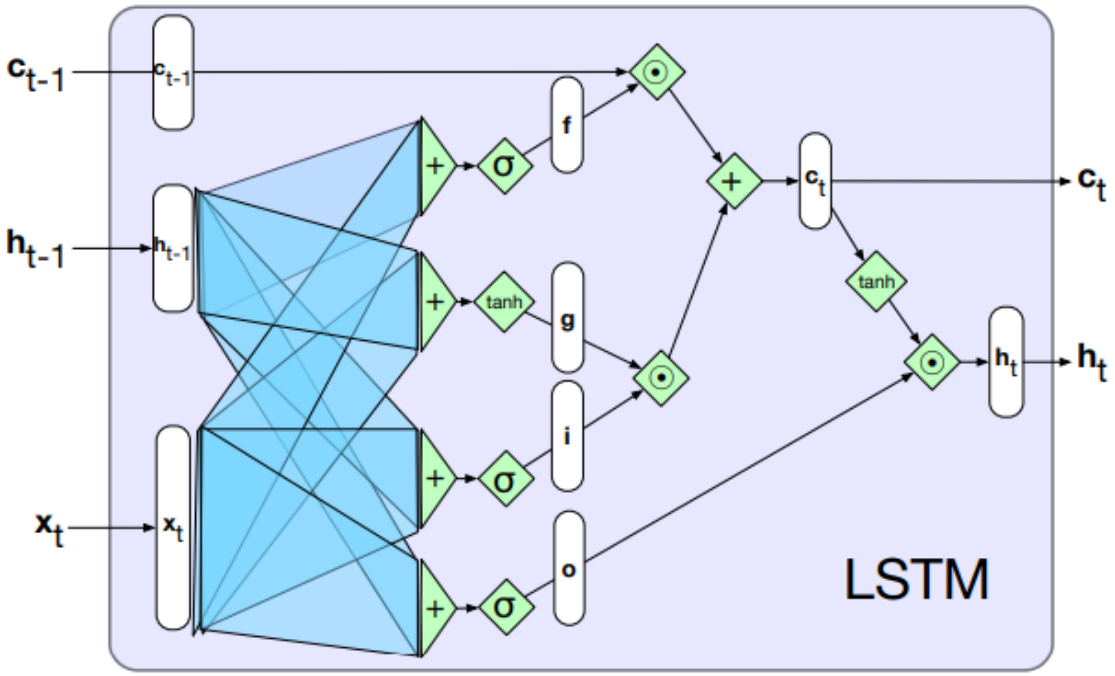


Figure 5: LSTM [22]

They use forget gates, which delete information that is not needed. It first averages and

adds the last hidden state and the input and passes it through a sigmoid activation function. Then, a pointwise multiplication with the last context vector.

$$f_t = sigmoid(U_f h_{t-1} + W_f x_t)$$
$$k_t = c_{t-1} \odot f_t$$

The add gate retains the needed information:

$$g_t = tanh(U_g h_{t-1} + W_g x_t)$$
$$i_t = sigmoid(U_i h_{t-1} + W_i x_t)$$
$$j_t = g_t \odot i_t$$

The new context vector is:

$$c_t = j_t + k_t$$

The output gate calculates the hidden state:

$$o_t = sigmoid(U_o h_{t-1} + W_o x_t)$$
$$h_t = o_t \odot tanh(c_t)$$

LSTMs present problems. Their sequential characteristics make training sequential and slow it down. Also, there is some information loss. An alternative method are **Transformers** blocks. They map sequences of input vectors to output vectors of the same length without the need for recurrent connections. They consist of combinations of linear layers, feedforward networks and self-attention layers.

A **self-attention** block compares the elements of the input sequence to rank them in relevance using similarity metrics like the dot product. The comparisons are made using the similarity score and then the results are passed over a softmax layer to obtain normalized results.

$$a_{ij} = softmax(score(x_i, x_j)) \ \forall j \leq i$$

Then the output is calculated by weighting the input by the comparison vector computed.

$$y_{ij} = \sum_{j \leq i} a_{ij} x_j$$

In an attention mechanism three different embeddings are used in which the input vectors are projected. The **Query** focuses the attention on the current input compared to all the previous inputs. The **Key** focuses the attention on previous input compared to the current focus of attention. The **Value** computes the output of the current focus of attention. They are calculated using weight matrices.

$$q_i = W^Q x_i;\ k_i = W^K x_i;\ v_i = W^{k_i} x_i$$

The score metric to be used can be updated using the query (current information) and the key (previous input). It is also scaled using the square root of the dimensionality of those vectors.

$$score(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

$$a_{ij} = softmax(score(x_i, x_j)$$

$$y_i = \sum_{j \le i} a_{ij} \cdot v_j$$

In matrix multiplication, it would be:

$$Q = XW^Q;\ K = XW^K;\ V = XW^V$$

$$SelfAttention = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

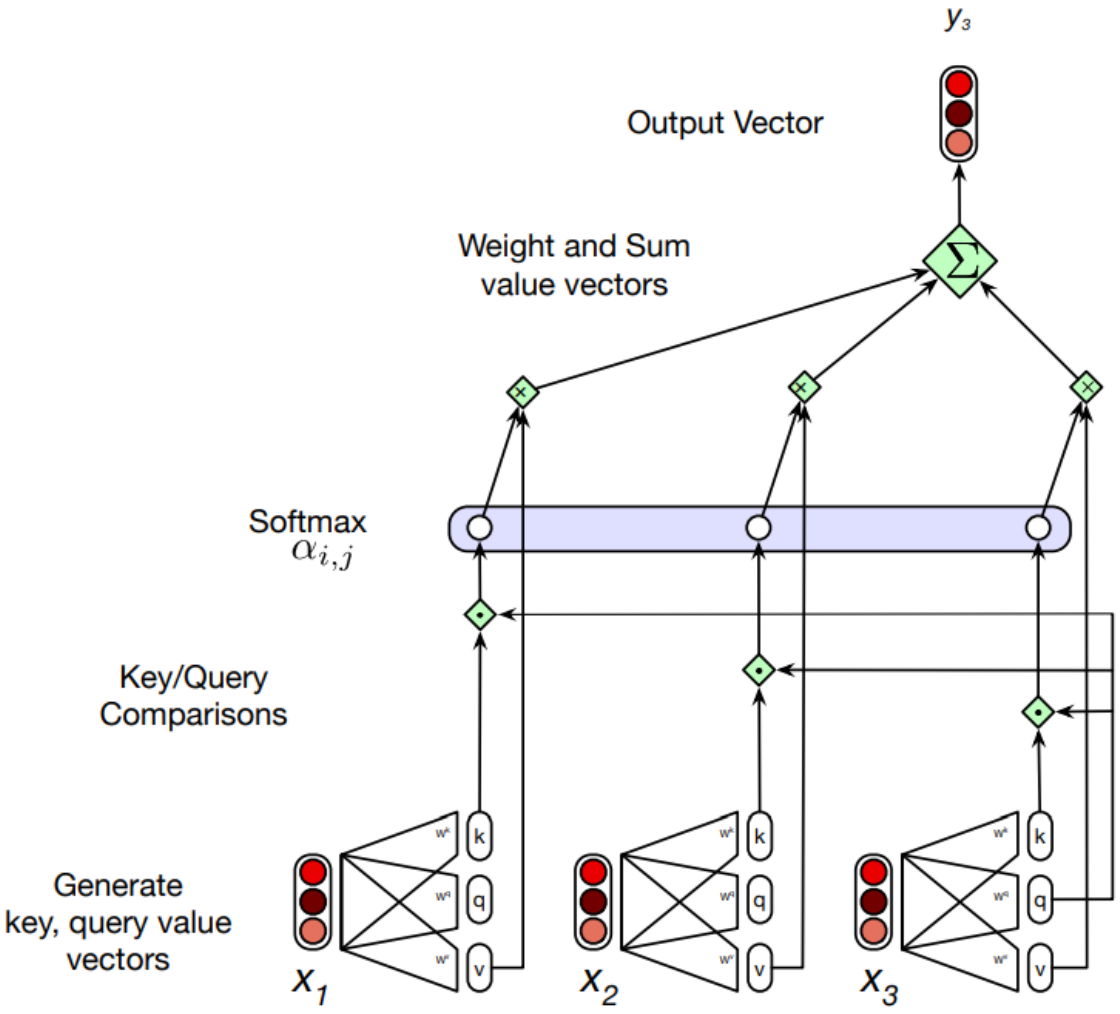


Figure 6: Self-attention layer [22]

A **Transformer Block** is composed by a self-attention layer and various feedforward layers with residual connections and normalization layers. Residual connections use information that doesn't pass through some layers by adding it to the information that did. This improves the efficiency of learning. The dimension of the output is the same as the dimension of the input. Blocks are normally stacked so that they can reach deeper levels of abstraction.

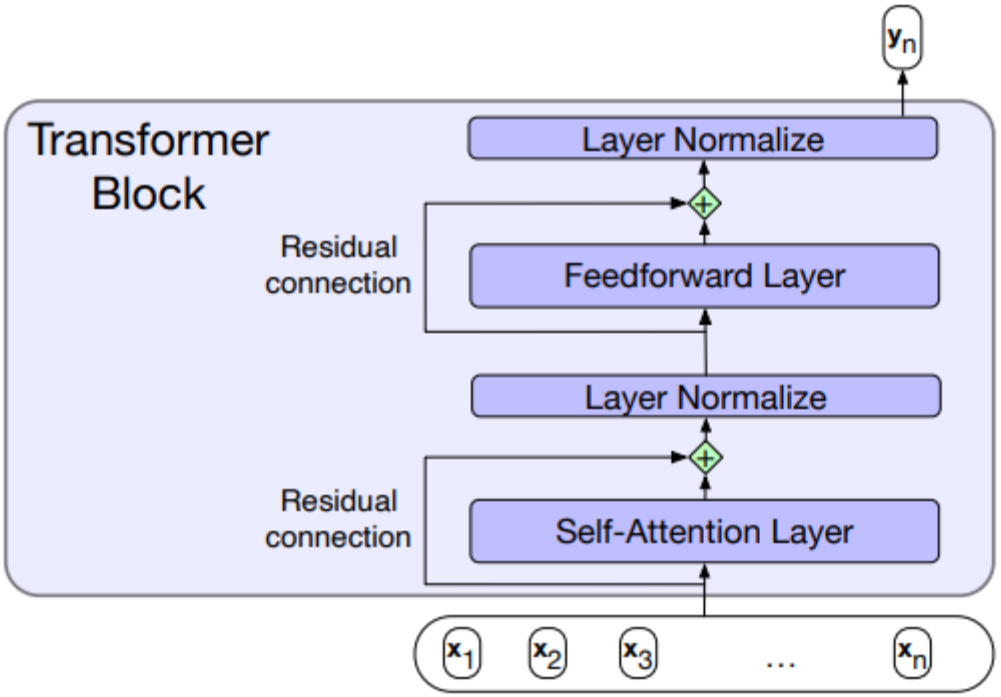


Figure 7: Standard Transformer Block [22]

$$z = LayerNormalize(x + SelfAttention(x))$$
$$y = LayerNormalize(z + FeedforwardLayer(z))$$

The layer normalization is used to improve the efficiency of the training process. It can be based on z-score:

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i - \mu)^2}$$

$$\hat{x} = \frac{(x-\mu)}{\sigma}$$

$$LayerNormalized = \gamma x + \beta\ (learnable)$$

Often, a **multihead attention** mechanism is used. It combines multiple self-attention layers combined at the same level so that each head can capture different relationships over the input sequences. Each of the heads have their own matrices W^K_i, W^Q_i, W^V_i so that they can produce different embeddings for the same input. At the end, all the heads are concatenated and projected into the output:

$$MultiHeadAttention(X) = (head_1 \oplus head_2 \oplus \ldots head_n \oplus) W^O$$

Currently, there are highly effective models based on transformers. **BERT (Pre-training of**

Deep Bidirectional Transformers) is one example of such models. It is based on bidirectional transformer encoders and the method of masked language modeling. [23]

In normal transformers the information comparison of the input was made from left to right. However, this could be a limitation for some tasks, where right to left information could also be important. That is why the self attention layer also performs comparison from right to left of the input tokens [24]:

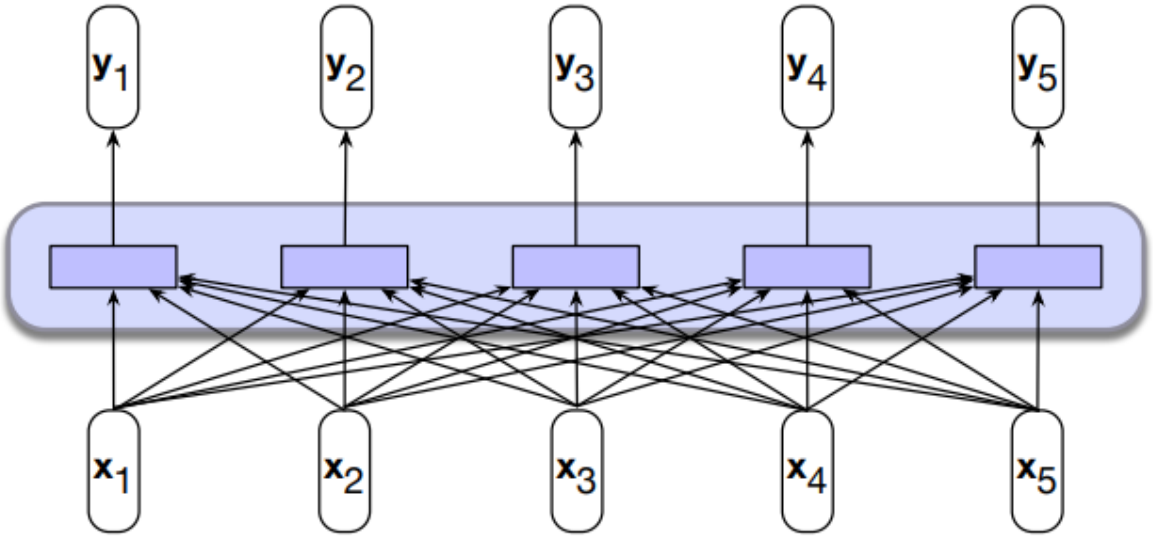


Figure 8: Bidirectional self-attention layer [24]

The training process of BERT is performed by using two tasks with a corpus made over 800 million words over BookCorpus, 2.5 Billion words of Wikipedia:

One is **Masked Language Modeling** [23]. Normal transformers would be trained on predicting the next word, as in the task of language modeling. Here, the task is to predict words which are in the middle of a sentence, knowing the preceding words and the next ones. For that purpose, they replaced 12% of the input tokens with a [MASK] token. The input sentence is tokenized using a subword tokenizer from the WordPiece tokenizers [25]. Concretely, the **BPE tokenizer**. After the input sequence is passed into the Bidirectional Transformer Encoder, it is multiplied by a classification weight so that the output has the dimension of the vocabulary. Then, a softmax layer over the vocabulary is applied and the cross entropy loss function is used to drive the learning process.

The other task is **Next Sentence Prediction** [23] where BERT is presented with pairs of sentences and it is asked whether the pair is really adjacent or not in the corpus. Two special tokens are added: [CLS], added at the beginning of the input; and [SEP], which separates both sentences. In a similar way, a softmax activation function and cross entropy loss is used after the transformer.

The usefulness of BERT training is that it can be used in other tasks by a process called **fine-tuning**, in which the model parameters are slightly modified and additional training processes are made over some specific dataset.

2.2.2 Natural Language Processing

It is a subfield of artificial intelligence and linguistics that tries to build mechanisms able to understand, interpret and process natural language employed by humans. It focuses its attention on a wide range of tasks such as language translation, machine comprehension, sentiment analysis, or summarization. [26]

One important task for this project is **Answer Span Extraction**, which consists on finding from a passage of text and a question, the span of text in the passage that answers this question. Another one is **POS Tagging**. This one consists in assigning a tag to each of the tokenized words in a sequence of text. Receiving a sequence x1, x2, x3 … xn words, it outputs y1, y2, y3, …, yn tags. Some common tags are grammatical ones (NOUN, VERB, ADJ, PROP, DET). The challenge is to solve the disambiguities between words. It is therefore a classification task inside a defined tagspace.

Information Retrieval is the field that tries to retrieve media taking into account the user's information needs. Concretely, the relevant task is ad hoc retrieval, in which a query is formulated and an ordered set of documents is returned from a collection.

Vector semantics are the way in which we can represent the meaning of words from a document. When we represent the meaning of words using vectors, we call them **embeddings.**

These vector representations can be sparse or dense. Sparse representations are those that can have a non fixed vector length and might vary with the vocabulary. A bag-of-words representation, which is based on occurrence counts, is a good example. A dense representation is a fixed length representation based on real numbers instead of counts. Some examples are word2vec, fastext or GloVe.

One of the most basic forms of representations is the **Term-document matrix.** It has a matrix shape in which the rows represent a word in the vocabulary and the columns are the documents in the corpus. Each cell represents the frequency of co-occurrence of both word and document.

Similarly, the **Term-term matrix** representation is a table representation in which columns and rows are words. It takes into account the co-occurrence of words. The rows would mean to represent the target word and the column the context word. The context can be defined as the +- n words surrounding the target word.

One widely used representation form is the **Bag-of-words.** This treats every document as a bag of words. It consists of an unordered set of words that only keeps the occurrence of each word in the vocabulary.

A more complex version of the Bag-of-words is **TF-IDF.** It is a sparse bag-of-words representation in which every term is weighted according to frequency and document

appearance. It is composed of two elements:

- The term frequency, which measures the co-occurrence of word and document.

$$tf_{t,d} = log_{10}(count(t,d) + 1)$$

- The inverse document frequency, which penalizes that a word appears in many documents as it is less discriminating.

$$idf_t = log_{10}\frac{\#\ documents}{df_t}$$
$$tf\ idf(t,d) = tf_{t,d} idf_t$$

More complex versions of TF-IDF exist. This is the case of **BM25.** It adds two parameters, k, to balance between term frequency and inverse document frequency and b, to control the importance of document length.

$$BM25 = \sum_{t \epsilon q} log(\frac{N}{df_t}) \frac{tf_{t,d}}{k(1-b+b(\frac{|d|}{|d_{avg}|}))+tf_{t,d}}$$

Regarding dense representations **word2vec** [45] must be mentioned. This model generates word embeddings based on the context of words so that they are fixed length and they are always the same. word2vec embeddings are in reality the weights of a classifier after being trained. The classifier is presented with a target word and a context window of words that are around that target word. Its task is to determine how likely the word is to be found in that context. The process of training is to treat target words and neighboring context as a positive example from a corpus of text, randomly sampling other words as negative context, use a logistic regression classifier to distinguish both classes and use the resulting weights as embeddings.

Other different approaches to dense static embeddings are **fasttext** [46] and **GloVe** [47]. fasttext uses a subword representation to generate the embeddings while GloVe uses multiple approaches like that of word2vec with a cooccurrence matrix and PPMI models [27].

Once a corpus is represented using one of these representations, they can be compared using **cosine similarity.** It is a similarity metric between two vectors that computes the cosine of the angle that is formed between them. The mathematical formula is based on dot product between vectors and the formula would be:

$$cosine(v,w) = \frac{v\,w}{|v||w|} \cdot \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

In order to train Machine Learning and NLP algorithms it is often necessary to perform a data preprocess. This normally facilitates the training process of the algorithms and improves the performance. **Tokenization** is one example of a preprocessing task. It consists of separating a text into smaller units like sentences, words or lemmas. Oftentimes, regular expressions are used for tokenization. These are algebraic expressions that characterize strings. They define the pattern of text and is useful to search for those patterns in a specific text or for tokenization. One example of a tokenizer is the **BPE Tokenizer**. It is an algorithm that tokinzes texts. It builds a vocabulary which initially is formed by the individual characters that appear. It then examines the corpus, chooses the two elements that are most frequently adjacent, creates a merged symbol and adds it to the vocabulary, replacing the prior individual symbols. This merging process is repeated k times more. The vocabulary is composed of the original characters and the new symbols generated [27].

2.2.3 Evaluation Metrics

Before deploying Machine Learning models it is crucial to test whether the performance is appropriate. Here there are outlined the evaluation metrics that are relevant for measuring the performance of the different models to be developed as well as the whole system.

A **Confusion matrix** is a table that allows the visualization of how an algorithm performs compared to how it is expected to perform. It contains two dimensions. One axis describes the output of the system while the other the expected outputs. True positives are instances that the system classifies as positive and are indeed positives. True negatives are instances that the system classifies as negative and are indeed negative [28].

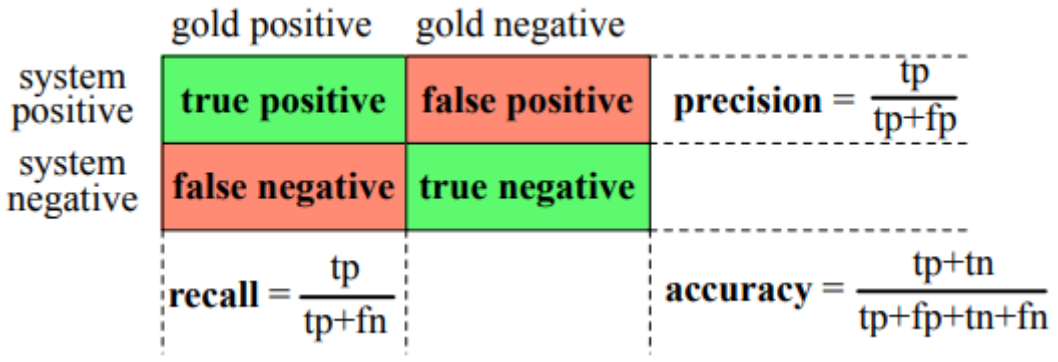


Figure 9: Confusion Matrix [28]

From the confusion matrix several metrics can be extracted.

- **Accuracy:** Performance metric that computes the percentage of the predictions that were correct [28].

$$Accuracy = \frac{true\ positives + true\ negatives}{all\ predictions}$$

- **Precision:** Measures the percentage of instances that the system detected correctly [28].

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

- **Recall:** Measures the percentage of the expected data that was correctly identified by the system [28].

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

- **F1-score:** It is a performance metric that combines both precision and recall [28].

$$F1 - score = \frac{2\ precision\ recall}{precision + recall}$$

Two metrics specific to information retrieval are Mean Reciprocal Rank and Mean Average precision:

- **Mean Reciprocal Rank** is a measure used by ranking systems in information retrieval to assess the performance of systems that return a ranked number of elements in response to a query. It returns a number based on the position of the first relevant document that is retrieved and averages that result over the number of questions performed.

[29]

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i}$$

- **Mean Average Precision** is a similar metric to mean reciprocal rank. This one takes into account the position of the relevant documents retrieved and the number of relevant ones found until that position. It is used for a system that returns a ranked number of documents in which some might be relevant and others not. Mean average precision averages the precision of each of the questions asked. The precision is computed by, for each query, P(k) is the precision at rank number k. The factor relevance(k) returns one or zero depending on the value of whether the document at rank k is relevant or not. Then the summation is divided by the number of ground true positives, the number of relevant documents retrieved. [30]

$$MAP = \frac{\sum_{q=1}^{Q} AvgP(q)}{Q}$$

$$AvgP(q) = \frac{1}{GTP}\sum_{k}^{n} P(k) * relevance(k)$$

$$P = \frac{|\{relevant\ docs\} \cap \{retrieved\ docs\}|}{\{retrieved\ docs\}}$$

There are also metrics which are targeted towards measuring Question Answering performance:

- **Exact Match** measures the capabilities of the system getting exactly the perfect answer. Knowing the answer to the context from which it is extracted, if the predicted answer is exactly the same, character by character to the solution, then the value of EM is 1. Otherwise, it is 0 [31].

- **F1-score** can also be adapted to Question Answering by using the words in the prediction and in the expected answer. Precision here is the ratio between shared words and the words in the prediction and recall is the ratio between shared words and the words in the expected answer. [31]

2.2.4 Relevant Datasets

Here are some datasets that were helpful in understanding and solving the task to be solved.

- **SQuAD:** The Stanford Question Answering Dataset is a reading comprehension dataset with more than 100.000 context, question and answer entries crowdsourced from more than 500 wikipedia articles. There are currently two versions of the dataset; the 1.1 version and the 2.0 version. The latter adds 50.000 non answerable questions, which the model must identify as non-answerable [32].

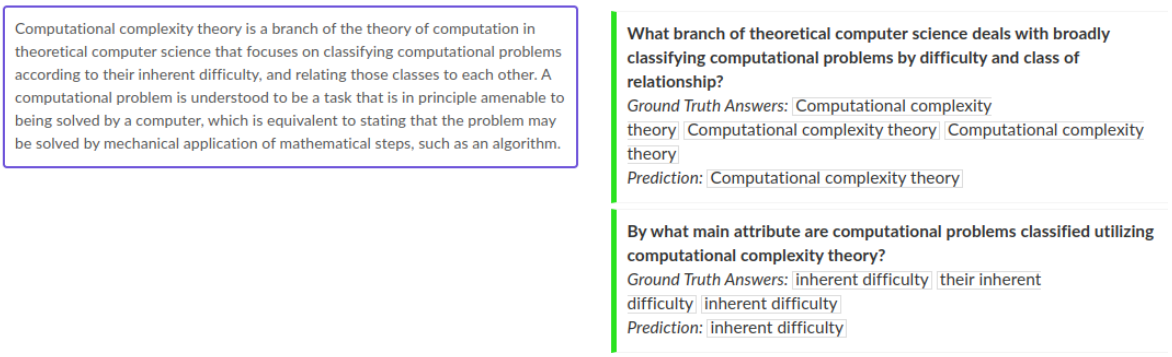


Figure 10: SQuAD dataset explorer

- **TriviaQA:** TriviaQA is a reading comprehension dataset which contains more than 650.000 entries with question, answer and evidence. It is also crowsourced and it contains on average six potential source documents for each question. It is more challenging than SQuAD as the authors demonstrated, models that performed well on SQuAD, nearly human performance only achieved from 20-40%. [33]

- **searchQA:** A general question answering dataset with over 140.000 question

answers pairs augmented with 49.6 snippets on average. The goal of this dataset is to evaluate the full pipeline of open domain question answering by including noisy source snippets. [34]

- **Natural Questions:** Dataset created by Google with over 300.000 real Google search engine queries. Annotators were presented with the query and a wikipedia article from the top 5 results. A short answer and a long answer were annotated if they were present. [35]

- **MS MARCO:** Reading comprehension dataset extracted from anonymized queries to Bing and Cortana. It includes query, human generated answers and passages from which answers can be extracted. [36]

- **Penn Treebank:** The Penn Treebank corpus is a dataset created from articles of the Wall Street Journal. It is an annotated dataset, meaning that the words have been annotated with Part of Speech (POS) information and skeletal syntactic structure [37]. An example of a data entry is:

```
( (SBARQ
   (INTJ (UH So) )
   (WHNP-1
    (WHADJP (WRB how) (JJ many) )
    (, ,)
    (INTJ (UH um) )
    (, ,) (NN credit) (NNS cards) )
   (SQ (VBP do)
    (NP-SBJ (PRP you) )
    (VP (VB have)
     (NP (-NONE- *T*-1) )))
   (. ?) (-DFL- E_S) ))
```

- **Language Detection dataset:** A small language detection dataset with 17 different languages available on Kaggle. [38]

- **wikicorpus:** It is a trilingual corpus with English, Spanish and Catalan Wikipedia articles. [39]

2.3 Technology

In this section, the libraries used to develop the project are mentioned as well as all the technologies needed to deploy the models through a web application.

2.3.1 Machine Learning Frameworks

During the development of this project, several already existing frameworks were used. To develop neural networks there are several Application programming interfaces (API) available for the programming language Python.

- **PyTorch** [40] is an optimized deep learning framework based on Python and has as main features tensor computation, GPU (Graphical Processing Unit) and TPU (Tensor Processing Unit) acceleration support and automatic differentiation for neural networks.

- **Tensorflow** [41] is also an open source deep learning framework with an easy and intuitive model building framework, powerful enough for research and geared for production environments and deployment. Comparing both of them, PyTorch has a lower level more complex interface which is becoming more prominent in academia.

- **Huggingface Transformers:** It is a state of the machine learning framework that works with both PyTorch and Tensorflow. It contains an API to easily download, train and deploy state-of-the-art pretrained machine learning models. It has a social hub where you can upload your trained models and datasets or see the ones from other users. [42]

- **Natural Language Toolkit:** Python library specialized in programs that work with human language data. It contains interfaces to work with text classification, tokenization, stemming, parsing, tagging as well as related corpora such as treebank corpus or WordNet. [43]

- **Gensim:** It is a Python library for natural language processing and information retrieval. It offers easy interfaces for semantic analysis with topic modeling, document indexing and similarity retrieval. It is interesting in this case for the inclusion of word2vec models. [44]

- **Numpy:** Scientific package implemented in C with interface for Python. Counts with N-dimensional arrays, high performance and mathematical functions. [48]

- **Pandas:** Open source Python library used for data analysis and manipulation. [49]

- **Sentence Transformers:** Interface to use Sentence-BERT. It is a pretrained BERT model that is able to derive meaningful sentence embeddings that can be used with cosine-similarity. [50]

2.3.2 Interface and frontend

The interface and frontend are one of the most important parts of any platform because the final user interacts with the system through it.

In web based platforms, the most common languages are HTML, CSS and Javascirpt.

The content on the web is meant to be dynamic and interactive. For that, there are two options.

- **Server Side Scripting**, in which the server is in charge of serving the changes in the HTML and layout desired by the user. Some common programming languages that are used in the backend are PHP, Ruby, Node.js or Python [51].

- In **Client Side Scripting**, the effort is delegated to the browser. To do that javascript scripts are run to do things like changing between tabs on a website, validating forms and modifying data [51].

Web interfaces can be implemented using simply Javascript CSS and HTML, however, normally frameworks are used to aid in the development of web interfaces. These frameworks are based on components, which are small, modularized and reusable packages of code [52]. Three frameworks can be highlighted such as React.js [53], AngularJs [54] and Vue.js [55]. All of them need to be compiled before they can be deployed.

Some frameworks have also been useful during the construction of the project.

- **i18next:** It is an internationalization framework which provides features like detecting the user language and loading or caching translation. It is originally written in Javascript and available in frontend frameworks like React, Angular or Vue.js but it has expanded also to the backend for PHP or even devices like Android or iOS. [56]
- **Bootstrap:** It is an open source framework for the development of web applications which provides a CSS to style websites in a consistent way being reactive across multiple devices. [57]
- **react-speech-kit:** It is an react package which act as a hook for The SpeechSynthesis [58] and SpeechRecognition [59] APIs. They allow the inclusion of text-to-speech and speech recognition functionalities in an easy way. [60]
- **Material Design:** Similarly to Bootstrap, it is a set of components and guidelines that help the users in developing interfaces. [61]

2.3.3 Backend

Normally it is necessary for the web application to communicate with a server so the

necessary files and information can be fetched at the frontend interface. That is normally accomplished with the use of APIs and built over the HTTP protocol, over which the internet is based [62].

There are two main architectural styles to develop APIs:

- **SOAP (Simple Object Access Control):** SOAP is a stateful web based access protocol that uses XML as a format to write the messages. It is a protocol well defined in the sending and receiving of packets and the packets sent have some headers and a body. However, the use of XML formats can become complex and slow through HTTP connections [63].

- **REST (Representational State Transfer)**: REST is a stateless protocol focused on the data. The data itself defines how the communication protocol must take place. It uses the HTTP defined methods like GET, POST, PUT, OPTIONS, DELETE, ANY. It also supports any file format like CSV, JSON, RSS which make it adaptable to any application. All those factors make the exchange of data lighter and faster [64].

2.3.4 Deployment

When deploying an application for user usage it is important to take into consideration that the system is able to respond to variable amounts of demands in terms of processes, network, time or storage. The proper response to client demands is a sign of competitiveness. [65].

The scalability of a system can be viewed in two ways. One is called **vertical scalability**, which means to modify the application so that components can be updated and replaced. **Horizontal scalability** deals with the need of running more instances that might trigger the use of more servers and the load balancing on them. [66]

The deployment of applications can be done **on-premise**, which is the traditional enterprise building computing model. Here the company was in charge of owning the hardware and software to be able to run the application, perform the scalability, security and maintenance themselves. In recent years, businesses rely more on cloud computing. The servers are owned by external companies and they are in charge of any maintenance, networking and security issues and they can be utilized by paying for their use [67].

There are different types of cloud computing that depend on the business needs [68] [69].

- **Infrastructure as a service (IaaS):** Companies use hardware infrastructures from a third party by paying for them. The contracts used allow companies to select how much memory, storage and processing capacity that the instances they are buying can have and they can be scaled depending on their needs. Some companies that offer their hardware as hosts are Amazon Web Services [70], Microsoft Azure [71] or Google Cloud [72].

- **Platform as a service (PaaS):** Here companies offer platforms where users can launch all kind of applications and they just have to focus on their development. Some examples are Google App Engine [73] or Bungee Connect [74].
- **Software as a service (SaaS):** In this model the company delegates its data or software to servers external to it so that employees can access from any point. Some examples are Microsoft Office 365 [75] or WordPress [76].

To deploy applications is widespread the use of **containers.** They allow the use of a unique host instance by multiple application components in a way that they are isolated from one another thanks to the Linux kernel virtualization capabilities. They work in a similar way as **Virtual Machines** but with the advantages that they are lighter, with better resource efficiency and are faster to deploy, provision and restart, which benefits developers. **Docker** is a platform that offers the tools for developers to create, deploy and share their own containers [77].

3. SYSTEM ANALYSIS

In this section the application through which the model developed is going to be available described. To do so, the design will be explained, with the user personas, requirements, use cases and class diagrams.

3.1 Users

It is important to know the potential users of the application so that the system can reflect the needs of the users. Here some examples of personas are described from which later requirements, use cases, and prototypes will be further developed.

- **Persona 1: JOHN MATTHEWS**
    - **Personality and bio**
      John is a PHD research student attending University of Waterloo. He is a hard working student that tries to live life to the fullest enjoying activities like ice skating and grabbing a beer with his friends. He is a technology enthusiast and likes to be up to date with the latest cutting edge discoveries. The program he is enrolled in is demanding and time consuming and is always looking for productivity tools that can make more free time for himself.
    - **Technology skills**
      Being a computer science student he is quite an expert in all technological domains. In every system he uses, he likes to immerse himself in the settings options and customize as much as possible to achieve maximum productivity. He always likes to own the latest technological devices and although he carries his phone always with him, he spends most of his time on his desktop PC.
    - **Goals and objectives**
      Right now he is searching for a tool that allows him to search for information efficiently so that he can search for information he doesn't know while working on his PHD. He also is interested in knowing how the tools he uses work so that he can think on how to improve everything he uses.

- **Persona 2: SUSANA MARTÍNEZ**
    - **Personality and bio**
      Susana is a middle aged woman who works as a kindergarten teacher. She has always felt comfortable around children which easily led her to become a teacher. She is naturally cheerful and fun to be around.
    - **Technology skills**
      In high school it became clear to her that she would not have a future in the technological world. She always found it difficult to keep up to date with technology, and now always relies on his son to help her get unstuck with her phone. She likes to use highly intuitive and easy interfaces that provide lots of help and guides. She uses the assistant a lot on her phone as she finds it quite natural.

- **Goals and objectives**
  The children in her school always ask her questions which most of the time she doesn't know. She has to take her phone to try to answer the children but she finds it hard to find the answers on the web. She would love to have an application that answers her questions like his son would reply on her chat app.

3.2 Requirements

Requirements define how the system must behave. Requirements can be of several types:

- User: They describe the needs of users
- Software: They describe the functionalities and characteristics of the software to be implemented. They can be of two types:
  - Functional: They describe the activities and functions that the system must perform.
  - Non functional: They describe the restrictions and characteristics of the system.

Table 1: Requirement Format

ID	(U-X F-X NF-X)
Description	
Origin	(RU-X)
Priority	(1-5)
Necessity	(1-5)
Verifiability	(Yes-No)

The elements on the table are the following:

- **ID:** Serves to uniquely identify a requirement. They follow this format: A-B where A indicates whether the requirement is of type User (U), Functional (F) or non functional (NF). B is an integer number.
- **Origin:** Serves to know where the source of one requirement. It may be a user persona in case of user requirement or a user requirement in case it is a software requirement.
- **Necessity:** Indicates how much is needed to be implemented. It is an integer from one to five.
- **Priority:** Indicates the order of implementation. It is a value from 1 to 5.
- **Verifiability:** Indicates whether it must be tested once implemented. Its value must be Yes or No.

3.2.1 User requirements

Table 2: User Requirement U-1

ID	U-1
Description	The system shall allow the user to ask a question.
Origin	Persona 1 and Persona 2

Priority	5
Necessity	5
Verifiability	Yes

Table 3: User Requirement U-2

ID	U-2
Description	The system shall provide answers to the user's question
Origin	Persona 1 and Persona 2
Priority	5
Necessity	5
Verifiability	Yes

Table 4: User Requirement U-3

ID	U-3
Description	The system shall provide a chat interface.
Origin	Persona 2
Priority	4
Necessity	4
Verifiability	Yes

Table 5: User Requirement U-4

ID	U-4
Description	The system shall include a guide on how to use the system.
Origin	Persona 2
Priority	4
Necessity	4
Verifiability	Yes

Table 6: User Requirement U-5

ID	U-5
Description	The system shall include references to the resources used.
Origin	Persona 1
Priority	4
Necessity	4
Verifiability	Yes

Table 7: User Requirement U-6

ID	U-6
Description	The system shall include advanced options to adjust the query.
Origin	Persona 1
Priority	4
Necessity	4
Verifiability	Yes

Table 8: User Requirement U-7

ID	U-7
Description	The system shall give the possibility of having a speech interface.
Origin	Persona 2
Priority	2
Necessity	3
Verifiability	Yes

Table 9: User Requirement U-8

ID	U-8
Description	The system shall be usable in phone devices.
Origin	Persona 1 and Persona 2
Priority	5
Necessity	5
Verifiability	Yes

Table 10: User Requirement U-9

ID	U-9
Description	The system shall be usable in computer devices.
Origin	Persona 1
Priority	5
Necessity	5
Verifiability	Yes

Table 11: User Requirement U-10

ID	U-10
Description	The system shall be available in English.
Origin	Persona 1
Priority	5
Necessity	5
Verifiability	Yes

Table 12: User Requirement U-11

ID	U-11
Description	The system shall be available in Spanish.
Origin	Persona 1
Priority	5
Necessity	5
Verifiability	Yes

3.3.2 Functional Requirements

Table 13: Functional Requirement F-1

ID	F-1
Description	The system shall have a presentation section.
Origin	U-4
Priority	3
Necessity	4
Verifiability	Yes

Table 14: Functional Requirement F-2

ID	F-2
Description	The system shall have a section where the user can ask a question.
Origin	U-1, U-2
Priority	5
Necessity	5
Verifiability	Yes

Table 15: Functional Requirement F-3

ID	F-3
Description	The system shall have a section with all the references used.
Origin	U-4
Priority	3
Necessity	1
Verifiability	Yes

Table 16: Functional Requirement F-4

ID	F-4
Description	The system shall have a list of information about the page that serves as a guide.
Origin	U-4
Priority	3
Necessity	4
Verifiability	Yes

Table 17: Functional Requirement F-5

ID	F-5
Description	The system shall have a list of information facts about the page that serves as a guide.
Origin	U-4
Priority	3
Necessity	4
Verifiability	Yes

Table 18: Functional Requirement F-6

ID	F-6
Description	Each information fact must contain a title, a description, a picture and an optional link button.
Origin	U-4
Priority	3
Necessity	4
Verifiability	Yes

Table 19: Functional Requirement F-7

ID	F-7

Description	Each information fact shall contain a title, a description, a picture and an optional link button.
Origin	U-4
Priority	3
Necessity	4
Verifiability	Yes

Table 20: Functional Requirement F-8

ID	F-8
Description	Each reference fact shall contain a title, a description, a picture and a link button to the original source.
Origin	U-5
Priority	3
Necessity	4
Verifiability	Yes

Table 21: Functional Requirement F-9

ID	F-9
Description	The chat interface shall have an input text box.
Origin	U-3
Priority	5
Necessity	5
Verifiability	Yes

Table 22: Functional Requirement F-10

ID	F-10
Description	The chat interface shall have a send button.
Origin	U-3
Priority	5
Necessity	5
Verifiability	Yes

Table 23: Functional Requirement F-11

ID	F-11

Description	The chat interface shall have a send record button that while is pressed listens to the user and stops when it is released.
Origin	U-7
Priority	2
Necessity	3
Verifiability	Yes

Table 24: Functional Requirement F-12

ID	F-12
Description	The system shall show the user's questions on the right part of the chat.
Origin	U-3
Priority	4
Necessity	3
Verifiability	Yes

Table 25: Functional Requirement F-13

ID	F-13
Description	The system shall show its answers on the left part of the chat.
Origin	U-3
Priority	4
Necessity	3
Verifiability	Yes

Table 26: Functional Requirement F-14

ID	F-14
Description	The system shall have a button to activate and deactivate the voice of the system.
Origin	U-7

Priority	4
Necessity	3
Verifiability	Yes

Table 27: Functional Requirement F-15

ID	F-15
Description	The system shall have a button that shows the advanced options.
Origin	U-6
Priority	4
Necessity	4
Verifiability	Yes

Table 28: Functional Requirement F-16

ID	F-16
Description	The advanced options section shall allow the user to select among the models developed.
Origin	U-6
Priority	4
Necessity	4
Verifiability	Yes

Table 29: Functional Requirement F-17

ID	F-17
Description	The advanced options section shall allow the user the number of results to be returned.
Origin	U-6
Priority	2
Necessity	1

Verifiability	Yes

Table 30: Functional Requirement F-18

ID	F-18
Description	The advanced options shall include a table with the answer provided, the title of the article, the source link, the context passage and the score of the passage.
Origin	U-6
Priority	4
Necessity	4
Verifiability	Yes

Table 31: Functional Requirement F-19

ID	F-19
Description	The user shall be able to select the number of answers to be displayed at the table being an integer from 1 to 10.
Origin	U-6
Priority	2
Necessity	3
Verifiability	Yes

Table 32: Functional Requirement F-20

ID	F-20
Description	The system must include a button to change the interface between English and Spanish
Origin	U-6
Priority	2
Necessity	4
Verifiability	Yes

Table 33: Functional Requirement F-21

ID	F-21
Description	The system shall indicate in the message that it is fetching the answer and then change it to the actual answer.
Origin	U-3
Priority	4
Necessity	4
Verifiability	Yes

Table 34: Functional Requirement F-22

ID	F-22
Description	The system shall indicate in its message when an error occurred while fetching the answer.
Origin	U-3
Priority	4
Necessity	3
Verifiability	Yes

Table 35: Functional Requirement F-23

ID	F-23
Description	The interface shall transmit in JSON format to the prediction model the query, the reader model to use, the ranker model to use and the number of results.
Origin	U-1
Priority	4
Necessity	3
Verifiability	Yes

Table 36: Functional Requirement F-24

ID	F-24
Description	The inference model shall return to the interface a list of answers containing the answer, the title of the article, the context fragment, the link to the article and the score.
Origin	U-2
Priority	4
Necessity	4
Verifiability	Yes

3.3.3 Non-Functional Requirements

Table 37: Non-Functional Requirement NF-1

ID	NF-1
Description	The system shall be available as a web application.
Origin	U-8 U-9
Priority	4
Necessity	4
Verifiability	Yes

Table 38: Non-Functional Requirement NF-2

ID	NF-2
Description	The system shall be responsive and adaptable to all devices.
Origin	U-8 U-9
Priority	4
Necessity	4
Verifiability	Yes

Table 39: Non-Functional Requirement NF-3

ID	NF-3
Description	The machine learning model shall communicate with the interface with an API REST.
Origin	U-2
Priority	5
Necessity	5
Verifiability	Yes

Table 40: Non-Functional Requirement NF-4

ID	NF-4
Description	The API REST and the models shall be implemented in Python.
Origin	U-2
Priority	5
Necessity	5
Verifiability	Yes

Table 41: Non-Functional Requirement NF-5

ID	NF-5
Description	The application must communicate through HTTPS.
Origin	U-2
Priority	5
Necessity	5
Verifiability	Yes

Table 42: Non-Functional Requirement NF-6

ID	NF-6
Description	The platform shall be deployed using docker containers.

Origin	U-2
Priority	3
Necessity	5
Verifiability	Yes

Table 43: Non-Functional Requirement NF-7

ID	NF-7
Description	The interface shall be implemented using React.js
Origin	U-8 U-9
Priority	5
Necessity	2
Verifiability	Yes

Table 44: Non-Functional Requirement NF-8

ID	NF-8
Description	The transmission of data between model and interface shall be made through the POST HTTP method.
Origin	U-2
Priority	5
Necessity	5
Verifiability	Yes

3.3 Mockup

After the analysis of users and the elicitation of user requirements, a prototype of type High Visual and Low Functionality of the interface has been created. The tool used to create this prototype was Figma, which is a web-based vectorial graphics editing tool [78].

In Figure 11, the initial design of the homepage can be seen. This is the first thing that the user can

see. It contains the name of the application with a link to the Playground section, where the user can perform queries. Below, the user guide can be found, where the user can see explanations on every section of the page and hints on how to use them.

The navigation bar contains the links to the different sections of the web interface and also a flag that allows the user to switch the language of the interface from English to Spanish.

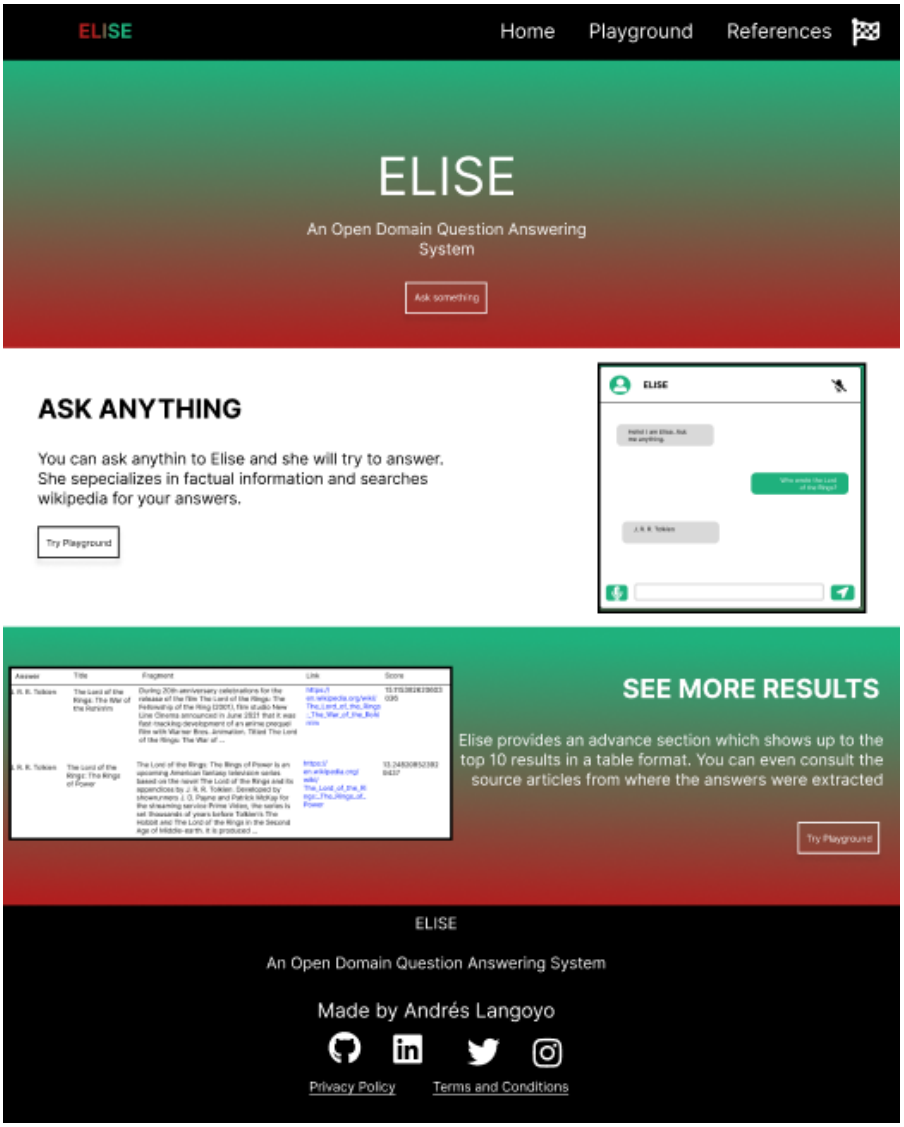


Figure 11: Mock-up design of the homepage with the tutorial.

Figure 12, shows the reference sections. Here there are listed the different resources used during the development of the project including a title, some explanation, and a link to the source.

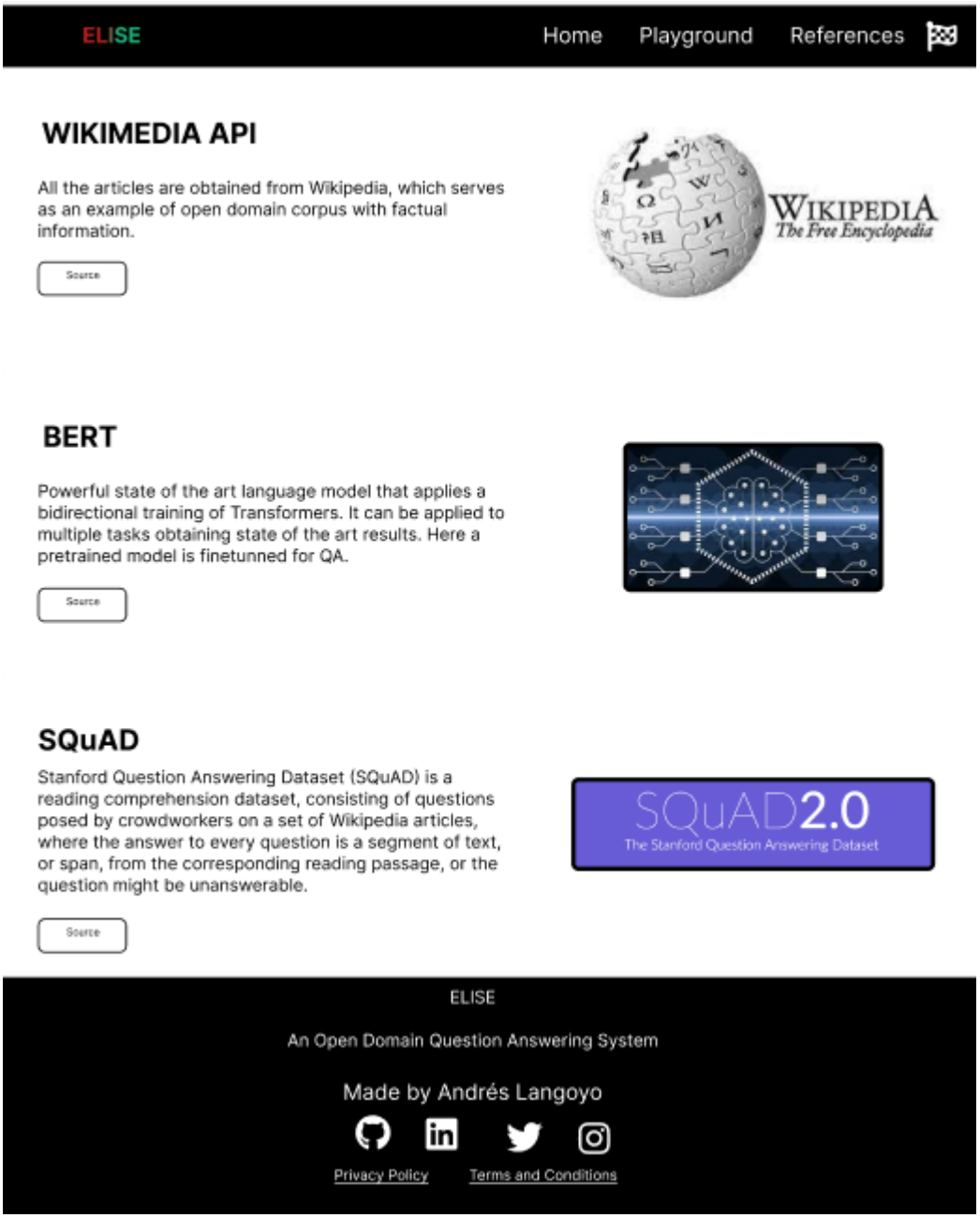


Figure 12: Mock-up design of the references section

Figure 13, shows the first part of the Playground section. The chat interface allows the user to type the questions on a text box. Also, there is a microphone button that while pressed synthesizes the user's words into the text box. Then with the send button the query is sent. The user questions appear on the right side of the chat box on green bubbles while the system response, on the left, with gray bubbles. On the top of the chat box there is a microphone button which can unmute the system and allow it to use text to speech to transmit the responses in audio format.

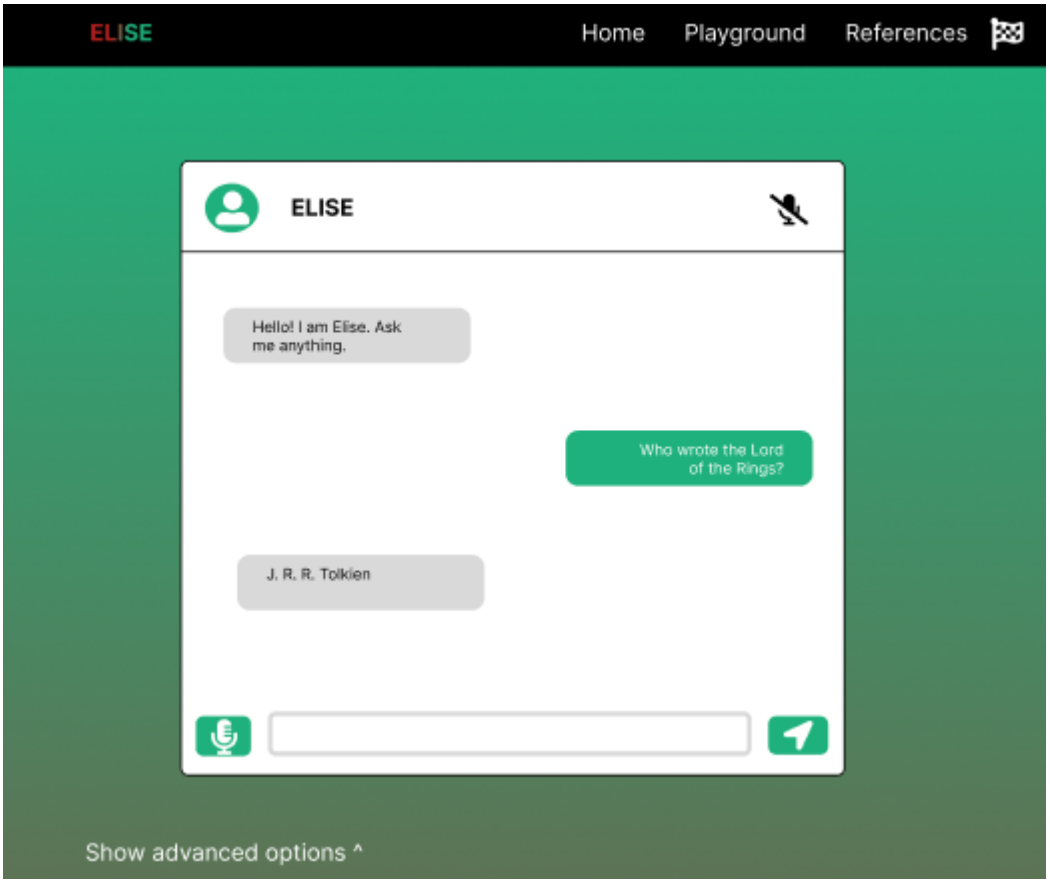


Figure 13: Mock-up page of the chat interface.

In Figure 14, the advanced section is shown, which appears once the user's actions the collapsible button. The form allows the user to select among reader and ranker models, the number of results to be displayed and also an input text box for the query. The results appear in a table format where the user can see the fragment where it comes from, a link to the article and also the score given to the answer.

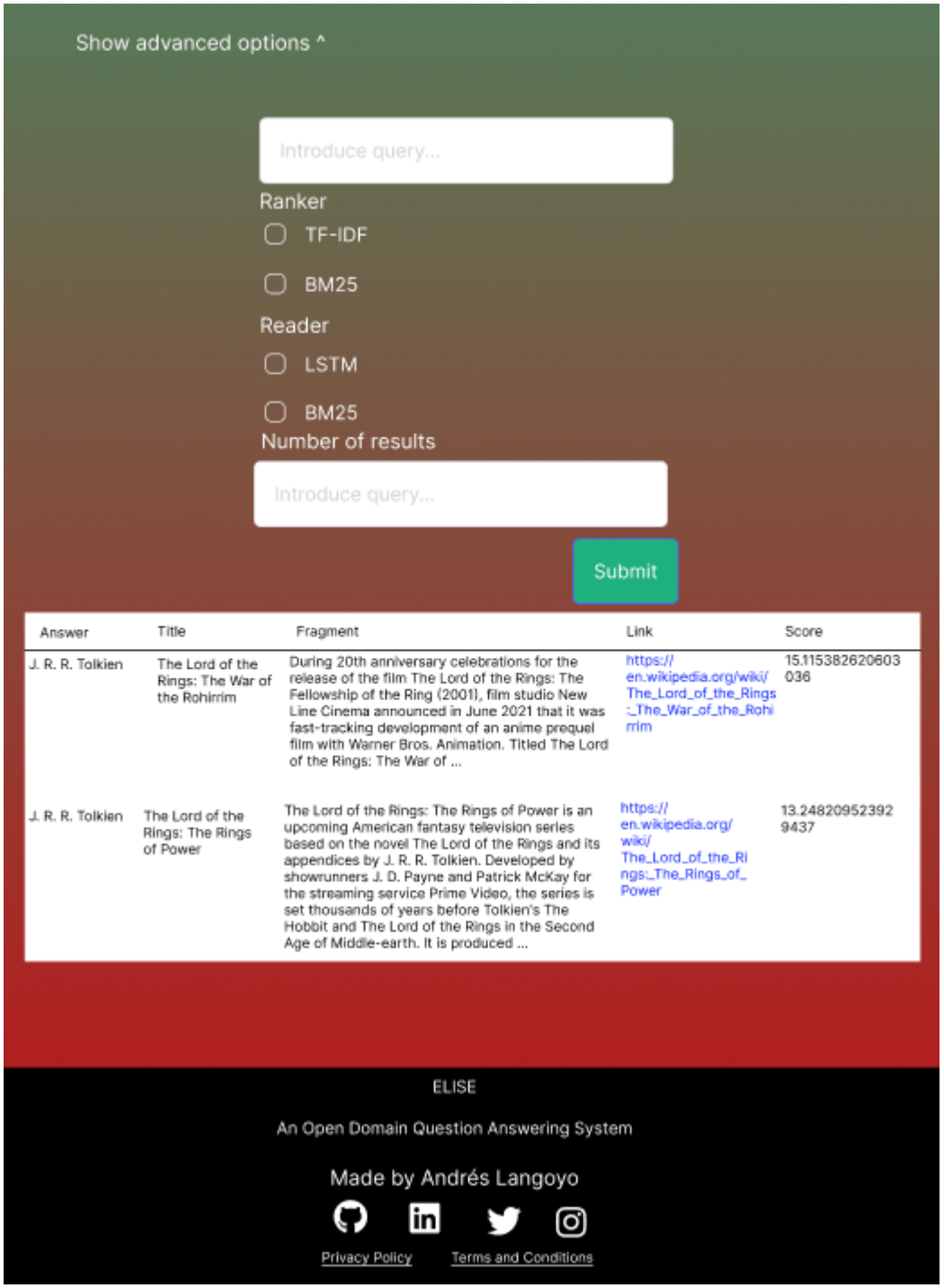


Figure 14: Mock-up page of the advanced section.

3.4 Use Cases

Use cases describe the actions the users can do in the platform giving a description of the user actions and the responses from the system. Two use cases can be identified in the current system that are related with the formulation of a query.

***UC-01*: User formulates a question in the chat**

- **Actors**: User
- **Purpose**: Allow the user to send the question with the chat interface.
- **Overview**: The user introduces a query in the interface and sends it. The system fetches the answer and presents it in the chatbox.

Typical Course of Events:

Table 45: Use Case 1

Actor's Action	System Response
1. The user types a question in the chat box text input and hits the send button.	2. As the user hasn't used the advanced option, the system defaults to the best model with 5 answers. The system displays the user's answer in the chatbox. It shows in a message that it is looking for the answer while computes the result.
	3. Once the system has the answer, it displays the answer span of the top result in the chat box replacing the wait message.

Alternative Route of Events:

- The user decides to use the speech synthesis and maintains the microphone button pressed while speaking.
- If an internal error in the model occurs, the system sends a message to the user notifying the error.

***UC-02*: User formulates a question in the chat**

- **Actors**: User
- **Purpose**: Allow the user to send the question with the chat interface.
- **Overview**: The user introduces a query in the interface and sends it. The system fetches the answer and presents it in the chatbox.

Typical Course of Events:

Table 46: Use Case 2

Actor's Action	System Response
1. The user presses the advanced option collapsible button.	2. The system shows the advanced input form and the results table.
3. The user writes the question and selects the models he desires with the number of results.	4. The system displays the user's answer in the chatbox. It shows in a message that it is looking for the answer while computes the result.
	5. Once the system has the answer, it displays the answer span of the top result in the chat box replacing the wait message. Also it displays the top results in the table.

Alternative Route of Events:

- If an internal error in the model occurs, the system sends a message to the user notifying the error.

3.5 Classes

In this section, a description of the classes of the system is included. The classes with their attributes are described as shown in the template.

Table 47: Class Template

Class name		CL-X
Attribute	Type	Description

Table 48: Class CL-1

CharFeatures		CL-1
char_to_idx	Dictionary	Maps the the vocabulary of characters to indexes
idx_to_char	Dictionary	Maps indexes to vocabulary of characters

Table 49: Class CL-2

LanguageID		CL-2
features	CharFeatures Object	Preprocesses text and creates unigram vocabulary
model	Pytorch Classification model	Model that predicts the language

Table 50: Class CL-3

Tagger		CL-3
features	Features object	Preprocesses text and creates unigram vocabulary
model	Pytorch tagger model	Model that predicts the tags of a given sentence

Table 51: Class CL-4

TaggerFeatures		CL-4
word_to_idx	Dictionary	Maps words from the vocabulary into indexes
tag_to_idx	Dictionary	Maps tags from the tagset into indexes
char_to_idx	Dictionary	Maps chars from the vocabulary into indexes
idx_to_tag	Dictionary	Maps index to tags from the tagset

Table 52: Class CL-5

QueryGenerator		CL-5
model	Tagger Object	Maps words from the vocabulary into indexes
relevant	list	contains a list with relevant POS tags

Table 53: Class CL-6

DocumentExtractor		CL-6
wikipedia	Wikimedia API object	Object that serves to make calls to the Wikimedia API

Table 54: Class CL-7

PassageRanker		CL-7
model	TF-IDF, Bert Embeddings, Word2Vec, SentenceTransformers	Depending on the user choice it can be several ranker models

Table 55: Class CL-8

TF-IDF		CL-8
unigram	Dictionary	Vocabulary of words to perform TF-IDF or BM25
b	Number	Parameter in BM25 algorithm
k	Number	Parameter in BM25 algorithm
corpus	List	Set of texts introduced to create the vocabulary
word_count	Dictionary	Has the occurrences of each word in the vocabulary

Table 56: Class CL-9

Word2Vec		CL-9
model	Word2Vec object model	Custom trained Word2Vec model using gensim

Table 57: Class CL-10

BERTEmbeddings		CL-10
model	BertModel or LSTMReader object	Maps words from the vocabulary into indexes
tokenizer	Tokenizer object	Tokenizes inputs to be passed into the BERT model

Table 58: Class CL-11

DocumentReader		CL-11
model	BertModel or LSTMReader object	Maps words from the vocabulary into indexes
tokenizer	Tokenizer object	Tokenizes inputs to be passed into the BERT model

3.6 Operation Contracts

In this section the methods included in each of the classes are specified according to the following format.

Table 59: Contract Template

CN-X	
Class	class where it belongs
Name:	name_of_method(parameters)
Responsibilities:	Description of what are the tasks the method performs.
Pre-conditions:	Specification of how the state of the class must be and the format of the parameters when the method is invoked.
Post-conditions:	Specification of how the state of the class results and what arguments are returned after the method finishes its execution.

Table 60: Contract CN-1

CN-1	
Class	CharFeatures
Name:	fit(text_set)
Responsibilities:	Creates a vocabulary of characters out of the corpus of texts given.
Pre-conditions:	It must be given a list of tokenized sentences.
Post-conditions:	It creates a mapping between words and integers.

Table 61: Contract CN-2

CN-2	
Class	CharFeatures
Name:	transform(text)
Responsibilities:	Transforms a sentence tokenized into a bag-of-words vector.
Pre-conditions:	It must be given a tokenized string.
Post-conditions:	It returns a vector of the dimension of the vocabulary with the frequencies of the words in the string that are also in the vocabulary.

Table 62: Contract CN-3

CN-3	
Class	CharFeatures
Name:	save(fiflePath)
Responsibilities:	Saves the file into a pickle file so that the vocabulary can be loaded for inference.
Pre-conditions:	The vocabulary must have been created using the fit method.
Post-conditions:	It saves the class into a pickle file in the specified directory.

Table 63: Contract CN-4

CN-4	
Class	CharFeatures
Name:	load(fiflePath)
Responsibilities:	Loads the file into a pickle file so that the vocabulary can be loaded for inference.
Pre-conditions:	The vocabulary must have been previously saved into a pickle file.
Post-conditions:	It loads into memory the class instance with the saved vocabulary.

Table 64: Contract CN-5

CN-5	
Class	LanguageID
Name:	constructor( )
Responsibilities:	Creates an CharFeatures class and loads the vocabulary file. It also creates the classification model.
Pre-conditions:	The vocabulary and the classification model must have been previously saved.
Post-conditions:	The vocabulary and the model are loaded into memory.

Table 65: Contract CN-6

CN-6	
Class	LanguageID
Name:	predict(query)
Responsibilities:	It predicts whether the input query is in English or Spanish.
Pre-conditions:	The input parameter is in string format.
Post-conditions:	Returns ‘es’ or ‘en’ if the input is in Spanish or English.

Table 66: Contract CN-7

CN-7	
Class	LanguageID
Name:	predict(query)
Responsibilities:	It predicts whether the input query is in English or Spanish.
Pre-conditions:	The input parameter is in string format.
Post-conditions:	Returns ‘es’ or ‘en’ if the input is in Spanish or English.

Table 67: Contract CN-8

CN-8	
Class	QueryGenerator
Name:	constructor(language)
Responsibilities:	Loads a model based on the language to perform POS tagging.
Pre-conditions:	Language is either ‘en’ or ‘es’.
Post-conditions:	Initializes the attribute model.

Table 68: Contract CN-9

CN-9	
Class	QueryGenerator
Name:	generate_query(query)
Responsibilities:	Obtains the tags predicted from the query tokens.
Pre-conditions:	The input query must be tokenized.
Post-conditions:	Returns a list with the keywords to search based on the relevant tags.

Table 69: Contract CN-10

CN-10	
Class	DocumentExtractor
Name:	constructor(language)
Responsibilities:	Initializes the wikipedia library in the language specified.
Pre-conditions:	Language is either ‘en’ or ‘es’.
Post-conditions:	Initializes a wikipedia object.

Table 70: Contract CN-11

CN-11	
Class	DocumentExtractor
Name:	getDocuments(queries)
Responsibilities:	Uses the wikipedia API to search for relevant documents.
Pre-conditions:	Receives a list of keywords to search in wikipedia.
Post-conditions:	Returns the documents as a dictionary with three lists with keys ‘titles’, ‘contents’ and ‘urls’

Table 71: Contract CN-12

CN-12	
Class	DocumentExtractor
Name:	getDocuments(queries)
Responsibilities:	Uses the wikipedia API to search for relevant documents.
Pre-conditions:	Receives a list of keywords to search in wikipedia.
Post-conditions:	Returns the documents as a dictionary with three lists with keys ‘titles’, ‘contents’ and ‘urls’

Table 72: Contract CN-13

CN-13	
Class	PassageRanker
Name:	constructor(language, model)
Responsibilities:	Initializes the model to be used.
Pre-conditions:	Language can have values ‘en’ or ‘es’. Model can have values ‘ST’, ‘BERT’, ‘TF-IDF’ or ‘BM25’
Post-conditions:	Initializes the required model.

Table 73: Contract CN-14

CN-14	
Class	PassageRanker
Name:	text_splitter(text, n)
Responsibilities:	Splits a document into passages of n words to avoid Reader errors with the number of tokens.
Pre-conditions:	n is a positive integer and text is a tokenized string.
Post-conditions:	Returns a list of strings of the desired size.

Table 74: Contract CN-15

CN-15	
Class	ResultsCreator
Name:	results_creator(library)
Responsibilities:	Creates the final results data structure which contains score, title, content of the passage and url. It also returns the list with all passages created.
Pre-conditions:	library is composed of the documents retrived from the

	DocumentExtractor.
Post-conditions:	Returns final results data structure which contains score, title, content of the passage and url. It also returns the list with all passages created.

Table 75: Contract CN-16

CN-16	
Class	ResultsCreator
Name:	rank(library, query)
Responsibilities:	Creates the vector representation of the passages and ranks them using cosine similarity metric.
Pre-conditions:	library is composed of the documents retrieved from the DocumentExtractor and query is the original query introduced by the user.
Post-conditions:	Returns a list of tuples with the shape (score, title, context, url) ordered by score from highest to lowest.

Table 76: Contract CN-17

CN-17	
Class	ResultsCreator
Name:	cosine_similarity(doc1, doc2)
Responsibilities:	Computes the cosine similarity metric between two vectors.
Pre-conditions:	Both documents must be represented by a float type vector of the same dimension.
Post-conditions:	A real numeric score.

Table 77: Contract CN-18

CN-18	
Class	DocumentReader

Name:	constructor(language, model)
Responsibilities:	Initializes the model and the BERT tokenizer in case of being BERT.
Pre-conditions:	language must be ‘en’ or ‘es’ and the model must be ‘BERT’ or ‘LSTM’
Post-conditions:	Initializes the model and the BERT tokenizer in case of being BERT.

Table 78: Contract CN-19

CN-19	
Class	DocumentReader
Name:	answer_question(question, context_text)
Responsibilities:	Computes the answer to the question on the context_text applying the corresponding method.
Pre-conditions:	question and context_text must be strings.
Post-conditions:	Returns a string as an answer

Table 79: Contract CN-20

CN-20	
Class	DocumentReader
Name:	answer(question, results,n_results)
Responsibilities:	Computes the answer to the top n_resutls questions and contexts in results.
Pre-conditions:	question is a string with the original query. results is a list of tuples of the form (score, title, context, url) n_results is an integer between 1 and 10
Post-conditions:	Returns a list of Dictionaries with the keys ‘score’ ‘title’, ‘content’, ‘url’, ‘answer’

3.7 Sequence Diagram

This section presents a sequence diagram related to the two use cases specified in prior sections. It exemplifies the flow of function calls that the various classes perform when a query is being

processed.

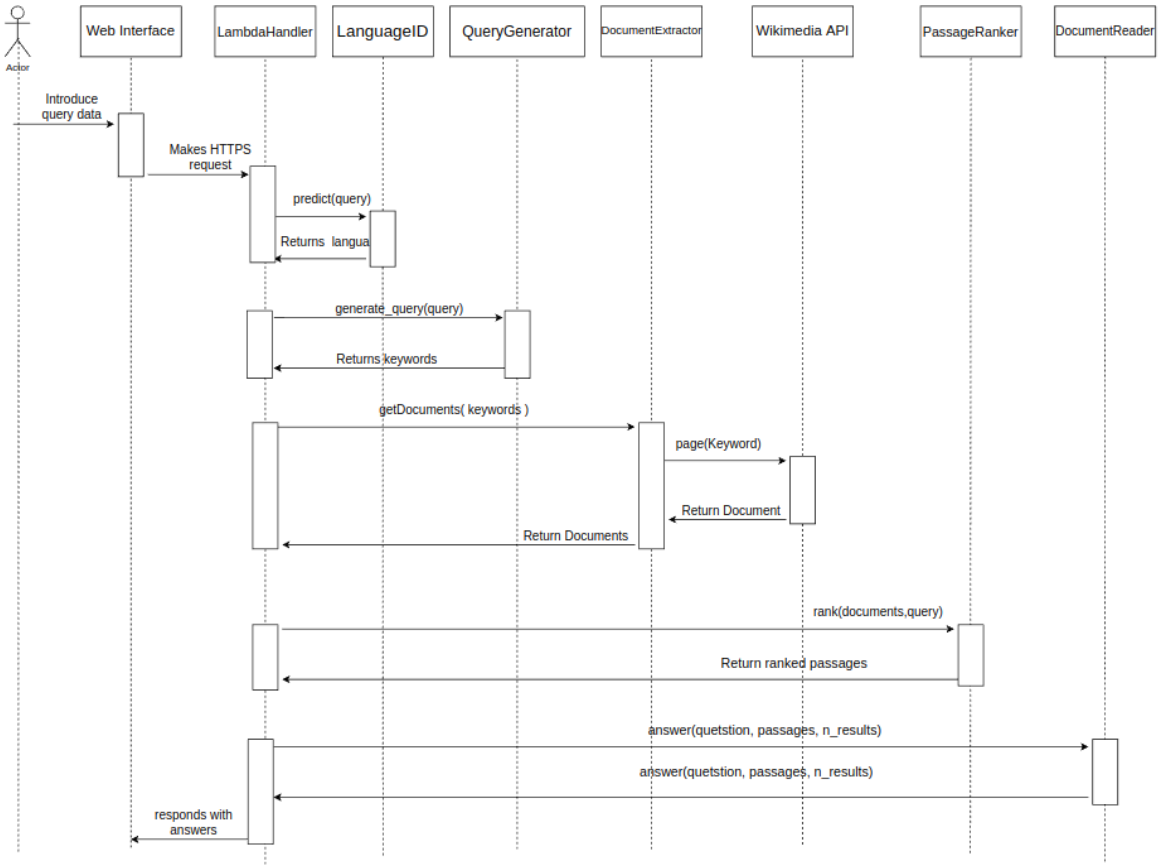


Figure 15: Sequence Diagram.

4. DESIGN AND TRAINING PROCESS

This chapter is devoted to the explanation of how the Question Answering system is constructed. First, an introduction on the whole inference pipeline is made. Later, all the elements are described one by one describing the model architecture, the data used and the training process with its validation.

4.1 Inference Pipeline

To obtain a set of answers from the query the user introduces, there are several components involved. These can be seen in Figure 16.

The modules involved are

- **Language Identifier:** As the system is supposed to support two languages, Spanish and English, there must be a module that identifies when a query is in either of those languages.
- **Part Of Speech Tagger:** To search for relevant texts that could contain the answers, it is necessary to extract keywords from the query. This can be done by identifying grammatical

information of words and eliminating the ones that are less relevant like articles, pronouns or punctuation.

- **Repository of documents:** It is necessary to contain a source from where documents can be extracted. In this case wikipedia is chosen because it has constantly updated, generic and factual information documents which serve the purpose of Open Domain Question Answering. However, it could be substituted for other applications.
- **Ranker:** This module is in charge of fragmenting the documents and ranking them by likelihood of containing the answer.
- **Reader:** The purpose of the reader is to be able to extract the answer to a question from a fragment of text that contains that answer.

In the following sections, it is explained how each of the modules are developed.

4.2 Language Identifier

The language identifier is the model with the purpose of identifying if an input query is in one of either English or Spanish, the two languages which the system must support. In this section, it is described the data, model and experiments performed to obtain the best possible model.

4.2.1 Data Preprocessing

The data used to train the language identifier was extracted from a public dataset from kaggle [38]. The dataset contains a total of 10337 sentences from a total of 17 languages. An example of some entries in the dataset is:

	Text	Language
0	Nature, in the broadest sense, is the natural...	English
1	"Nature" can refer to the phenomena of the phy...	English
2	The study of nature is a large, if not the onl...	English
3	Although humans are part of nature, human acti...	English
4	[1] The word nature is borrowed from the Old F...	English

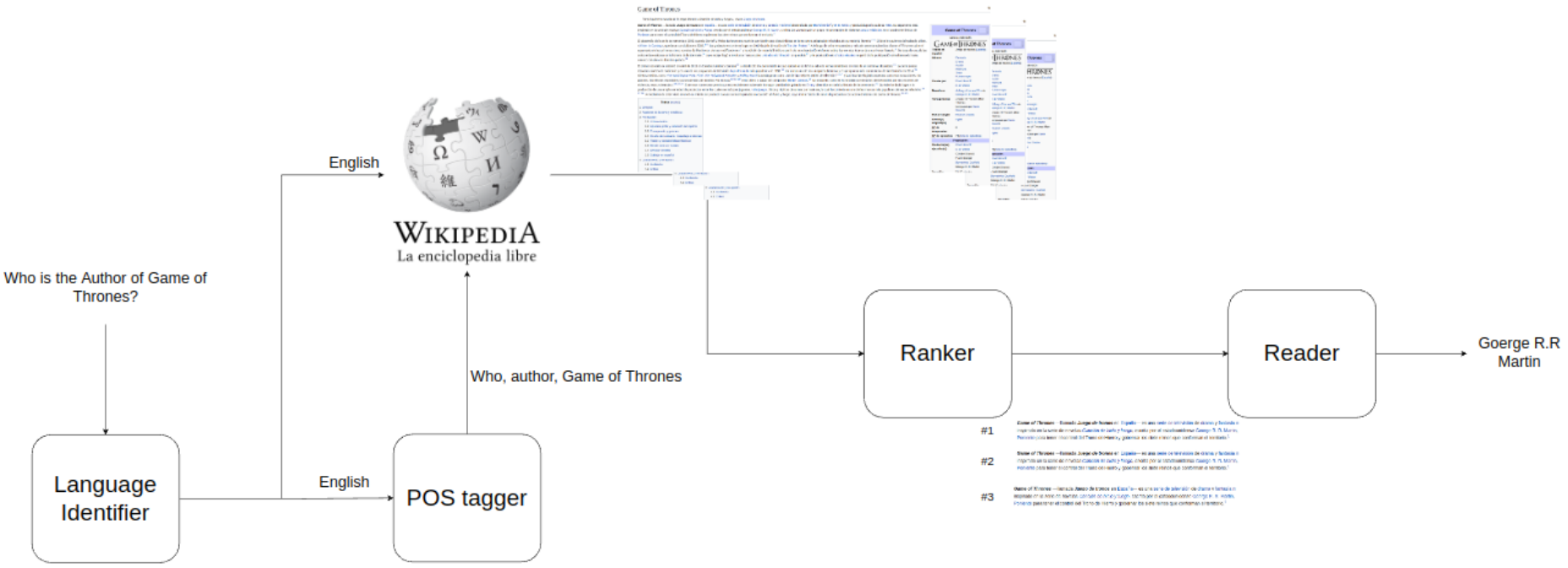


Figure 16: Inference Pipeline

Also it was used another dataset which contained documents in French, Spanish and English [79] from sources like Wikipedia, conference papers, Amazon Reviews, JRC-Acquis and Europarl.

During the preprocessing stage, for the first dataset, the sentences from other languages were filtered. That left a total of 2204 sentences in English and Spanish. The second dataset contained documents formatted in sentences splitted every line. This is how one document would look like:

```
i read this book because in my town, everyone uses it and order.
this is my pharmacist who advised me she was so thin i asked her
what she had done and instead of just selling snake oil capsules,
she advised me this book to 5 euros.
of course, we must make an effort to lose 25 pounds but with the
book, i had a companion.
the author was able to talk to me just with strong arguments and
above all i felt he knew many cases like mine.
```

After extracting all the sentences from the second dataset and merging it with the first one,
there was obtained a dataset with 1,956,462 values. The new dataset was highly unbalanced with counts of 1,586,621 English sentences and 369,841 Spanish sentences. Then, it extracted a random subset of 600000 sentences which left the counts as 250102 for English and 249898 for Spanish.

Several articles like [80] and [81] suggest that the frequency distribution of the letters varies across languages and can be used to identify them.

For that reason, here it is chosen a character-level representation for the sentences in the dataset. It is a bag-of-words like representation in which each sentence is converted into a vector of the size of the vocabulary of characters found. Each index of the vector represents a character of this vocabulary and the value is the frequency observed in that sentence. This, if obtains a good accuracy in performance, is a more lightweight representation than it would be a word-based representation, especially, with such a large number of sentences in the dataset, as the vocabulary would rapidly grow.

In this case, punctuation symbols and uppercase letters were maintained as the use of some punctuation like '¿' differ and could be a determinating factor.

The dataset is also divided into three subsections, one for training the model, one for validating the model while training and observing its progress and one for testing models and reporting values. The percentage split is 80%, 10% and 10%. The final model is then trained with all the data.

Lastly, pytorch data loaders were created to wrap the data in order to begin the training process.

4.2.2 Model architecture

For this model, a multilayer perceptron architecture was envisioned to be able to classify the sentences. The multilayer perceptron had an entry layer and an output layer with several hidden layers in between. The hidden and input layers had a ReLU activation function while the output layer had a softmax activation function. This is because an output from zero to one was desired 0 representing English and 1 Spanish with a boundary decision value of 0,5. In Figure 17, the model architecture can be seen.

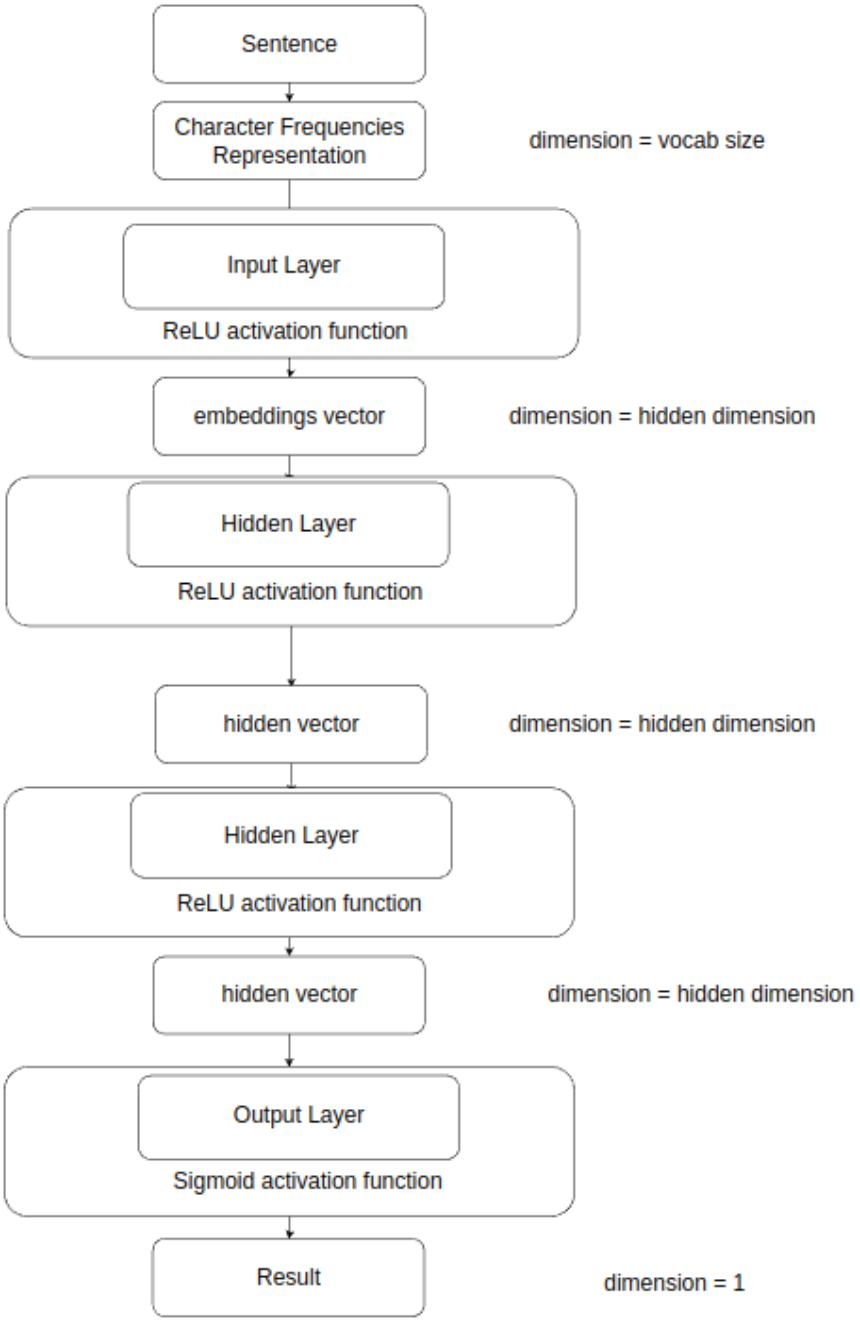


Figure 17: Language Identifier best model

4.2.3 Experiments and Results

The experiments are aimed at finding the best architecture and the inner structure of the multilayer perceptron was modified both in hidden dimension and number of hidden layers. The metrics used to test the performance are the accuracy and the F1-score.

The number of epochs and the learning rate were adjusted for each experiment. They were stalled when the validation loss stagnated and the training started to increase. The loss function used was binary cross entropy loss and the optimizer was stochastic gradient descent.

Table 80: Experiments Language Identifier

Hidden layers	Hidden dimensions	Val Accuracy	Val F1	Test Accuracy	Test F1
1	8 - 8	0.9789	0.9679	0.9679	0.9679
1	16 - 16	0.9683	0.9683	0.9683	0.9683
1	32 - 32	0.9648	0.9648	0.9642	0.9642
1	64 - 64	0,9682	0,9682	0,9682	0,9682
1	128 - 128	0.9642	0.9642	0,9655	0.9667
1	512 512	0.9676	0.9676	0.9671	0.9671
2	8 -8 -8	0.9669	0.9669	0.9669	0.9667
2	16-8-16	0.9672	0.9672	0.9668	0.9668
2	16 16 16	0.9665	0.9665	0.9662	0.9662
2	32 16 32	0.9672	0.9672	0.9662	0.9662
2	32 32 32	0.9673	0.9672	0.9668	0.9668
2	64 64 64	0.9673	0.9673	0.9670	0.9670
2	256 256 256	0.9683	0.9683	0.9681	0.9681
2	**512 512 512**	**0.9769**	**0.9769**	**0.9752**	**0.9752**
3	64 64 64 64	0.9688	0.9688	0.9687	0.9687
3	128 128 128 128	0.9691	0.9690	0.9685	0.9685
3	512 512 512 512	0.9711	0.9711	0.9705	0.9705

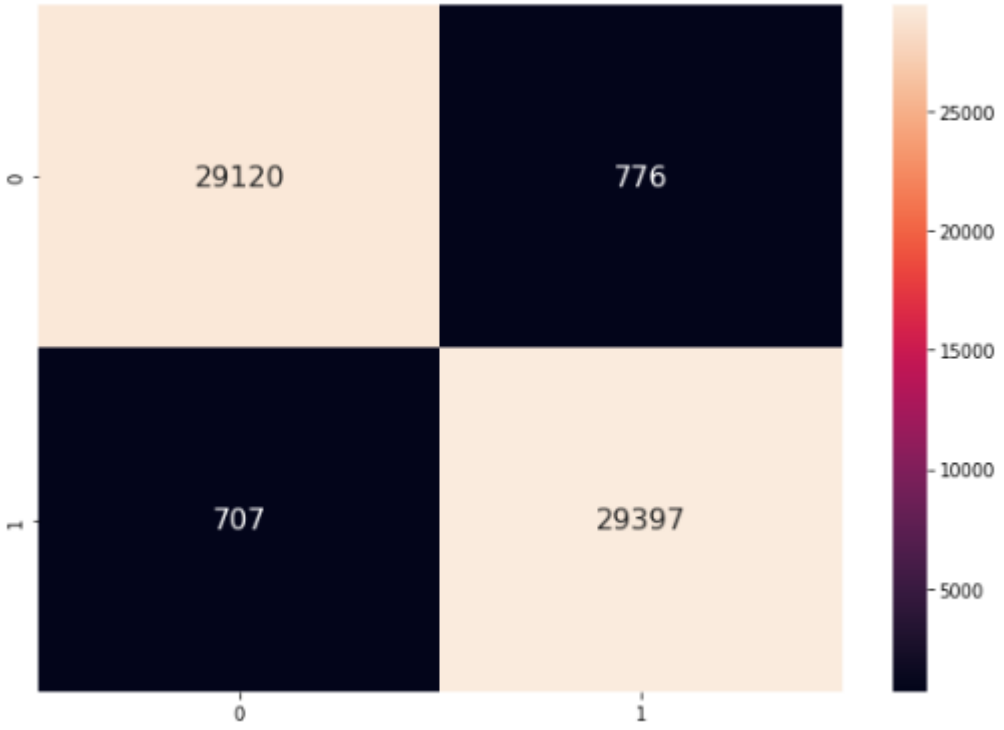


Figure 18: Confusion Matrix from best language identifier experiment

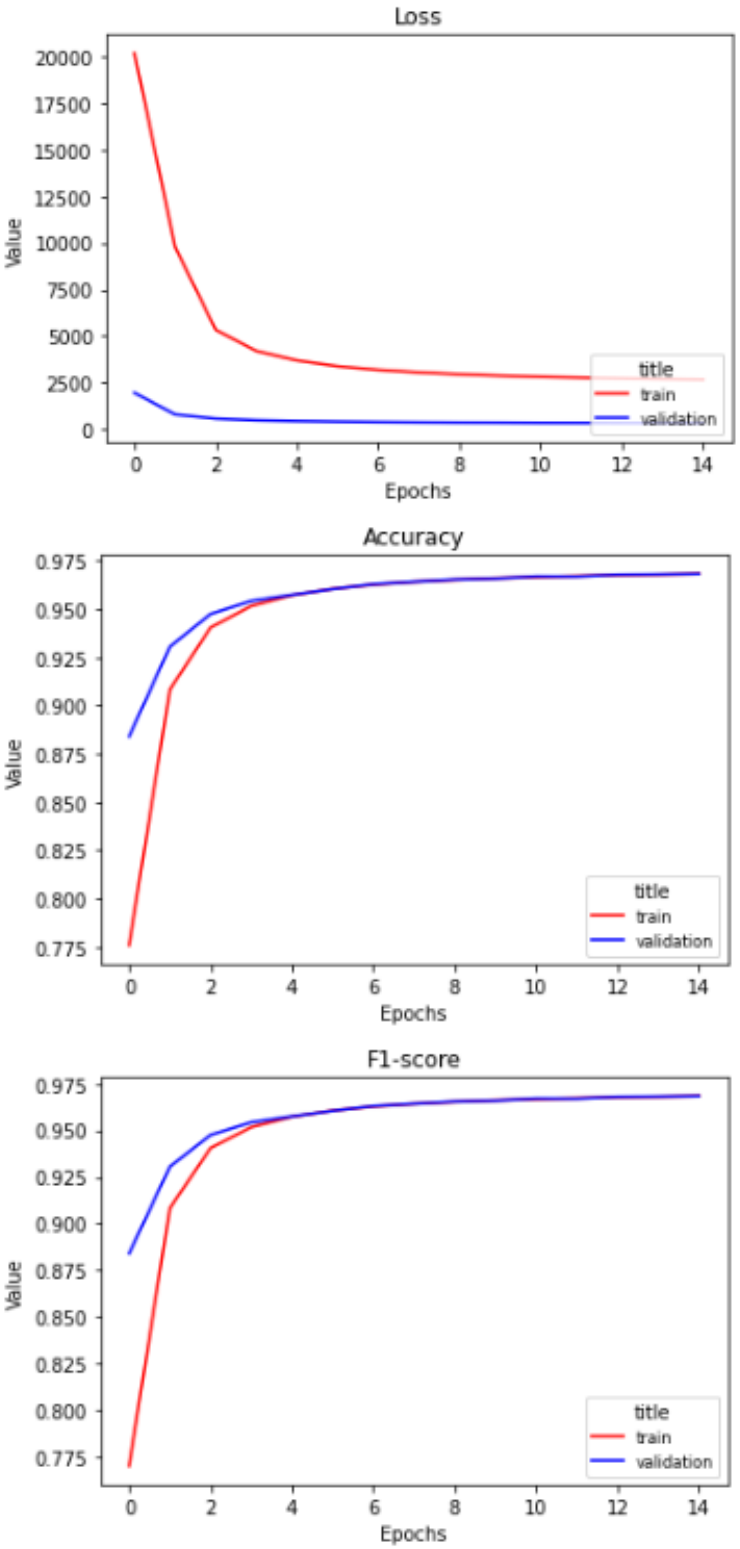


Figure 19: Training progress from best language identifier experiment

Figure 18 shows the confusion matrix of the best experiment while Figure 19 shows the training process comparing the validation and training losses. The best model obtained was a model with two hidden layers and a hidden size value of 512 between layers.

4.3 POS Tagger

The language identifier is the model with the purpose of identifying the words that appear in the sentence with the purpose of eliminating the non relevant ones to perform the query. In this section, it is described the data, model and experiments performed to obtain the best possible model.

4.3.1 Data Preprocessing

The proper dataset to perform POS tagging would have been the Penn Treebank dataset [37]. The problem is that it doesn't have free access and the licenses are very expensive. As an alternative, the library NLTK has some reduced versions of the same dataset. The original Penn Treebank dataset has a tree structure of sentences and tags are introduced at different levels:

```
( (SBARQ
        (INTJ (UH So) )
        (WHNP-1
          (WHADJP (WRB how) (JJ many) )
          (, ,)
          (INTJ (UH um) )
          (, ,) (NN credit) (NNS cards) )
        (SQ (VBP do)
          (NP-SBJ (PRP you) )
          (VP (VB have)
            (NP (-NONE- *T*-1) )))
        (. ?) (-DFL- E_S) ))
```

The simplified NLTK dataset, instead of a tree structure it simply contains tagged sentences, with tuples of tokenized words and tags:

[('El', 'DET'), ('grupo', 'NOUN'), ('estatal', 'ADJ'), ('Electricité_de_France', 'NOUN'), ('-Fpa-', '.'), ('EDF', 'NOUN'), ('-Fpt-', '.'), ('anunció', 'VERB'), ('hoy', 'ADV'), (',', '.'), ('jueves', 'X'), (',', '.'), ('la', 'DET'), ('compra', 'NOUN'), ('del', 'ADP'), ('51_por_ciento', 'NUM'), ('de', 'ADP'), ('la', 'DET'), ('empresa', 'NOUN'), ('mexicana', 'ADJ'), ('Electricidad_Águila_de_Altamira', 'NOUN'), ('-Fpa-', '.'), ('EAA', 'NOUN'), ('-Fpt-', '.'), (',', '.'), ('creada', 'ADJ'), ('por', 'ADP'), ('el', 'DET'), ('japonés', 'ADJ'), ('Mitsubishi_Corporation', 'NOUN'), ('para', 'ADP'), ('poner_en_marcha', 'VERB'), ('una', 'DET'), ('central', 'NOUN'), ('de', 'ADP'), ('gas', 'NOUN'), ('de', 'ADP'), ('495', 'X'), ('megavatios', 'NOUN'), ('.', '.')]

Also, the tags are greatly simplified. The number of tags considered are listed in Table 81.

Table 81: Simplified tagset

ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary
CCONJ	coordinating conjunction
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	other

From the whole set of documents, the tags considered relevant are proper nouns, nouns, verbs, adjectives, numerals and adverbs.

The size of the dataset is a limiting factor as there are only 3914 tagged sentences in English and 6030 in Spanish. This is limiting because it might not be enough general to perform well in deployment.

In order to process the data, all the words, characters and tags are registered and a vocabulary is made for each of them. To handle out of vocabulary words or characters a special token ‘UNK’ is used. This vocabulary is used to map words into numbers.

The dataset is again split into three sets: train, validation and test. Also with the same 80%, 10% and 10% split.

4.3.2 Model Architecture

Regarding models, two of them were attempted. Initially, some inspiration was used from a Pytorch tutorial to sequence processing [82]. This model is based on recurrent neural networks.
First, it contains an embedding layer. This layer condenses the information of the vocabulary into a

denser representation which takes into account similarity of words and is trained alongside the rest of the model. As a result, a sentence is introduced, it gives a new vector representation for each word.

Then, a bidirectional LSTM layer is introduced which takes the output of the embedding layer. A recurrent neural network is chosen because it is necessary to learn some sequence dependencies as the tag of a word could be influenced by the neighboring words.

Finally, the output of the LSTM layer is introduced into a linear layer, which has the output dimension of the tagset. A softmax layer takes this output and gives the probabilities of each tag for every word.

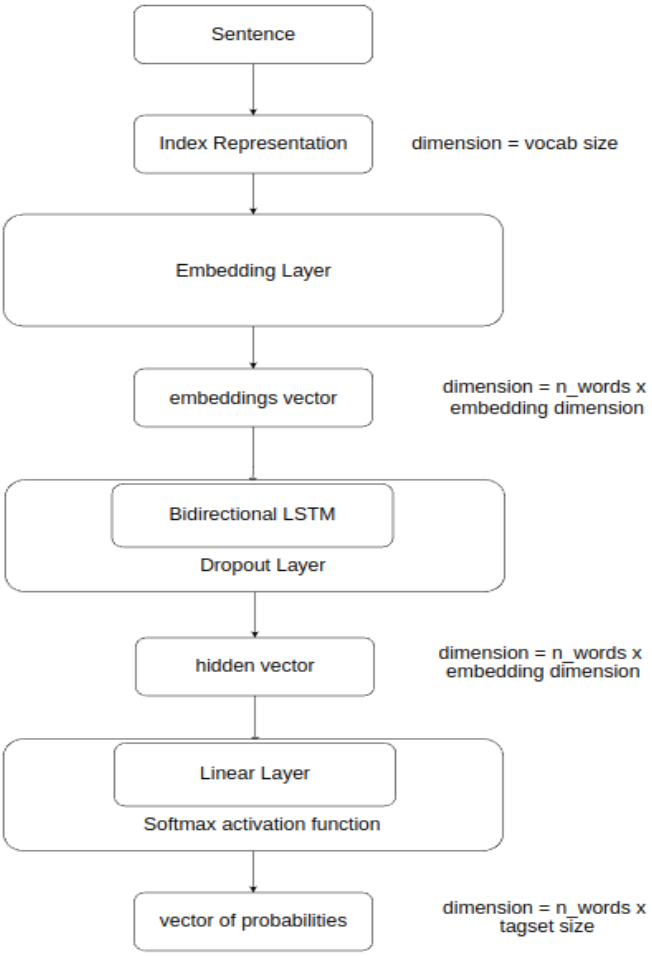


Figure 20: Diagram of word level POS tagger.

The second model is an extension of the first one. The difference is that it contains character level information, which could be useful to identify certain kinds of words. For instance, this is the case of

adverbs in English, which end in '-ly'.

To do so, an extra embedding layer and LSTM layer is used for characters. When an input sentence is presented to the model, it obtains the embeddings for each word as it did before. Now, however, it also passes the characters of each word through the character embedding layer and, then, through the character LSTM. The hidden output of the LSTM could serve as a character embedding as it condenses the information of all the characters of the word. The original word embedding is concatenated with the new generated hidden vector.
Then, the concatenated vector is passed through the main LSTM layer, the linear layer and the softmax layer.

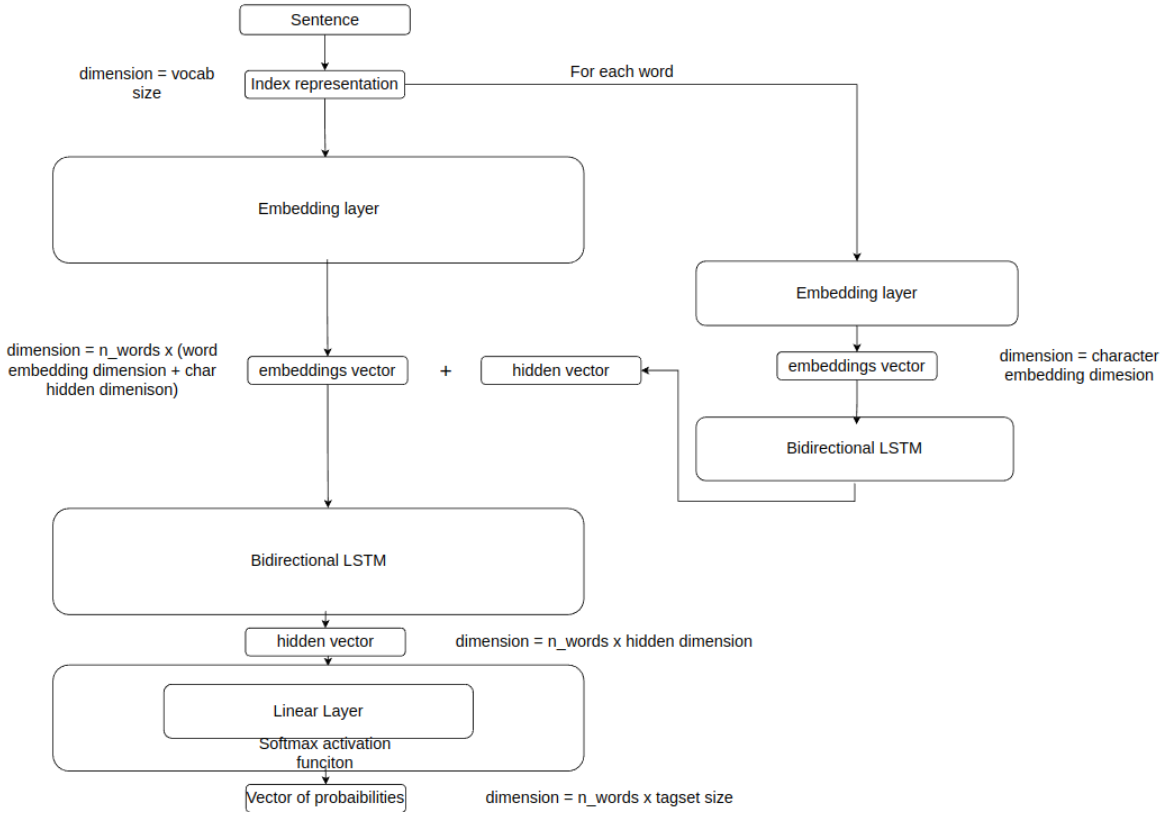


Figure 21: Diagram of word and character level POS tagger

4.3.3 Experiments and Results

To perform the experimentation, the target hyperparameters are the embedding dimensions and the hidden dimensions. The loss function used is Cross Entropy Loss and the optimizer is stochastic gradient descent.
The learning rate and the number of epochs are adjusted in every experiment. The experiment is stopped when overfitting starts to happen, when the validation loss stagnates and the training loss continues decreasing.

It is tested in both Spanish and English datasets and the metrics used for comparison are accuracy and F1-score.

Table 82: Experiments POS Tagger English

Model	Embedding Dimensions	Hidden Dimensions	Val Accuracy	Val F1	Test Accuracy	Test F1
word	32	32	0.9035	0.9031	0.8934	0.8928
word	64	64	0.9220	0.9215	0.9170	0.9172
word	128	128	0.9231	0.9225	0.9207	0.9198
word	256	256	0.9355	0.9355		
word	256	256	0.9488	0.9486	0.9461	0.9465
word+ chars	w:128 c:32	w:128 c:32	0.9496	0.9497	0.9439	0.9442
word+ chars	w:128 c:64	w:128 c:64	0.9510	0.9508	0.9388	0.9395
word+ chars	**w:128 c:128**	**w:128 c:128**	**0.9580**	**0.9585**	**0.9477**	**0.9474**
word+ chars	w:512 c:64	w:512 c:64	0.9963	0.9547	0.9465	0.9465
word+ chars	w:1024 c:128	w:1024 c:128	0.9499	0.9500	0.9383	0.9383
word+ chars	w:1024 c:64	w:1024 c:64	0.9511	0.9507	0.9423	0.9421
word+ chars	w:400 c:60	w:256 c:128	0.9487	0.9485	0.9415	0.9416

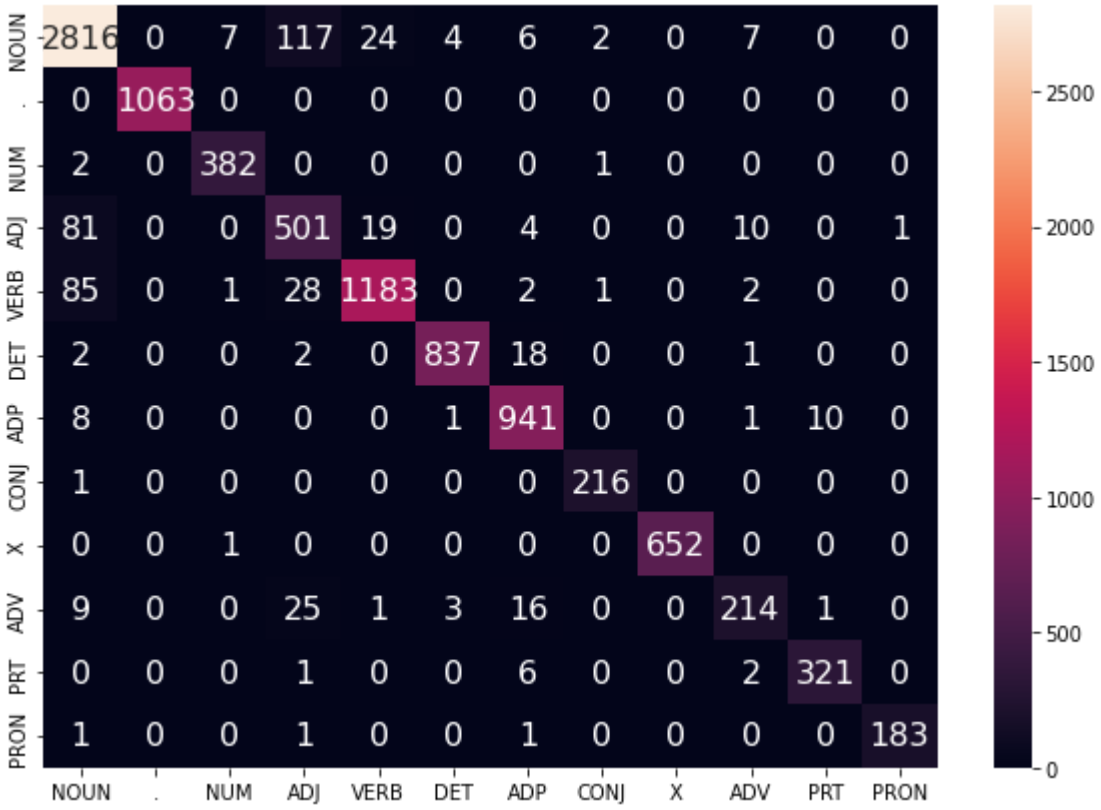


Figure 22: Confusion matrix English POS tagger.

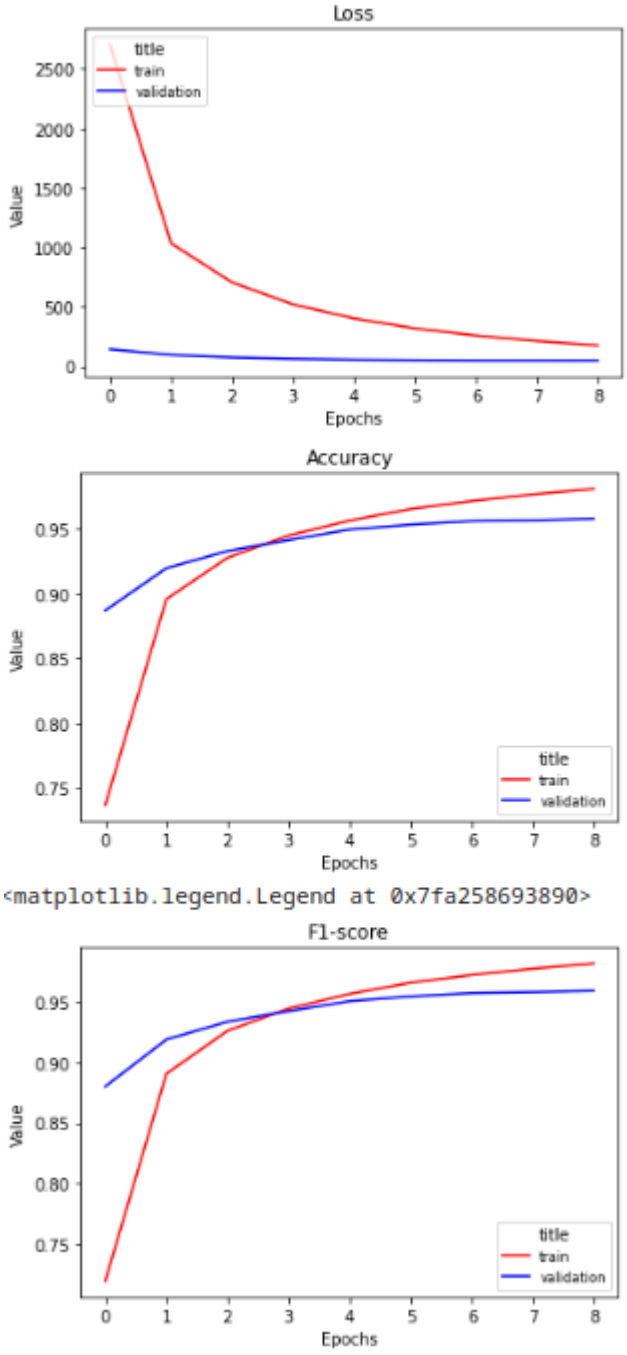


Figure 23: Training progress from English POS tagger.

Table 83: Experiments POS Tagger Spanish

Model	Embedding Dimensions	Hidden Dimensions	Val Accuracy	Val F1	Test Accuracy	Test F1
word	32	32	0.8772	0.8761	0.8737	0.8737
word	64	64	0.8939	0.8934	0.8885	0.8886
word	128	128	0.8873	0.8849	0.8821	0.8836
word	256	256	0.9060	0.9055	0.8993	0.8998
word	512	512	0.9160	0.9156	0.9101	0.9089
word	1024	1024	0.9202	0.9199	0.9114	0.9110
word+chars	w:128 c:64	w:128 c:64	0.9221	0.9215	0.9174	0.9166
word+chars	w:128 c:64	w:128 c:64	0.9419	0.9413	0.9294	0.9295
word+chars	w:300 c:100	w:1024 c:64	0.9385	0.9382	0.9289	0.9282
word+chars	**w:300 c:100**	**w:1024 c:128**	**0.9417**	**0.9412**	**0.9313**	**0.9303**
word+chars	w:300 c:100	w:1024 c:256	0.9511	0.9507	0.9257	0.9244

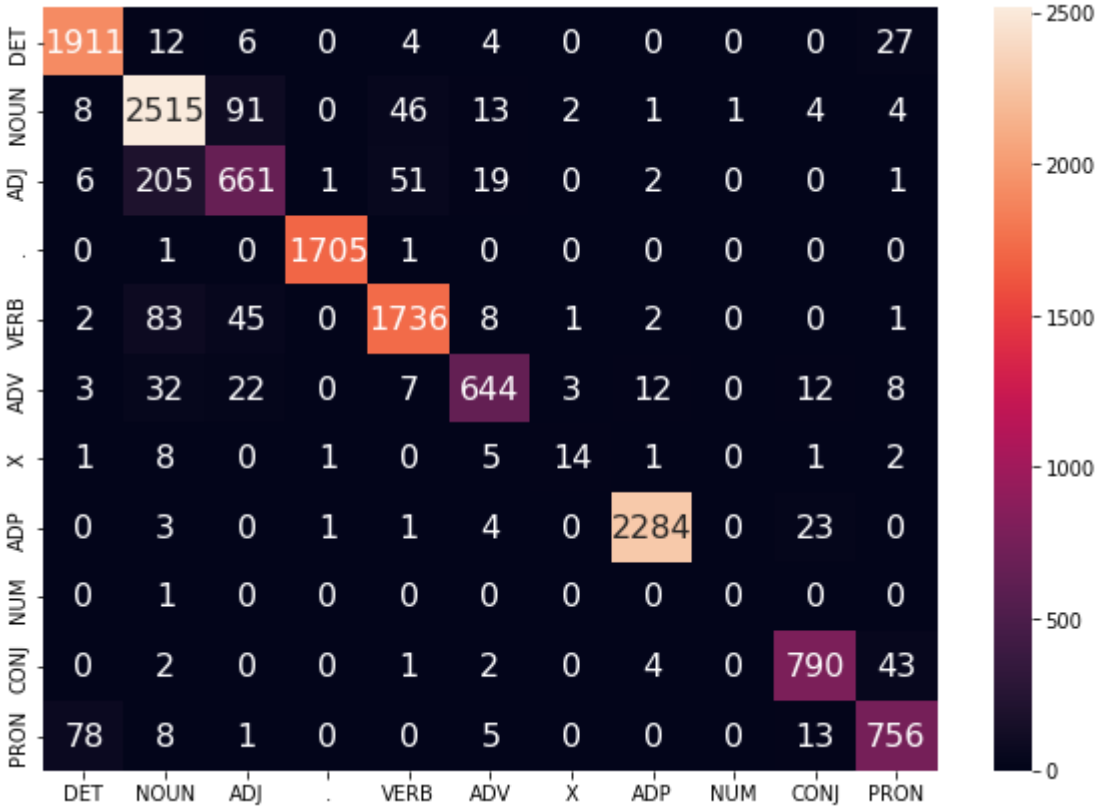


Figure 24: Confusion matrix Spanish

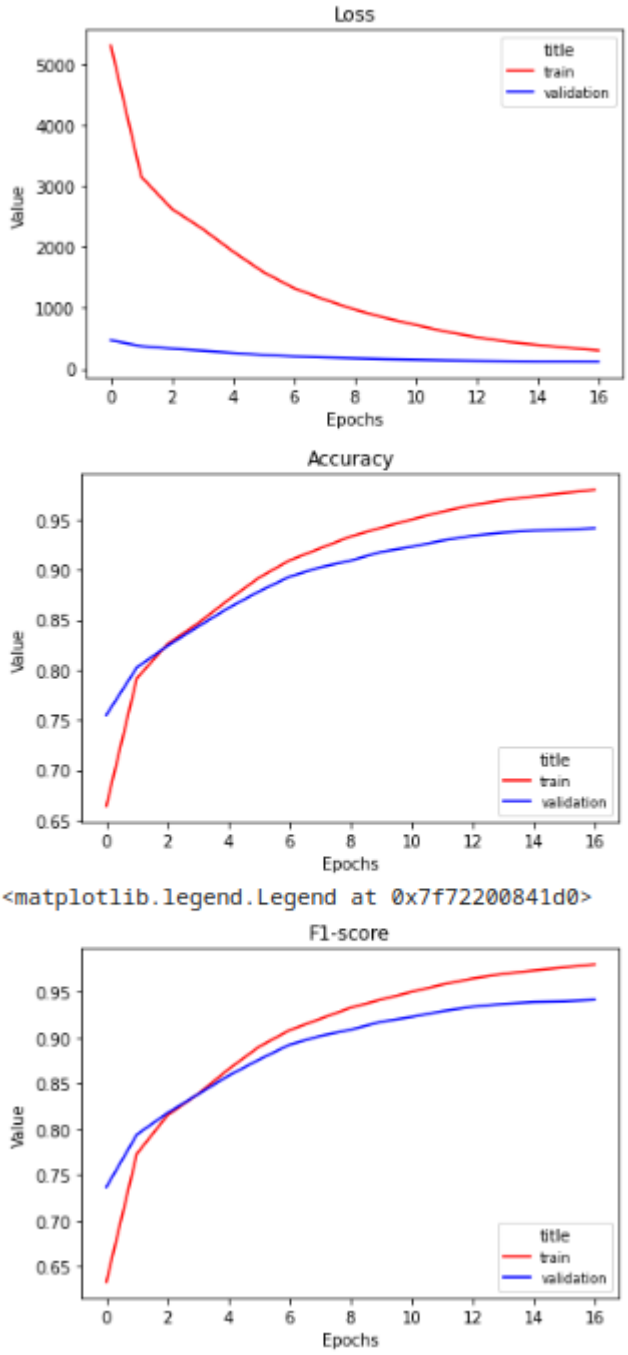


Figure 25: Training progress from Spanish POS tagger.

Several differences can be observed between models. First, is that the character level representation seems to improve the performance of the base model. It is only similar when the parameters of the base model become larger.

The model also obtained slightly better results with the English dataset than in the Spanish dataset. In Spanish, it can be appreciated using the confusion matrix in Figure 24 that the model confuses nouns

with adjectives. Also has problems differentiating between nouns, verbs and adverbs and between conjunctions and pronouns. This could be the reason for lower performance.

The model that performed the best on the test set is the word and character model with embedding dimensions of 128 for both words and characters and hidden dimensions of 128 for both words and characters. In the case of the Spanish, it works better with embedding sizes of 300 and 100 and hidden dimensions of 1024 and 128.

4.4 Ranker

In order to select a method to rank the documents retrieved, several ones were implemented and compared. The models attempted are a TF-IDF and BM25 ranker, one based on word2vec embeddings and another based on BERT embeddings. Also, the Python library Sentence Transformers, contains high performance embedding deep learning models, which were used to compare the efficacy of all the other mentioned models.

4.4.1 Data Preprocessing

TF-IDF and BM25

Rankers based on TF-IDF and BM25 don't need a dataset to be trained on. A vocabulary is built at run time and afterwards the documents are weighted according to the respective formulas.

word2vec

For word2vec, it is necessary to train a classifier to identify words and their contexts. As a result, the wikicorpus dataset [39] was used, which contained Spanish, English and Catalan documents from Wikipedia. In this case only the Spanish and English versions were used. Also, 100,000 fragments were used to train each of the models. For each of the fragments, tokenization was performed by words and all the characters were lowercased.

BERT

In the case of BERT embeddings no extra dataset was used. An already trained model [23] was adapted to perform the generation of a dense vector representation.

4.4.2 Model Architecture

TF-IDF and BM25

Both models operate in the same manner. The difference lies at the end, where BM25 performs a more complex weighting.
Firstly, both models receive a corpus of texts. From them, the vocabulary of the model is registered and the document frequency of each word is calculated from the beginning.

Then, the model is ready to give a score for documents. A faster simplification of the cosine similarity with tf-idf suggested in [83]. The cross product can be expressed in terms of the tf-idf:

$$score(q,d) = cos(q,d) = \frac{q \cdot d}{|q| \cdot |d|}$$

$$score(q,d) = cos(q,d) = \sum_{t \epsilon q} \frac{tf-idf(t,q)}{\sqrt{\sum_{q_i \epsilon q} tf-idf^2(q_i,q)}} \sum_{t \epsilon q} \frac{tf-idf(t,q)}{\sqrt{\sum_{d_i \epsilon d} tf-idf^2(d_i,d)}}$$

This is generally simplified by removing the query part as queries are normally very short and all the terms are likely to have frequency 1. Also, the division by the length of the query is the same for every document, which is not a differentiating factor. Then, the score can be simplified as:

$$score(q,d) = \sum_{t \epsilon q} \frac{tf-idf(t,d)}{|d|}$$

So, in the end, the tf-idf is computed with an input set of documents to be ranked and the terms of the query, using the following formulas. The term frequency and the idf are still computed with the same expressions:

$$tf_{t,d} = log_{10}(count(t,d) + 1)$$

$$idf_t = log_{10} \frac{\#\ documents}{df_t}$$

$$tf\ idf(t,d) = tf_{t,d} idf_t$$

If the metric was BM25, it is necessary to compute the lengths and the average length of the documents. Also, the score is updated according to its expression:

$$score(q,d) = \sum_{t \epsilon q} log(\frac{N}{df_t}) \frac{tf_{t,d}}{k(1-b+b(\frac{|d|}{|d_{avg}|}))+tf_{t,d}}$$

word2vec

For word2vec, it is necessary to train a classifier to identify words and their contexts. This method is called Skip-Gram with negative sampling. This model tries to predict context words given an input word. It consists of a neural network model with a unique hidden layer. It takes an input vector and condenses down into a smaller dimension to compute probabilities over the vocabulary of words.
The negative sampling means that random words from the vocabulary are selected as negative examples of context words during training. The positive examples are extracted from the training corpus. After training the hidden vector that is generated when the input is given is the embedding vector.

The training process of this model was made using the gensim module [84]. In this a hidden dimension of 300 was selected for the embeddings and a window size of 7 context words was chosen.

After having trained them, word2vec gives embeddings for words. The approach to generate embeddings for documents is by averaging word embeddings.

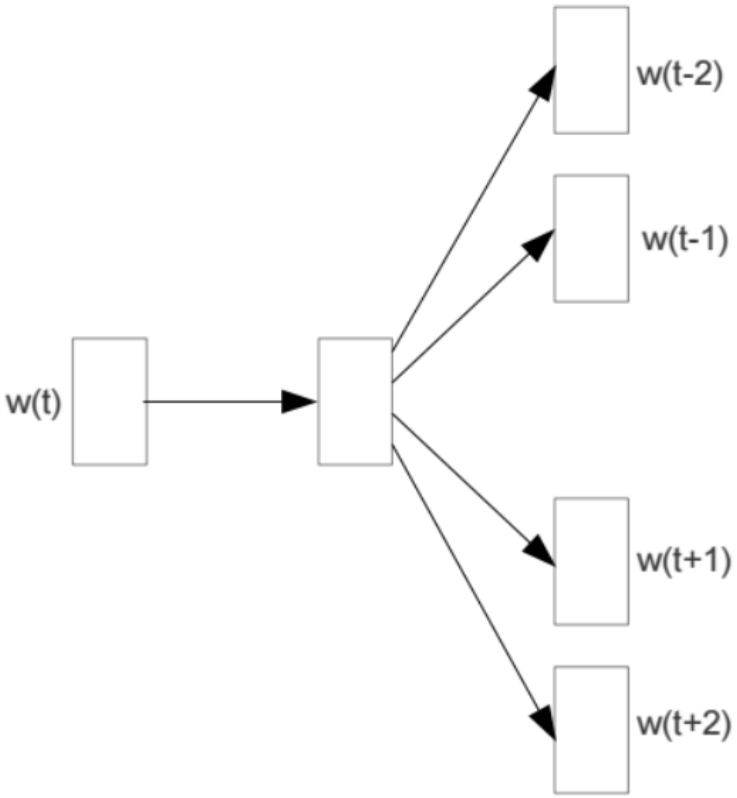


Figure 26: Skip-gram model [45].

BERT

In the case of BERT embeddings no extra dataset was used. An already trained model [88] was adapted to perform the generation of a dense vector representation. In this case, it consists of simply tokenizing the query or document and adding the [CLS] tag at the beginning and the [SEP] tag at the end. Then the input is passed into the model and it outputs an encoded representation for each token. In [85] and in [83] It is proposed to use the representation of the [CLS] token, with dimension 768. It can also be done by averaging the representation of all the tokens. In this case, the first approach was chosen.

$$h_q = BERT(q)[CLS]$$
$$h_d = BERT(d)[CLS]$$
$$score(q,d) = h_q \cdot h_d$$

Sentence Transformers

A third party library already uses this BERT embedding models to compute semantic similarity [50]. They use the same approach as described in the previous method but they add a finetunning layer

over the SNLI dataset [86].
Initially the sentences are passed into a BERT model and they extract the embedding making the average of the tokens which the model outputs. Then it performs a classification task to determine the inference relation between two texts. They test several activation functions and obtain a model that outperforms the current state of the art models and also is more computationally efficient.
They released several models that can be used with their module. In this case, a multilanguage one was selected for this project [87].

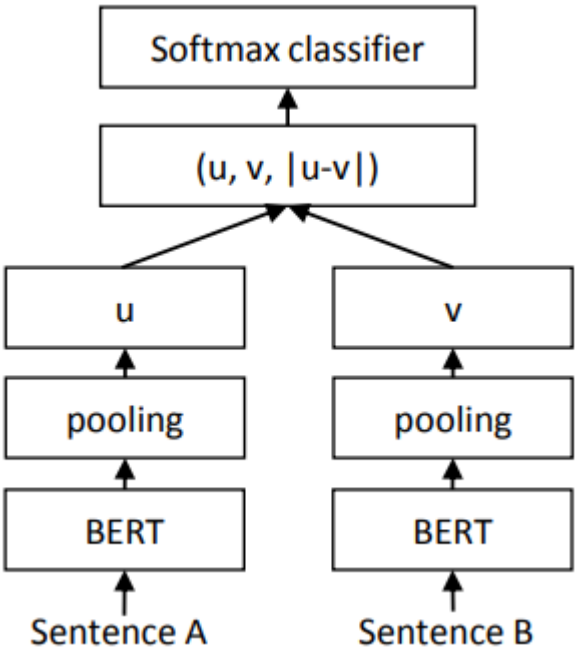


Figure 27: Sentence Transformers training diagram

4.4.3 Experiments and Results

The experimentation of the models were tested on a dataset created to test information retrieval specifically [124] [125]. This dataset consists of 1469 documents and 112 associated queries specifying for each query, a list of those documents that are relevant.

The experiment performed consists of running those queries and documents with all the algorithms implemented and measuring the results using Mean Reciprocal Rank and Mean Average Precision.
In this part of the inference pipeline it begins to be interesting computation time so it is also measured. The results can be observed in Table 84.

Table 84: Ranker Experiments

Model	MRR	MAP	Execution time (seconds)

TF-IDF	0.3828	0.3511	107.4870
w2v embeddings	0.3050	0.2577	340.9206
BERT embeddings	0.0783	0.0783	26171.1205
Sentence Transformers	0.6311	0.5885	4697.8775
BM25 (b=0.1/k=0.1)	**0.4325**	**0.3873**	**110.6599**
BM25 (b=0.25/k=0.25)	0.4150	0.3754	“
BM25 (b=0.5/k=0.5)	0.4198	0.3635	“
BM25 (b=0.75/k=0.75)	0.4058	0.3353	“
BM25 (b=1/k=1)	0.2234	0.2074	“
BM25 (b=1/k=1.25)	0.4114	0.3758	“
BM25 (b=1/k=0.1)	0.4155	0.3711	“
BM25 (b=1.25/k=0.1)	0.4089	0.3627	“
BM25 (b=1.75/k=0.1)	0.3821	0.3277	“
BM25 (b=2/k=0.1)	0.3350	0.3005	“
BM25 (b=1.2/k=1.5)	0.1095	0.1033	“
BM25 (b=0.75/k=1.2)	0.3719	0.3054	“
BM25 (b=0.75/k=2)	0.3353	0.2749	“
BM25 (b=0/k=0)	0.0888	0.0946	“

In Table 84, it can be observed that the model from sentence transformers is clearly the best by some margin. The next best model is BM25 with chosen parameters b=0.1 and k=0.1. It is clear that the use of semantic meaning in text representations can outperform the ranking methods based on matching words. However, the methods attempted here were not good enough.

The Word2vec model probably doesn't work that well because of averaging all the word representations on a document. Another possible reason is a deficient training method or corpus.

The BERT embeddings method attempted is promising but probably some kind of fine-tuning over a dataset should have been performed. It is also very probable that a mistake in the implementation was made as the performance is too low. However, no error could be identified that could have improved

the efficiency.

4.5 Reader

In terms of Reader, two methods were created. One is an LSTM encoder-decoder with attention and the other one is a fine-tuned BERT for question answering.

4.5.1 Data Preprocessing

For both models the fine-tuning is performed on the same dataset: SQuAD version 1.1.0. This dataset has a 2.0.0 version. which adds questions that cannot be answered from the context fragment. However, it is preferred that the model gives an answer from the fragment even though it is incorrect as the fragments selected are selected by the ranker. In addition, a Spanish automatic translation of the dataset was used [126].

Here is an example of a dataset entry.


```
{

    "answers": {

        "answer_start": [404, 356, 356],

         "text": ["Santa Clara, California", "Levi 's Stadium", "Levi 's Stadium en la Bahía de San Francisco en Santa Clara, California."]

    },

      "context": "\"El Super Bowl 50 fue un partido de fútbol americano para determinar al campeón de la NFL para la temporada 2015. El campeón de ...",

    "id": "56be4db0acb8001400a502ee",

    "question": "¿Dónde tuvo lugar el Super Bowl 50?",

    "title": "Super Bowl _ 50"

}
```


LSTM

To preprocess for this model, first the question, answer and context are tokenized by whitespaces. Also, it is necessary to calculate the end index of the answer in the context. That is done using the tokens and length of the answer. Then, a vocabulary of words is created from the whole corpus so

that the word tokens can be converted to numbers. The dataset is already given with a train and validation split. The preprocessed data is then organized into batches.

BERT

Using the same dataset, the data is preprocessed using a BERT tokenizer which uses the word-piece tokenizer [89] to break the words into subwords. For instance, with the word 'sleeping' it would do 'sleep' and '##ing'. It also uses the vocabulary which was used during the pretraining process to convert the new tokens to indexes. It handles the out of vocabulary problem with an [UNK] token as well. As it uses batched inputs with specific sequence lengths, it adds padding values at the right of shorter sequences and attention masks which tell the BERT model which inputs are actual tokens or padding values. Here there is an example of a padded sequence with a mask that indicates the non padding elements:

```
input_ids = tensor([[ 105, 3002,  112,  9, 85, 1996, 15,  903,  8012,    97,    5003,    0,
        0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
        0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
        0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
        0,    0,    0,    0]])

mask = tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

4.5.2 Model Architecture

LSTM

Here, it seemed like a promising idea to use an encoder-decoder architecture. This architecture is based on recurrent neural networks like the LSTM. Initially, the input sequence is passed into an encoder RNN and the hidden vector is kept as an internal representation. Then, this vector is used as an input to another RNN which analyzes this hidden vector and converts it. The decoder's output is then used for the desired task. One very common task is machine translation between languages. The problem is that these networks fail to obtain deep contextual relationships, especially in long sequences. That is why an attention mechanism is normally introduced in between the encoder and decoder, so that it obtains some significance and emphasizes elements of the hidden sequence. Figure 28 poses an example of this.

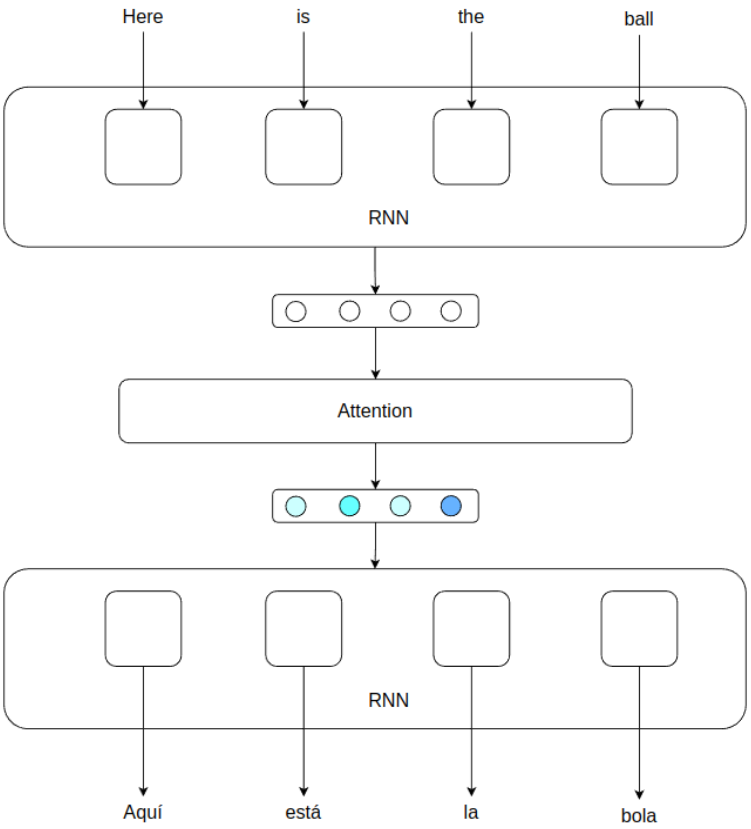


Figure 28: Encoder-Decoder with attention mechanism in Machine Translation

This architecture can also be adapted to other tasks, like question answering. BiDAF is an example of such a model [90]. The model implemented takes inspiration from this one.

The model implemented receives the context and question concatenated being the tokens represented in numeric format. The input is introduced into a trainable embedding layer, to obtain a dense vector representation. This embedding vector is then passed into a bidirectional LSTM which acts as the decoder. Once the sequence is processed, the output of the LSTM has twice the same dimension as the specified hidden dimension. To return this hidden vector to the desired hidden dimension, it is passed through a linear layer. Then a self-attention layer is used. Here, the input is projected into the key, query and value vectors. The key and query are multiplied and passed through a softmax layer. The output of the softmax is then multiplied by the value vector and the output is obtained. The output of the attention layer is sent to the decoder bidirectional LSTM. The output of the LSTM passes through a new Linear layer which has two outputs: one for the start position of the answer and another for the end position.

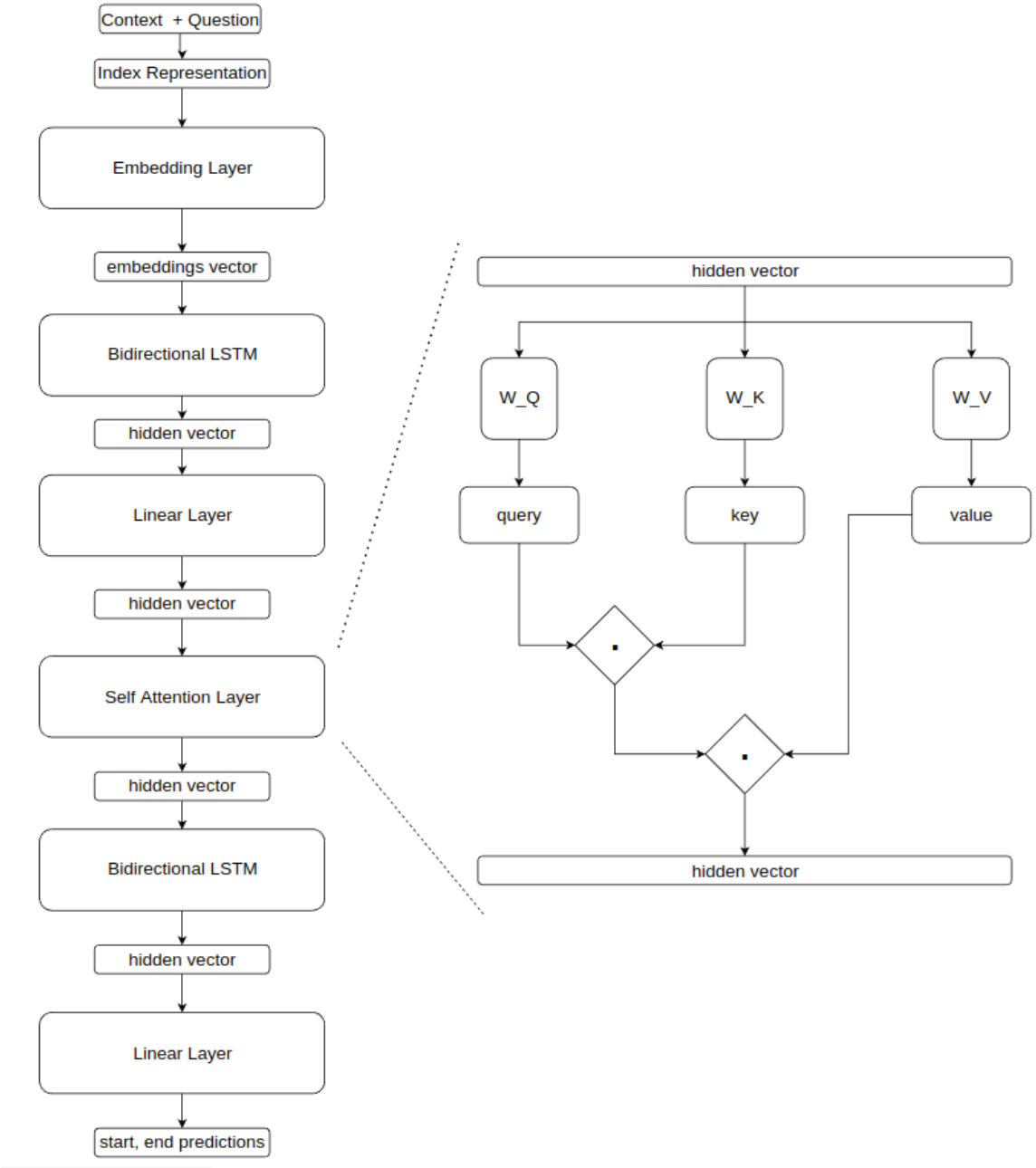


Figure 29: Encoder-Decoder with attention mechanism in Question Answering

BERT

BERT models can also be finetuned to solve question answering tasks. In this case, it is necessary to format the input. The way to do this is by concatenating the question and the reference text together with two special tokens. The [CLS] token is added at the beginning of the sequence and the [SEP] token is added to separate the question from the reference. After being processed by the model, two extra classifiers are used to predict the start and the end of the answer sequence. It uses, for that purpose, a start vector S and an end vector E with the size of every output token. The probability of a token being the start or the end is computed by multiplying every token by each vector followed by a

softmax activation function [23]. This is graphically represented in Figure 30.

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot j}}$$

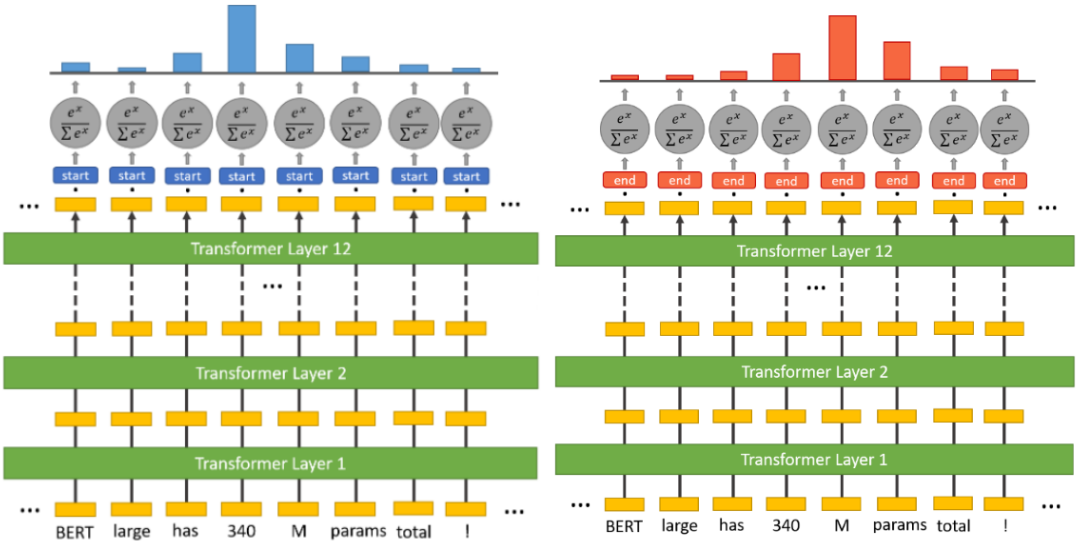


Figure 30: BERT fine-tuning for Question Answering [91]

4.5.3 Experiments and Results

LSTM

To train the Encoder-Decoder model with attention, several configurations were tried of embedding dimension, hidden dimension number of layers in each LSTM and bi-directionality in the LSTMs.

The learning rate was set to 0.1 with a Stochastic gradient descent classifier and a gradient descent loss function. The experimentation was done in the English dataset and then the best model was used also with the Spanish one. The number of training epochs was adjusted to avoid overtraining.

The training metrics are Exact Match, which measures extracting exactly the answer, and F1-score, which compares the number of words that are matched between the real and the predicted answer.

Table 85: Reader LSTM Experiments

Lang	embedding dim	hidden dim	num_layers	bidirectionality	val EM	val F1

English	**300**	**128**	**2**	**True**	**47.24**	**58.91**
English	300	128	3	True	34.54	45.21
English	300	256	3	True	34.54	43.04
English	100	256	2	True	27.38	36.34
English	100	256	2	False	20.43	29.34
Spanish	**300**	**128**	**2**	**True**	**29.45**	**44.97**

BERT

For BERT model, the training was made using a script provided by hugging face transformers [92] Also several base pretrained models were tried like BERT tiny, BERT mini, BERT base and BERT large [93]. Various parameters were changed during the experimentation like the number of transformers layers stacked, the hidden dimension of the vectors, the maximum sequence length of the input and the batch size of the input. The results are also measured with Exact Match and F1-score over the validation set. The Spanish version was trained over a pretrained model made by another user [94] as the ones provided by google are only available in English.

Table 86: Reader BERT Experiments

Lang	num layers(L)	hidden dim(H)	max sequence length	Epochs	batch size	val EM	val F1
English	2	128	512	3	5	37.2753	49.6880
English	4	256	512	3	5	65.4210	75.4484
English	8	512	384	2	12	80.8136	88.2712
English	8	512	512	2	6	81.3245	88.6687
English	8	512	256	2	16	79.0444	87.0220
English	**12**	**768**	**256**	**1**	**6**	**83.7559**	**90.5116**

Spanish	8	512	512	2	6	65.0214	76.665

By the results it can see that the BERT models perform significantly better than the Encoder Decoder LSTM architectures. The superior performance can be due to the transformers characteristics and extended pre-training period of time. The encoder decoder model was self constructed which might not be the optimal approach and surely some mistakes were made while making it.
In terms of language the models perform better for English than Spanish. One reason for this could be the automatic translation of the SQuAD dataset.

5. IMPLEMENTATION AND DEPLOYMENT

This chapter describes the operational environment where the project was developed as well as the final interface and backend architecture.

5.1 Operational Environment

In this section, the hardware in which the project is executed is described.
The machine in which the project is implemented is a Modern 14 A10RAS with

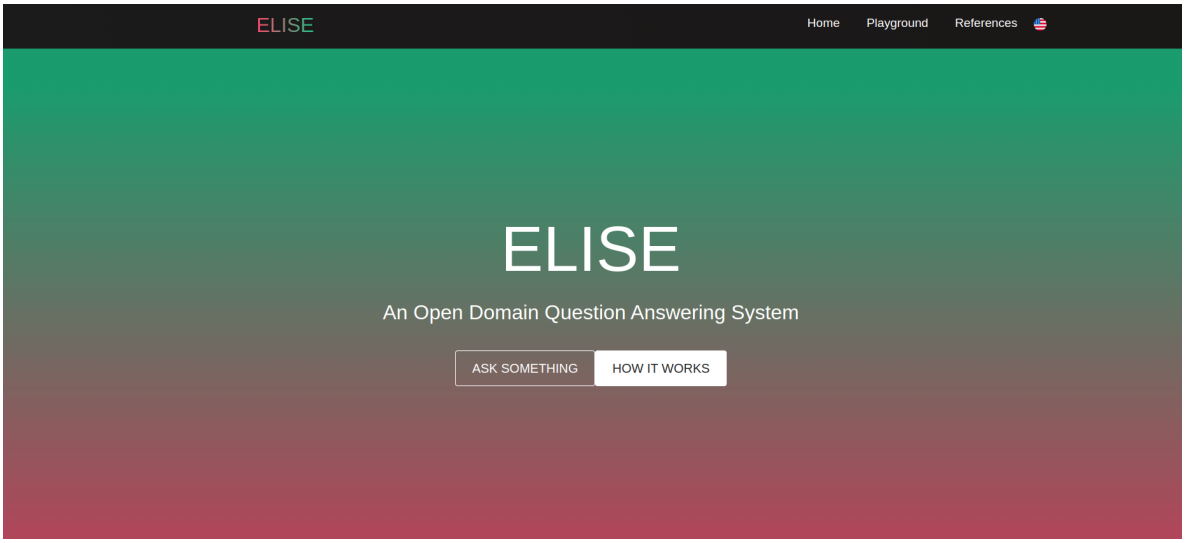
- CPU I7-10510U.
- RAM memory DDR IV 16GB (2666MHz).
- A storage unit of 1TB NVMe PCIe SSD
- In an Ubuntu 20.04.3 LTS OS.

The training process of all the deep learning models was too computationally expensive for this machine in terms of time and hardware. For that reason, for a period of 4 months the pro tier of Google Collaborate was used in order to have access to GPU computing capabilities and longer runtimes.

5.2 User Interface

Here it is shown the final web interface implemented using the React.JS framework and the languages HTML, CSS and Javascript. Screenshots of the different sections are shown with both desktop and mobile versions.

Figure 31 shows the landing page of the application in which the name of the system appears and two buttons are shown, one which leads to the page where the user can ask questions and the tutorial part on the same page.



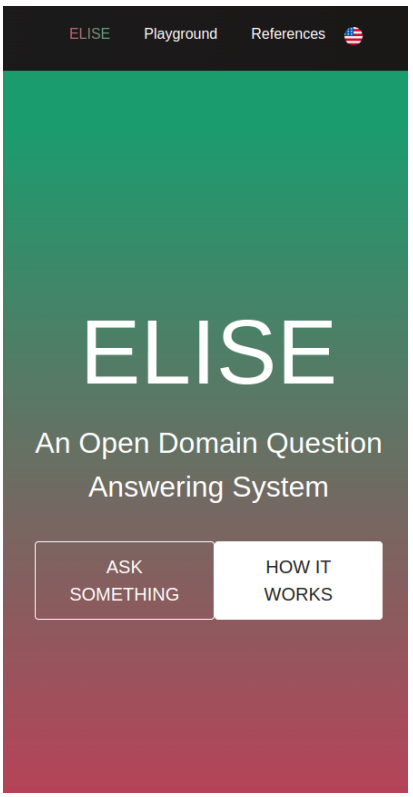
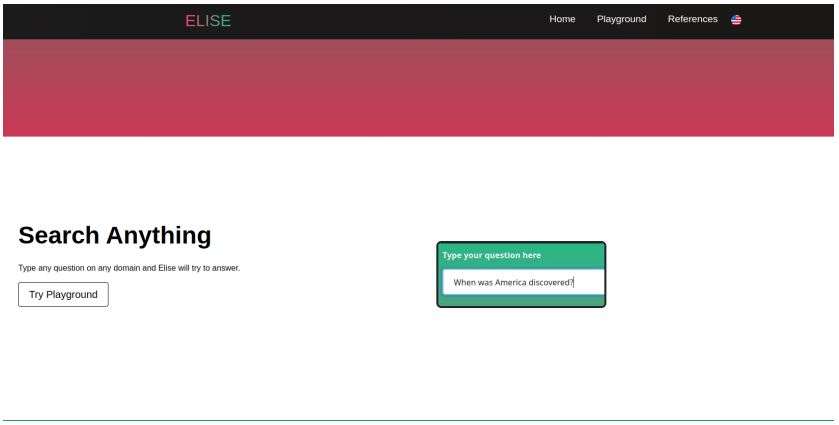


Figure 31: Landing page in the web interface

Figure 32 It can be shown some examples of the entries of the tutorial, where a brief description of what can be done is accompanied by an image and a link to that page in the website.



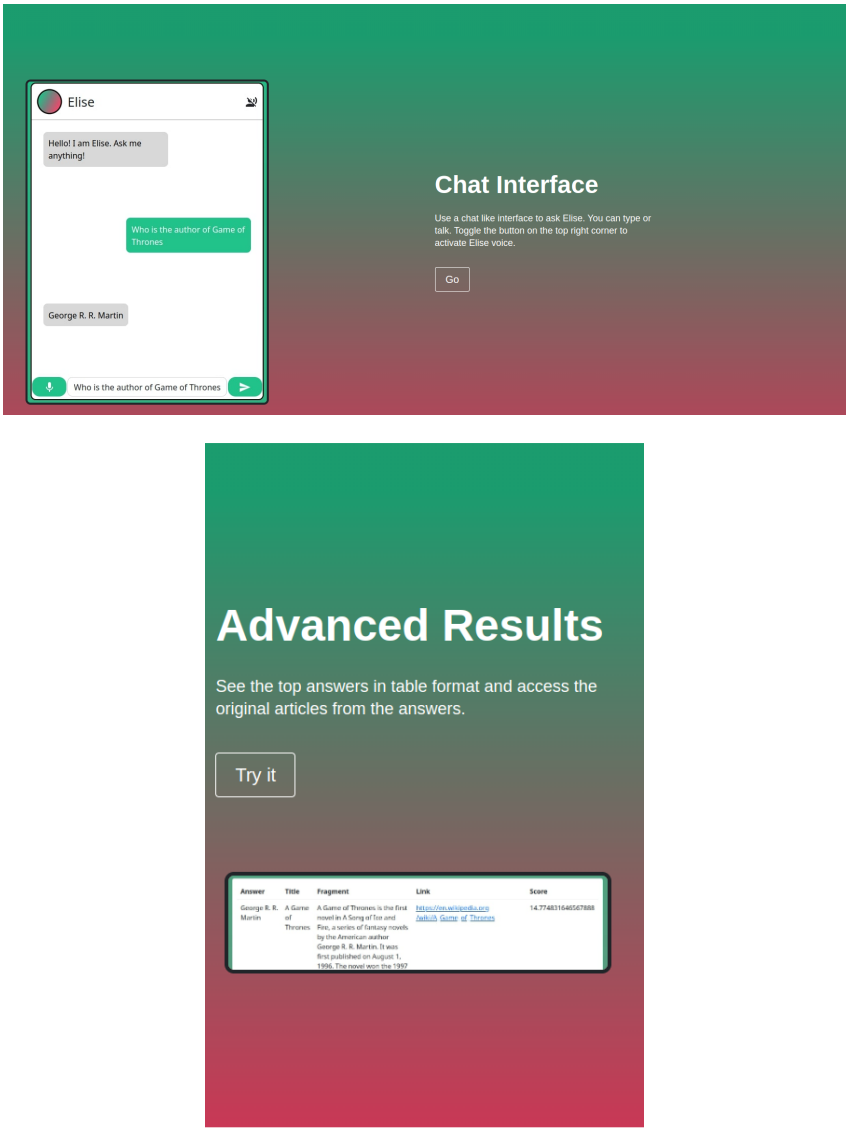


Figure 32: Tutorial from the web interface

In figure 33, it appears the references section on the web interface, where the user can see a brief description of each resource, an image and a link to the source.

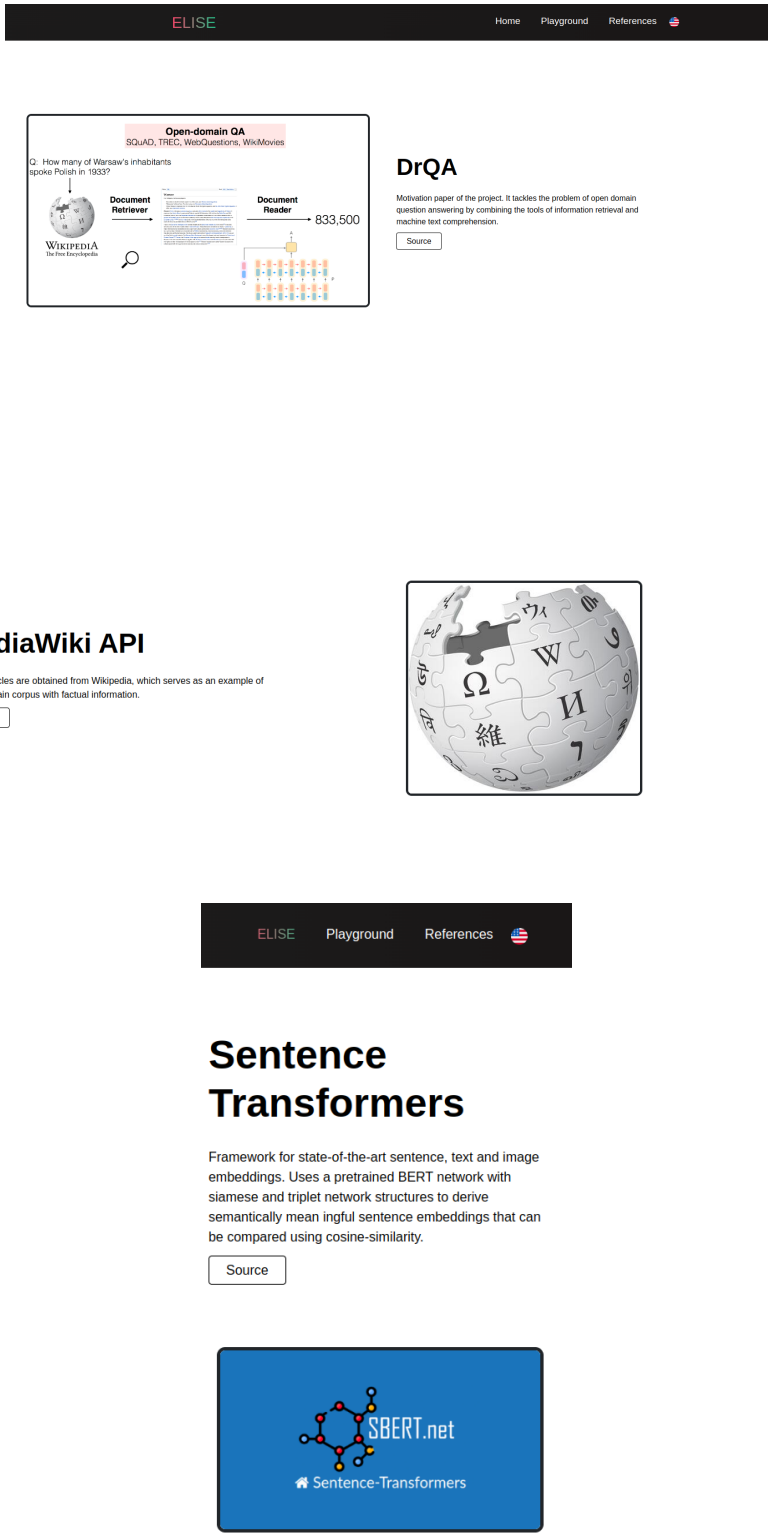


Figure 33: References page from web interface

In Figure 34, the chat interface is shown, where the user can type questions in the text box below. The user can use his speech with the microphone button at the right. It is needed to maintain the button pressed to be recording. On release, the text appears in the text box. The query is sent when the user presses the arrow button. Also, the text-to-speech of the answers can be activated by pressing the cross icon in the top right corner.

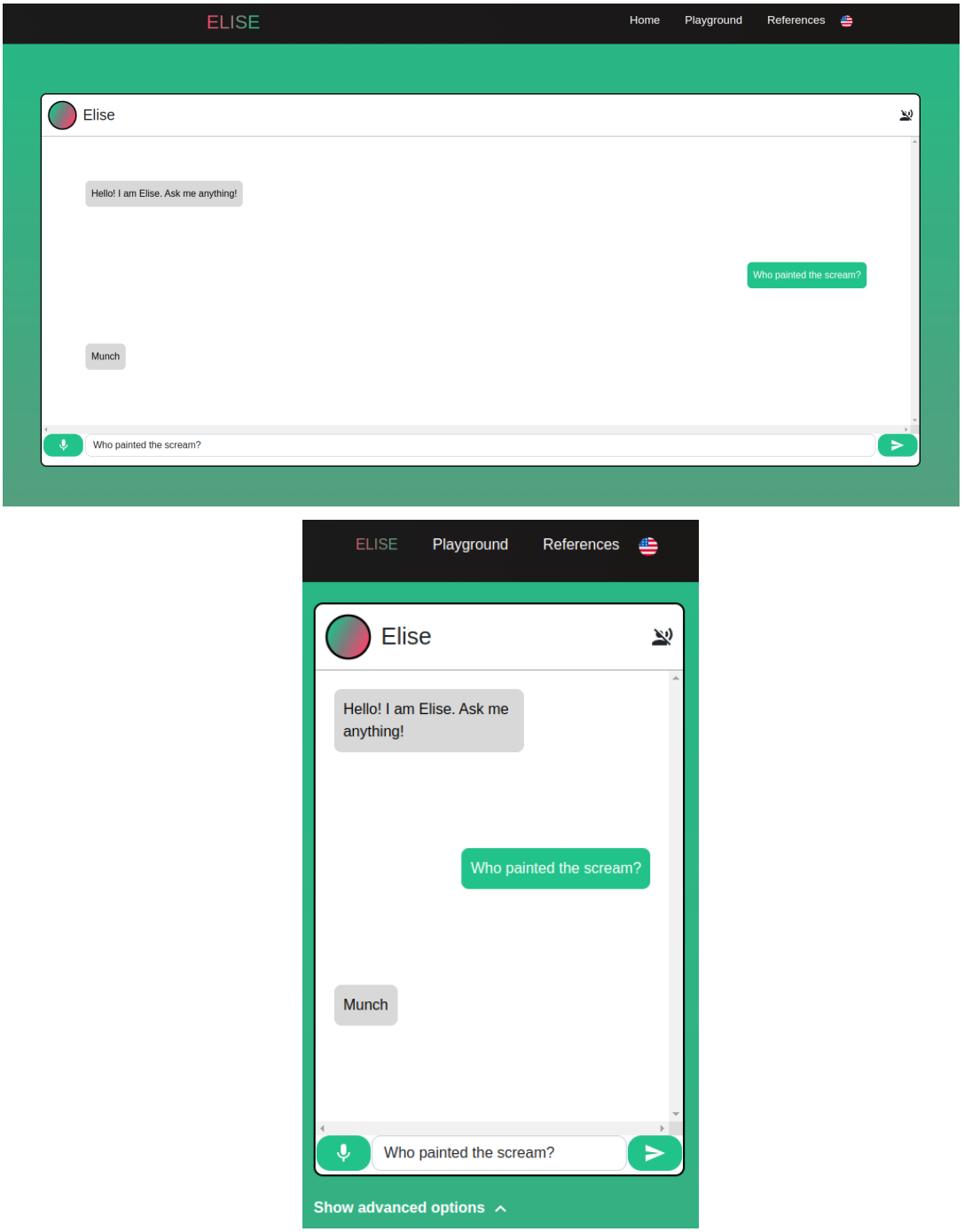
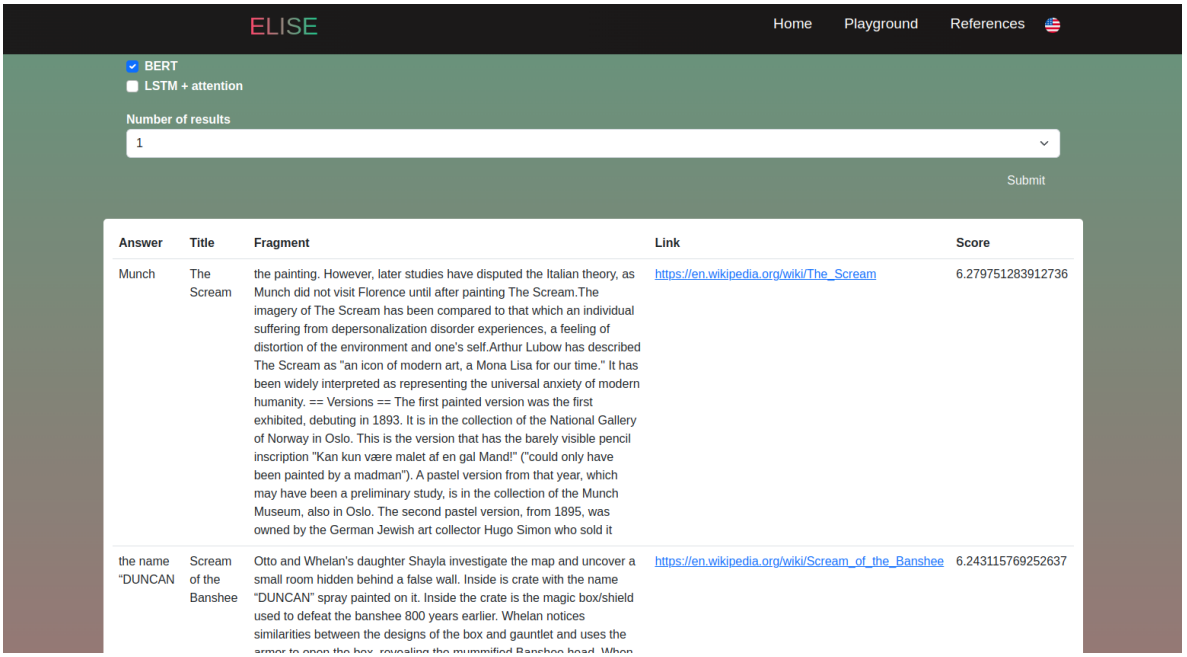


Figure 34: Chat box in the web interface

In Figure 35, the advanced options section appears. Here, by clicking the advanced options text a

new form and table appears. The form allows the user to select a few rankers and reader models while a selector has from 1 to 10 results to be returned. The table shows the answers by order of score given by the ranker. For each answer the Title of the article, the fragment from the article and the link to the article are included. In the mobile version the table is adapted by making it scrollable.



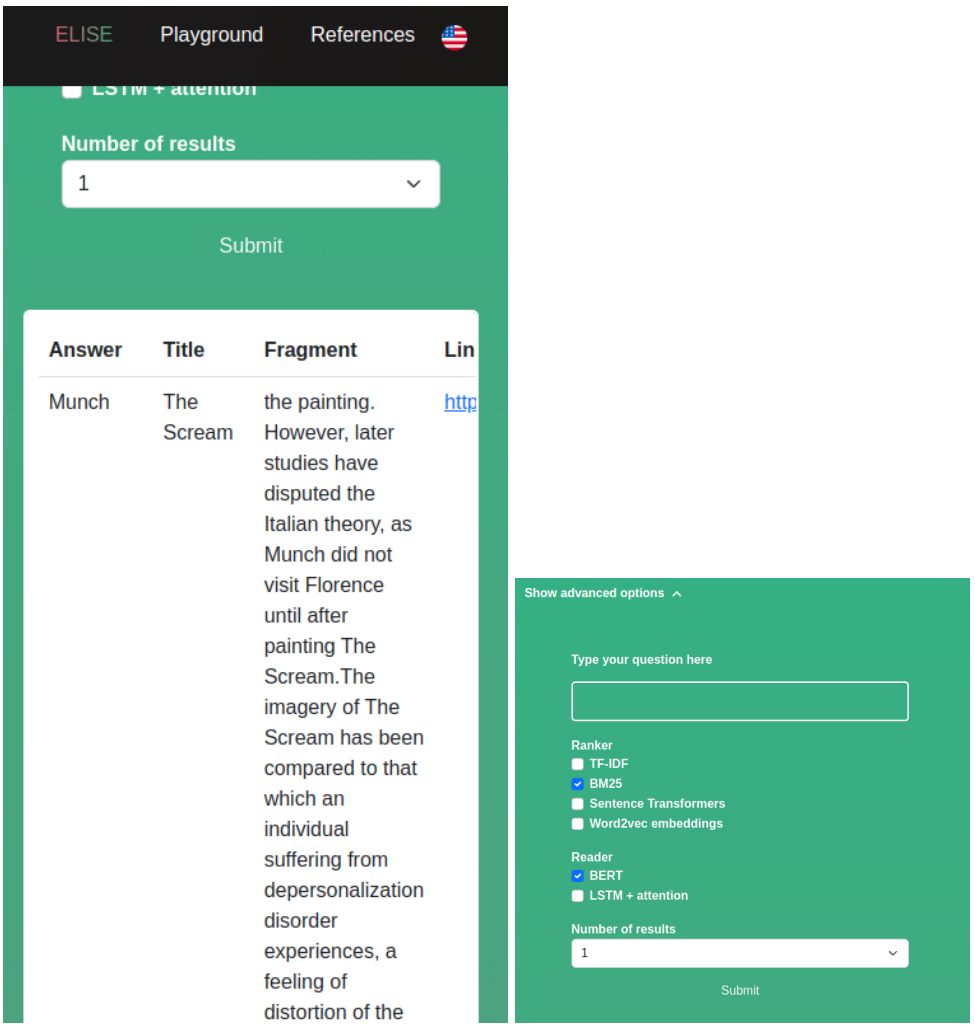
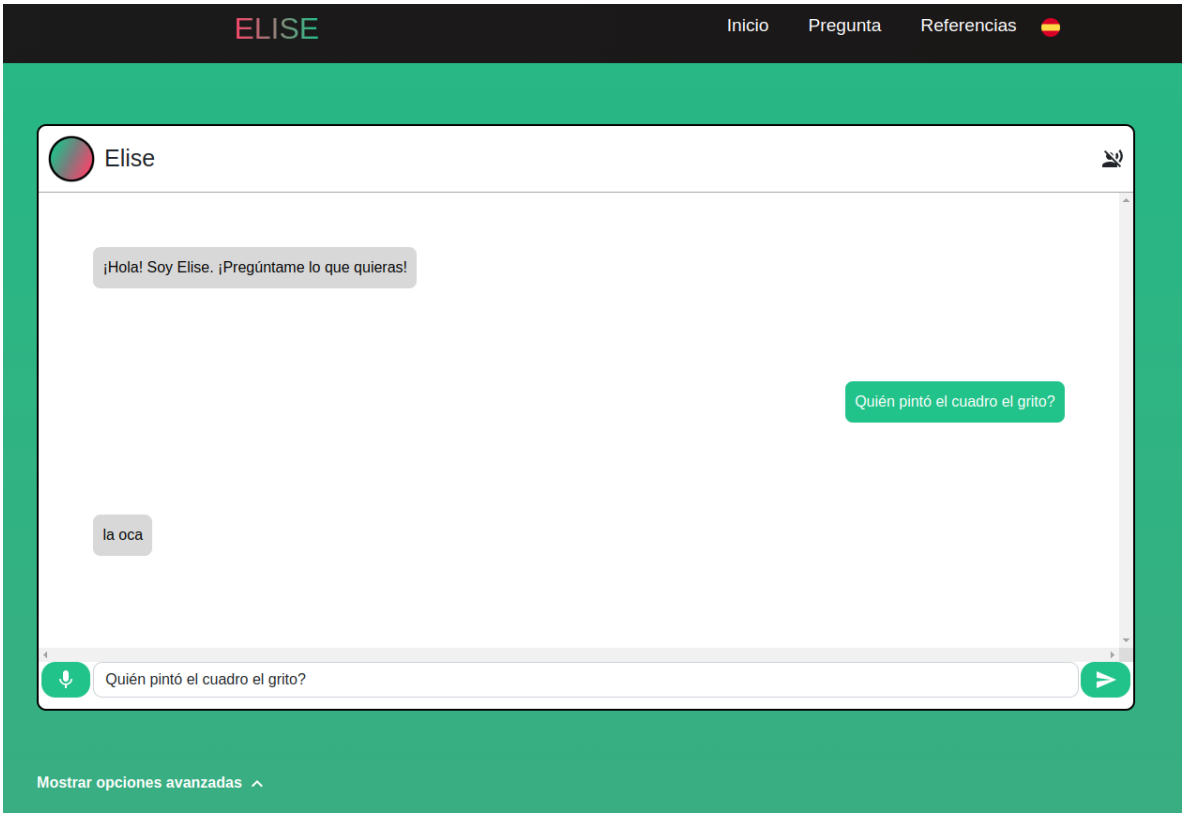
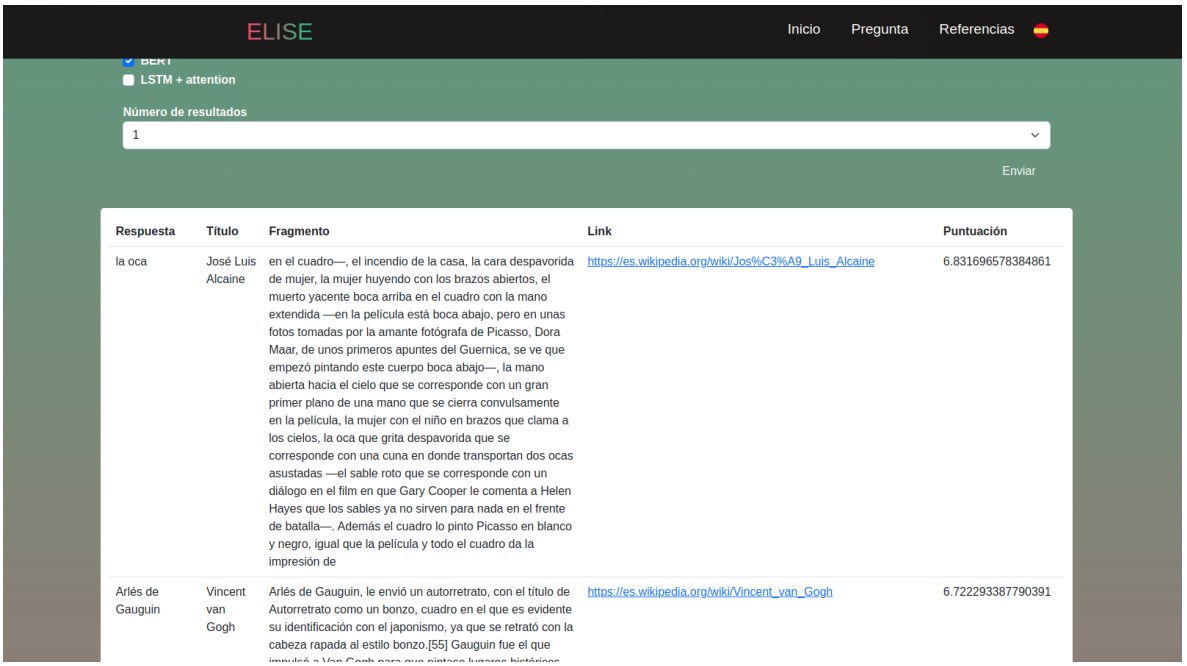


Figure 35: Advanced options in the web interface

Figure 36 shows the translation of the page into Spanish.





Respuesta	Título	Fragmento	Link	Puntuación
la oca	José Luis Alcaine	en el cuadro—, el incendio de la casa, la cara despavorida de mujer, la mujer huyendo con los brazos abiertos, el muerto yacente boca arriba en el cuadro con la mano extendida —en la película está boca abajo, pero en unas fotos tomadas por la amante fotógrafa de Picasso, Dora Maar, de unos primeros apuntes del Guernica, se ve que empezó pintando este cuerpo boca abajo—, la mano abierta hacia el cielo que se corresponde con un gran primer plano de una mano que se cierra convulsamente en la película, la mujer con el niño en brazos que clama a los cielos, la oca que grita despavorida que se corresponde con una cuna en donde transportan dos ocas asustadas —el sable roto que se corresponde con un diálogo en el film en que Gary Cooper le comenta a Helen Hayes que los sables ya no sirven para nada en el frente de batalla—. Además el cuadro lo pinto Picasso en blanco y negro, igual que la película y todo el cuadro da la impresión de	https://es.wikipedia.org/wiki/Jos%C3%A9_Luis_Alcaine	6.831696578384861
Arlés de Gauguin	Vincent van Gogh	Arlés de Gauguin, le envió un autorretrato, con el título de Autorretrato como un bonzo, cuadro en el que es evidente su identificación con el japonismo, ya que se retrató con la cabeza rapada al estilo bonzo.[55] Gauguin fue el que impulsó a Van Gogh para que pintase lugares históricos	https://es.wikipedia.org/wiki/Vincent_van_Gogh	6.722293387790391

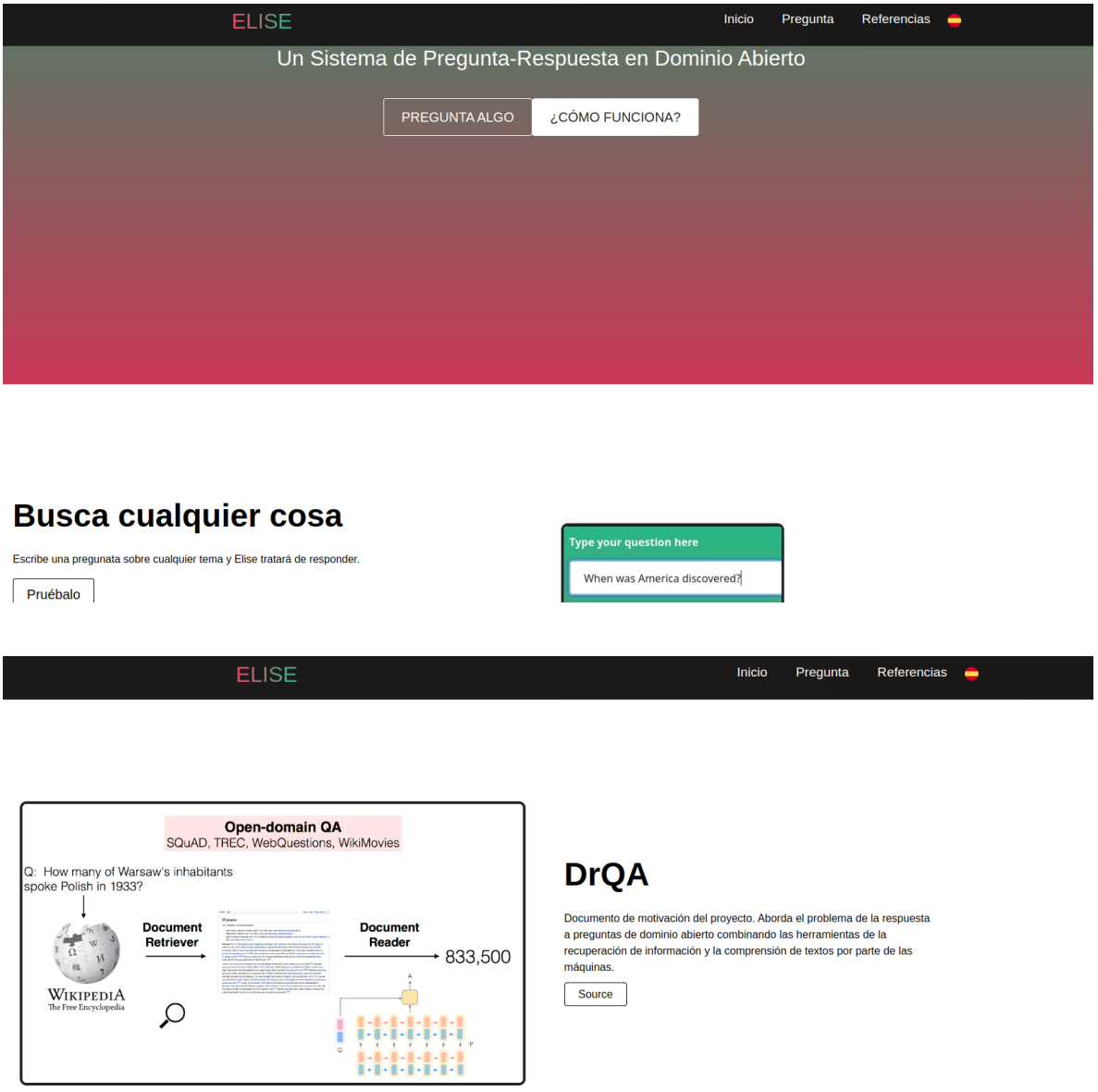


Figure 36: Spanish version of the web interface

5.3 Backend

For the backend, initially, the testing process has been done in a local server implemented in Flask with an HTTP internal protocol. Later, when the deployment phase was to begin, several instances were considered in order to deploy the models like DigitalOcean droplets or Amazon AWS EC2. However, the size of the models, most of them consisted of several hundred MB, made the need for larger instances to load the models in memory. Also, a single instance made it look like scalability would be a problem.

For that reason, it seemed a good idea to look over a serverless solution like AWS lambda or GCP cloud run. This kind of services allow you to deploy containerized images of your application and serve them as a REST API over HTTPS. This way, when a request is received, the instance is launched and starts running. It handles scalability as it can deploy multiple images over several instances depending on the incoming demand. The good thing is that only the running time is billed and the resources, maintenance is all handled by AWS. Also, it offers a free tier that suffices for the demands of the project for the implementation of the prototype.

However, even though it is advertised that AWS can handle containers of size as big as 10GB, it produced errors surpassing 3GB. Apparently, the service is capped for new users. This drawback didn't allow the deployment of all the models as it was desired. Some models had to be left out of the final deployment. Even though the best reader found was BERT(L12/H768), it is also the largest one. Having to upload a Spanish and English version of it, it was preferred to upload two BERT(L8/H512) alongside with the Spanish and English LSTMs versions. For the ranker, the Sentence Transformer and the Word2Vec embeddings couldn't fit as well for storage matters. Bert embeddings weren't good enough also, so it was preferred to leave it out. It resulted in including only the BM25 and TF-IDF which are the lighter and faster ranker models. Even though it is not the desired situation, at least the second best performing models were deployed.

Regarding the web application, it is in reality a static website that makes an HTTPS petition to a REST API for data. Searching for a place to deploy its firebase was a good option as it offers a free hosting tier for static websites and allows linking a custom domain and an SSL certificate. A .tk domain was selected as it added no extra cost to the project as well. The website was deployed under the name of elise.tk.

As for the data, the application communicates with the Wikimedia API through a python module that [95]. This allows the system to extract different articles from a general domain from which to attempt to answer the question to the user. This module could be changed for another source to make the system useful in a specific domain or application. In terms of performance the wikipedia module certainly acts a little as a bottleneck as it adds an average of 10 seconds of computation time.

Docker was used to deploy the lambda application. The container starts from a basic python 3.8 public image and then it sets up the working environment by installing the required python dependencies and setting the handler program.

Finally, the compiled image is pushed into AWS Elastic Container Registry (ECR) [96] and deployed using serverless [97].

6. TESTING AND VALIDATION

In order to test the efficacy of the system, two types of validations have been done. One with a specific dataset designed for open domain question answering in which objective metrics can be applied to understand which of the retriever-reader combinations works best. The other aims to evaluate the user interface and how the model performs with actual users with the goal of receiving feedback and applying it to the system.

6.1 Dataset Evaluation

There are two datasets that from the ones described before that have the structure to test open domain question answering. This structure is formed by a question, a set of valid answers and a set of documents for each question. The set of documents contain noise to hinder the extraction of the answer. From the datasets researched, TriviaQA and searchQA are the most suitable.

It seems like there is no current standard way of testing open domain question answering systems. For that reason, in the presentation paper of searchQA they create this dataset and propose to measure the accuracy from the top answer and the accuracy from the top 5 answers [34]. Here this is taken as inspiration but with a few variations.

The testing process followed was to use the validation set of the TriviaQA. As the number of triplets in the dataset are very large and computation expensive, a random subsample of 1000 triplets was selected and all of the combinations were tested on this subset.
It is worth noting that this is only for comparative performance between the models created in this project. So that the best one can be selected. For each of the answer-question-documents triplets, the question and documents are presented to the rankers which extract the most likely fragments to contain the answers. Then, the top ten fragments are presented to the readers, which extract an answer from each of the fragments.
As in traditional Question Answering, the Exact Match and F1-score are used to compare the predicted answer with the real answers. In this case, the measurements are taken taking into consideration the top k answers where k can be 1,2,5 and 10. This way we can use the metrics of Question Answering with the ranking of Information Retrieval. The way to use EM and F1 with several potential answers is to take the one the answer that produces a maximum value. Then, the best EM and F1 of every query is averaged.
In Table 87 and Figures 37 and 38 the results from the experimentation are presented.

Table 87: TriviaQA full pipeline evaluations

Reader	Ranker	Top 1 answer		Top 3 answers		Top 5 answers		Top 10 answers	
		EM	F1	EM	F1	EM	F1	EM	F1
BM25	BERT (L8/H512)	0.121	0.174	0.307	0.413	0.407	0.521	0.572	0.696

BM25	BERT (L2/H128)	0.031	0.063	0.098	0.189	0.145	0.266	0.244	0.409
BM25	BERT (L4/256)	0.07	0.114	0.214	0.314	0.302	0.421	0.439	0.578
Sentence Transformers	BERT (L8/H512)	0.286	0.358	0.432	0.537	0.5	0.622	0.58	0.702
TF-IDF	BERT (L8/H512)	0.057	0.103	0.212	0.311	0.305	0.426	0.446	0.583
w2v	BERT (L8/H512)	0.114	0.169	0.267	0.357	0.347	0.461	0.466	0.600
BERT embeddings	BERT (L8/H512)	0.026	0.034	0.056	0.076	0.074	0.097	0.1	0.136
BM25	BERT (L12/H768)	0.200	0.211	0.4	0.454	0.520	0.569	0.660	0.720
BM25	LSTM	0.025	0.058	0.091	0.181	0.14	0.262	0.237	0.406
TF-IDF	LSTM	0.032	0.068	0.099	0.186	0.15	0.267	0.15	0.267
ST	LSTM	0.13	0.188	0.203	0.324	0.263	0.389	0.303	0.452
ST	**BERT (L12/H768)**	**0.353**	**0.419**	**0.537**	**0.630**	**0.603**	**0.703**	**0.673**	**0.776**

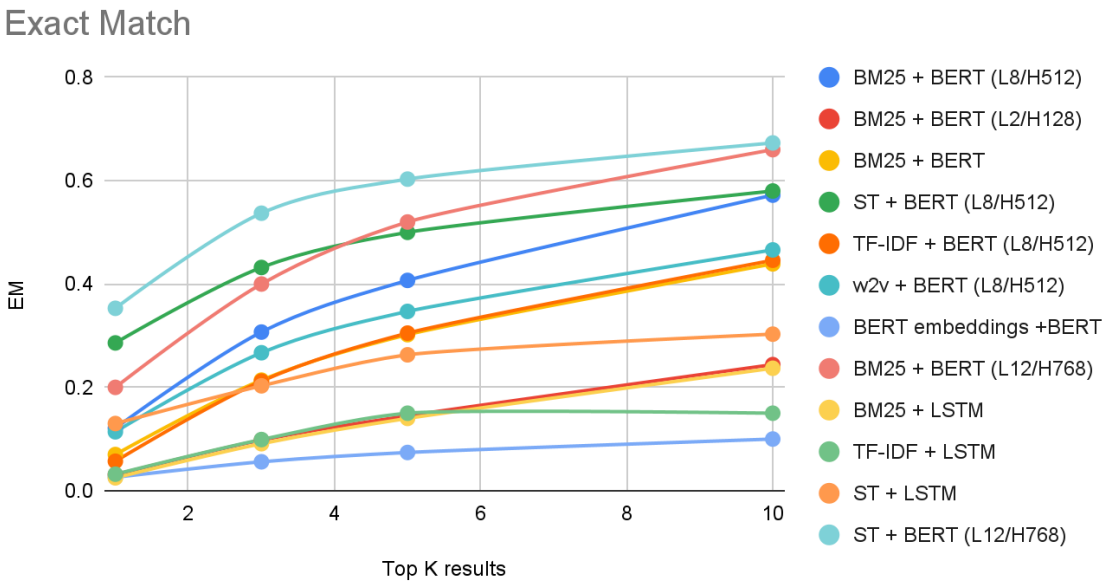


Figure 37: EM graphical comparison across models

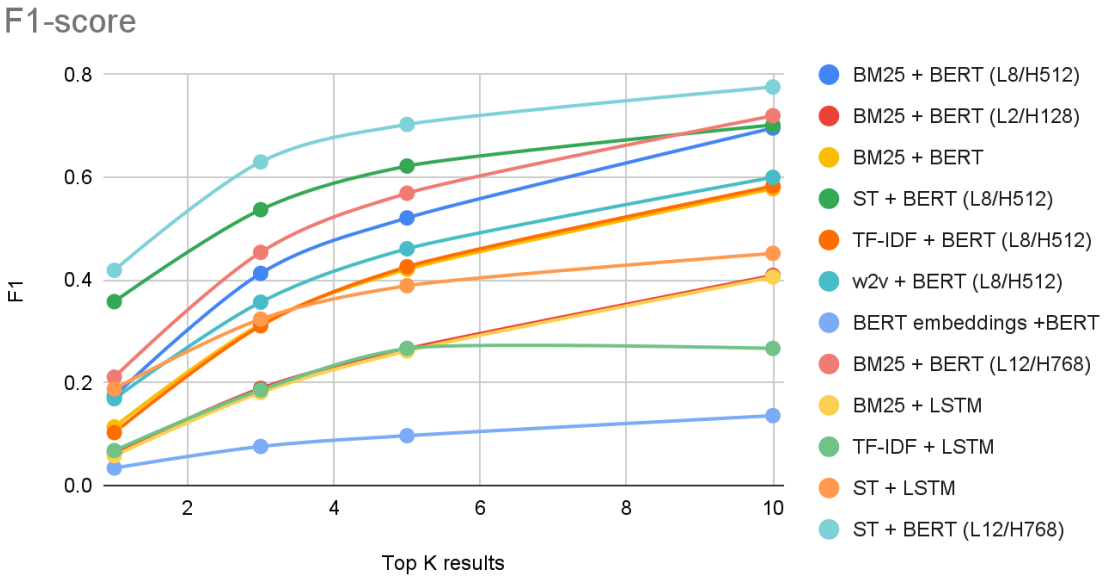


Figure 38: F1 graphical comparison across models

The best results are obtained using the library Sentence Transformers as the ranker and the BERT

(L12/H768). However, as Sentence Transformers is an external library, the second best model considered is BM25 + BERT(L12/H768) which is the one that was meant to be defaulted in the web application. In Spanish these very same combination was also included.
From the results, it can be observed that the transformers based models obtain better results than the LSTM based models which is congruent with the experimentation of the readers individually. Moreover, rankers also follow this kind of pattern showing that BM25 and ST are the best rankers.
One of the limitations of the dataset is that it wasn't found in Spanish so that would need to be addressed in the user evaluation.

6.2 User Evaluation

An evaluation with users was also carried out in order to see problems with the web application and the possible improvements. The evaluation has two parts.
In one the user is invited to ask six questions of their own choosing to Elise. Three in English and three in Spanish to see how the Spanish model is compared with the English one. For each question, the user introduces the question asked, the answer he/she expected and the answer that the system gives. In addition, it is asked if a valid answer can be found in the top 1, top 3, top 5 or top 10 questions shown in the table.
In the second, part a CSUQ (Computer System Usability Questionnaire) [98] inspired questionnaire was asked to the user in which a Likert scale from 1 to 5 was used as a possible answer being 1 completely disagree and 5 completely agree.
An initial question in the questionnaire asks the user for permission on his/her data usage and it is explained how the data is going to be used:

You have been invited to participate in a research project carried out by a student at Universidad Carlos III de Madrid for the purpose of preparing the Final Degree Project. In order for me to use any information I collect, I must have your consent.
You must use a device with internet access to fill in the form. In this survey you will:

- *Access a web page*
- *Answer questions*
- *Share ideas on how you could improve the design of the application*
- *Comment on other observations about design improvements*

The researcher will be able to observe your comments and feedback. You will be able to leave the process at any time.

The evaluation was performed with 9 users, who tested the web application and were explained the workings of the web before asking questions. They were asked to write the results on a Google Form questionnaire.

In the APPENDIX B all the questions with their answers are listed.

Comparing the English and the Spanish answers, 27 questions in English were asked and 26 in Spanish. Looking at the question to see in which rank a valid answer could be found, an average was taken. When the user didn't find a question in the table, the result was substituted by 20. The English

model obtained an average of 7,07 and the Spanish model 10,92. That means that the English model is more likely to find an answer and rank it better.

The reader works quite well and finds the answer if it is contained within the context text. It looks like there are more problems with the ranker because often the answer would be included in the table but the ranker gave priority to the wrong fragment.

Regarding the usability of the system, the results of the CSUQ questionnaire were averaged for each question in order to extract possible areas to improve. The results are shown in table X

Table 88: Results CSUQ questionary

Question	Average
The system responds correctly to my questions	2,22
In general I was satisfied using the website.	3,78
It was easy to use this website.	4,33
I am comfortable using this website.	4,11
It was easy to learn to use this website.	4,11
I think I am now an expert using this website.	3,78
The website shows error messages that explain how to solve the problems.	1,89
Every time I make a mistake I solve it rapidly.	2,67
The information that this website uses is clear.	4,33
It is easy to find in the web the information I need.	4,33
The information that the web uses was useful to complete the tasks.	4,33
The organization of the web was clear.	4,67
The interface was pleasant.	2,22
I liked using the website.	4,33
The website responds quickly to my questions.	3,22
In general I was satisfied with the web	4,00

Analyzing the results, It seems that the users were not very satisfied with the system accuracy in answering as it obtained 2.22 in the question *The system responds correctly to my questions*.

Also it seems that the latency of response is a little slow as the question *The website responds quickly to my questions* has an average of 3.22.

Questions *Every time I make a mistake I solve it rapidly* and *The website shows error messages that*

explain how to solve the problems and shows with averages of 1.89 and 2.87 that the application fails to notify errors when they happen. The appearance of the interface was also a weak point as the question *The interface was pleasant* has an average of 2.22.

The other answers had higher averages which might imply that the system is intuitive and simple enough and that the tutorial was helpful.

Also the users gave specific advice on what they found. In Table 89 some of the suggestions were included.

Table 89: Suggestions

Suggestion	Implemented/Solved
The button of ‘how it works’ on the homepage doesn't work.	No
The table shows the results of previous questions.	Yes
The order of the answers could be included in the table.	Yes
Mic button doesn't work on mobile or firefox browser.	No
Improve Spanish translations in references.	Yes
The bot hangs when it has some error and should notify it.	Yes
Wikipedia links could be opened in another tab.	Yes
Remove scrollability in the table in mobile.	No
The text should be removed from the input box when the question is sent.	No
The colors of the interface could be more appealing. Too many gradients.	Yes

Regarding the suggestions, the interface was changed to show less gradients. They were changed by a green background. Only in the homepage, and in some details it was maintained.

The bugs regarding the table showing previous questions and the missing error message were also fixed.

The homepage tutorial button couldn’t be fixed and the microphone button is limited as the library component doesn’t have support on firefox browser. A solution couldn’t be found to the clearing of the input text in the chat box. The scrollability on mobile was found to be one of the best options although some users might dislike it. It is also the approach used by Elicit [9].

7. PROJECT ORGANIZATION

This chapter explores the socioeconomic environmet that surrounds the project as well as the related regulation. The planification followed is also described and the associated financial costs are listed.

7.1 Socioeconomic Environment

Machine learning models and research have become a part of our daily lives. Their results are used everywhere from recommendations in social media to the analysis of customers for marketing purposes and the prediction of 3D models of protein structures [99].

There is also a business interest in Machine Learning. The Global market revenue for ML is expected to grow to US$20 billion by 2025 [100]. It poses a competitive advantage as it frequently ends up in return of investing for companies that try to integrate them [101].

Neural networks training has an impact on the environment also. Just like cryptocurrency uses GPUs for the mining processes, in Machine Learning GPUs have become accessible for the general public and problem challenges and enterprise performance requirements push ML engineers to train models for longer periods of time with larger datasets with more GPUs. These GPUs are normally connected to grids powered by fossil fuels which urge the community to take into account the carbon emission of their models. In this project, at least 100 hours of GPU compute time have been used with Google cloud T4 GPUs which roughly translates to 1.89 kg of CO2 equivalent. [102]

It is not only the training of Machine Learning models, but using the internet and storing data in servers. The internet alone accounts for 4% of greenhouse gas emissions every year [103] and emits 1.6 thousand million of annual tons of greenhouse gas emissions [104].

This project relies on Machine Learning models, which in terms of interpretability are still a mystery to the research community. This could pose a problem to humanity as a whole in the near future.
Machine Learning models are becoming increasingly complex and able to perform tasks which surpass humans in a wide variety of tasks. Large language models such as GPT-3 and LaMDA [105] are able to perform at state of the art level in multiple tasks like Question Answering, Summarization, Sentiment Analysis. Also, the appearance of text-conditional image generation models like DALL-E 2 [106], Stable Diffusion [107] or Midjourney [108] are able to generate images from text. Researchers have been wondering about when the development of AI models would finally reach a point of General Artificial Intelligence, where the models are able to surpass human comprehension. For that reason, recent efforts of researchers are devoted to AI safety. A field of research in which they establish the framework and regulations to develop safe and aligned models with human goals so that they don't pose a risk for humanity. Big companies like DeepMind, OpenAI, Google or Meta are opening positions for these researchers as they understand the urgency of this area [109] [110].

7.2 Regulations

Regarding artificial intelligence there is no current regulation of what can and cannot be done although several proposals have been made. This is an example of the AI act [111]. Here they classify AI applications in three different risk categories with unacceptable risk, like government social scoring; high risk applications, like CV scanning applications and unregulated activities.

In terms of privacy it is important to respect user's privacy and their ownership of data.
In the Spanish national level, there is the Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. [112]. It states that every citizen has the right to access their data. Companies cannot make an indiscriminate use of users' data. A privacy policy must be specified by the company. It mandates the deletion of data after being processed and to inform every user how the information is processed and what information is stored from his or her personal data. In addition, it establishes that there must be a data controller and data processor and regulates infringements for non-compliance with this Law.
On the european level, there is the European General Data Protection Regulation [113], which specifies digital rights for users belonging to the European Union. It establishes that companies with European users must ask for and receive user consent, have the right to ask for the deletion of their data and prohibition of the transmission to third party companies without the consent of the user.

For this system, a privacy policy is selected:

- It complies with the Ley Orgánica de Proteccion de Datos Personales y garantía de los derechos digitales.
- It complies with the European General Data Protection Regulation.
- No information from the user is stored.
- Information introduced by the user as a query is only used to fetch answers.
- No information is shared to third parties.
- No information is used for personal profit.
- No information is used to understand behavioral user trends.

For licenses and intellectual property, Intellectual property is protected in law. It enables people to earn recognition or financial benefit from what they invent or create. By striking the right balance between the interests of innovators and the wider public interest. [114]

In this project, the materials and the software used has tried to respect all possible licenses and copyrights referencing them when possible.

7.3 Planning

In this project, the Agile development methodology has been followed. Agile is an approach to software development that came to replace the traditional waterfall model, which emphasized a logical progression of rigid stages [115].

The industry later transitioned to Agile because of the lack of feedback, rigidity to necessary changes and difficulties and no overlapping phases. The Agile methodologies promote a more flexible approach in which working software is the primary measure of progress. It is based on self organizing teams with frequent meetings to evaluate progress and set goals for the following intervals. It aims to deliver software early with continuous updates and takes into account the feedback of users, management and the rest of stakeholders [116].

Following the Agile manifesto there are several frameworks. The one used for this project is kanban [117]. In it there was a classification of tasks which are ready to start, in progress and done. The tracking of those tasks has been made with the application Trello [118]. The tasks were also divided into several sprints or time periods. However, the periods were only orientative and flexible and the process was more feature driven. At the beginning of every week, there was a reflection session to see the progress of the previous one and to schedule tasks for the following one. The project also was made thinking on the Minimum Viable Product [119], by which the idea was to develop a basic functional prototype and continuously make improvements and adding features.

The planification followed is explained here. Also, Figure 39 shows the planification as a Gnatt chart.

- **Project definition and proposal** (23/12/21 - 03/01/22)
  Process of researching various topics and looking for resources where to research the state of the art as well as the generation of the proposal.

- **State of the art research** (03/01/22 - 15/03/22)
  Researching the different topics to understand the machine learning models to implement and the technologies to implement and deploy a web application that gives access to the inference models. Also, necessary tutorials were followed to get familiarized with Pytorch and React.JS.

- **System analysis** (28/02/22 - 21/03/22)
  Process of defining the users of the system with the derived requirements, use cases and class design.

- **Initial prototype with third party modules** (21/03/22 - 28/03/22)

Construction of a basic initial prototype with basic TF-IDF from NLTK and RoBERTa [120] models already implemented.

- **Basic structure of web interface** (04/04/22 - 11/04/22)
  This sprint consists of setting up the React.JS environment and project structure, designing the basic structure of components and the implementation of the navigation bar, the footer and the different pages.

- **Advanced options web interface** (11/04/22 - 02/05/22)

Implementation of the table to show the results and the form to send the queries selecting the models.

- **References web interface** (25/07/22 - 31/07/22)
  Implementation of the references section of the web interface in both English and Spanish.

- **Chat web interface** (02/05/22 - 31/05/22)
  Creation of the chat interface in the web interface.

- **Homepage web interface** (02/05/22 - 16/05/22)
  Implementation of the initial page of the web interface and the guide section.

- **Spanish translation** (06/06/22 - 13/06/22)
  Integration of the i18next and translation of the text strings into Spanish.

- **Text to speech and speech synthesis** (13/06/22 - 27/06/22)
  Implementation of the text to speech and speech synthesis by integrating the react-speech-kit.

- **Mock server development** (23/05/22 - 06/06/22)
  Implementation of a local server integrating the already developed models with flask [121]. The purpose is to test the connection between frontend and the models using HTTP requests.

- **API REST in aws lambda** (04/06/22 - 18/06/22)
  Process of migrating the current models to the aws lambda.

- **Language Identifier** (11/04/22 - 25/04/22)
  Process of obtention of the data, preprocessing of the data, implementation of the models, experimentation with parameters and evaluation of the different configurations.

- **POS tagger** (16/05/22 - 30/05/22)
  Process of obtention of the data, preprocessing of the data, implementation of the models, experimentation with parameters and evaluation of the different configurations.

- **LSTM model (**06/06/22 - 04/07/22)
  Process of obtention of the data, preprocessing of the data, implementation of the models, experimentation with parameters and evaluation of the different configurations.

- **BERT model** (4/06/22 - 25/06/22)
  Process of obtention of the data, preprocessing of the data, implementation of the models, experimentation with parameters and evaluation of the different configurations.

- **TF-IDF (**4/04/22 - 11/04/22)
  Implementation of the TF-IDF algorithm.

- **BM25** (11/04/22 - 21/04/22)
  Implementation of the BM25 algorithm.

- **word2vec** (11/07/22 - 18/07/22)
  Process of obtaining data and training word2vec embeddings.

- **BERT embeddings** (18/07/22 - 25/07/22)
  Process of adapting a pretrained BERT model to obtain embeddings from a sentence.

- **Evaluation dataset** (01/08/22 - 14/08/22)
  Evaluation of the different configurations of ranker and reader tuple together with a specialized dataset to identify the best configuration.

- **Evaluation users** (01/08/22 - 14/08/22)
  Evaluation with users of the interface and the question answering capabilities.

- **Documentation** (15/08/22 - 04/09/22)
  Redaction of the final document describing the project to be delivered.

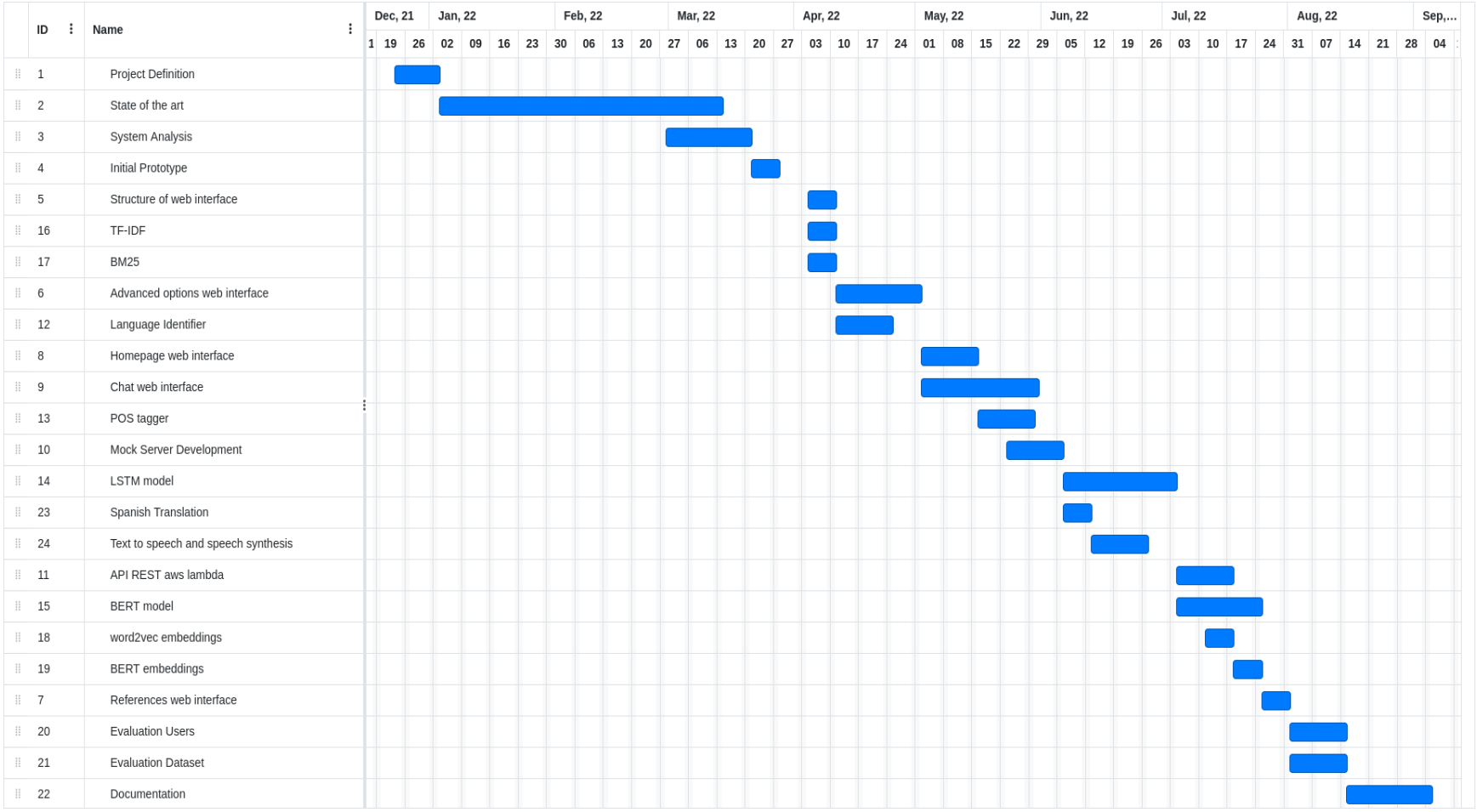


Figure 39: Gnatt Chart with the planification of the project.

7.4 Budget

This section describes the financial resources spent to develop this project. The costs are mostly related with the purchase of computational resources to train deep learning models and also their deployment.

Regarding the physical hardware, it was developed using a MSI Modern 14 A10RAS-870XES laptop with a second monitor Samsung LF24T350FHRXEN 1.002,95€ 119.99€

The cost of registering for the project is also taken into account, which is 233.16€

For the implementation, training and experimentation with the deep learning models the google platform platform was used and the pro tier was purchased during a time period of 4 months, which amounted to 37.00€.

Additionally, for the deployment 5€ were spent for a monthly droplet in DigitalOcean as an attempt to deploy the model there. Finally, Amazon was chosen as the Iaas provider. Aws offers free trials for some of their solutions. In case of aws lambda, their serverless computing service, it allows for 1000000 million invocations and 400000 GB-seconds of compute per month. After that, it is 0.000016€ for each GB-second used and 0.0000002€ for each request. We could imagine an approximate cost by neglecting the free tier and estimating 10000 petitions in a month with an upper bound of 60 seconds each request launching an image of 3GB. That would add up to 0.002€ per launching petitions and 28.8€ for compute time.

In addition the web interface has been deployed with firebase, that offers 10GB of storage and 360 MB of data transfer free per day which suffices for our static page hosting. The domain .tk has also no added cost.

The project also requires some personnel. In this case, a Project Manager, a DevOps engineer, a Front End developer and a Machine Learning engineer could be necessary for the project. Table 89 summarizes the personnel cost. The hours are extracted from the plannification while the salaries from taken from the 2022 HAYS salary survey by positions [122]. Table 89 shows the total Personnel costs and the complete cost of the project is summarized in Table 90.

Table 90: Personnel costs

Position	Hourly Salaries	Hours	Cost
Project Manager	20.19€	192	3876.01€
DevOps Developer	19.23€	224	4692.12€

FrontEnd Developer	14.42€	536	7729.12€
ML Engineer	20.19€	752	15182.88€
			TOTAL 31480.13€

Table 91: Spending summary

MSI Modern 14		1.002,95€
Monitor Smasung		119.99€
TFG fees		233.16€
AWS lambda invocations		0.002€
AWS lambda compute time		28.8€
Personnel costs		31480.13€
	TOTAL	**32865.03€**

8. CONCLUSIONS AND FUTURE WORK

In this section the conclusions and the future directions are described.

8.1 Conclusions

In this project a Open Domain Question Answering system was developed and deployed as a web application for users to ask questions.

To answer user questions, research was made on the basics of Machine Learning and Natural Language Processing and several methods were implemented like TF-IDF, BM25, word2vec, BERT or Encoder Decoder architectures. The models were trained and a comparison was made. From them it resulted that the system works best with a BM25 document ranker and a BERT(L=12/H=768) document reader.

Also, a web application was designed to allow the users to use the model. To do so, users were defined and use cases identified. From them, a requirement elicitation was made in order to define the functionality of the system. Lastly the classes were defined and the methods with their interactions in a sequence diagram.

The application was built using the React.JS framework and it was deployed using Google's Firebase. The web interface accessed the models using an API REST thanks to AWS lambda, a serverless solution which allows the application to be scalable. This solution had some limitations with the memory size of the container image which caused the use of BM25 ranker and the BERT(L=8/H=512).

An evaluation of the whole question answering pipeline was performed with the use of the dataset TriviaQA and found that the best configuration was to use the library Sentence Transformers to order documents and to use BERT(L=12/H=768). The user evaluation provided useful information about what features should be added or modified in the interface. Also showed evidence that the models generated in English are superior to those in Spanish.

8.2 Future Work

During the project there were some decisions made that could be improved in the future.

1. The REST API was implemented using AWS lambda. The memory limit should be troubleshooted or an alternative to this solution should be found like Google's Cloud Run.
2. In terms of the ranker, the creation of a dense representation based ranker was attempted but not successful. In the future a BERT model should be finetuned for sentence similarity to achieve near performance to the Sentence Transformers library.
3. With respect to the Spanish language, they underperformed the English ones which could be because the datasets were automatically translated. More research should be made in order to find or create better training resources.
4. For evaluation, F1-score and Exact Match were used. However, these are metrics that rely on matching tokens exactly but they don't take into account semantic relations between words that are not exactly the same like synonimity. In this paper [123] they

present a score based on BERT embeddings: BERTScore. More research should be done on this area and try to evaluate models based on contextual embeddings.

BIBLIOGRAPHY

[1] "acl2020-openqa-tutorial/simmons1965.pdf at master · danqi/acl2020-openqa-tutorial", GitHub, 2022. [Online]. Available: https://github.com/danqi/acl2020-openqa-tutorial/blob/master/docs/simmons1965.pdf. (Accessed: 05- Sep- 2022).

[2] D. Ferrucci et al., "Building Watson: An Overview of the DeepQA Project", AI Magazine, vol. 31, no. 3, p. 59, 2010. Available: 10.1609/aimag.v31i3.2303.

[3] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to Answer Open-Domain Questions," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, Canada, 2017, pp. 1870–1879. doi: 10.18653/v1/P17-1171.

[4] D. Chen and W. Yih, "Open-Domain Question Answering," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, Online, Jul. 2020, pp. 34–37. doi: 10.18653/v1/2020.acl-tutorials.8.

[5] B. Mott, J. Lester, and K. Branting, "Conversational Agents," Chapman & Hall/CRC Computer & Information Science Series, 2004, Accessed: Sep. 05, 2022. [Online]. Available: https://www.academia.edu/2756336/Conversational_agents

[6] J. Prager, "Open-Domain Question–Answering," FNT in Information Retrieval, vol. 1, no. 2, pp. 91–231, 2006, doi: 10.1561/1500000001.

[7] T. B. Brown et al., "Language Models are Few-Shot Learners." arXiv, Jul. 22, 2020. doi: 10.48550/arXiv.2005.14165.

[8] A. Chowdhery et al., "PaLM: Scaling Language Modeling with Pathways." arXiv, Apr. 19, 2022. doi: 10.48550/arXiv.2204.02311.

[9] https://elicit.org/ (accessed Sep. 05, 2022).

[10] "Ought." https://ought.org/ (accessed Sep. 05, 2022).

[11] C. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." arXiv, Jul. 28, 2020. doi: 10.48550/arXiv.1910.10683.

[12] https://elicit.org/faq (accessed Sep. 05, 2022).

[13] "Google Assistant, your own personal Google," Assistant. https://assistant.google.com/ (accessed Sep. 05, 2022).

[14] "Siri," Apple (España). https://www.apple.com/es/siri/ (accessed Sep. 05, 2022).

[15] "Amazon Alexa Official Site: What is Alexa?," Amazon (Alexa). https://developer.amazon.com/es-ES/alexa.html (accessed Sep. 05, 2022).

[16] “Introduction to Machine Learning.” https://ai.stanford.edu/people/nilsson/mlbook.html (accessed Sep. 05, 2022).

[17] D. Jurafsky, “Neural Networks and Neural Language Models” in Speech and Language Processing.: 3rd Edition. Upper Saddle River, NJ, 2021.

[18] V. Jain, “Everything you need to know about ‘Activation Functions’ in Deep learning models,” Medium, Dec. 30, 2019. https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253 (accessed Sep. 05, 2022).

[19] “Softmax Activation Function: Everything You Need to Know,” Pinecone. https://www.pinecone.io/learn/softmax-activation/ (accessed Sep. 05, 2022).

[20] D. Jurafsky, “Logistic Regression” in Speech and Language Processing.: 3rd Edition. Upper Saddle River, NJ, 2021.

[21] A. V. Srinivasan, “Stochastic Gradient Descent — Clearly Explained !!,” Medium, Sep. 07, 2019. https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31 (accessed Sep. 05, 2022).

[22] D. Jurafsky, “Deep Learning Architectures for Sequence Processing” in Speech and Language Processing.: 3rd Edition. Upper Saddle River, NJ, 2021.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.

[24] D. Jurafsky, “Transfer Learning with Pretrained Language Models and Contextual Embeddings” in Speech and Language Processing.: 3rd Edition. Upper Saddle River, NJ, 2021.

[25] "GitHub - google-research/bert: TensorFlow code and pre-trained models for BERT", GitHub, 2022. [Online]. Available: https://github.com/google-research/bert. [Accessed: 05- Sep- 2022].

[26] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, “Natural language processing: an introduction,” J Am Med Inform Assoc, vol. 18, no. 5, pp. 544–551, Sep. 2011, doi: 10.1136/amiajnl-2011-000464.

[27] D. Jurafsky, “Vector Semantics and Embeddings” in Speech and Language Processing.: 3rd Edition. Upper Saddle River, NJ, 2021.

[28] D. Jurafsky, “Naive Bayes and Sentiment Classification” in Speech and Language Processing.: 3rd Edition. Upper Saddle River, NJ, 2021.

[29] N. Craswell, “Mean Reciprocal Rank,” in Encyclopedia of Database Systems, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 1703–1703. doi: 10.1007/978-0-387-39940-9_488.

[30] R. J. Tan, “Breaking down Mean Average Precision (mAP),” Medium, Mar. 02, 2022. https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52 (accessed Sep. 05, 2022).

[31] “Evaluating QA: Metrics, Predictions, and the Null Response,” NLP for Question Answering, Jun. 09, 2020. https://qa.fastforwardlabs.com/no%20answer/null%20threshold/bert/distilbert/exact%20match/f1/robust%20predictions/2020/06/09/Evaluating_BERT_on_SQuAD.html (accessed Sep. 05, 2022).

[32] “The Stanford Question Answering Dataset.” https://rajpurkar.github.io/SQuAD-explorer/ (accessed Sep. 05, 2022).

[33] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, “TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension,” in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, Canada, 2017, pp. 1601–1611. doi: 10.18653/v1/P17-1147.

[34] M. Dunn, L. Sagun, M. Higgins, V. Guney, V. Cirik and K. Cho, "SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine", arXiv.org, 2022. [Online]. Available: https://arxiv.org/abs/1704.05179. [Accessed: 05- Sep- 2022].

[35] “Google’s Natural Questions.” https://ai.google.com/research/NaturalQuestions/ (accessed Sep. 05, 2022).

[36] P. Bajaj et al., “MS MARCO: A Human Generated MAchine Reading COmprehension Dataset.” arXiv, Oct. 31, 2018. doi: 10.48550/arXiv.1611.09268.

[37] Marcus, Mitchell P., Santorini, Beatrice, Mary Ann Marcinkiewicz, and Taylor, Ann, “Treebank-3.” Linguistic Data Consortium, 1999. doi: 10.35111/GQ1X-J780.

[38] “Language Detection.” https://www.kaggle.com/datasets/basilb2s/language-detection (accessed Sep. 06, 2022).

[39] “Wikicorpus, v. 1.0: Catalan, Spanish and English portions of the Wikipedia.” https://www.cs.upc.edu/~nlp/wikicorpus/ (accessed Sep. 06, 2022).

[40] “PyTorch.” https://www.pytorch.org (accessed Sep. 06, 2022).

[41] “TensorFlow,” TensorFlow. https://www.tensorflow.org/?hl=es-419 (accessed Sep. 06, 2022).

[42] “🤗 Transformers.” https://huggingface.co/docs/transformers/index (accessed Sep. 06, 2022).

[43] “NLTK :: Natural Language Toolkit.” https://www.nltk.org/ (accessed Sep. 06, 2022).

[44] “Gensim: topic modelling for humans.” https://radimrehurek.com/gensim/ (accessed Sep. 06, 2022).

[45] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space.” arXiv, Sep. 06, 2013. doi: 10.48550/arXiv.1301.3781.

[46] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information.” arXiv, Jun. 19, 2017. doi: 10.48550/arXiv.1607.04606.

[47] J. Pennington, R. Socher, and C. Manning, “GloVe: Global Vectors for Word Representation,” in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, Oct. 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.

[48] “NumPy.” https://numpy.org/ (accessed Sep. 06, 2022).

[49] “pandas - Python Data Analysis Library.” https://pandas.pydata.org/ (accessed Sep. 06, 2022).

[50] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” arXiv, Aug. 27, 2019. doi: 10.48550/arXiv.1908.10084.

[51] “Difference Between Server-side Scripting and Client-side Scripting (with Comparison Chart),” Tech Differences, May 29, 2018. https://techdifferences.com/difference-between-server-side-scripting-and-client-side-scripting.html (accessed Sep. 06, 2022).

[52] “Web Components | MDN.” https://developer.mozilla.org/es/docs/Web/Web_Components (accessed Sep. 06, 2022).

[53] “React – Una biblioteca de JavaScript para construir interfaces de usuario.” https://es.reactjs.org/ (accessed Sep. 06, 2022).

[54] “Angular.” https://angular.io/ (accessed Sep. 06, 2022).

[55] “Vue.js - The Progressive JavaScript Framework | Vue.js.” https://vuejs.org/ (accessed Sep. 06, 2022).

[56] “i18next” https://www.i18next.com/ (accessed Sep. 06, 2022).

[57] M. O. contributors Jacob Thornton, and Bootstrap, “Bootstrap.” https://getbootstrap.com/ (accessed Sep. 06, 2022).

[58] “SpeechSynthesis - WebAPIs | MDN.” https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis (accessed Sep. 06, 2022).

[59] “SpeechRecognition - Web APIs | MDN.” https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition (accessed Sep. 06, 2022).

[60] “react-speech-kit,” npm. https://www.npmjs.com/package/react-speech-kit (accessed Sep. 06, 2022).

[61] “Material Design,” Material Design. https://material.io/designhttps://material.io/design (accessed Sep. 06, 2022).

[62] H. Nielsen et al., “Hypertext Transfer Protocol – HTTP/1.1,” Internet Engineering Task Force, Request for Comments RFC 2616, Jun. 1999. Accessed: Sep. 06, 2022. [Online]. Available: https://datatracker.ietf.org/doc/rfc2616/

[63] “Simple Object Access Protocol (SOAP) 1.1.”

https://www.w3.org/TR/2000/NOTE-SOAP-20000508/ (accessed Sep. 06, 2022).

[64] “Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST).” https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (accessed Sep. 06, 2022).

[65] “What is Scalability? - Definition from Techopedia,” Techopedia.com. http://www.techopedia.com/definition/9269/scalability (accessed Sep. 06, 2022).

[66] “Software scalability and how is it better in custom software.” https://massmediagroup.pro/blog-mmg/software-scalability-&-how-is-it-better-in-custom-software (accessed Sep. 06, 2022).

[67] N-able, “On-Premises vs. Cloud Security: Key Differences,” N-able, Oct. 10, 2019. https://www.n-able.com/blog/on-premise-vs-cloud-security (accessed Sep. 06, 2022).

[68] “Best Practices for Application Deployment in 2020,” EngineYard, Oct. 15, 2020. https://www.engineyard.com/blog/best-practices-for-application-deployment/ (accessed Sep. 06, 2022).

[69] B. Antoniony, “Definición de IaaS, PaaS y SaaS ¿En qué se diferencian?” https://www.ambit-bst.com/blog/definición-de-iaas-paas-y-saas-en-qué-se-diferencian (accessed Sep. 06, 2022).

[70] “Cloud Computing Services - Amazon Web Services (AWS),” Amazon Web Services, Inc. https://aws.amazon.com/ (accessed Sep. 06, 2022).

[71] “Cloud Computing Services | Microsoft Azure.” https://azure.microsoft.com/en-us/ (accessed Sep. 06, 2022).

[72] “Cloud Computing Services,” Google Cloud. https://cloud.google.com/ (accessed Sep. 06, 2022).

[73] “Plataforma de aplicaciones App Engine | App Engine,” Google Cloud. https://cloud.google.com/appengine?hl=es (accessed Sep. 06, 2022).

[74] “Bungee Connect.” https://www.bungeeconnect.com/ (accessed Sep. 06, 2022).

[75] “Office 365 | Microsoft Office.” https://www.office.com/ (accessed Sep. 06, 2022).

[76] “Herramienta de blog, plataforma de publicación y CMS,” WordPress.org España. https://es.wordpress.org/ (accessed Sep. 06, 2022).

[77] “What is Docker?” https://www.ibm.com/cloud/learn/docker (accessed Sep. 06, 2022).

[78] “Figma: the collaborative interface design tool.,” Figma. https://www.figma.com/ (accessed Sep. 06, 2022).

[79] J. Ferrero, F. Agnès, L. Besacier, and D. Schwab, “A Multilingual, Multi-Style and Multi-Granularity Dataset for Cross-Language Textual Similarity Detection,” Portorož, Slovenia,

May 2016.

[80] H. Takcı and İ. Soğukpınar, “Letter Based Text Scoring Method for Language Identification,” in Advances in Information Systems, Berlin, Heidelberg, 2005, pp. 283–290. doi: 10.1007/978-3-540-30198-1_29.

[81] R. H. Abbas, F. A. E. A. Kareem, R. H. Abbas, and F. A. E. A. Kareem, “Text Language Identification Using Letters (Frequency, Self-information, and Entropy) Analysis for English, French, and German Languages,” Journal of Southwest Jiaotong University, vol. 54, no. 4, 2019, Accessed: Sep. 06, 2022. [Online]. Available: http://jsju.org/index.php/journal/article/view/334

[82] “Sequence Models and Long Short-Term Memory Networks — PyTorch Tutorials 1.12.1+cu102 documentation.” https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html (accessed Sep. 06, 2022).

[83] D. Jurafsky, “Question Answering.” in Speech and Language Processing.: 3rd Edition. Upper Saddle River, NJ, 2021.

[84]“Gensim: topic modelling for humans.” https://radimrehurek.com/gensim/models/word2vec.html (accessed Sep. 06, 2022).

[85] V. Karpukhin et al., “Dense Passage Retrieval for Open-Domain Question Answering.” arXiv, Sep. 30, 2020. doi: 10.48550/arXiv.2004.04906.

[86]“The Stanford Natural Language Inference (SNLI) Corpus.” https://nlp.stanford.edu/projects/snli/ (accessed Sep. 06, 2022).

[87] “sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 · Hugging Face.” https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 (accessed Sep. 06, 2022).

[88] “bert-base-uncased · Hugging Face.” https://huggingface.co/bert-base-uncased (accessed Sep. 06, 2022).

[89] Y. Wu et al., “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.” arXiv, Oct. 08, 2016. doi: 10.48550/arXiv.1609.08144.

[90] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, “Bidirectional Attention Flow for Machine Comprehension.” arXiv, Jun. 21, 2018. doi: 10.48550/arXiv.1611.01603.

[91] N. N, “Question Answering System with BERT,” Analytics Vidhya, Jul. 30, 2020. https://medium.com/analytics-vidhya/question-answering-system-with-bert-ebe1130f8def (accessed Sep. 06, 2022).

[92] “transformers/examples/legacy/question-answering at main · huggingface/transformers,” GitHub. https://github.com/huggingface/transformers (accessed Sep. 06, 2022).

[93] "GitHub - google-research/bert: TensorFlow code and pre-trained models for BERT", GitHub, 2022. [Online]. Available: https://github.com/google-research/bert. (Accessed: 06- Sep- 2022).

[94] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, and J. Pérez, “Spanish Pre-Trained BERT Model and Evaluation Data,” 2020.

[95] J. Goldsmith, “wikipedia: Wikipedia API for Python.” Accessed: Sep. 06, 2022. [Online]. Available: https://github.com/goldsmith/Wikipedia

[96] “Registro de contenedores completamente administrado - Amazon Elastic Container Registry - Amazon Web Services,” Amazon Web Services, Inc. https://aws.amazon.com/es/ecr/ (accessed Sep. 06, 2022).

[97] “Serverless: Develop & Monitor Apps On AWS Lambda.” http://serverless.com// (accessed Sep. 06, 2022).

[98] “Computer System Usability Questionnaire.” https://garyperlman.com/quest/quest.cgi (accessed Sep. 06, 2022).

[99] “AlphaFold.” https://www.deepmind.com/research/highlighted-research/alphafold (accessed Sep. 06, 2022).

[100] “Business Impacts of Machine Learning,” Deloitte Turkey. https://www2.deloitte.com/tr/en/pages/strategy-operations/articles/business-impacts-of-machine-learning.html (accessed Sep. 06, 2022).

[101] “Machine Learning: The New Proving Ground for Competitive Advantage,” MIT Technology Review. https://www.technologyreview.com/2017/03/16/106260/machine-learning-the-new-proving-ground-for-competitive-advantage/ (accessed Sep. 06, 2022).

[102] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, “Quantifying the Carbon Emissions of Machine Learning.” arXiv, Nov. 04, 2019. doi: 10.48550/arXiv.1910.09700.

[103] T. S. Project, “‘Climate crisis: The Unsustainable Use of Online Video’ : Our new report,” The Shift Project, Jul. 10, 2019. https://theshiftproject.org/en/article/unsustainable-use-online-video/ (accessed Sep. 06, 2022).

[104] S. Griffiths, “Why your internet habits are not as clean as you think.” https://www.bbc.com/future/article/20200305-why-your-internet-habits-are-not-as-clean-as-you-think (accessed Sep. 06, 2022).

[105] R. Thoppilan et al., “LaMDA: Language Models for Dialog Applications.” arXiv, Feb. 10, 2022. doi: 10.48550/arXiv.2201.08239.

[106] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical Text-Conditional Image Generation with CLIP Latents.” arXiv, Apr. 12, 2022. doi: 10.48550/arXiv.2204.06125.

[107] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-Resolution Image Synthesis With Latent Diffusion Models,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2022, pp. 10684–10695.

[108] “Midjourney,” Midjourney. https://www.midjourney.com/home/ (accessed Sep. 06, 2022).

[109] “AI risk - EA Forum.” https://forum.effectivealtruism.org/topics/ai-risk (accessed Sep. 06, 2022).

[110] “Paul Christiano: Current Work in AI Alignment | Effective Altruism.” https://www.effectivealtruism.org/articles/paul-christiano-current-work-in-ai-alignment (accessed Sep. 06, 2022).

[111] “Home,” The Artificial Intelligence Act, Sep. 07, 2021. https://artificialintelligenceact.eu/ (accessed Sep. 06, 2022).

[112] Jefatura del Estado, Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, vol. BOE-A-2018-16673. 2018, pp. 119788–119857. Accessed: Sep. 06, 2022. [Online]. Available: https://www.boe.es/eli/es/lo/2018/12/05/3

[113] “General Data Protection Regulation (GDPR) – Official Legal Text,” General Data Protection Regulation (GDPR). https://gdpr-info.eu/ (accessed Sep. 06, 2022).

[114] “What is Intellectual Property (IP)?” https://www.wipo.int/about-ip/en/index.html (accessed Sep. 06, 2022).

[115] “Software Engineering | Classical Waterfall Model,” GeeksforGeeks, Mar. 18, 2018. https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/ (accessed Sep. 06, 2022).

[116] “12 Principles Behind the Agile Manifesto | Agile Alliance,” Agile Alliance |, Nov. 04, 2015. https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/ (accessed Sep. 06, 2022).

[117] “Different Types of Agile Methodologies: Find Which One Fits Best Your Needs,” Kanbanize Blog, Nov. 02, 2021. https://kanbanize.com/blog/different-types-of-agile-methodologies/ (accessed Sep. 06, 2022).

[118] “Manage Your Team’s Projects From Anywhere | Trello.” https://trello.com/ (accessed Sep. 06, 2022).

[119] H. McCloskey, “Minimum Viable Product (MVP).” https://www.productplan.com/glossary/minimum-viable-product/ (accessed Sep. 06, 2022).

[120]“deepset/xlm-roberta-base-squad2 · Hugging Face.” https://huggingface.co/deepset/xlm-roberta-base-squad2 (accessed Sep. 06, 2022).

[121] “Welcome to Flask — Flask Documentation (2.2.x).” https://flask.palletsprojects.com/en/2.2.x/ (accessed Sep. 06, 2022).

[122] M. D. A. SL and MOTTO, “Guía del Mercado Laboral 2022 - Hays.” https://guiasalarial.hays.es/empresa/home (accessed Sep. 06, 2022).

[123] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT.” arXiv, Feb. 24, 2020. doi: 10.48550/arXiv.1904.09675.

[124] “Glasgow IDOM - CISI collection.” http://ir.dcs.gla.ac.uk/resources/test_collections/cisi/ (accessed Sep. 06, 2022).

[125] “University of Glasgow - Schools - School of Computing Science - Research - Research sections - IDA-Section - Information Retrieval.” https://www.gla.ac.uk/schools/computing/research/researchsections/ida-section/informationretrieval/ (accessed Sep. 06, 2022).

[126] C. P. Carrino, M. R. Costa-jussà, and J. A. R. Fonollosa, “Automatic Spanish Translation of the SQuAD Dataset for Multilingual Question Answering.” arXiv, Dec. 12, 2019. doi: 10.48550/arXiv.1912.05200.

APPENDIX A. GLOSSARY

QA	Question Answering
TF-IDF	Term Frequency - Inverse Document Frequency
LSTM	Long Short-Term Memory
TFG	Trabajo de Fin de Grado
UC3M	Universidad Carlos III de Madrid
BERT	Bidirectional Encoder Representations from Transformers
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
REST	Representational State Transfer
POS	Part Of Speech
AWS	Amazon Web Services
NLP	Natural Langauge Processing
ML	Machine Learning
GPT-3	Generative Pre-trained Transformer 3
PaLM	Pathways Language Model
SVM	Support Vector Machine
ReLU	Rectified Linear Unit
CEL	Cross Entropy Loss
SGD	Stochastic Gradient Descent
RNN	Recurrent Neural Network

ST	Sentence Transformers
BPE	Bynary Pair Encoding
EM	Exact Match
MRR	Mean Reciprocal Rank
MAP	Mean Average Precision
SQuAD	Stanford Question Answering Dataset
CSS	Cascading Style Sheets
JS	Javascript
SOAP	Simple Object Access Control
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
NLTK	Natural Language Toolkit

APPENDIX B. USER SURVEY AND RESULTS

You have been invited to participate in a research project carried out by a student at Universidad Carlos III de Madrid for the purpose of preparing the Final Degree Project. In order for me to use any information I collect, I must have your consent.
You must use a device with internet access to fill in the form. In this survey you will: Access a web page
Answer questions
Share ideas on how you could improve the design of the application Comment on other observations about design improvements

The researcher will be able to observe your comments and feedback. You will be able to leave the process at any time.

*Obligatorio

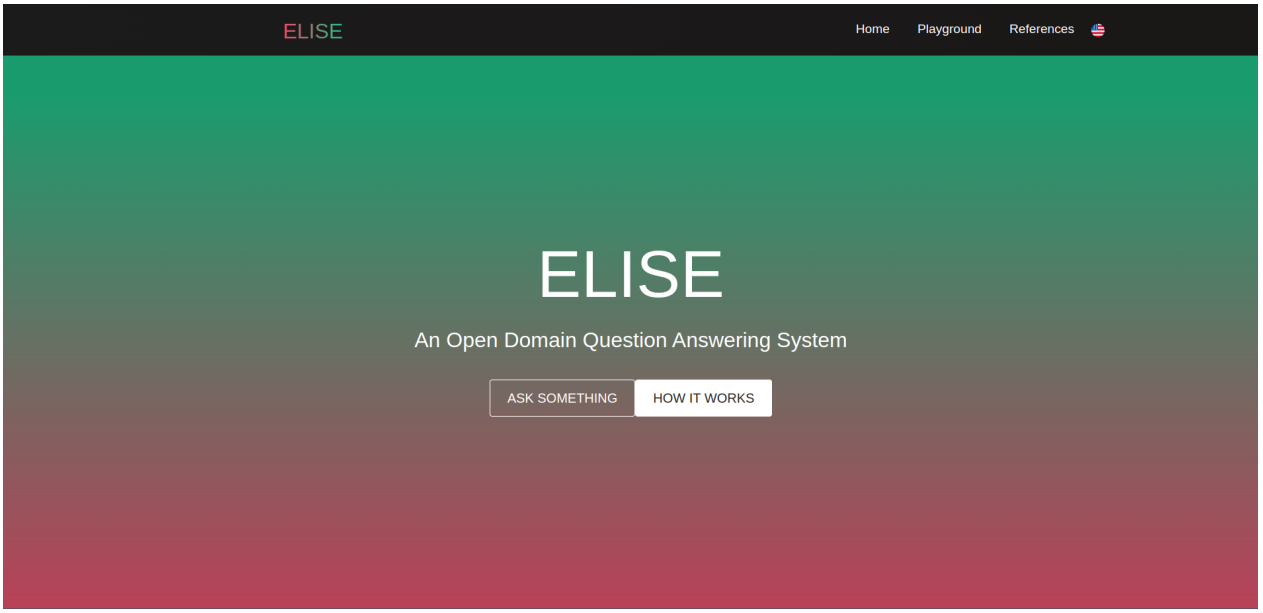
1. I agree that the information I provide on this form may be processed for purposes of assisting in the design of the application. *

   Marca solo un óvalo.

   Yes

   No

https://elise.tk/



The study will have two parts. One is aimed at evaluating the accuracy of Elise and the other is to evaluate the interface and its usability.

2. Which device are you using?

Marca solo un óvalo.

PC

Cell phone

Tablet

Otro: _____

Chat interface

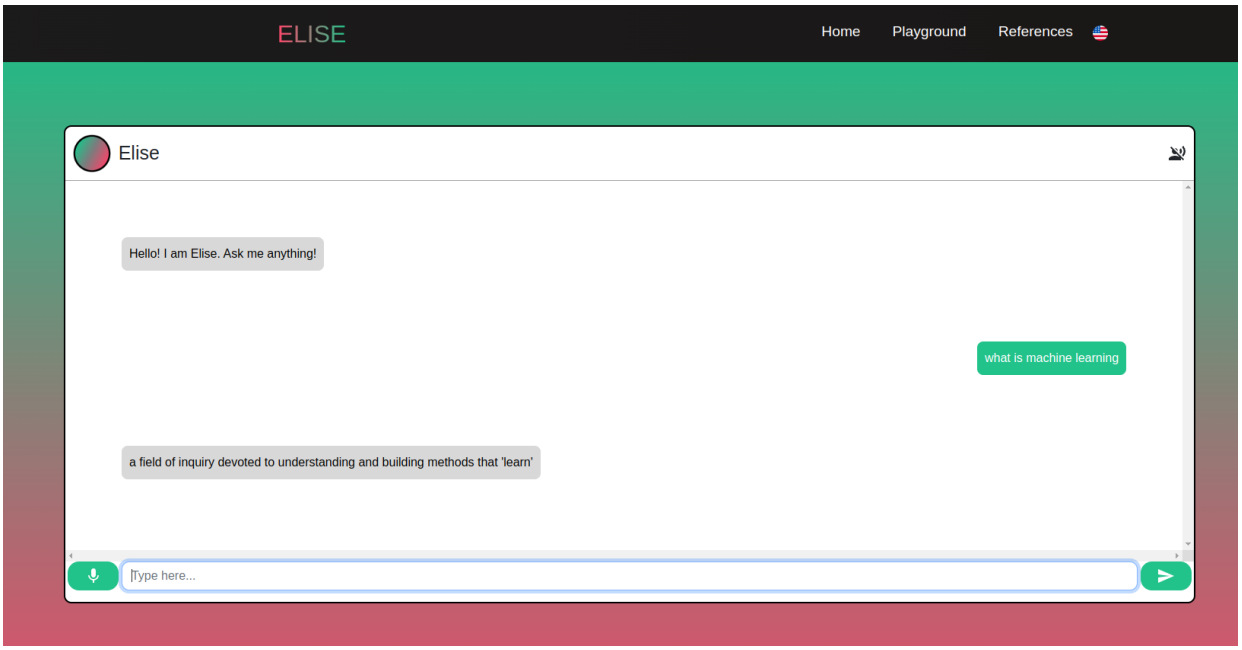
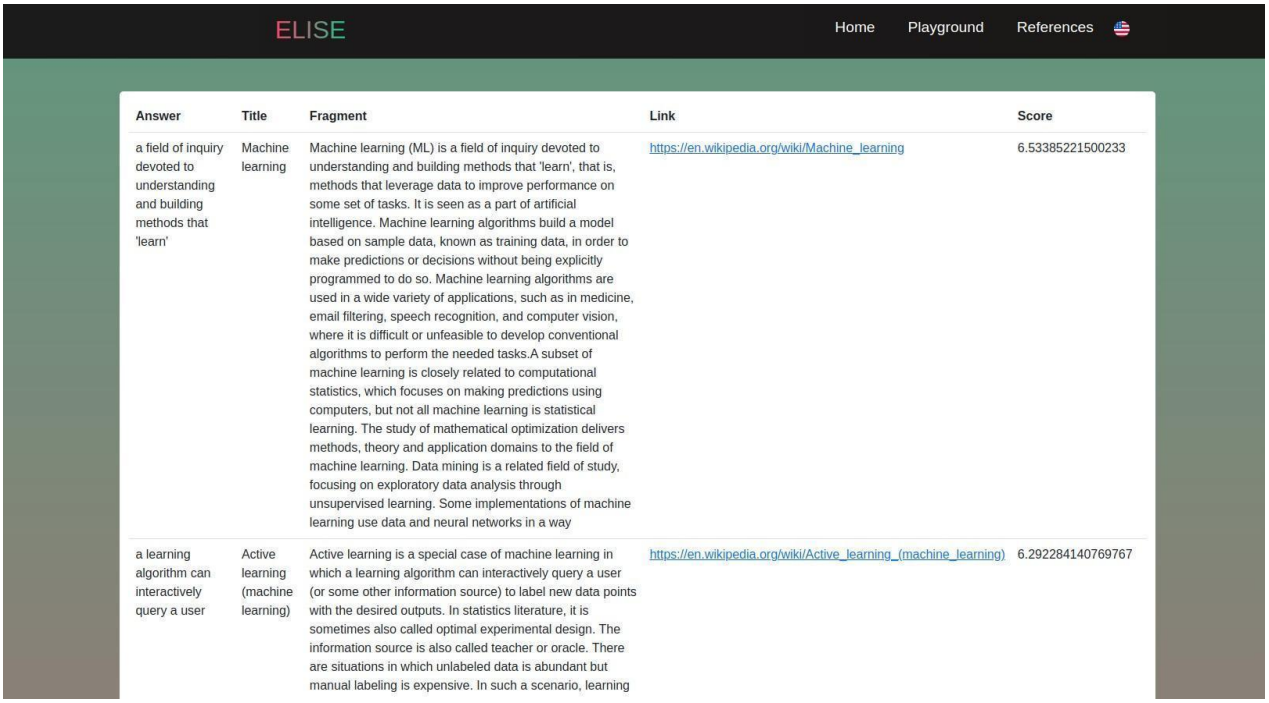


Table results



3. Write question, expected answer and Elise answer #1 in English

4. Does a valid answer to the question #1 appear in the___?

   Marca solo un óvalo.

   - Top 1 result
   - Top 3 results
   - Top 5 results
   - Top 10 results
   - It doesn't appear

5. Write question, expected answer and Elise answer #2 in English

6. Does a valid answer to the question #2 appear in the___?

   Marca solo un óvalo.

   - Top 1 result
   - Top 3 results
   - Top 5 results
   - Top 10 results
   - It doesn't appear

7. Write question, expected answer and Elise answer #3 in English

10. Does a valid answer to the question #1 appear in the___?

Marca solo un óvalo.

- Top 1 results
- Top 3 results
- Top 5 results
- Top 10 results
- It doesn't appear

11. Write question, expected answer and Elise answer #2 in Spanish

14. Does a valid answer to the question #3 appear in the___?

Marca solo un óvalo.

- Top 1 results
- Top 3 results
- Top 5 results
- Top 10 results
- It doesn't appear

15. If you asked any other questions it would be great to know them and how did Elise answer.

17. In general I was satisfied using the website.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

18. It was easy to use this website.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

19. I am comfortable using this website.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

21. I think I am now an expert using this website.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

22. The website shows error messages that explain how to solve the problems.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

23. Every time I make a mistake I solve it rapidly.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

25. It is easy to find in the web the information I need.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

26. The information that the web uses was useful to complete the tasks.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

27. The organization of the web was clear.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

29. I liked using the website.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

30. The website responds quickly to my questions.

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

31. In general I was satisfied with the web

Marca solo un óvalo.

1 2 3 4 5

Completely disagree Completely agree

Thank you for your time taking this survey :)

32. If you found any problem or bug that you would like to report or any suggestion for improvement, you can do this here.

Which device are you using?
PC
Cell phone
Cell phone
Cell phone
PC
Cell phone
Cell phone
PC
Cell phone

Write question, expected answer and Elise answer #1 in English
Who is the author of the manga one piece?, Eichiro Oda, Oda?
who was the first yo land on the moon, apollo 11 or Neil Amstrong, Lunokhod 1
q:Who won the champions league on 2022? ea: Real Madrid Ea: Who won the champions league on 2022?

what is the meaning of life? I just want to see what answers What is the meaning of my life
When will be the end of the world a new era
who are the members of guns and roses? Axl Rose,Tracii Guns, Izzy Stradlin,Ole Beich and Rob Gardner Gilby Clarke
when did Leonardo Dicaprio win the oscar? 2017 February 26, 2017
Who is the creator of Dark Souls? Miyazaki. Lord Gwyn
Where is oviedo? North of Spain Red Bug Lake Road

Does a valid answer to the question #1 appear in the __ ?
1
3
3
3
20
3
1
3
1

Write question, expected answer and Elise answer #2 in English
What is the capital of Argentina?, Buenos Aires, Srinagar
Who is the antagonist in Phineas and Ferb?, Dofensmirth, Candace

q: Where does Cristiano Ronaldo play? ea: Manchester United Ea: convention
when was America discovered? 1492 when she was young
What is the function of the hypocampus in the brain? it doesn't respond
what is music, , The Sound of the 80's
Who is the fastest man ever? Usain Bolt Justin Gatlin
What is the record of a Dark Souls speedrun? Around 20 minutes, 2011
what is covid? A pandemic. Asthma

Does a valid answer to the question #2 appear in the __ ?
5
1
3
10
20
20
20
20
3

Write question, expected answer and Elise answer #3 in English
What are the colors of the Spanish flag?, red and yellow, three equally wide horizontal bands coloured light blue and white
distance from the Earth to the Moon, 384.400km, one-half a degree

q: Who is bad bunny? ea: artist/singer Ea:begging"
who built the eiffel tower Gustave Eiffel Gustave Eiffel
what is osteoporosis a disease of bone where there is reduced bone mineral density
What is the debut album of billie eilish. When We All Fall Asleep, Where Do We Go?, 1 Happier Than Ever
who painted the mona lisa? Leonardo Da Vinci Fernando Botero
what is coffee? Specialty coffee is a term for the highest grade of coffee available
Who created covid-19? It is uncertain. US National Institute for Occupational Safety and Health

Does a valid answer to the question #3 appear in the __ ?
20
5
3
1
1
3
3
5
10

Write question, expected answer and Elise answer #1 in Spanish
Quién es el hombre más rico del mundo?, Elon Musk, Hafid

esperanza de vida media en Japón, no lo se, 13]
q:Quién escribió el quijote? ea: Cervantes Ea: Tulio Febres Cordero
quién es la mejor actriz española? Penélope cruz Katina Paxinou
En qué año fue la guerra de independencia de Estados Unidos. 1811
número de cuerdas de una guitarra, 6, cinco órdenes
Quién mató a la mujer de Ned Flanders Homer Wayne Slater
por qué es Einstein famoso? Por ser uno de los mejores físicos de la historia. el radio de Einstein
quién es el dueño de amazon? Jeff Bezos. Arlene Fowler

Does a valid answer to the question #1 appear in the __ ?
20
20
20
3
5
10
20
10
20

Write question, expected answer and Elise answer #2 in Spanish

Cuántos gramos de proteína tiene un huevo?, 13 gramos, 56
donde está Francia, europa, Francia continental europea
q:Quién escribió el quijote? ea: Cervantes Ea: Pierre Menard
cuántos anillos ganados tiene Michael Jordan? 6 seis anillos
qué es la belleza? una idea
cómo se creó el sol?, a partir de una nube de gas y otras partículas, segunda terminal de pasajeros
Dónde se puede comprar una Big Mac Macdonads 2013
Esiste Dios? No, «Dios existe porque dice en este libro que existe
Cuántos minutos dura un partido de fútbol? 90. 15 minutos

Does a valid answer to the question #2 appear in the __ ?
20
3
20
1
1
20
20
1
3

Write question, expected answer and Elise answer #3 in Spanish

Cuál es el sentido de la vida?, no esperaba ninguna, Platonismo
que es la bachata, un tipo de baile, «música de amargue
q:Cuáles son los orígenes del rap ea: las calles de nueva york Ea: rap metal
Quién es Pau Gasol? Jugador de baloncesto Pau Gasol aportó con 33 puntos y 14 rebotes
quién era blas de lezo Fragata Blas de Lezo
Cuántas caloría debe consumir una persona diariamente?, 2.000 y 2.500, dieta específica
Cómo se hace un mortal Saltando (2011
Cuántas piezas de fruta hay que comer al día? 5, 4

Does a valid answer to the question #3 appear in the __ ?
10
3
5
3
5
20
20
1

The system responds correctly to my questions	3	2	2	4	2	2	1	1	3
In general I was satisfied using the website.	4	3	4	4	4	3	4	4	4
It was easy to use this website.	4	4	5	2	5	4	5	5	5

I am comfortable using this website.	4	4	4	4	5	3	4	4	5
It was easy to learn to use this website.	3	5	3	4	4	5	4	5	4
I think I am now an expert using this website.	2	4	4	3	4	4	4	4	5
The website shows error messages that explain how to solve the problems.	1	2	1	2	3	3	3	1	1
Every time I make a mistake I solve it rapidly.	2	2	3	2	3	3	4	3	2
The information that this website uses is clear.	4	4	4	4	5	4	5	4	5
It is easy to find in the web the information I need.	4	4	4	5	5	5	5	4	3
The information that the web uses was useful to complete the tasks.	4	4	4	5	5	4	4	5	4
The organization of the web was clear.	4	5	4	4	5	5	5	5	5
The interface was pleasant.	2	2	1	1	2	3	4	2	3
I liked using the website.	4	5	3	4	4	5	5	5	4
The website responds quickly to my questions.	3	3	4	2	4	2	2	4	5
In general I was satisfied with the web	4	3	4	4	4	4	3	5	5

If you found any problem or bug that you would like to report or any suggestion for improvement, you can do this here.
The button of how it works doesn't work. The table shows results of previous questions. It is nice if you include the order of the answer in the table.
Creo que usar otros colores en la página la podría quedar mejor. No responde muy bien a las preguntas pero me ha resultado entretenido.
- El botón del micro no funciona en el móvil. - La tabla tiene resultados de preguntas anteriores también. - En las referencias algunas traducciones suenan un poco raras en español.
sometimes the bot hangs and it doesn't say why, it just shows three points and the table contained answers from other questions as well.

It would be best if the table was shown entirely and not have to move it, at least on mobile phone.
Al pular el link the la wikipedia la página cambia. Estaría bien que se abriese en una nueva pestaña. También cuando mandas un mensaje, el texto se queda en la barra. Quedaría bien si se eliminase.
The table has bugs. It shows answers from before. If you don't write a word well it just hangs.