TÉCNICO LISBOA

Neural Retrieval Models for Matching Patients to Clinical Trials

João da Costa Pereira

Thesis to obtain the Master of Science Degree in

Data Science and Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. João Miguel da Costa Magalhães

Examination Committee

Chairperson: Prof. Maria da Conceição Esperança Amado
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Prof. Fernando Manuel Marques Batista

November 2022

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisors, Prof. Bruno Martins and Prof. João Magalhães, for their invaluable advice, continuous support, and dedicated involvement during my dissertation. Their immense knowledge and plentiful experience have encouraged and inspired me throughout my academic research. I also would like to thank INESC-ID for providing the hardware that made possible the concretization of this work. Finally, I would like to express my gratitude to my parents, my sister, my girlfriend, and all my family and friends that supported me over the past months. Without their tremendous understanding and encouragement, things would have been much different.

Abstract

Recruiting clinical study participants is the most critical and one of the most challenging parts of the clinical trial process. Thus, it is crucial to investigate the barriers to recruitment and define strategies to improve these processes. One way to overcome this challenge is to utilize the available patient data and information retrieval techniques to match patients to clinical trials.

This study aims to experiment with different retrieval techniques for matching clinical trials and patient descriptions using the TREC 2021 Clinical Trials Track test collection. These retrieval methods receive input queries corresponding to patient descriptions and produce a ranked list of clinical trials using dense and sparse term-based retrieval approaches. A Sentence-BERT bi-encoder model was used for the dense retrieval task, trained explicitly for semantic search on a large and diverse dataset. Transfer learning methods were also explored to improve the ranking results by adapting this model, trained on a large generic corpus, to the clinical domain using complementary data from the SIGIR-CT and TREC-PM test collections. In addition, random negative sampling techniques were tested.

Overall, experiments with dense retrieval methods led to better results compared to sparse techniques. Fine-tuning the models with small in-domain datasets proved to be more effective in retrieval compared to the experiments with pre-trained models in a zero-shot scenario. The best run used a model fine-tuned on the two complementary datasets reserved for training, the SIGIR-CT and the TREC-PM. The strategy was to fine-tune the model on the SIGIR-CT dataset and train it again on the TREC-PM.

Keywords

Information Retrieval, Neural Models for Ranking Results, Bi-Encoder, BERT

Resumo

O recrutamento de participantes para ensaios clínicos é um processo bastante crítico, por isso é importante definir estatégias para melhorar estes processos. Uma alternativa para ajudar a mitigar este problema está em usar dados de pacientes juntamente com técnicas de recuperação de informação para encontrar pacientes adequados a ensaios clínicos.

Este estudo pretende experimentar diferentes técnicas de recuperação de informação para fazer corresponder descrições de pacientes a ensaios clínicos utilizando para isso dados da tarefa TREC 2021 Clinical Trials. Estes métodos recebem como entrada um conjunto de informações dos pacientes e produzem uma lista ordenada com os ensaios clínicos mais relevantes através de técnicas baseadas em representações densas e esparsas. Para os métodos com representações densas foram usados modelos Sentence Transformers, de arquitetura bi-codificador, treinados para pesquisa semântica num conjunto de dados grande e diversificado. Foram ainda exploradas técnicas de transferência de aprendizado com o objetivo de melhorar os resultados do ranking através da adaptação do modelo ao domínio clínico. Foram ainda testadas técnicas de amostragem negativa aleatória.

No geral, as experiências com métodos de recuperação com representações densas obtiveram melhores resultados quando comparadas com as experiências feitas com técnicas com representações esparsas. Fazendo recuperação com modelos treinados em pequenos conjuntos de dados relacionados ao domínio do estudo mostrou ser o método mais eficaz. A melhor estratégia consistiu em treinar o modelo com os dados SIGIR-CT, seguido do treino com os dados TREC-PM.

Palavras Chave

Recuperação da Informação, Modelos Neuronais para Ordenação de Resultados, Bi-Codificadores, BERT

Contents

List of Figures

List of Tables

1

Introduction

Contents

Medical information attracts the attention of a wide range of users, including researchers, patients, and clinicians. Such data is available through multiple sources, including the world wide web, journal articles, and hospital records. A good amount of research has been dedicated to designing solutions for the problem of patient recruitment for clinical trials using publicly available medical information. This work studies this research area and aims to experiment with retrieving relevant clinical trials for patients descriptions using state-of-the-art techniques in information retrieval with the available test collections consisting of clinical trials, patient descriptions and relevance judgments.

1.1 Context

Clinical trials have been a part of the medical landscape for many years, creating opportunities for physicians to find viable treatments for a range of conditions. Since the beginning of clinical research, there have been various challenges. Through times, these trials have been suffering many changes and updates to how they are carried out and standards that must be followed to ensure they are ethical. One of the biggest challenges faced just before starting a clinical trial is patient recruitment. Patient recruitment is essential to the success of pharmaceutical research and, consequently, patient care. Yet, nearly 80% of clinical trials conducted in the United States fail to meet their enrollment timelines, and up to 50% of research sites enroll one or no patients [1]. This problem translates into millions in lost revenue each day a drug is delayed. More importantly, new cutting-edge medications are delayed in their journey to the patients who need them most.

A possible solution to overcome these difficulties, and help the task of matching patient to trials, is to use the vast amounts of patient data already available in the form of electronic health records (EHR).

The interest in patient recruitment through clinical trial search, as manifested in international conferences, as well as efforts to construct adequate datasets, provides the opportunity to investigate the different aspects of the creation and evaluation of systems for matching clinical trails to patients. One of these conferences is the Text REtrieval Conference (TREC), co-sponsored by the National Institute of Standards and Technology (NIST) and the U.S. Department of Defense, and has the purpose of supporting research within the information retrieval community. In 2021, it introduced the Clinical Trials Track [1]. The primary motivation for this task lies in building an automated system to help clinical trials meet their recruitment targets by suggesting trials relevant to a given patient. In addition, it enables the evaluation of patient matching systems and the building of a test collection for clinical trial search.

These patient matching systems should generally take input queries corresponding to patient descriptions and produce a ranked list of clinical trials. The core aspect of the trial descriptions is the inclusion/exclusion criteria. These criteria are a set of characteristics that subject patients must have/not

[1] https://www.trec-cds.org/2021.html

have to be suitable for the study, defining the trial eligibility. Notably, a clinical trial can be relevant for a patient based on their disease or condition, but they might be ruled out due to their age, gender, or other factors. These intricate criteria can be a challenge for retrieval systems. Another challenge for these automated systems is that patient and trial descriptions are often long, containing noise that may significantly reduce the accuracy of the retrieval results.

The TREC 2021 Clinical Trials Track offers a test collection containing 375,580 publicly accessible clinical trials obtained from ClinicalTrials.gov [2] on April 27, 2021, and 75 topics consisting of synthetic patient cases in the form of admission notes created by individuals with medical training. To participate in the task, each team had to submit their results file following the TREC format as follows:

TOPIC_NO Q0 ID RANK SCORE RUN_NAME

Where **TOPIC_NO** is the topic number (1–75), **Q0** is a required but ignored constant, **ID** is the identifier of the retrieved document (PMID or NCT ID), **RANK** is the rank (1–1000) of the retrieved document, **SCORE** is a floating point value representing the confidence score of the document, and **RUN_NAME** is an identifier for the run. The **RUN_NAME** is limited to 12 alphanumeric characters (no punctuation). The file is assumed to be sorted numerically by **TOPIC_NO**, and **SCORE** is assumed to be greater for documents that should be retrieved first.

1.2 Motivation

Clinicians had to constantly be aware of clinical trials pertinent to their field and assess patient eligibility based on semi-structured electronic health records (EHRs). While a physician can refer patients to a clinical trial, given the vast and ever-growing number of clinical trials available, it is challenging for even the best and brightest doctors to stay up to date on all the possible treatment options. Often, they will be unaware of trials being run outside their institution. Therefore, it is possible to reduce the time spent searching for suitable trials through automated systems using existing data from the EHR.

Some tools are already available for this purpose, for example, “Automated Clinical Trial Matching” [3] developed by Circlebase. Regardless, there is always room for improvement.

1.3 Problem Definition and Objective

This work assumed that matching clinical trials to patients descriptions could take advantage of advances in language modeling using transformers with limited training data, the often case in the biomed-

[2] https://clinicaltrials.gov
[3] https://circlebase.com/life-sciences-solutions/automated-clinical-trial-matching/

ical domain. Therefore, this work aims to study this premise by experimenting with document retrieval using transfer learning and transformer-based neural retrieval approaches, but also sparse term-based retrieval approaches and comparing them.

An essential component for building the implemented solutions is the test collection available from TREC 2021 Clinical Trials Track, containing data about clinical trials, patient descriptions, and relevance judgments indicating which trials are relevant for which patient descriptions.

This research provides evidence to the following questions when applying first-stage retrieval for ranking clinical trials to patient descriptions:

- *What is the effectiveness of sparse retrieval when ranking clinical trials using the entire collection?*
- *Compared to sparse models, how well do BERT-based pre-trained models perform applied in a zero-shot scenario for retrieval?*
- *Does training a model on small in-domain data affect the effectiveness of the downstream task and how it performs when compared to a zero-shot scenario with a pre-trained model?*

This study empirically shows that noisy terms can negatively influence retrieval with term-based models. Therefore, choosing the more likely exact terms is an effective and simple way to face the vocabulary mismatch problem. It also found that more classical retrieval approaches with sparse representations can obtain similar results compared to state-of-the-art transformer-based approaches using dense representations, despite the latter's complexity. Finally, the conducted experiments reveal that adapting a BERT-based model to the domain being studied, even with a small number of training examples, improves retrieval performance compared to applying a pre-trained model in a zero-shot scenario.

1.4 Document Structure

The remaining of this document is structured as follows:

- Chapter 2, Fundamental Concepts - Introduces fundamental concepts in deep learning and information retrieval, including techniques, models, and algorithms.
- Chapter 3, Related Work - Presents state-of-the-art models for the tasks of information retrieval and overviews the task of the TREC 2021 Clinical Trials Track and some of its submissions.
- Chapter 4, Clinical Trial Data - Describes the available dataset for the TREC 2021 Clinical Trials Track and complementary data used in this research work.
- Chapter 5, Approaches for Clinical Trial Search - Details the implementation of the experiments, the used models, indexing and addresses the framework for fine-tuning the bi-encoder Sentence Trasnformer's models.

- Chapter 6, Experimental Evaluation - Presents and discusses the obtained results in the retrieval task from the various experiments.
- Chapter 7, Conclusions and Future Work - Provides the conclusion for this dissertation and potential directions for further research.

2

Fundamental Concepts

Contents

This chapter presents a concise overview of the fundamental concepts and theory needed to comprehend the task of matching clinical trials to patient descriptions. The first section introduces essential Deep Learning concepts present direct and indirectly in methodologies and algorithms used in the various components of this research. The second section consists of a summary of information retrieval with some important definitions, the evaluation metrics commonly used to evaluate retrieval systems, and a term-based model commonly used in retrieval systems.

2.1 Fundamental Concepts in Deep Learning

Artificial intelligence is a wide-ranging field of computer science and refers to creating machines as intelligent as the human brain [2]. Informally, the term ”artificial intelligence” is used when a machine has the ability to execute operations such as ”learning” and ”problem-solving”, usually carried out by human minds [3]. Machine Learning (ML) and Deep Learning, frequently mentioned in conjunction with artificial intelligence, are composed of algorithms that typically make predictions or classifications based on input data. More specifically, Deep Learning is a subset of Machine Learning that uses artificial neural networks (ANNs) to mimic the human brain's learning process. These networks operate by simulating the way biological neurons communicate with each other.

Artificial neural networks consist of multiple node layers, including an input layer, one or more hidden layers, and an output layer [4]. Each node connects to another and has an associated weight and threshold. A node is activated when its output exceeds the specified threshold value. In this case, the artificial neuron sends data to the next network layer. Otherwise, no data is passed along to the next layer of the network [4]. A simple representation of a neural network is illustrated in Figure 2.1.

The appearance of deep learning dates back to 1943 when Walter Pitts and Warren McCulloch published an article describing the first mathematical model of a neural network based on the human brain [5]. Since then, only two significant breaks have occurred in the development of this field, both related to the infamous Artificial Intelligence winters [6].

By 2011, the speed of graphics processing units (GPUs) had increased significantly, allowing researchers to solve a wide variety of problems yet to be solved. The backpropagation and stochastic gradient descent algorithms, the outstanding advances in highly parallelizable GPUs, and the development of GPU-enabled algorithms brought deep learning from a research tool to a method present in daily life. Another factor that contributed to the success of deep learning is the amount of available data to train deep neural networks (DNNs) [7]. Figure 2.2 illustrates a performance comparison between deep learning and other machine learning methods with the amount of data.

This section addresses the theory present in the implementation of artificial networks and the main architectures employed in relevant fields to this research, such as NLP and information retrieval. First,

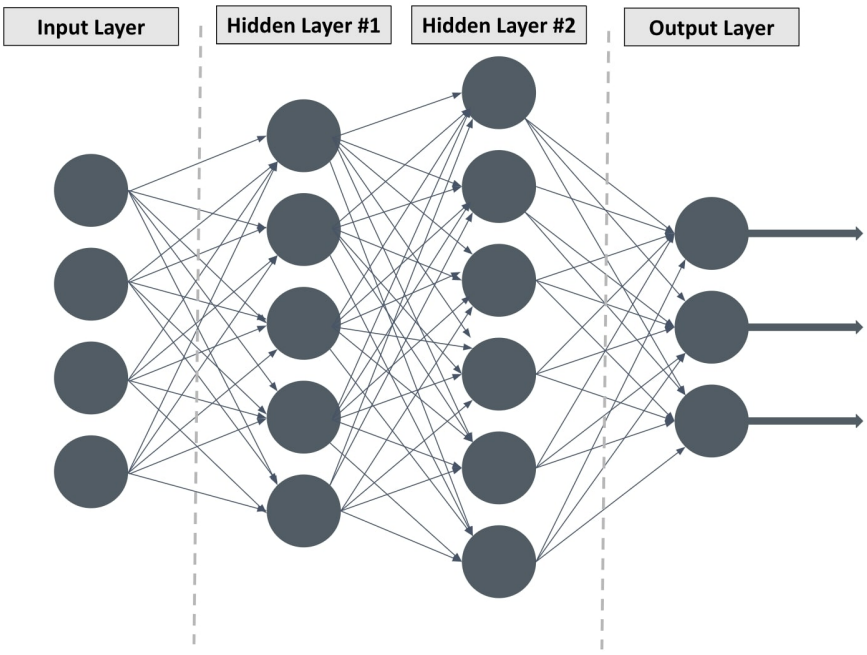

Figure 2.1: Illustration of an artificial neural network.

it briefly introduces the concept of Artificial Neural Networks, then details the Feed Forward Neural Network (FFNN) and the mathematics behind it. After this, it addresses the concept behind the attention mechanism, the transformer model, a revolutionary type of artificial neural network, and Bidirectional Encoder Representations from Transformers (BERT), a model developed by Google that learns bi-directional representations of text.

2.1.1 Artificial Neural Networks

An artificial neural network is a system inspired by the structure of biological nervous systems and how they process information. However, these networks' processing elements and architectures have diverged considerably from their biological motivation. These are networks of simple processing units (known as "neurons") operating on their local data and communicating between them. Each neuron can receive input signals, process them, and send an output signal. The most typical type of artificial neural network consists of three layers of units: an input layer of units connected to a layer of hidden units, which is connected to a layer of output units as shown in Figure 2.1. A real number evaluates each connection between neurons, called the weight coefficient, and reflects the relative importance of the given connection.

The principal benefit of neural networks is their ability to use some a priori unknown information hidden in data. The procedure of "capturing" unknown information is known as learning or training of

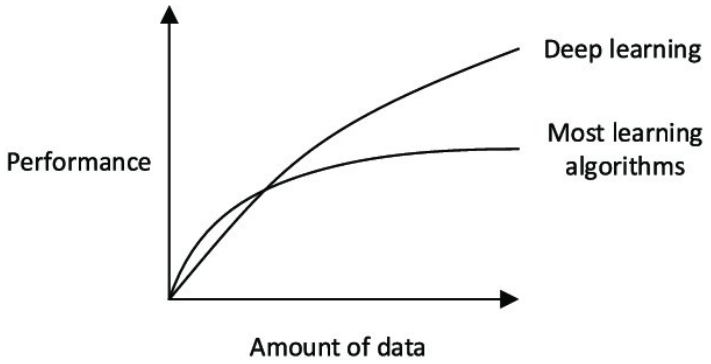

Figure 2.2: Comparison between performance of deep learning against that of most other machine learning algorithms [8].

neural network. It adjusts weight coefficients in such a manner that some conditions are fulfilled.

2.1.2 Feed Forward Neural Networks

The feed forward neural network, also known as multi-layer perceptron (MLP), is a type of artificial neural network in which nodes do not form loops. This networks follow the same architecture described in the previous section, illustrated in Figure 2.1. The input layer has as many neurons as the dimension of the input data. The number of neurons in the output layer depends on the type of problem the neural network is attempting to solve (e.g., regression[1] or a classification[2] problem). The number of hidden layers characterizes the depth of the network, and its complexity is measured by the number of free parameters in the network, that is, the number of neurons and the number and strength of connections between neurons (weights). The more hidden layers the network has (and the more neurons each hidden layer has), it can estimate more complex functions. However, the growth in complexity increases the training time (due to the increased number of parameters) and increases the likelihood of overfitting.[3]

Each neuron's connection has a weight and is connected to the hidden layer, where input is combined with the initial weights in a weighted sum and subjected to the activation function. The data is propagated forward through the network, and when it reaches the output layer, the network predicts an outcome.

It is important to define two distinct phases in the lifecycle of a neural network, the training phase and the prediction phase. During the training phase, the neural network tries to find the optimal coefficients and biases that would match the input to the desired output. In the prediction phase, the neural network processes the input to produce predictions by moving the information from the input layer, passing through the hidden layers until it reaches the output layer in a forward fashion.

[1] Regression is a supervised machine learning algorithm used to predict the continuous values of output based on the input.

[2] Classification is a supervised machine learning algorithm that predicts specific discrete values (categories or classes) to which the input belongs.

[3] Overfitting occurs when a model learns the details of the training data and performs well on this data but cannot perform well on unseen data (data not used during training). Hence, cannot generalize to unseen data.

Forward propagation

When simplified, the feed forward neural network can appear as a single layer perceptron [4], as shown in Figure 2.3.

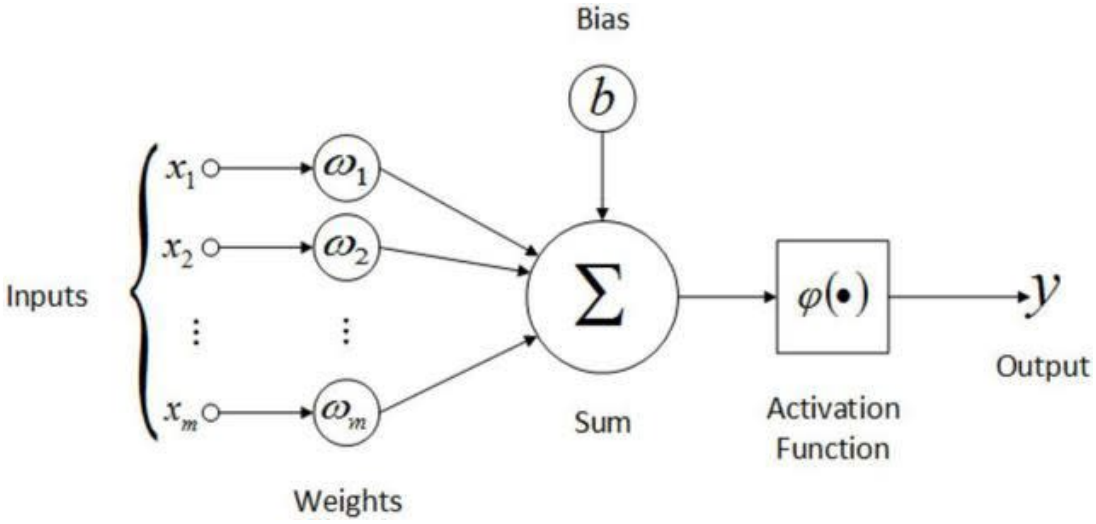

Figure 2.3: Illustration of a single layer perceptron [10].

In order to generate some output, the input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function, and passes the information to the successive layer. Each neuron is connected to every other neuron of the previous and consecutive layers by a trainable weight representing the strength of the connection. At each neuron in a hidden or output layer, the processing occurs in two steps:

1. **Pre-activation** - Is the weighted sum of the inputs from the previous layer plus bias. The output of this function is passed to the activation function.

2. **Activation** - An activation function is a mathematical function which adds non-linearity to the network. Some of the most popular activation functions are sigmoid, hyperbolic tangent(tanh), Rectifier Linear Unit (ReLU) and Softmax.

The following general equations are a mathematical representation of this process for a model with multiple ($L \geq 1$) hidden layers:

• **Hidden layer pre-activation**:

$$z^{(l)}(x) = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}, \tag{2.1}$$

with:

[4] Rosenblatt perceptron [9] is a binary single neuron model. The inputs integration is implemented through the addition of the weighted inputs that have fixed weights obtained during the training stage. If the result of this addition is larger than a given threshold, the neuron fires. When the neuron fires its output is set to 1, otherwise it's set to 0.

$\boldsymbol{W}^{(l)} \in \mathbb{R}^{K_l \times K_{l-1}}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{K_l}$.

• **Hidden layer activation**:

$$\mathbf{h}^{(l)}(x) = \mathbf{g}(z^{(l)}(x)). \tag{2.2}$$

• **Output layer activation**:

$$\mathbf{f}(x) = \mathbf{o}(z^{(L+1)}(x)) = \mathbf{o}(\mathbf{W}^{(L+1)}\mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}). \tag{2.3}$$

Backward propagation

Backpropagation is a learning mechanism that allows the Multilayer Perceptron to iteratively adjust the weights in the network, with the purpose of minimizing a cost function [11]. The mathematical process involved in backpropagation is the calculation of derivatives using the chain rule. The partial derivative of the cost function is calculated with respect to the weights, biases, and previous layer activations to determine which values affect the gradient of the cost function. Then, it updates the weights and biases in the opposite direction of the gradient, to find a local minimum, also known as gradient descent. An example of a gradient descent algorithm is the batch gradient descent:

$$\theta = \theta - \eta.\nabla(\theta)J(\theta), \tag{2.4}$$

where θ is the model's parameters, η is the learning rate and it controls the step size of the procedure, $J(\theta)$ is the objective function and $\nabla(\theta)J(\theta)$ is the gradient of the objective function.

The process is repeated for all of the examples in the training data. One round of updating the network for the entire training dataset is called an epoch.

2.1.3 Attention Mechanism

The attention mechanism was introduced, by Bahdanau et al. [12], to address the problem of using a fixed-length encoding vector that limits the input's information passed to the decoder stage in sequence-to-sequence models. This is particularly problematic for long sequences, where the dimensionality of their representation would be forced to be the same as for shorter sequences. The attention mechanism addresses this problem as it utilizes all the hidden states of the input sequence during the decoding process. It achieves this by creating a unique mapping between each time step of the decoder output to all the encoder hidden states. Thus, the decoder has access to the entire input sequence and selectively chooses specific elements from that sequence for each output that it produces. In broader terms, this mechanism enables the model to concentrate and give more ”attention” to certain parts of the input sequence as needed.

There are many versions of Attention as it suffered multiple adaptations through the years. This section introduces a major type of this mechanism, the Bahdanau Attention.

Bahdanau Attention

Commonly referred to as Additive Attention, it was first introduced in a paper by Dzmitry Bahdanau in 2014 [12]. The paper aimed to improve the sequence-to-sequence model in machine translation by aligning the decoder with the relevant input sentences and implementing attention. Here, the process of applying attention is divided into step-by-step computations as follows:

1. Computing the encoder hidden states - Encoder produces hidden states of each element in the input sequence.

2. Calculating Alignment Scores - The alignment model assigns a score $\alpha_{t,i}$ to the pair of input at position i and output at position t, (y_t, x_i), based on how well these elements match.

$$\alpha_{t,i} = \text{align}(y_t, x_i), \tag{2.5}$$

   where the set of $\{\alpha_{t,i}\}$ are weights defining how much of each source hidden state should be considered for each output. The alignment score α is parameterized by a feed-forward network with a single hidden layer. The score function is as follows:

$$score(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i]), \tag{2.6}$$

   where $tanh$ is used as a non-linear activation function and both $\boldsymbol{v}_a$ and $\boldsymbol{W}_a$ are weight matrices to be learned in the alignment model.

3. Softmax Operation to the Alignment Scores - The weights, $\alpha_{t,i}$, are computed by applying a softmax operation to the previously computed alignment scores:

$$\alpha_{t,i} = \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}. \tag{2.7}$$

4. Calculating the Context Vector - The encoder hidden states and their respective alignment scores are multiplied to form the context vector:

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i. \tag{2.8}$$

5. Decoding the Output - the context vector is concatenated with the previous output and fed into the Decoder for that time step along with the previous decoder hidden state to produce a new output.

6. Repeats steps 2-5 for each time step of the decoder.

Figure 2.4 is a graphical illustration of the encoder-decoder model with additive attention mechanism trying to generate an output y_t given an input sequence $(x_1, x_2, ..., x_T)$.

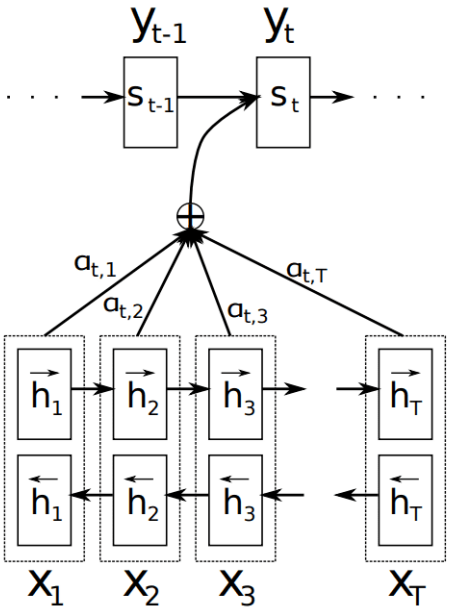

Figure 2.4: Illustration of attention mechanism from Bahdanau's paper [12]

A significant advantage of this mechanism is that it allows one to interpret and visualize the correlation between source and target words. For example, Figure illustrates the attention weight matrix α when a sentence is translated from French to English.

2.1.4 Transformers

Previous networks, such as recurrent architectures, depend on the sequential processing of the input at the encoding stage. Despite the attention mechanisms, the sequential nature of these networks prevents them from being parallelized, resulting in computational inefficiency [13]. These constraints, inherent to recurrent networks, limit their usefulness for transfer learning.

A new architecture was proposed, the Transformer architecture [13], that eliminates sequential processing and recurrent connections. It relies only on a self-attention mechanism to capture global dependencies between input and output.

The Transformer architecture is shown in detail in Figure 2.6, with each component, i.e., the encoder and decoder layers, each multi-head attention block, and the Feed-Forward Neural Network (FFNN) blocks. The subsections below will cover each component in more detail.

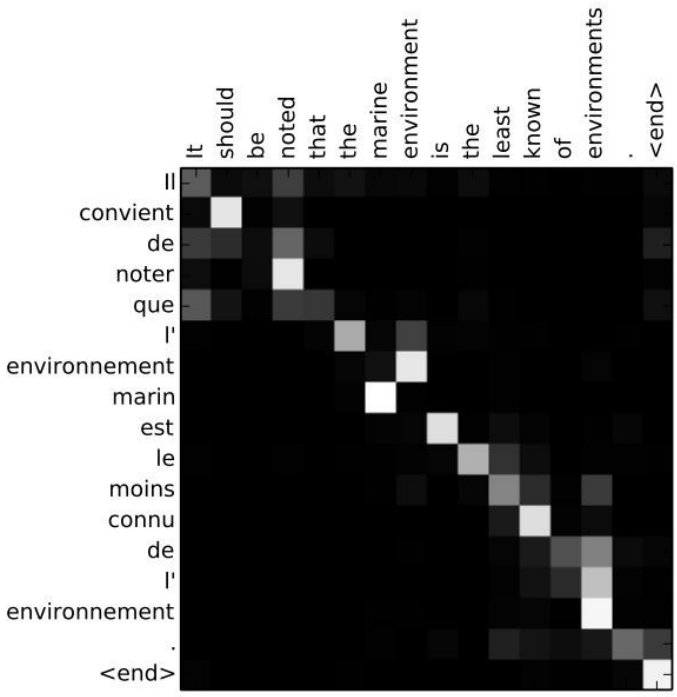

Figure 2.5: Alignment matrix of ”L’accord sur l’Espace économique européen a été signé en août 1992” (French) and its English translation ”The agreement on the European Economic Area was signed in August 1992” [12].

Input Embeddings

The input text must be processed and transformed into a suitable format before being fed to the transformer. First, the text goes through a tokenizer, which breaks it down into chunks of characters that can be processed separately. The tokenizer produces a list of numbers representing the input text’s token IDs. The tokens are then converted into word embeddings, created by embedding models, and trained separately from the transformer. Word embeddings are word representations that allow words with similar meanings to have exact representations, capturing their meaning. They help capture the semantics or context and help to understand the relationship between terms in the sentence. Before entering the transformer, each token in the input sequence is transformed into a vector of size d = 512.

Positional Encoding

Positional encoding defines the position of an entity in a sequence so that each position is assigned a unique representation. Transformers use a positional encoding system where each position is mapped to a vector. The positional embedding is a tensor of values, where each row represents the position of a word in a sequence, which are added to input embeddings to produce a final embedding with order information. Thus, the output of the positional encoding layer is a matrix, where each row represents an

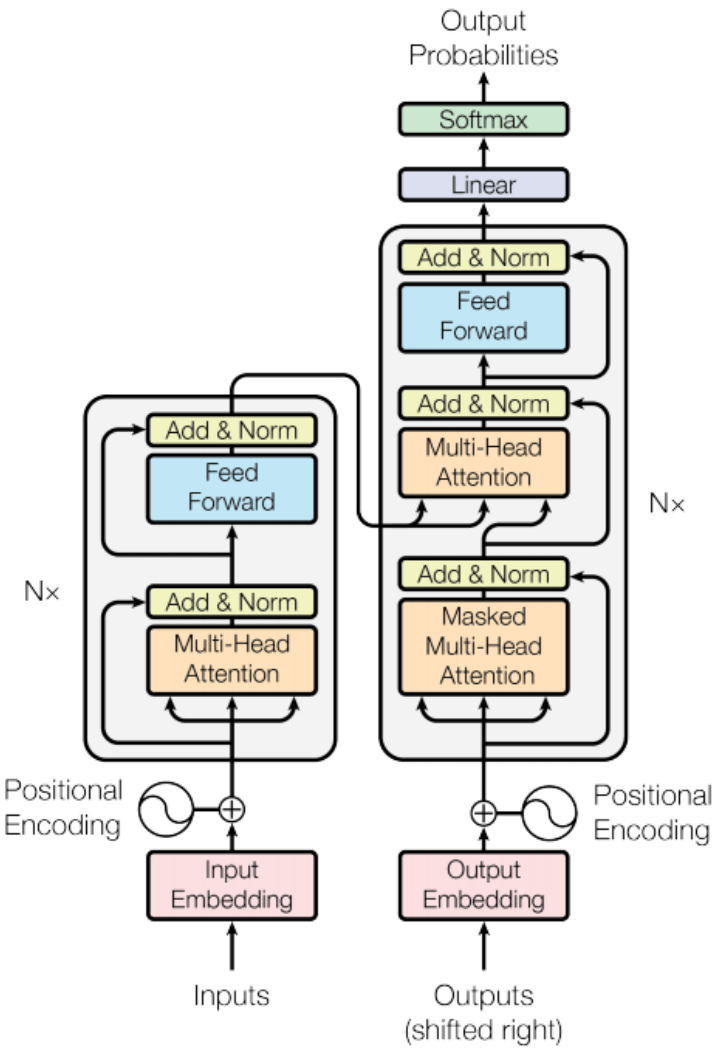

Figure 2.6: The Transformer - model architecture [13].

encoded object of the sequence summed with its positional information.

As shown in Figure 2.6, the positional encoding happens after input word embedding and before the encoder. For calculating positional embeddings, the authors of the transformer proposed the following:

$$\begin{aligned} \mathrm{PE}(pos, 2i) &= \sin\left(pos/10000^{2i/d_{model}}\right), \\ \mathrm{PE}(pos, 2i+1) &= \cos\left(pos/10000^{2i/d_{model}}\right), \end{aligned} \tag{2.9}$$

where pos is the position in time, $dmodel$ is the number of dimensions, and i is the dimension index in the input tensor. The above expression applies the sine function for even embedding indexes (i) and the cosine function for odd embedding indexes.

Multi-head Self-Attention

The Transformer implements the scaled dot-product attention, which uses three vectors: Query, Key, and Value. These are obtained through multiplication of the general input to the encoder/decoder with three different weight matrices learned while training. The vector is a word embedding of any input token obtained from the input (or could be the output of the previous attention layer). The matrices are initially randomly initialized weight matrices and are mostly rectangular to reshape the input vector into a smaller shape.

$$\begin{aligned} \mathbf{Q} &= \boldsymbol{X}\boldsymbol{W}_{\boldsymbol{q}}, \\ \mathbf{K} &= \boldsymbol{X}\boldsymbol{W}_{\boldsymbol{k}}, \\ \mathbf{V} &= \boldsymbol{X}\boldsymbol{W}_{\boldsymbol{v}}. \end{aligned} \tag{2.10}$$

A scaled dot product self attention is applied for every word in the input sequence. It computes the similarity score using the dot-product of one word, the query vector, with all other words in the sentence, each key vector. This result gives the relevance of all words to the query word, where the more similar pair will have the highest score. As the dimension of the queries and keys gets larger, the dot product grows large in magnitude, thus pushing the softmax function into regions where it has extremely small gradients [13]. Scaling the dot product by the query/key vectors length results in more stable gradients. To ensure the scores are all positive and add up to 1, these are normalized by applying a softmax function. Each result is concatenated, and the output is multiplied by a weight matrix to produce the layer's output. In practice, the attention function is computed simultaneously on a set of queries crammed together in a matrix Q. The keys and values are also packed together into matrices K and V. The attention matrix is given by:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}})\boldsymbol{V}. \tag{2.11}$$

The idea behind multi-head attention is to allow the attention function to extract information from different representation subspaces, which would otherwise not be possible with a single attention head. This is done by using h parallel attention layers, known as heads. The multi-head attention function can be represented as follows:

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(head_1, ..., head_h)\boldsymbol{W^o}, \tag{2.12}$$

where

$$head_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{Q}}, \boldsymbol{K}\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{K}}, \boldsymbol{V}\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{V}}). \tag{2.13}$$

Figure 2.7 illustrates the the scaled dot-product attention and the multi-head attention mechanism.

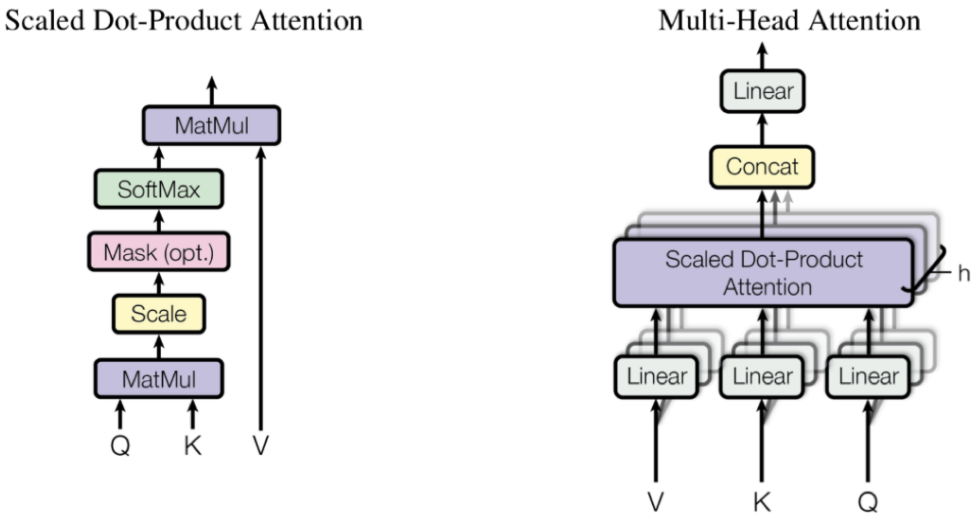

Figure 2.7: Illustration for the scaled dot-product attention (left) and for multi-head attention (right) [13].

Encoder Block

The encoder block is a stack of multiple layers, which themselves are a multi-head self-attention block and a position-wise Fully Connected Feed-Forward Network (FC FFN), a couple of linear layers with a $ReLU$ activation function in between them.

Researchers have used a residual connection by wrapping each of the two sub-layers, followed by layer normalization. Hence, the output of each sub-layer is $LayerNorm(x + Sublayer(x))$ and $Sublayer(x)$ is a function implemented within itself. The output dimension of all sub-layers, as well as embeddings, is d_{model} = 512.

Decoder Block

In addition to the sub-layers employed by the encoder, the decoder uses multi-head attention over the outputs of the encoder block. The residual connections are attached to the sub-layers followed by layer normalization, just like the encoder. The decoder component also has a Masked Multi-Head Attention Layer that guarantees that attention is applied to tokens up to the current position (index till which prediction is done by the model) and not future tokens, unlike the encoder where attention is calculated for the entire sequence at once.

2.1.5 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a multi-layer bidirectional structure based on a Transformer encoder. It can pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers [14]. It not only outperformed the state-of-the-art results on high-profile NLP tasks, but also made way for transfer learning from pre-trained language models to be the new standard in NLP [15].

Architecture

As proposed in [14], the implementation of the BERT system is based on the encoder component of the Transformer, following the same working principles described earlier. BERT has been presented in two sizes, the smaller one with 110 million parameters and the larger one with 340 million parameters. The larger ones require specialized hardware like TPUs or GPUs to train due to their considerable memory requirements and size. The smaller one, referred to as $BERT_{Base}$, has 12 encoder layers and 12 attention heads. The larger model, named $BERT_{Large}$, has 16 attention heads and 24 encoder layers. The two models use different sizes of hidden space, where the smaller models have 768 dimensions, and the larger models have 1024.

As shown in Figure 2.8, the system has two main steps, pre-training, where the task is Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), and fine-tuning, where the output layer, used during pre-training, is thrown away and replaced with one to be used for the fine-tuning task.

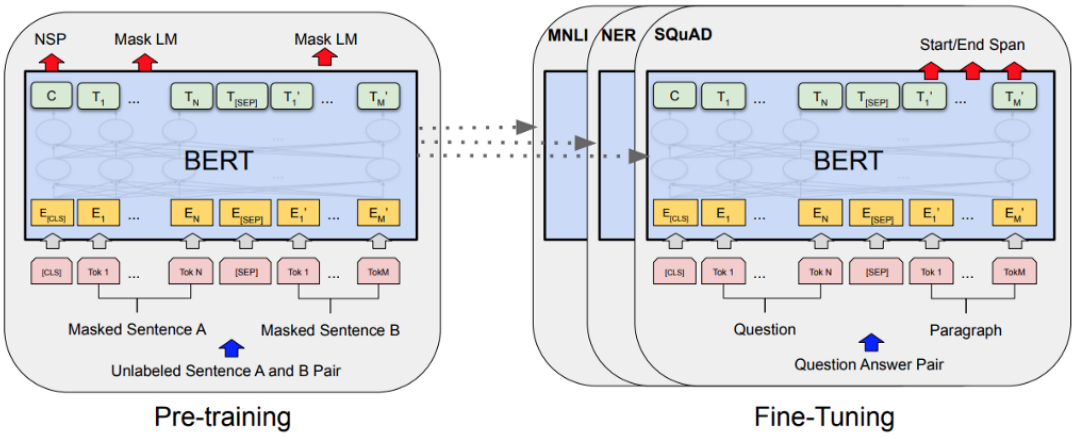

Figure 2.8: Pre-trainning and fine-tuning procedures for BERT [14].

Tokenization and Input Embeddings

As opposed to Transformers, where a positional encoding is used to generate positional embeddings, BERT learns the positional embedding during the training stage. It uses a word-piece tokenizer concept

that breaks some words into sub-words, the WordPiece tokenization algorithm [16]. The first token of each sequence is set to a special [CLS] classification token. Its final hidden state is used in classification tasks. It is trained and subsequently used as a representation of the complete input sequence in a fine-tuning classification layer (if added). For sentence pairs, as in the Next Sentence Prediction pre-training, sentences are separated with a special [SEP] token. Finally, for each token, adding its token embedding, its segment embedding, and its position embedding produces a final embedding. Where the token embedding refers to the content of the token, the segment embedding indicates the segment, A or B, to which the token belongs given that the input can corresponds to either a single sentence or a pair of sentences, and the position embedding is the absolute position of the token within the input sequence. The input representation for BERT is illustrated in Figure 2.9.

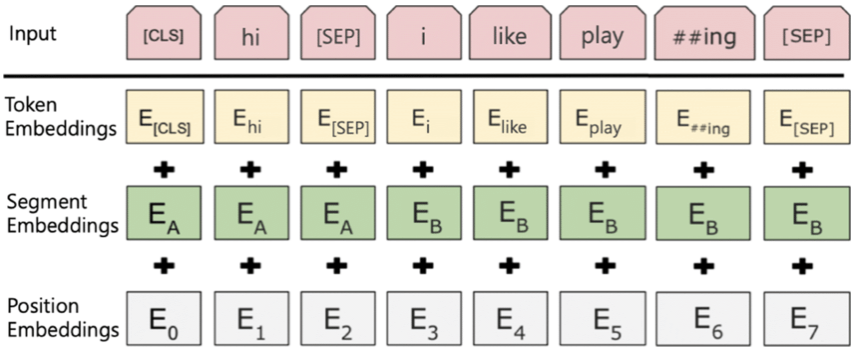

Figure 2.9: BERT input representation [14].

Pre-training

The pre-training of BERT consists in training the network before the architecture is used in other tasks, allowing the system to learn the language model of the data it is trained on. The pre-training is done by training the input tokens on a combination of two main NLP tasks, Masked Language Modelling and Next Sentence Prediction. The first consists of giving BERT a sentence and optimizing the weights inside BERT to output the same sentence on the other side. Before giving the input sentence to BERT, 15% of the input tokens are randomly masked with the [MASK] token. Increasing this percentage makes it difficult for the model to learn from the input since more information is hidden. By decreasing this value, learning enough context could take longer and require more data since it might predict the missing tokens easily. Having selected the 15% of input tokens to mask and to avoid inconsistencies with the fine-tuning procedure, only 80% of these tokens are masked with [MASK], while 10% are replaced with a random token, and the remaining 10% are left untouched. Combining random, masked, and original words helps the model generalize. The Next sentence prediction task aims at predicting whether a sentence is the sentence coming immediately after a given one. Given a sentence pair, 50% of the time, the second sentence is the following sentence, and the other 50% is a random sentence. The

hidden representation of the [CLS] token has an important role in this task since it encodes information from all over the sequence. BERT was pre-trained on unlabeled document-level data extracted from BooksCorpus (800M words) [17] and English Wikipedia (2,500M words) using a sequence length of 512 tokens and a batch size of 256 sequences for 1,000,000 steps.

Fine-tuning

Fine-tuning is a training procedure for downstream tasks, such as Natural Language Inference (NLI), Question Answering, and others. This procedure is very straightforward compared to pre-training since, for every fine-tuning task, it is only necessary to use inputs and the appropriate outputs while allowing all network parameters to be updated [14]. For example, in the case of a Question Answering task, the input-output pair would correspond to a question-passage pair, and the only required modification to the architecture is the addition of an output layer. This layer would be token-level in this case, while the [CLS] representation would be fed into a layer for classification in the case of, e.g., sentiment analysis [14]. According to BERT's original paper, all the results can be replicated in at most 1 hour on a single cloud TPU, or a few hours on a GPU [14]. When compared to pre-training, fine-tuning BERT is inexpensive.

2.2 Fundamental Concepts in Information Retrieval

Information Retrieval (IR) is the process of obtaining some information resources relevant to an information need, in which this information is extracted from unstructured, semi-structured, or even structured large datasets. The returned results are generally ranked concerning some relevance notion. Generally, requirements are defined in the form of queries, and the IR system should process this information to select and present the most relevant documents given those requirements. For instance, in web searching, users communicate their requirements in the form of keywords, concisely manifesting their information needs.

These systems are evaluated on their accuracy and ability to retrieve high-quality information, which maximize users' satisfaction, meaning that better systems have more answers corresponding to the user's expectations. Search engines such as Google, Bing, and Yahoo! are examples of some of the most commonly used information retrieval systems daily.

Information retrieval has been evolving through the past decades and has been taking advantage of fields such as Artificial Intelligence and its subfields. Thus, various ranking models have been proposed, including vector space models, probabilistic models, and learning to rank (LTR) models [18].

Recently, there has been a significant increase in work on applying deep neural networks for building ranking models due to their effectiveness at learning abstract representations and their capacity to tackle complex learning problems. Consequently, neural ranking models have been applied to conversational

search, community-based QA, ad-hoc retrieval, and other tasks [18].

In the following sections, some important definitions will be introduced shortly and will present concepts related to information retrieval techniques and the evaluation of these systems.

2.2.1 Definitions

This subsection aims to characterize some crucial components to understand ad-hoc retrieval and the problem of ranking texts concerning their relevance to a particular query.

Document Corpus

Inherently, text ranking presumes the existence of a collection of texts (can be arbitrarily large) or a corpus composed of mostly unstructured natural language text [19]. In other words, a collection is a set of documents from which a user wishes to get information. Its documents may be part of a narrowly defined subject domain or may pertain to a wide range of concepts covering several topic domains. These documents could have various formats, and their texts can vary in length, ranging from sentences (e.g., searching for related questions in a question answering application) to entire books [19].

Query

A query is a textual description of the user information need in the input language provided by the information system, usually a short sequence of words. In these systems, a query does not uniquely identify a single document in the collection. Instead, several documents may match the query with different degrees of relevance. According to some studies, the average query length is two to three words [20].

Relevance

In text ranking systems, relevance is an abstract measure representing how related a text and a particular information need are to each other. It is not limited only to a binary scale, where a text is said to be relevant if it addresses the information need and otherwise not. Relevance can also be characterized using ordinal scales in multiple dimensions. As an example, a three-way scale of not relevant, relevant, and highly-relevant is one common alternative [19].

Relevance Judgments

Relevance judgments, also called relevance labels, are composed of a set of (q, d, r) triples, where the q is a query, d a document, and relevance judgment r is a (human-provided) annotation on (q, d)

pairs [19]. These triples are contained in text files as part of a test collection and can be treated like "ground truth". Relevance judgments represent an assessor's judgment of what's relevant or not [19]. In broader terms, all relevance judgments are opinions and thus are subjective. These can be used to train ranking models in a supervised setting and can also be used to evaluate ranking models.

Assessor agreement on relevance judgments is relatively low. However, depending on various factors such as the information needs, the exact agreement metric, and the study design, the range of values reported in the literature differs [19].

Test Collection

Test collections comprise a set of information need descriptions or topics, information objects (a corpus or collection) to be searched, and relevance judgments indicating which objects are relevant for which topics. These tools are mainly used for evaluating the effectiveness of information retrieval, having a vital role in providing a basis for measuring and comparing the effectiveness of different information retrieval algorithms and techniques [21]. Events such as the Text Retrieval Conference[5] (TREC), the Cross-Language Evaluation Forum[6] (CLEF), the NII Testbeds and Community for Information Access Research project[7] (NTCIR), the Initiative for the Evaluation of XML Retrieval[8] (INEX), and the Forum for Information Retrieval Evaluation5 (FIRE) has raised the popularity of these collections in IR evaluation [19, 21]. In particular, TREC has developed and made several test collections available and allowed many teams from all over the world to participate in developing next generation retrieval technologies. Despite all the benefits, these collections can be expensive and complex to construct [21]. The test collections related to this work will be described later.

2.2.2 Evaluation in IR

Retrieval evaluation is systematically associating a quantitative metric to the results produced by an IR system in response to a set of queries to determine its effectiveness. In other words, retrieval evaluation measures how close the obtained ranking list is to the ideal condition. Ranking metrics quantify the quality of a ranking, thus, are essential to assess effectiveness improvements. They are computed from relevance judgments (qrels), described previously.

Different measures have been proposed to quantify effectiveness. Below are described some of the most common metrics used throughout this thesis for evaluating retrieval performance. This description follows the notation of [19]. Here, a ranked list $R = \{(d_i, s_i)\}_{i=1}^{l}$ of length l as $\{(i, d_i)\}_{i=1}^{l}$, retaining only the rank i induced by the score s_i's. Many metrics are computed at a certain cutoff, meaning that

[5] http://trec.nist.gov.
[6] http://www.clef-initiative.eu.
[7] http://research.nii.ac.jp/ntcir.
[8] http://inex.mmci.uni-saarland.de.

the ranked list R is truncated to a particular length k, $\{(d_i, s_i)\}_{i=1}^{k}$, where $k \leq l$: this is notated as Metric@k [19]. The difference between l and k is that the system determines l (i.e., how many results to return), whereas k is a property of the evaluation metric [19]. The following metrics can be computed:

Precision

Precision is defined as the fraction of documents in ranked list R that are relevant and can be writen as:

$$\text{Precision}(R, q) = \frac{\sum_{(i,d)\in R} \text{rel}(q, d)}{|R|}, \tag{2.14}$$

where $\text{rel}(q, d)$ indicates whether document d is relevant to query q, assuming binary relevance. Graded relevance judgments are binarized with some relevance threshold. For instance, in a three-grade scale, $\text{rel}(q, d) = 1$ for "relevant" and "highly relevant" judgments. Not taking into account graded relevance judgments is one of the disadvantages of this metric. Another disadvantage is the fact that it does not take into consideration rank positions [19]. Often, precision is evaluated at a cutoff k, notated as P@k. For instance, P@10, where the ranked list R is truncated to length 10.

Recall

Recall is defined as the fraction of relevant documents (in the entire collection C) for q that are retrieved in ranked list R and can be expressed as:

$$\text{Recall}(R, q) = \frac{\sum_{(i,d)\in R} \text{rel}(q, d)}{\sum_{d\in C} \text{rel}(q, d)}, \tag{2.15}$$

where $\text{rel}(q, d)$ indicates whether document d is relevant to query q, assuming binary relevance. As with precision, graded relevance judgments are binarized. Thus, it does not consider relevance grades nor the rank positions in which relevant documents appear. Recall is often evaluated at a cutoff k, notated as R@k.

Mean Reciprocal Rank (MRR)

The Reciprocal Rank (RR) computes the reciprocal of the rank at which the first relevant document is retrieved. When averaged across all evaluation queries, the measure is called the Mean Reciprocal Rank (MRR). It can be formulated as follows:

$$\text{MRR}(R, q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}, \tag{2.16}$$

where $rank_i$ is the smallest rank number of a relevant document. That is, if a relevant document appears in the first position, reciprocal rank = 1, 1/2 if it appears in the second position, 1/3 if it appears in the third position, and so on. If a relevant document does not appear in the top k, then that query receives a score of zero. As with precision and recall, RR is computed concerning binary judgments.

Normalized Discounted Cumulative Gain (nDCG)

As opposed to the other metrics above, nDCG was designed explicitly for graded relevance judgments, providing better output quality distinctions. First, it is important to define Discounted Cumulative Gain (DCG):

$$\mathrm{DCG}(R, q) = \sum_{(i,d)\in R} \frac{2^{rel(q,d)} - 1}{\log_2(i+1)}. \tag{2.17}$$

The discounting refers to the decay in the gain as the user consumes results lower and lower in the ranked list [19]. It penalizes relevant documents appearing lower on the list of ranked results. Introducing normalization:

$$\mathrm{nDCG}(R, q) = \frac{\mathrm{DCG}(R, q)}{\mathrm{IDCG}(R, q)}, \tag{2.18}$$

where IDCG represents the DCG of a ranked list that starts with all documents of the highest relevance grade, then the documents with the next highest relevance grade, and so on. Therefore, nDCG represents DCG normalized to a range of [0, 1] concerning the best possible ranked list, and is usually associated with a rank cutoff, commonly a value of 10 or 20 [19].

2.2.3 Okapi Best Matching 25 (BM25)

Traditional retrieval systems leverage lexical similarities to match queries and documents [22]. These classic ranking models, also known as sparse bag-of-words models, were state-of-the-art for decades and are still in use since they serve as important baselines for comparison with neural-based approaches [23, 24].

The following major models have been developed to retrieve information: the Boolean model, the Statistical model, which includes the vector space and the probabilistic retrieval model, and the Linguistic and Knowledge-based models. The first model is often referred to as the ”exact match” model, and the latter ones as the ”best match” models [25, 26].

Since this thesis aims not to give a comprehensible guide to traditional IR, it only presents the BM25 term-weighting and document-scoring function, an algorithm that stems from a framework for probabilistic models and document retrieval, the Probabilistic Relevance Framework (PRF) [27].

BM25 is considered the state-of-the-art of traditional IR [27] and is still widely used due to its simplicity and effectiveness in the retrieval of documents, despite contemporary approaches based on the

revolutionary BERT-based neural retrieval techniques [19]. It is a non-linear combination of three document attributes: document frequency, document length, and term frequency, and is used to estimate the similarity between a document and a given query. Even though through the years several BM25 versions have been developed [28], the basic formula is [19]:

$$\mathrm{BM25}(q,d) = \sum_{t \in q \cap d} \log \frac{N - \mathrm{df}(t) + 0.5}{\mathrm{df}(t) + 0.5} \cdot \frac{\mathrm{tf}(t,d) \cdot (k_1 + 1)}{\mathrm{tf}(t,d) + k_1 \cdot (1 - b + b \cdot \frac{l_d}{L})}, \tag{2.19}$$

where the first component of the summation is the idf (inverse document frequency) component, N is the total number of documents in the collection, and df(t) is the number of documents that contain term t (i.e., its document frequency). In the second component of the summation, tf(t, d) represents the number of times term t appears in document d (i.e., its term frequency). The expression in the denominator involving b is responsible for performing length normalization, since collections usually have documents that differ in length: ld is the length of document d while L is the average document length across all documents in the collection [19]. k_1 and b are free parameters.

3

Related Work

Contents

This chapter explores previous work on document retrieval, also known as ad-hoc retrieval, in general, followed by an overview of the 2021 edition of TREC Clinical Trials Track and its submissions.

3.1 Transformer-based Neural Ranking

Document retrieval is a standard retrieval task in which an IR system must respond to a previously unseen query by producing a ranked list of documents from a static collection, where documents at the top of the ranking are more likely to be relevant. The texts in the documents often come from a distinct set of authors, ranging from several phrases to numerous paragraphs. The query comes from a user and is usually succinct, ranging from a few terms to a few sentences [18, 29].

A significant characteristic of document retrieval is the heterogeneity of the query and the documents, leading to the critical vocabulary mismatch problem [18, 30, 31]. Matching terms and sentences with similar meanings could alleviate the problem, but exact matching is crucial, particularly with rare terms [18,32]. Relevance assessment is another challenge since it is a time-consuming and expensive process involving human beings [33]. It consists of creating relevance judgments for a subset of documents for each query [33].

In recent years, various neural ranking models (NRMs) based on BERT have been proposed in the information retrieval (IR) community. These BERT-based tend to be robust against the vocabulary mismatch problem because they learn semantic representations of query–document pairs in a latent space [29]. As a result, they have proven successful in both the retrieval and re-ranking stages.

Such neural ranking models come in two varieties [34]: bi-encoder models [23, 35], which learn separate embeddings for the query and document, and cross-encoder models [36], which learn a joint embedding for the query and document.

Bi-encoders are mainly used for retrieval, given that the individual representations can be indexed through methods supporting the fast execution of maximum inner product searches, such as FAISS [37]. On the other hand, both bi-encoder and cross-encoder models are applicable for re-ranking [38].

3.1.1 Ranking with Bi-Encoders

Bi-encoder [34] takes two inputs (e.g., a query and a document) and performs two independent self-attentions over these inputs, generating two different representations (embeddings) that can be compared with a comparison function [39] (Figure 3.1). More formally, given an encoder η_q for queries, an encoder η_d for texts from the collection, and a comparison function ϕ, dense retrieval with a bi-encoder design involves estimating the following over a collection $C = \{d_i\}$ [19]:

$$\mathrm{P}(Relevant = 1|d_i, q) = \phi(\eta_q(q)\eta_d(d_i)). \tag{3.1}$$

The ranker returns the top k texts from the collection based on these estimates of relevance. In addition, representations of texts from the collection ($\eta_d(d_i)$) can be computed in advance and indexed to facilitate low latency querying without depending on the queries ($\eta_q(q)$) [19].

Research on BERT-based dense representations for similarity comparisons appeared in 2019 from a few sources [19]. One example of a bi-encoder design for generating semantically meaningful sentence embeddings to be used in similarity comparisons is Sentence-BERT (SBERT), a modification of the BERT network using siamese and triplet networks [40]. It makes use of bi-encoders to produce representations for queries and documents separately. Moreover, the architecture incorporates two BERT encoders (with shared weights) followed by a pooling layer. The authors experimented with both BERT and RoBERTa[1] as the basis of the encoder and proposed three options to generate the representation vectors: Using the [CLS] token representation, Mean Pooling, and Max Pooling across all contextual output representations [40]. Different objective functions were experimented with depending on the task.

For classification, the representations for query, S_q, document, S_d, and their element-wise difference, $|S_q - S_d|$, are concatenated. This representation is fed to a softmax layer, trained using the cross-entropy loss. For regression, the cosine similarity between the representations S_q and S_d is computed using the mean-squared error loss as the objective. Lastly, A triplet-based objective using the triplet-loss formulated as follows [40]:

$$L_{triplet} = \max\left(\text{sim}(s_q - s_{d^+}) - \text{sim}(s_q - s_{d^-}) + \epsilon, 0\right), \tag{3.2}$$

with Sq as the representation embedding for a query, S_{d^+} as the representation embedding of the positive document, S_{d^-} as the representation embedding for the negative document, sim as the similarity function, and margin ϵ. The intent is to score the positive document at least ϵ higher than the negative one.

Bi-encoders need considerable training data and fine-tuning over the target task to achieve competitive performance. However, in many cases, there is only little training data available. An effective data-augmentation strategy known as Augmented SBERT (AugSBERT) [43] was released to solve this issue. It exceeded the previous SBERT in all tested tasks.

In this approach, sampled unlabeled pairs are labeled using a pre-trained cross-encoder. These weakly labeled examples, known as the silver dataset, are merged with the already labeled pairs, the gold training dataset. The pairs to be labeled using the cross-encoder are sampled using different strategies, such as random sampling (RS), kernel density estimation (KDE), BM25 sampling, semantic search sampling (SS), BM25 + Semantic Search Sampling (BM25-S.S.). In the first strategy, a sentence

[1] RoBERTa stands for Robustly Optimized BERT Pre-training Approach. It was presented by researchers at Facebook and Washington University. The goal of this paper was to optimize the training of BERT architecture in order to take lesser time during pre-training [41]

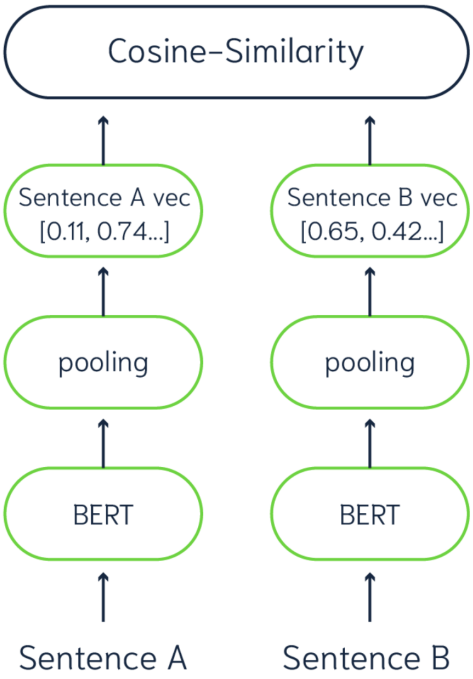

Figure 3.1: Structure of a bi-encoder model [42].

pair is randomly sampled and weakly labeled with the cross-encoder. It skews the label distribution of the silver dataset heavily towards negative pairs, given that positive pairs are scarce. The second strategy aims to get a similar label distribution for the silver dataset as for the gold training set. In the third strategy, the top-k passages for each query returned by BM25 are selected. The fourth strategy uses a trained SBERT model on the already labeled set to retrieve the top-k most similar sentences. The fifth strategy applies both BM25 and semantic search sampling techniques simultaneously.

The sampling strategy is critical to achieving an improvement using AugSBERT. The authors reported that BM25 sampling and KDE produce the best AugSBERT results, followed by semantic search (SS), while random sampling decreases performance compared to SBERT in most cases [43].

3.1.2 Ranking with Cross-Encoders

Cross-encoder models are encoder stacks that receive concatenated pairs of inputs and perform a full self-attention over the entire pair (e.g., query-document pair), learning a joint embedding for the query and document. Thus, the cross-encoder allows for rich interactions between the input context and candidate label [34].

A few approaches have been proposed for text ranking using cross-encoders based on pre-trained language models. For example, two approaches were introduced using a BERT cross-encoder for ranking, monoBERT, a pointwise approach, and duoBERT, a pairwise approach [44]. Both models were

fine-tuned on the document ranking task.

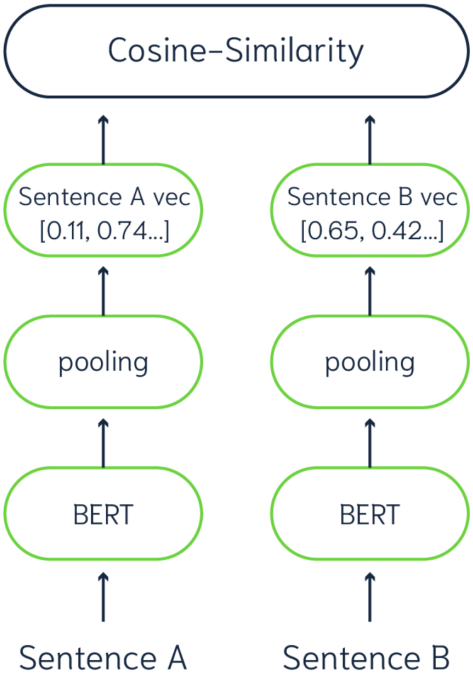

Figure 3.2: Structure of a cross-encoder model [42].

monoBERT is a BERT model that performs binary classification on query–document pairs into relevant/non-relevant. The task of relevance classification is to estimate a score quantifying how relevant a candidate document d_i is to a query q, which can be denoted as [19]:

$$\mathrm{P}(Relevant = 1 | d_i, q). \tag{3.3}$$

The model takes as input a sequence composed of the following:

$$[[\mathrm{CLS}], q, [\mathrm{SEP}], d_i, [\mathrm{SEP}]], \tag{3.4}$$

where q comprises the query tokens and d_i comprises tokens from the candidate text to be scored. The special tokens [CLS] and [SEP] are precisely those defined by BERT. The contextual representation of the [CLS] token is used as input to a fully-connected layer that produces the document score s that quantify how relevant a document d_i is to a query q [19].

The model was trained for re-ranking using cross-entropy loss:

$$L = -\sum_{j \in J_{\mathrm{pos}}} \log(s_j) - \sum_{j \in J_{\mathrm{neg}}} \log(1 - s_j), \tag{3.5}$$

where J_{pos} is the set of indexes of the relevant candidates and J_{neg} is the indexes of the non-relevant candidates, typically part of the training data.

An alternative is **duoBERT**, a pairwise approach, which compares pairs of documents. In this ranking model, BERT is trained to estimate the following:

$$\mathrm{P}\left(d_i \succ d_j \mid d_i, d_j, q\right), \tag{3.6}$$

where $d_i \succ d_j$ indicates that d_i is more relevant than d_j. It takes as input a sequence composed of a query and two documents as follows:

$$[[\mathrm{CLS}], q, [\mathrm{SEP}], d_i, [\mathrm{SEP}], d_j, [\mathrm{SEP}]]\,. \tag{3.7}$$

Given the length limitations of BERT, the query, candidates d_i and d_j are truncated to 62, 223, and 223 tokens, respectively, so that the entire sequence has at most 512 tokens when concatenated with the [CLS] token and the three [SEP] tokens [19]

Similar to monoBERT, the final representation of the [CLS] token is used as input to a fully-connected layer to obtain the probability $p_{i,j}$. The model is trained end-to-end with the following loss:

$$L = -\sum_{i \in J_{\mathrm{pos}}, j \in J_{\mathrm{neg}}} \log\left(p_{i,j}\right) - \sum_{i \in J_{\mathrm{neg}}, j \in J_{\mathrm{pos}}} \log\left(1 - p_{i,j}\right). \tag{3.8}$$

At inference time, the pairwise scores pi,j are aggregated so that each document receives a single score s_i. Several different aggregation methods were investigated, including SUM, BINARY, MIN, MAX, and SAMPLE. SUM and BINARY methods appear to have a better performance [44].

3.2 TREC 2021

The Text Retrieval Conference (TREC) is an annual workshop co-sponsored by the National Institute of Standards and Technology (NIST) and the Intelligence Advanced Research Projects Activity with the purpose of building the infrastructure required for extracting relevant information from large volumes of electronic documents. TREC starts each spring with a call for participation. The evaluation today is divided into different information retrieval (IR) research areas, or tracks, that examine different information access problems. Beyond providing the organizational framework for exploring different tracks at TREC, NIST also contributes evaluation resources and expertise. TREC has researched various tasks in different domains, including search in biomedicine.

This section focus on the TREC 2021 edition. More specifically, it overviews the Clinical Trials track.

3.2.1 Overview Clinical Trials Tracks

Unfortunately, the vast majority of clinical trials fail to recruit sufficient patients or to recruit them in time for the study. However, more trials may be run if patients can be matched to appropriate clinical trials efficiently using automated systems. This track aims to identify appropriate clinical trials given a patient descriptions.

Although over a hundred runs were submitted to the Clinical Trials track from 26 teams, this section only addresses a couple of these submissions and their results. The described submissions approach retrieval through transformer-based models, sparse retrieval models such as BM25, and complementary approaches to improve performance using techniques such as keyword extraction.

DoSSIER team

This team submitted three runs coming from subsequent stages of the same architecture [45]. Before any experiment, they pre-processed the documents and topics using ScispaCy, a specialized NLP library for processing biomedical texts [46]. Then, fields such as title, brief summary, and eligibility criteria from the clinical trials were concatenated into a single text representing every clinical trial.

For the first submission, which the authors called baseline, the top-ranked 1000 clinical trials were returned for each topic using the BM25 model with its default parameters.

In the second submission, called postproc, 2000 trials were retrieved by the BM25 and post-processed using age, gender, and health status data extracted from patient descriptions. Retrieved trials were then filtered based on the stored metadata to remove the mismatches. Authors claim that using the filtering removed 34% of potentially ineligible trials on average.

The third submission uses the output from the postproc run as an initial pool. It uses a custom formula for re-ranking using the SPECTER language model [47]. In the first step, the inclusion and exclusion criteria were encoded one by one using SPECTER. The same was done for the patient descriptions. Next, the cosine similarity between the inclusion criteria and the patient encoding and exclusion criteria and patient was computed. Both of them were sorted, and they took the three highest inclusion and exclusion similarity scores. Finally, they calculated the final score based on the reasoning that the patient's description should overlap with the inclusion criteria and be as distinct as possible from the exclusion criteria.

The authors found that the post-processing improves the model performance over the baseline run, but the custom re-ranking negatively impacts average results. Thus, the submission with the best results was the postproc with 0.455 on the NDCG@5, 0.413 on the NDCG@10, 0.252 on the PREC@10, and 0.478 on the reciprocal rank (RR). The results for the other submissions are shown in Table 3.1.

Table 3.1: Performance comparison for submitted runs.

Submission	NDCG		P@10	RR
	@5	@10		
1 baseline	0.401	0.380	0.208	0.406
2 postproc	**0.455**	**0.413**	**0.252**	**0.478**
3 rerank	0.322	0.293	0.187	0.368

WisPerMed Text team

This team compared classical information retrieval options with transformer-based methods through five submissions. Before ranking, each topic went through a pre-processing step in which age and gender information were extracted using regular expressions, using the standard re[2] module in python. This extracted information was used to restrict the trials to a set of trials for which a patient is eligible, using the fields min age, max age, and gender of the trial. The authors considered the trials that did not contain sufficient information to exclude a patient as candidate trials for that patient. Text pre-processing, with removed punctuation and document tokenization, was only done for the third submission.

For the first submission, the brief summaries of each available clinical trial and the different topics were embedded using a clinical-oriented pre-trained BERT model, Clinical BioBERT [48]. Then, the cosine similarity between the embeddings of the topics and those of the brief summaries was computed. Finally, the clinical trials for each topic were ranked based on the obtained similarity scores.

In the second submission, the Clinical BioBERT model was used with KeyBERT [49], which finds the keywords that best describe the entire document. KeyBERT uses the document embedding and word embeddings of a specific document to find the words that are the most similar to the document based on cosine similarity. Thus, the top 10 keywords were extracted from each topic using this model. These keywords were used as the search query, and the brief summaries of the clinical trials as the documents to search from using an implementation of BM25 in Rank-BM25[3] with the parameter k1 set to 1.5 and b set to 0.75. The scores for each query were then calculated, and a ranking of the clinical trials was identified for each topic.

For the third submission, the model used to encode the topics and the trials was BioBERT. The fields used for the trails were the title and the brief summary. Apart from the dense BioBERT embeddings, a weighted similarity based on sparse representation was also included in the final similarity calculation. The semantic match between topics and clinical trials and the similarity based on character-level representations were incorporated into this ranking.

The fourth submission uses a multi-stage pipeline, where the first-stage retrieval uses Elasticsearch[4]

[2] https://docs.python.org/3/library/re.html
[3] https://github.com/dorianbrown/rank_bm25
[4] https://hub.docker.com/_/elasticsearch

Lucene [50] BM25, with the parameter k_1 set to 1.2 and b to 0.75. The results of this ranking were then re-ranked based on the transformers approach described in the third submission. The re-ranking was achieved by sorting the top 1000 trials from the initial retrieval using the similarity scores between dense embeddings and sparse TF-IDF representation.

The last submission prioritized medical conditions for which only a few trials were available. It used the Unified Medical Language System (UMLS)[5] in an attempt to find relevancy between patient topics and clinical trials based on biomedical concepts.

Apparently, in their experiments, the authors proved that transformer-based models performed poorly compared to simple methods such as BM25, to their surprise. They suspected that a zero-shot retrieval setting using pre-trained language models might not achieve good results without additional fine-tuning the dense retrieval model, especially in semantic similarity. A summary of the results obtained for each run is shown in Table 3.2.

Table 3.2: Results of the presented runs in comparison to best runs for the Clinical Trials Track.

Submission	NDCG		P@10	RR
	@5	@10		
1 Clinical BioBERT + Cosine	0.0760	0.0763	0.0493	0.1000
2 KeyBERT + BM25	**0.3432**	**0.3336**	**0.2133**	**0.4069**
3 BioBERT + TF-IDF	0.2047	0.1948	0.1427	0.2863
4 Elasticsearch + BioBERT	0.1852	0.1583	0.0973	0.2630
5 ULS	0.1449	0.1372	0.0840	0.2122

Table 3.3: DoSSIER, WisPerMed, and Clinical Trials Track best results.

Submission	NDCG		P@10	RR
	@5	@10		
(DoSSIER) postproc	**0.455**	**0.413**	**0.252**	**0.478**
(WisPerMed Text) KeyBERT + BM25	0.3432	0.3336	0.2133	0.4069
All TREC Submissions: Median	–	0.3040	0.1613	0.2942
All TREC Submissions: Best	–	0.8491	0.7480	1.0000
Best Team	–	**0.7118**	**0.5933**	**0.8162**

[5]The UMLS, or Unified Medical Language System, is a set of files and software that brings together many health and biomedical vocabularies and standards to enable interoperability between computer systems.

3.2.2 Summary

The described submissions from the DoSSIER and WisPerMed Text team used classical information retrieval approaches as well as more modern approaches based on transformer models. In both cases, the submissions using BM25 outperformed the ones using transformer-based models. Despite the similar results, the DoSSIER team obtained the best results with its second submission (postproc). Both teams obtained results above the median of the results for this track. However, these results are far below the ones obtained by the best team. The comparison of both teams best results and the the median results of this track are presented in Table 3.3.

4

Clinical Trial Data

Contents

The available data is a critical component for the development of this work. Thus, it is important to comprehend this data. This chapter describes the test collections used in this study to retrieve relevant clinical trials for patient descriptions and train a dense retrieval model to improve retrieval effectiveness.

4.1 TREC-CT

For this task, a collection of 375,580 publicly accessible clinical trials XML files with descriptions was used, which was obtained from https://clinicaltrials.gov on April 27, 2021. These XML files correspond to current and historical clinical trials in the United States and other places.

Additionally, 75 topics were made available, consisting of synthetic patient cases in the form of an admission note created by individuals with medical training. These provided topics are unstructured text documents ranging from 5 to 10 sentences as illustrated in Figure 4.1.

The clinical trials corpus includes different fields describing the trials, including the official title, a brief summary, a detailed description of the trial, eligibility criteria, gender, and others. Despite the amount of available fields, only some of these are found to be relevant, including:

- **brief_title**: Contains only essential keywords from the title.
- **official_title**: The title plus expanded biomedical terms.
- **brief_summary**: A summary of the clinical trial containing a brief description of the study.
- **detailed_description**: Detailed description of the study.
- **criteria**: This fields correspond to the trial eligibility criteria. It comprises the inclusion and exclusion criteria, often divided in its respective header (see Figure 4.2)

<topic number="9">
41 year old man with history of severe intellectual disability, CHF, epilepsy presenting with facial twitching on the right and generalized shaking in at his NH which required 20 mg valium to cease seizure activity. Per outside medical patient was felt to have focal epilepsy with secondary generalization, likely due to anoxic brain injury at birth, and probably related to the atrophic changes seen on MRI, particularly in the left temporal lobe. The patient first developed seizures at age 13 found by family to have a generalized convulsion. He had a second seizure two years after his first episode. He was maintained on Dilantin and phenobarbital. The patient went 20 years without another seizure. He was recently tapered off Dilantin, and it was felt that perhaps this medication was necessary to maintain him seizure free. The patient had no further events during the hospital course and was back at his baseline at the time of discharge. Full EEG reports are pending at the time of dictation. Past Medical History: Epilepsy as above, CHF, depression
</topic>

Figure 4.1: Example of a topic from the TREC-CT test collection.

Each topic-trial pair is judged using three labels: eligible (relevant and the patient does not satisfy any exclusion criteria), excluded (relevant but the patient satisfies some of the exclusion criteria), and not relevant. These relevances r are comprised in a set of relevance judgments, (q, d, r) triples, where q is the topic, and d a clinical trial document. This test collection has 35,832 relevance judgments in a text file, known as qrels file.

This test collection was made available for the participants' runs to be evaluated on the task. Nowadays, it is used to test retrieval methods in the clinical trials domain, as in this study's case.

```
<brief_title>
    Congenital Adrenal Hyperplasia: Calcium Channels as Therapeutic Targets
</brief_title>
<brief_summary>
    <textblock>
        This study will test the ability of extended release nifedipine (Procardia XL), a blood pressure
medication, to permit a decrease in the dose of glucocorticoid medication children take to treat con-
genital adrenal hyperplasia (CAH).
    </textblock>
</brief_summary>
<detailed_description>
    </textblock>
        This protocol is designed to assess both acute and chronic effects of the calcium channel
antagonist, nifedipine, on hypothalamic-pituitary-adrenal axis in patients with congenital adrenal hy-
perplasia. The multicenter trial is composed of two phases and will involve a double-blind, placebo-
controlled parallel design. The goal of Phase I is to examine the ability of nifedipine vs. placebo
to decrease adrenocorticotropic hormone (ACTH) levels as well as to begin to assess the dose-
dependency of nifedipine effects. The goal of Phase II is to evaluate the long-term effects of nifedip-
ine; that is, can attenuation of ACTH release by nifedipine permit a decrease in the dosage of
glucocorticoid needed to suppress the HPA axis? Such a decrease would, in turn, reduce the dele-
terious effects of glucocorticoid treatment in CAH.
    </textblock>
<detailed_description>
<eligibility>
    <criteria>
        <textblock>
            Inclusion Criteria:
            - diagnosed with Congenital Adrenal Hyperplasia (CAH)
            - normal ECG during baseline evaluation
            Exclusion Criteria:
            - history of liver disease, or elevated liver function tests
    </textblock>
</criteria>
</eligibility>
```

Figure 4.2: Example of a clinical trial from the TREC-CT with some of its free-text fields.

4.2 Complementary Data

One of the experiments in this work focuses on improving the effectiveness of dense retrieval with a transformer-based model by fine-tuning it with in-domain data. The idea was to use the Clinical Trials Track test collection just for retrieval, find complementary data related to the track's task, and use it for fine-tuning the neural ranking model.

Although several works have approached the task of matching patients to clinical trials, there are almost no test collections available with a similar format to the TREC 2021 CT Track.

The most similar collection to this track comes from [51], and is known as SIGIR 2016 Clinical Trials (SIGIR-CT). In this setup, free-text patient descriptions are matched with a snapshot of trials from ClinicalTrials.gov. In addition to the longer patient descriptions, the authors also obtained ad-hoc queries from medical professionals, representing search terms that those professionals would use to find appropriate trials for each patient.

Another collection used for retrieving relevant clinical trials for a set of patients came for the 2017 edition of the TREC Precision Medicine (PM) Track[1]. Again, the collection of trials is derived from ClinicalTrials.gov. However, the topics available were limited to cancer patients represented with semi-structured data (disease, variant, and demographics) instead of unconstrained free-text descriptions like the TREC 2021 CT-Track. Despite the differences, this collection was considered regardless due to the lack of available data.

The following sections briefly describe these two test collections.

4.2.1 SIGIR-CT

The dataset includes 204,855 publicly available clinical trials and 60 patient case reports describing a patient with certain conditions and observations.

The trials are again crawled from ClinicalTrials.gov, but as they were collected on December 16, 2015, fewer trials were included than in the TREC-CT collection from April 27, 2021.

The topics were adopted from the TREC Clinical Decision Support Track, 30 from 2014 and 30 from 2015. These topics are in three different formats: ad-hoc queries (short queries generated by medical assessors), summaries (average of 22 words), and descriptions (average of 78 words). Both topics and clinical trials are very similiar to the ones presented in Figure 4.1 and Figure 4.2 respectively. Despite the similarities, Figure 4.3 illustrates a topic from the SIGIR-CT dataset.

[1] http://www.trec-cds.org/2017.html

<topic number="20141">
A 58-year-old African-American woman presents to the ER with episodic pressing/burning anterior chest pain that began two days earlier for the first time in her life. The pain started while she was walking, radiates to the back, and is accompanied by nausea, diaphoresis and mild dyspnea, but is not increased on inspiration. The latest episode of pain ended half an hour prior to her arrival. She is known to have hypertension and obesity. She denies smoking, diabetes, hypercholesterolemia, or a family history of heart disease. She currently takes no medications. Physical examination is normal. The EKG shows nonspecific changes.
</topic >

Figure 4.3: Example of a topic from the SIGIR-CT test collection.

The relevance assessment file contains 3,870 relevance judgments, however, the labeling scheme differs from that of the TREC CT track. In both test collections, some trials are labeled as entirely relevant, e.g., "Highly likely to refer this patient for this clinical trial" or "eligible," and some are labeled as not relevant. However, this collection includes an intermediate label indicating, "Would consider referring this patient to this clinical trial upon further investigation." In contrast, TREC's intermediate label indicates that patients are excluded, as they meet the inclusion criteria and some exclusion criteria.

4.2.2 TREC-PM

This test collection has 241,006 clinical trials extracted again from ClinicalTrials.gov on April 2017 and 30 synthetic patient cases created by precision oncologists at the University of Texas MD Anderson Cancer Center. These topics include the disease, genetic variants, demographic, and potentially other patient information as shown in Figure 4.4.

The relevance assessment file contains 13,019 relevance judgments in a three-scale labeling scheme, where the trials can be labeled as definitely relevant, partially relevant, or not relevant. The partially relevant is primarily the same as definitely relevant, but with the exception that disease can also be more general, where the form of cancer in the trial is more general than the one in the topic. Gene can also be a missing variant, where the trial does not focus on the particular gene in the topic, or a different variant, where the trial focuses on the particular gene in the topic but a different variant than the one in the topic.

```
<topic number="1" >
    <disease>Acute lymphoblastic leukemia</disease>
    <gene>ABL1, PTPN11</gene>
    <demographic>12-year-old male</demographic>
    <other>No relevant factors</other>
</topic>
```

Figure 4.4: Example of a topic from the TREC-PM test collection.

4.3 Summary

This study utilizes the three test collection described previously. The TREC-CT data for the retrieval task and the other two collections for fine-tuning a pre-trained dense retrieval model with in-domain data. Despite the multiple attempts in the task of matching patients to clinical trials, the scarcity of quality data is a reality. Table 4.1 summarizes the three available test collections.

Table 4.1: Summary of the available test collections.

Dataset	Year	#Topics	#Trials	#Judgments	#Relevants
TREC-CT	2021	75	375,580	35,832	5,570
SIGIR-CT	2016	60	204,855	3,870	421
TREC-PM	2017	30	241,006	13,019	436

5

Approaches for Clinical Trial Search

Contents

This chapter addresses the implementation of the different approaches used to match patient case descriptions to eligible clinical trials, such as the TREC 2021 Clinical Trials track.

As described previously, TREC 2021 Clinical Trials track participants were challenged with retrieving clinical trials for patient case descriptions within a given test collection. In the case of this study, several retrieval experiments were tested to address this task. The implementation of these experiments can be divided into two main parts. The first part investigated a sparse retrieval approach, whereas the second focused on exploring neural retrieval techniques and compared them with the previous approach. In addition, some techniques were tested to improve effectiveness, including fine-tuning the pre-trained model used for dense retrieval, experimenting with negative sampling for constructing the training batches, and a method to penalize pairs in a ranked list based on regular expression rules.

Even though multi-stage ranking pipelines have been a solution in modern retrieval systems due to a balance between effectiveness and efficiency, this study focuses on first-stage retrieval. It investigates the effect of transformer-based models on first-stage retrieval since this architecture plays an essential role in almost all large-scale IR applications and has been long dominated by term-based models [52].

The implementation of the various experiments used two Python libraries. Pyserini [53], an easy-to-use toolkit that provides effective first-stage retrieval in a multi-stage ranking architecture, and FAISS[1], a tool that enables efficient similarity search. The methods used to retrieve were the following:

- Sparse retrieval using BM25.
- Dense retrieval using pre-trained models trained for semantic search on a large corpus.

The retrieval process can be divided into three steps. The first stage concerns pre-processing the available clinical trials and topics, while the second step indexes these clinical trials and saves them on a local disk. Each retrieval experiment used the TREC-CT test collection. The last step is to search through these indexes and retrieve the top-1000 clinical trials for each topic. While the third step is necessary whenever a retrieval approach is being explored, the second stage occurs whenever a new combination of fields is being tested, or a new model is used.

It is necessary to evaluate the obtained results, in this case, the ranking list, to assess the effectiveness of the retrieval. For evaluation, it was used ranx[2], a python library for fast ranking evaluation.

5.1 General Architecture

All the retrieval approaches use a first-stage architecture, where a set of documents are returned from a large collection. As mentioned before, each experiment uses a python library to index clinical trials and search. It uses the Pyserini toolkit for sparse retrieval, whereas in the case of dense retrieval, it

[1] https://github.com/facebookresearch/faiss
[2] https://amenra.github.io/ranx/

uses the FAISS library. The architecture and the test collection used are always the same throughout all the experiments. Therefore, the only changes are in the indexing process and the model used in the experiment. The general architecture of the retrieval process is shown in Figure 5.1.

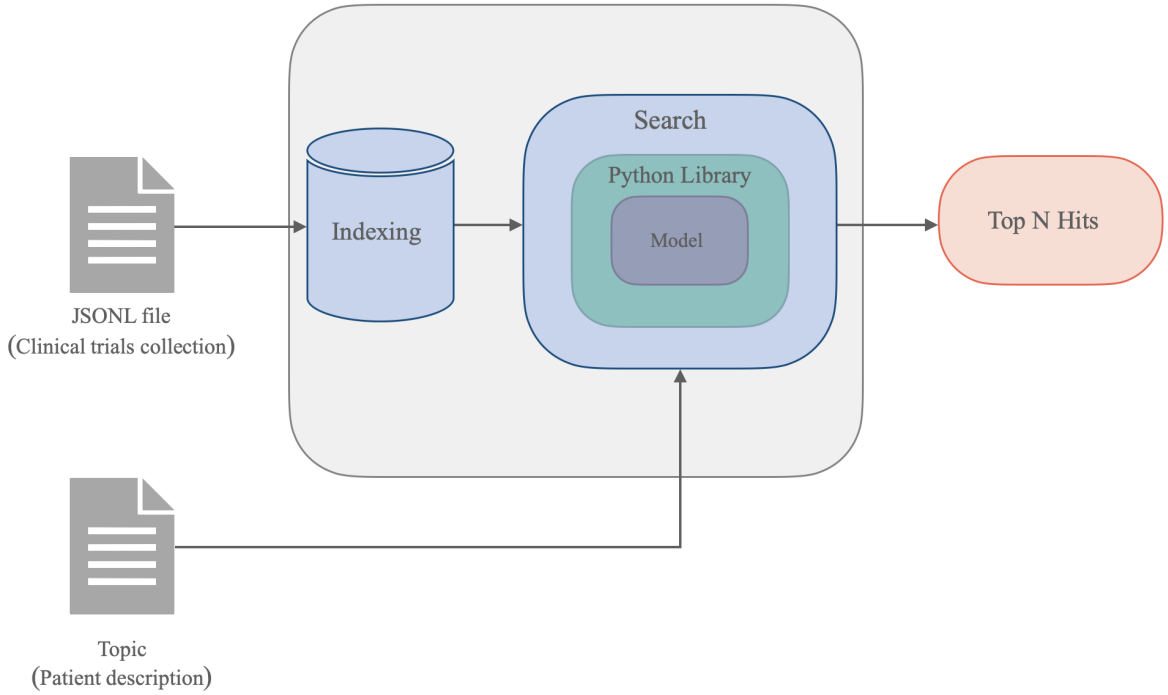

Figure 5.1: General representation of the implemented retrieval architecture.

5.2 Data Pre-Processing and Indexing

The first step before exploring retrieval methods was to examine the test collection to understand how the clinical trials and topics are organized and process these documents if necessary. As explained in Section 4.1, the TREC-CT comprises 375,580 clinical trials and 75 topics with unstructured text in an XML file scheme as illustrated in Figure 4.2 and Figure 4.1, respectively. Since the sparse retrieval implementation uses Pyserini and requires a specific format for the documents to be indexed and posteriorly searched, it was necessary to process these files and save them locally in the required format. Pyserini (via Anserini) provides ingestors for document collections in three formats:

- A JSON file for each document in the collection
- An array of JSON documents in a JSON file
- A JSON on an individual line for each document in a unique file (often called JSONL format)

Given this, the opted format was the JSONL format, in which a JSON line defined each clinical trial document in a unique file representing the entire collection. However, before each trial (XML file) is stored in a JSON representation, all the texts contained in the various tags are extracted recursively. The result is a unique string containing information on the different fields of the trial, with no more than a single neighboring whitespace between terms and each field separated by a newline character. This string can aggregate the entire information of the clinical trial, i.e., the texts contained in all the fields, or a combination of certain fields. Processing the texts and choosing the right combination of fields can improve retrieval performance and reduce the vocabulary mismatch between the trials and topics.

As stated, each line in the JSONL file represents the clinical trial. It is organized in two keys: *id* containing the trial's unique identifier, and 'contents' composed of a unique text containing all the trial information. Figure 5.2 represents a clinical trial in this format.

After parsing each clinical trial XML file and storing its information in the required format, the next step in the implementation is to index the documents.

{"id": "NCT04838431", "contents": "A First Trimester Prediction Model for Large for Gestational Age Infants: a Preliminary Study\nDevelopment of a Prediction Model for Large for Gestational Age Infants at First Trimester: a Preliminary Study\nLarge for gestational age infants (LGA) have increased risks of adverse short-term perinatal outcomes. This study aims to develop a multivariable prediction model for the risk of giving birth to a LGA baby, by using biochemical, biophysical, anamnestic, and clinical maternal characteristics available at first trimester. We decided to conduct this prospective study that include all singleton pregnancies attending the first trimester aneuploidy screening at the Obstetric Unit of the University Hospital of Modena, in Northern Italy."}

Figure 5.2: JSON representation of a clinical trial.

5.2.1 Data Configuration

Before indexing the clinical trials to be searched, it is essential to choose the way they will be indexed, i.e., the combination of fields to use. Selecting the appropriate fields is a relatively easy strategy that could improve the retrieval results. However, it should be addressed early in the retrieval process or at least considered when attempting to improve results before trying far-fetched strategies. This strategy can be particularly important when ranking with term-based models since these are prone to the vocabulary mismatch problem. It should also be considered when performing dense retrieval, e.g., with transformer-based models, since these limit the length of the input sequences.

In this study, experiments were conducted in two scenarios. The first consisted in executing a retrieval run with sparse representations using BM25, in which all the fields presented in the clinical trials were

indexed. The second scenario was to find a good combination of fields to improve the retrieval results.

The combination of fields that provided the best results was using the free text fields, i.e., the brief title of the trial, official title, brief summary, detailed description, and the eligibility criteria containing the inclusion and exclusion criteria for the trial.

This combination of fields proved ti be more effective than using all the fields in the trials. Therefore, it was the starting point for the dense retrieval experiments. Before starting the ranking process, it was necessary to know if this combination of fields would fit the transformer-based models. In other words, how much information would be lost when encoding the clinical trials to produce their dense representations, having a maximum sequence length of 512 tokens.

Using the model's tokenizer, it was possible to analyze the tokens distribution for the trials and the topics since they are also encoded to produce embeddings. This analysis focused on the individual fields from the combination described above but also on new combinations of fields. As the title fields are often short, they prove to be under the model's maximum limitation. The same applies for the topics. More descriptive fields such as brief summary, detailed description and the eligibility criteria exceed the maximum sequence length for a good number of trials.

The defined strategy was to combine title fields with more descriptive fields in pairs to avoid as much as possible to exceed the length limitations and, at the same time, preserve relevant information. Table 5.1 presents the number and percentage of trials that exceed the length limitation per field and the combination of fields experimented. Despite exceeding the maximum input sequence limitation of the model, the combination of the five free text fields was the one that performed best on the dense retrieval task when compared with the other combinations reported in Table 5.1. The results of the other combinations will not be reported because they are not relevant for the discussion.

Table 5.1: Distribution of tokens for single fields and combination of fields. Percentage of trials with more than 512 tokens for a single field and for a combination of fields.

Fields	# Tokens				# Trials
	Min	Max	Mean	Standard Deviation	>512 tokens (%)
Single					
brief_title	4	113	21.17	9.49	0
official_title	4	1358	34.74	29.52	0.05
brief_summary	3	6109	134.84	132.64	2
detailed_description	3	9209	369.96	433.79	20.46
criteria	3	4411	315.89	373.01	11.84
Combination					
brief_title + criteria	4	4439	228.28	338.92	12.53
brief_title + detailed_description	5	9246	383.53	433.06	21.78
official_title + brief_summary	6	6736	169.27	137.67	2.72
Combination of all 5 single fields	23	10931	759.92	603.14	57.79

As mentioned above, first-stage retrieval ranks on large-scale documents in the collection. Thus, the efficiency of the first-stage retrieval models is a crucial aspect to consider. In practice, retrieval systems need to build an index to support storing and fast retrieval of documents in the entire collection. Despite multiple indexing techniques, the most popular indexing scheme for term-based search is the inverted index due to its simplicity and efficiency. One of the reasons why the inverted index works well is that the documents–term matrix is very sparse [52].

With the appearance of dense retrieval models and their dense representations of documents, the inverted index method is no longer feasible to retrieve documents efficiently from a large collection. Hence, dense vector index based on approximate nearest neighbor search algorithms appeared.

Having all the data configured in the appropriate format it was time to index the collection. This study uses the two schemes to explore retrieval with sparse and dense representations.

5.2.2 Gererating the Inverted Index

Before building an inverted index, each document in the collection is parsed and segmented into a list of tokens. After having this list of tokens, the inverted index is created, mainly consisting of a dictionary and a collection of posting lists. This dictionary comprises all the terms in the collection and their document frequencies. Each posting list stores document identifiers and term occurrence frequencies.

To generate the inverted index, the sparse retrieval implementation uses Pyserini. It builds inverted indexes in the Lucene format[3] [53]. The behaviour is the same described above, in which all the terms in the collection of documents are saved in a dictionary. For each term in the dictionary, it records which document it occurs in and the specific location. By default, Pyserini builds an index that only stores term frequencies, which is sufficient for simple ”bag of words” querying (and yields the smallest index size).

5.2.3 Generating the Dense Vector Index

For the implementation of dense retrieval approaches, to index the dense representations vectors (embeddings) obtained from the encoded trials in the collection, it uses FAISS. The vectors representing each trial are stored in a FAISS index, more precisely, a flat index. This index doesn't optimized the stored vectors and come with perfect search-quality at the cost of slower search speeds when compared with other type of indexes, but still guarantees fastness. This is the ideal index when search quality is an unquestionably high priority and search speed is less important.

[3] https://lucene.apache.org/core/3_6_2/fileformats.html

5.3 Sparse Retrieval

The first approach uses a term-based model for retrieving. This case uses Pyserini's default implementation of sparse retrieval with BM25. For each query (topic), the top 1000 most similar documents (clinical trials) are fetched in turn with the help of the inverted index previously computed and stored on disk. Concretely, each query is processed with one term at a time. Initially, each document has a similarity of zero to the query. Then, for each query term, the similarity score of each document in the term's posting list increases by the contribution of the term to the similarity of the query-document pair. Finally, after processing all query terms, the 1000 largest similarity scores are determined, and the corresponding document list is returned for the given query.

In practical terms, to perform a search, Pyserini receives the number of documents to retrieve (e.g., 1000) and a given query used to search the index. This process occurs for each topic, and the final result is a list comprising a ranking with the most 1000 significant trials for each topic and its scores. Figure 5.3 is a visual representation of the sparse retrieval process.

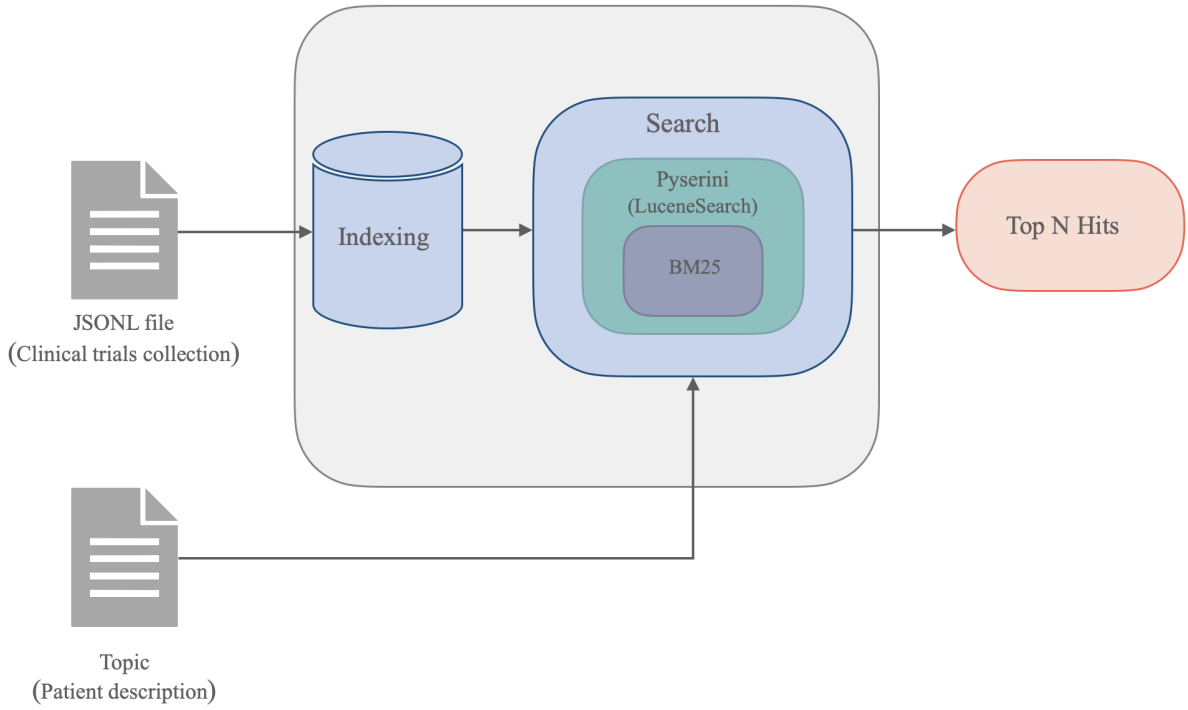

Figure 5.3: Representation of the sparse retrieval architecture.

5.4 Dense Retrieval

For retrieval with dense representations, the implementations use Facebook's FAISS library, which is an efficient tool for similarity search across dense vectors. Having the vector representations of the clinical trials from the entire collection and a FAISS index containing these vectors, the Faiss library is responsible for searching and retrieving the top 1000 relevant trials for each topic.

During the search, all the indexed vectors are decoded sequentially and compared to the vector for which nearest neighbors are being computed, the query (topic) vector. Both the clinical trials and the topics were encoded, independently, into dense representations using the same BERT-based model.

The similarity measure between vectors depends on the FAISS index that is being used. As described previously, the generated index for dense retrieval is flat in this case. More precisely, it is an IndexFlatIP index, which calculates the dot product between vectors. A visual representation of this dense retrieval process can be seen in Figure 5.4.

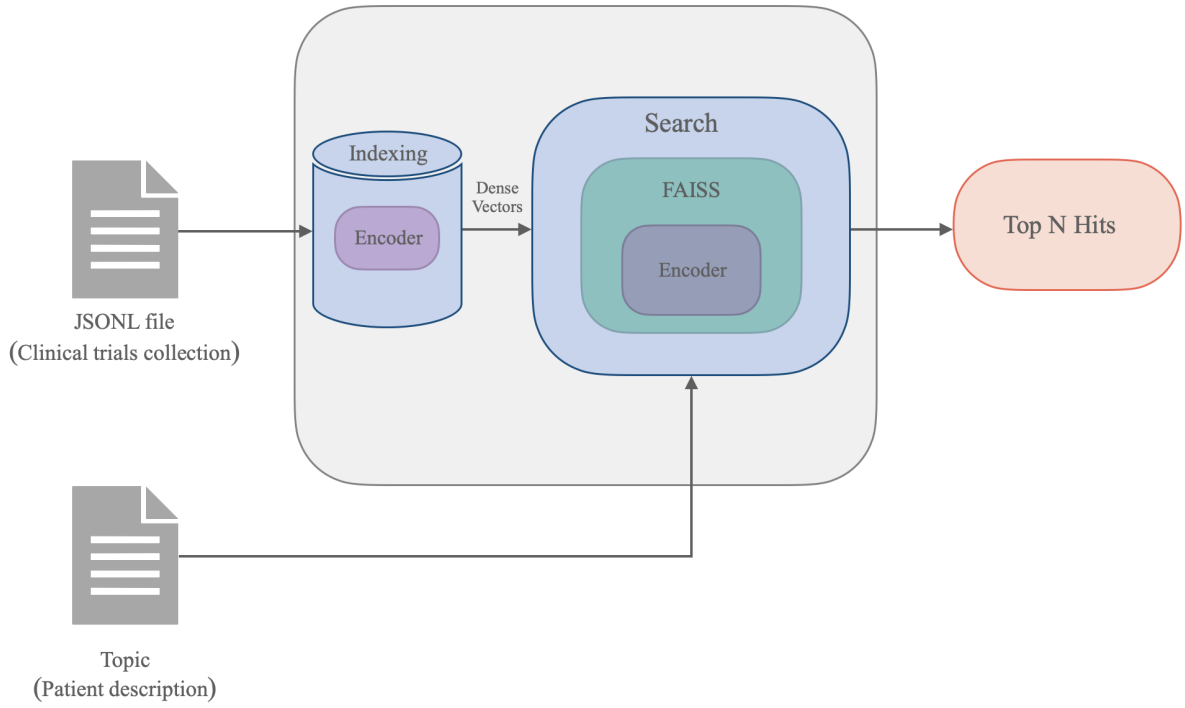

Figure 5.4: Representation of the dense retrieval architecture

5.4.1 Bi-Encoders

The first model tested was the msmarco-bert-base-v5, specifically trained for semantic search on the MSMARCO Passage Ranking dataset, consisting of 500k real queries from the Bing search engine and the relevant passages from various web sources. From the available bi-encoders on the Sentence Transformers trained for semantic search on the MSMARCO dataset, the msmarco-bert-base-v5 model was the one to yield the best performance.

The second model to be tested was the all-distilroberta-v1, a distilled version of RoBERTa [41]. The all-distilroberta-v1 is an all-round model tuned for many use cases and designed as a general-purpose model. It was trained on all available training data, a large and diverse dataset of over 1 billion training pairs. It includes data from broader sources, such as Reddit, WikiAnswers, Stack Exchange, and others, to more scientific sources, including S2ORC and SPECTER.

5.4.2 Model Fine-Tuning

A couple of the experiments used a transformer bi-encoder model to perform dense retrieval in a zero-shot scenario. In this scenario, the model was pre-trained on a large corpus not related to the domain field in this study, the biomedical domain. This pre-trained model was fine-tuned on a small set of in-domain labeled data. In fact, a couple of pre-trained models were tested and fine-tuned in multiple strategies with different data that will further discuss in the next chapter. Due to its effectiveness, the motivation behind fine-tuning these models is that this process is the facto standard in state-of-the-art NLP [15], and consequently, the will to improve results obtained in previous tested experiments in this study.

The experimented models used in the semantic-based retrieval were Sentence Transformers models pre-trained on large datasets using the Sentence Transformer[4] framework. The key is twofold:

- Prepare the dataset accordingly to input data into the model.
- Choose an appropriate loss function.

The dataset configuration often consists of pairs of sentences and labels indicating their similarities. In the case of this work, the input data would be pairs of topic-trial and their relevance judgments. For example, several samples were created to configure data before training comprising the topic-trial pair and its relevance judgment. As will be explained later, adding the relevance label to the pair is not necessary with the loss function in this study's experiments.

The relevance judgments in the qrels file are converted from a 3-point scale into binary scales where all the not relevant and intermediate label pairs are labeled as 0, and the eligible ones are labeled as

[4] https://www.sbert.net

1. Then, these relevance judgments are iterated, and each topic and respective trial is searched and collected from its files (i.e., the JSONL file containing the collection and the file comprising the topics). The final result is a single file containing each pair with its relevance. The number of pairs is bounded to the number of available relevance judgments, meaning that, considering P the number of topic-trial pairs and R the number of relevance judgments, $P \leq R$.

The chosen strategy for fine-tuning the models is training with in-batch negatives [35, 54, 55]. It is an effective strategy for learning a bi-encoder model that boosts the number of training examples. It is an alternative to reduce computational cost [35, 55] and surpass the problem of having a small dataset (low number of relevance judgments) and, consequently, a small number of training examples (see Table 4.1 for a summary of the available test collections).

Multiple Negative Ranking Loss

As described previously, the fine-tuning of bi-encoder model consists of training with in-batch negatives. The idea is to use documents from other query-document pairs in the same batch as negative samples.

Assuming that there are B queries in a batch, each one is associated with a relevant document. Let $\boldsymbol{Q}$ and $\boldsymbol{D}$ be the $(B \times d)$ matrix of query and document embeddings in a batch size of B. $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{P}^\top$ is a $(B \times B)$ matrix of similarity scores, where each row corresponds to a query paired with B documents. In doing so, the model is trained on $B^2(q_i, d_j)$ query-document pairs in each batch. When $i = j$, any (q_i, d_j) pair is a positive instance, and it is a negative one when $i \neq j$. It creates B training instances in each batch, where $B - 1$ are negative documents for each query. In rare cases, for a given query, some documents from other query-document pairs could be relevant.

Note that with this technique, the dataset configuration consists of positive topic-trial pairs, meaning only the relevant instances from the relevance judgements file are considered for training examples.

The Sentence Transformers framework uses the Multiple Negative Ranking loss (MNR loss) for this strategy. The authors state that this loss function is a good option for training embeddings for retrieval setups, and its performance usually increases with increasing batch sizes. It can be formalized as:

$$L = -\frac{1}{B}\sum_{i=1}^{B}\frac{\exp\left(\text{sim}\left(q_i, d_i\right)\right)}{\sum_j \exp\left(\text{sim}\left(q_i, d_j\right)\right)}. \tag{5.1}$$

It optimizes the cross-entropy loss, where the similarity between a query q_i and a document d_i and is normalized over all the similarities between q_i and d_j. The sim is the similarity function between the query and document vectors and can be the cosine similarity or dot product. The similarities are multiplied by a scaling factor of 20, making the differences larger.

Having the embeddings and computed the B^2 similarity scores, i.e., the predictions within the batch, it sees the task as a classification task. It applies the cross-entropy loss between the predictions and the

gold labels. In this case, the gold labels are in a square matrix, represented by 1 in the diagonal when $i = j$ (and 0 otherwise).

Negative Sampling and Batching

Negative sampling is a technique for finding non-relevant documents for a combination of query and relevant documents. Sampling techniques are necessary given that it is expensive to use every other document in the collection as a negative sample for a query when training a neural ranking model. This study experimented with random negative sampling to collect negative samples for pairs queries and relevant documents.

The first negative sampling experiment occurred when fine-tuning the pre-trained models used for dense retrieval with in-batch negative sampling. Since negative samples are obtained within each training batch and not previously selected, these samples are collected randomly for a query-document pair.

Another random technique was used to obtain negative samples. The method used was very straightforward and consisted of randomly choosing documents that were judged as not relevant for a given query in the relevance judgment file. When building batches, triples are first created by associating a query to its positive document and a negative document, sampled as described above. Then, batch-wise negative pairing is applied to maximize the effective training data by using a given query's positive and negative documents as negatives for the others, as described previously. This way, 2 x B x B pairs can be extracted for a batch of B triples.

Although the MNR loss in the Sentence Transformer framework allows using multiple negative samples for each pair of query-document, the conducted experiments only used one sampled negative due to a lack of GPU memory.

Framework

Each experiment with training was operated within the following framework: between each epoch (approximately 52 to 54 steps, depending on the dataset), the trained model performs a retrieval run on the TREC-CT test collection, and the effectiveness of the ranking is evaluated (using the metrics described in Section 2.2.2). The trained model is stored to be used in the next epoch. When a trained model is less effective than its previous version, it is no longer trained. It took on average four epochs to train the model (longer fine-tuning did not yield improvement on the retrieval task).

Each model was trained with batches of size 8, a linear warmup schedule for 50% of an epoch, and Adam optimizer with learning rate 2e-5. The maximum input sequence was set to 512.

5.5 Filtering Results with Regular Expression Rules

As described previously, each topic-trial pair is judged as eligible, excluded, or not relevant. The exclusion depends on the patient's characteristics and the eligibility criteria specifying what characteristics or conditions a patient must have/not have to be suitable for the trial. For example, a clinical trial can be relevant for the patient based on their disease or condition, but patients might be ruled out due to simple factors such as age and gender, or more complex factors. Thus, apart from the relevance of the semantic content, the retrieved clinical trial should be as close as possible to the patient's description in terms of the inclusion criteria and distant as possible in terms of exclusions.

Since choosing a correct clinical trial largely depends on matching its eligibility criteria in terms of some factors (e.g., age and gender), in some of the conducted experiments, a simple strategy was used to penalize the relevance of a given trial to a topic when the patient meets the exclusion criteria. The idea was to extract and standardize metadata from the patient descriptions using regular expressions, as implemented in other thesis relevant to this study [56]. The extracted metadata was the age and gender of the patients. For the clinical trials, the latter is represented by *male* and *female*. However, it can have different representations for patient descriptions, for instance, *man*, *gentleman*, *boy*, or *M* in the case of a male. Therefore, these values were normalized to the values of the clinical trials.

For extracting the metadata from the patient descriptions was used the *re*[5] module from Python, which provides regular expression matching operations. The used method for searching takes a regular expression pattern, a patient description and searches for that pattern within the string. If the search is successful, search() returns a match object or None otherwise. In the cases where the age information or the gender information was not found, it was attributed age 0 and gender None. Figure 5.5 illustrates the two patterns used for extracting the metadata.

With these rules, it was possible to compare patients' demographics with the eligibility demographics of a clinical trial and understand if a patient is eligible for the trial. When one of these patients' metadata is incongruent with the trial, the patient is considered not eligible for the given trial. In this case, the score between the patient-trial pair is penalized by discounting some factor.

The process starts with a ranking list obtained from a chosen retrieval run, and the demographics are compared for each patient-trial pair in the list. Whenever a patient does not meet the trial's age and gender information, the pair's score is decremented by a value.

```
r"\d+ *(yo—year old—-year-old—year-old—y/o—-year old—year—-day-old—months old)?"
r"(( *(\w*—African-American) *),6(man—woman—female—gentleman—male—boy—girl—F—M))?"
```

Figure 5.5: Illustration of the regular expression patterns to extract metadata from patient descriptions.

[5] https://docs.python.org/3/library/re.html

5.6 Overview

This chapter described the techniques used to address the task of matching clinical trials to patient descriptions. First, it briefly presented the problem task as a reminder of the study and gave a high-level view of the experiments, their implementation steps, and the technologies used. Then, the necessary steps to implement the retrieval solutions and their implementation were addressed. Following, the training steps were described for the bi-encoder architecture, focusing on the required data configuration to train the model, the loss function, the negative sampling techniques, and the framework used. Finally, it addresses the implementation of a technique to penalize the score between topic-trial pairs based on the extracted metadata from the topics using regular expression rules [56].

6

Experimental Evaluation

Contents

In this chapter, the experimental setups and results will be discussed. It starts by reporting the results obtained for the task of retrieving clinical trials for patient descriptions following the TREC 2021 Clinical Trials Track scheme. These experiments include ranking with sparse and dense representations and some strategies for improving the results. Then, it addresses each experiment and discusses its results and challenges. The last section addresses the questions this research investigated and closes with an overview of the chapter.

6.1 Experimental Results

During the course of this work, several experiments were conducted to retrieve the most relevant clinical trials for patient descriptions using traditional and more recent approaches using deep learning and state-of-the-art architectures in NLP and information retrieval.

6.1.1 Sparse Retrieval

The first run was an experiment using a sparse retrieval model. The selected model was BM25 due to its simplicity and effectiveness, but also because it has been a standard in information retrieval research. This model was widely used in all the TREC Clinical Trials track's submissions explored to develop this study. In addition, some teams used BM25 as a term of comparison for more complex models that operate with dense contextualized representations.

Table 6.1 reports the obtained results from two experiments using BM25. The metrics used to evaluate each run's performance were described in Section 2.2.2.

In the two presented runs, the retrieval model was implemented using Pyserini's default implementation (hyperparameters k = 0.9 and b = 0.4) and the TREC-CT test collection.

Table 6.1: Results of the retrieval task with the TREC-CT test collection using BM25. $^{(5)}$ indicates that the combination of fields included five fields: brief title, official title, brief summary, detailed description, criteria.

Run	Model	Fields	TREC-CT				
			P@10	R-Prec	MRR	Recall	NDCG@10
(1)	BM25	entire	0.1053	0.0682	0.2441	0.2431	0.2018
(2)	BM25	combination$^{(5)}$	0.1640	0.0925	0.3128	0.2576	0.2917
(3)	BM25 + Regex Filter	combination$^{(5)}$	**0.1920**	**0.1023**	**0.3471**	**0.2576**	**0.3145**

Run (1)

In the first run, the clinical trials were indexed entirely using all the available fields. In the second run, the clinical trials were indexed with the following combination of fields: brief title, official title, brief summary, detailed description, and eligibility criteria. These free text fields contain relevant information about the study and its essential criteria. The results still have a lot to improve.

Run (2)

In the second run, results show an improvement in all the evaluated metrics. In both experiments, the fraction of relevant trials retrieved is very similar, with more than one point of difference in recall. However, there is a more significant difference when comparing the NDCG@10 for both runs. This difference could prove that selecting specific fields impacts the quality of the retrieved list and can be explained by the fact that many fields are noisy and thus negatively affect the retrieval. This results also supports that term-based retrieval models are predisposed to the vocabulary mismatch problem.

The second run could be compared to some of the TREC Clinical Trials track's submissions (Section 3.2.1), given that the results are not so different when compared to more default implementations [57] or more complex ones that applied query extraction techniques with BERT-based models [45].

This result is used as a term of comparison for the experiments with dense retrieval.

Run (3)

The third run experimented with regular expression rules to create a method for penalizing the score of patient-trial pairs in a ranking list whenever a patient meet exclusion criteria for the trial in the pair. In this case, the ranked list used was the one obtained from the second run (2).

The obtained results had improved gains between 1 to 3% depending on the evaluated metric. As expected the recall did not change since the elements in the list are the same, however the quality of the ranking list improved for the top 10 as shown by the improvements on the NDCG@10. Overall, the results obtained in this run are the best ones for the sparse retrieval experiments.

6.1.2 Dense Retrieval

Having experimented with a term-based model for ranking, the following runs focused on ranking with dense representations using two different BERT-based models pre-trained on a large datasets. The two models experimented were very different. As described previously, the first model is the msmarco-bert-base-dot-v5, a BERT-base model trained for semantic search on the MSMARCO dataset.

Table 6.2: Results of the retrieval task with the TREC-CT test collection using the experimented transformer-based models, pre-trained and fine-tuned. The best BM25 run is included for comparison. (1) indicates that training used random negative sampling with one sample. (6) indicates that the model trained was the fine-tuned model in run (6).

Run	Model	Data	Encoder	TREC-CT				
				P@10	R-Prec	MRR	Recall	NDCG@10
(2)	BM25	-	-	0.1640	0.0925	0.3128	0.2576	0.2917
	Pre-trained							
(4)	msmarco-bert-base-dot-v5	MSMARCO	Bi-Encoder	0.1347	0.0740	0.3117	0.2701	0.2343
(5)	all-distilroberta-v1	Multiple	Bi-Encoder	0.1973	0.1280	0.4501	0.4221	0.3220
	Fine-tuned							
(6)	all-distilroberta-v1	SIGIR-CT	Bi-Encoder	0.2293	0.1630	0.3971	0.5937	0.3389
(7)	all-distilroberta-v1(1)	SIGIR-CT	Bi-Encoder	0.1200	0.0854	0.2896	0.4322	0.1613
(8)	all-distilroberta-v1	TREC-PM	Bi-Encoder	0.2360	0.1777	0.4572	0.5995	0.3501
(9)	all-distilroberta-v1(6)	TREC-PM	Bi-Encoder	0.2720	0.1832	0.4459	0.5980	0.3672
	Regex Filter							
(10)	all-distilroberta-v1(6) + Regex Filter	TREC-PM	Bi-Encoder	**0.2973**	**0.2036**	**0.4573**	**0.5980**	**0.3858**

The other tested model is the all-distilroberta-v1, a distilled version of RoBERTa tuned for many use-cases in a large and diverse dataset.

Table 6.2 presents the various runs and the obtained results for the different scenarios tested. These results can be divided into two parts: the two experiments with the pre-trained models in a zero-shot scenario, where the models were trained in data that is not related to the domain in this study, and four other runs where the pre-trained models were fine-tuned. These four runs differ from each other in the data and techniques used for training. As a reminder, in all these runs, the clinical trials were encoded and indexed using the combination of fields used in the second run of the sparse experiments.

Pre-trained Models

Run (4)

The fourth run ranks the clinical trials for the patient descriptions using the model msmarco-bert-base-dot-v5. Comparing the evaluated metrics with the best run for the sparse retrieval, the results show a small increasing of almost one and a half points for recall, however, they show a decreasing in the other metrics. The most significant difference is in the NDCG@10, indicating that the quality of the ranked list is worst. Moreover, it is closer to the first run of the sparse experiments where all the trials fields were used, than to the second run.

This run ranks the clinical trials for the patient descriptions using the model msmarco-bert-base-dot-v5. When comparing the evaluated metrics with the best run for the sparse retrieval, the results show a slight increase of almost one and a half points for recall, however, they show a decrease for the other

metrics. The most significant difference is in the NDCG@10, indicating that the quality of the ranked list got worst. Moreover, it is closer to the first run of the sparse experiments, where all the trial fields were used, than to the second run. These results corroborate the fact that not all transformer-based models are perfect for a ready-to-use situation and that BM25 could be a strong baseline for retrieval. Although BM25 is a term-based model tested in a default scenario, it outperformed the semantic-based approach with dense representations produced by a transformer-based model pre-trained with a large dataset. In defense of dense retrieval, this first model was tested in a zero-shot scenario.

These intriguing results were the motto for fine-tuning pre-trained models on a small in-domain dataset and investigating how much it can improve the results. However, before that, it also motivated the will to test another pre-trained model and rank again in a zero-shot scenario to see if it could surpass the sparse experiments with BM25 and be a possible candidate for a first-stage retrieval.

Run (5)

Another pre-trained model was tested in the fifth run, the all-distilroberta-v1. From the available models in the Sentence Transformers library designed for general purposes, this model has the best average performance in producing embeddings and in semantic search. It also has a maximum sequence length of 512, which is superior to the other models.

The results show a significant overall improvement compared to the previous run (4). Moreover, some evaluated metrics show an increase of almost 10 points (e.g., NDCG@10) and others more than 10 points (e.g., MRR and recall). For recall, it almost doubled the values of the previous runs, making it the best candidate for a first-stage retrieval so far. This improvement in the results could be explained by the data used to train the all-distilroberta-v1. According to the Sentence Transformers authors, the model was trained with all their available training data (more than 1 billion training pairs). One important factor is that the training data includes sections from scientific medicine and biology papers. Also to support these results, the authors in [58] noticed a significant performance gain when using distillation in pre-training stage for a bi-encoder when compared to a model that does not use distillation.

Comparing this run with other runs submitted on the TREC track, these results outperformed some other teams' results. Teams that used more complex approaches, such as named entity recognition (NER), to produce semantically meaningful sentence embeddings [59].

Fine-tuned Models

Having tested with pre-trained models in a zero-shot scenario and improved the results over the sparse retrieval experiments, the following tests focused on improving results by fine-tuning the model used in the fifth run (5) on small in-domain datasets and with different batch strategies.

Run (6)

In the sixth run, the strategy was to fine-tune the all-distilroberta-v1 on the SIGIR-CT dataset. This dataset is very similar to the one used to evaluate the model (TREC-CT). In-batch negative sampling was used to increase the number of training examples.

Comparing its results with the previous run (5), the NDCG@10 improved very little. However, only recall had a significant improvement, with an increase of 17%.

Run (7)

In the seventh run, the model was fine-tuned using a simple random negative sampling technique together with in-batch negatives. The results were the worst for dense retrieval, with a significant decrease in all the evaluated metrics, but mainly the NDCG@10. The authors in [60] show that random negative sampling methods, such as random negatives and in-batch negatives, minimize the total pairwise errors but cannot effectively minimize the top-k pairwise errors. Moreover, these techniques allow difficult queries to dominate the training easily, resulting in a serious loss of top-ranking performance [60], which supports the decreased value for NDCG@10.

Run (8)

The eighth run, the applied strategy was very similar to the sixth run (6). These two runs differ on the dataset used to fine-tuned the model, where in this run the data used comes from the TREC-PM test collection. The idea of these experiment was to investigate how could a very different dataset affect training and consequently the ranking task

The results obtained for this run outperform all the other experiments. However, these are very similar to the results obtained in the sixth run (6), where the all-distilroberta-v1 model was fine-tuned on the SIGIR-CT dataste.

Run (9)

This experiment used the TREC-PM dataset for training as in the previous run, but instead of training the pre-trained version of the all-distilroberta-v1, it trained the fine-tuned model obtained in run (6).

The ranking result from using this fine-tuned model yields the best performance overall. However, the gains were minimal when compared to the previous run.

Run (10)

As in the third run (3), this run experimented with regular expression rules to extract metadata from the topics and create a method that penalizes relevance scores between topic-trial pairs in a ranking list whenever the extracted information from the topic does not satisfies the trial's eligibility demographics. For this run, the ranking list used was the one that led to the best results, the run (9).

Similar to the run (3), the obtained results had gains between 1 and 2% depending on the evaluated metric. Overall, the results obtained in this run are the best ones for the sparse and dense retrieval experiments.

6.2 General Discussion

What is the effectiveness of sparse retrieval when ranking clinical trials using the entire collection? The results presented in Table 6.1 show a decrease in performance when analysing the evaluated metrics and compared them to the second run with a small combination of fields. Moreover, the quality of the ranking list got worst as the NDCG@10 almost 10% lower. However, first-stage retrieval aims to recall potentially relevant documents as many as possible [52]. Thus, both options would have a very similar behaviour for a first-stage or recall ranking.

Compared to sparse models, how well do BERT-based pre-trained models perform when applied in a zero-shot scenario for retrieval? The obtained results prove that depending on the pre-trained model used, the performance could be worst (e.g., third run) or could outperform significantly more classical approaches with sparse models such as the one tested in this study, a default implementation of BM25.

It is important to consider both options and select according to some factors. For instance, using BM25 is less computationally expensive when compared to BERT-based models. In addition, BM25 would be a much better option than training BERT from scratch with a small dataset. However, many tools, e.g., HuggingFace, enable users to build, train and deploy ML models based on open-source code and technologies. Hence, one can easily find a pre-trained BERT-based model for any domain without worrying about training the model and the associated costs.

Another factor to be considered is the index size, as the performance of dense representations decreases quicker than sparse representations for increasing index sizes [61]. In addition, lets not forget that term-based models may suffer from the vocabulary mismatch problem and ignores term ordering information [52].

Does training a model on small in-domain data affect the effectiveness of the downstream task and how it performs compared to a zero-shot scenario with a pre-trained model? The experiments with fine-tuning show an improve in effectiveness compared to ranking with pre-trained models on data

not related to the domain in study. In fact, the best run comes from one of these experiments with a model fine-tuned on two very different in-domain datasets.

6.3 Overview

This chapter described the experimental setup and results. It started addressing the experiments using BM25. Then, it focused on the dense retrieval approaches and their results. The fine-tuned models were trained on the SIGIR-CT and TREC-PM datasets. Techniques to improve performance, such as random negative sampling, were also evaluated.

The results are summarized in two tables, one for results with sparse retrieval (Table 6.1), and the other for dense retrieval (Table 6.1). The best model was achieved by training the pre-trained model on the SIGIR-CT dataset and then training that model on the TREC-PM dataset. The experiment on training with one negative for each trial-topic pair, sampled from the relevance judgments, led to the worst results for the experiments conducted with BERT-based models, making this model the best candidate for a first-stage retrieval.

Using a strategy based on regular expressions to update the ranking list obtained form the best runs of the sparse and dense retrieval experiments led to an overall improvement in the results for these runs, however, these improvements do not make much difference when considering the first-stage retrieval model since they not changed the fraction of relevant trials in the ranking list.

7

Conclusions and Future Work

Contents

This dissertation presented a fine-tuning setup for ranking clinical trials to a given patient description. This chapter overviews the main contributions, and addresses directions for future work in this task.

The attention of this study was address to transformer-based models and investigate their ability to integrate first-stage retrieval and replace traditional approaches with term-based models. These models have a very important role in the whole effectiveness of a multi-stage pipeline, right from the beginning, since they are responsible for retrieving a ranking list of relevant documents to be re-ranked in the following stages of the pipeline. Thus, when first-stage retrieval lacks of performance, it affects the entire system.

In recent years, with the deep learning revolution brought the attention for dense representations in NLP and IR. Consequently, there is a surge of research interest in applying deep-learning techniques for first-stage retrieval. Having the opportunity to contribute to these research topic and investigating in a very important domain that is the recruitment of patients for clinical trials were some of the main motivations to dedicate to this study.

7.1 Contributions

Fine-Tuning Methods

A fine-tuning setup for bi-encoders, for the task of clinical trials document retrieval (i.e., retrieve relevant trial to patient descriptions) was proposed. The model fine-tuning setup focuses on batch construction through random negative sampling procedure, more specifically, in-batch negative sampling, and in one experiment another random sampling technique was used based on easy negatives from the relevance judgments. In addition, two very different datasets were used to investigate the effect of a model fine-tuned with small in-domain data and its effect in the retrieval task.

Results

Experiments showed that fine-tuning a model with in-domain data improved the results obtained in other experiments with a sparse model and with pre-trained dense models that were not related to the domain in study. From all the evaluated metrics, recall was the one that showed the best improvements. It would be interesting to compare recall with other experiments submitted to the TREC 2021 Clinical Trails track. Unfortunately, all the submissions found did not report this metric.

7.2 Future Work

An extension of this study could be using extraction techniques using transformer models to summarize the clinical trials and patient descriptions producing semantically meaningful sentence embeddings. This could help mitigating the vocabulary mismatch problem, but also resolve the problem of the maximum sequence limitation in the case transformer-based models. This way relevant information could fit the transformer models without losing information.

It would be relevant to test pre-trained models on more related domains or in the same domain if possible to compare their performance with the already experiments. In addition, these models could be fine-tuned on the datasets used in this study and once more compare their performance with previous experiments.

Finally, it would be interesting to experiment with hard negative sampling techniques and see much improvements they can bring, given that the experience with random negatives led to the worst results for dense retrieval.

Bibliography

[1] "Clinical trial delays: America's patient recruitment dilemma - Clinical Trials Arena," 2022. [Online]. Available: https://www.clinicaltrialsarena.com/marketdata/featureclinical-trial-patient-recruitment/

[2] "What is Artificial Intelligence (AI)?" 2022. [Online]. Available: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence

[3] A. K. Bishnoi, "Applications of Deep Learning and Machine Learning," *Journal of Emerging Technologies and Innovative Research*, vol. 5, no. 11, 2018.

[4] "What are Neural Networks?" 2021. [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks

[5] W. S. Mcculloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biology*, vol. 5, 1943.

[6] K. D. Foote, "A Brief History of Artificial Intelligence," 2022. [Online]. Available: https://www.dataversity.net/brief-history-artificial-intelligence/

[7] M. Kraus, S. Feuerriegel, and A. Oztekin, "Deep learning in business analytics and operations research: Models, applications and managerial implications," *European Journal of Operational Research*, vol. 281, no. 3, 2020.

[8] I. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Computer Science*, vol. 2, no. 420, 2021.

[9] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological review*, vol. 65, no. 6, 1958.

[10] R. M. e Silva de Oliveira, R. C. F. Araújo, F. J. B. Barros, A. P. Segundo, R. de Freitas Zampolo, W. da Silva Fonseca, V. A. Dmitriev, and F. de Souza Brasil, "A System Based on Artificial Neural Networks for Automatic Classification of Hydro-generator Stator Windings Partial Discharges," *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, vol. 16, no. 3, 2017.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back-propagating Errors,” *Nature*, vol. 323, no. 6088, 1986.

[12] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate.” ICLR, 2015.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” NAACL-HLT, 2019.

[15] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, “Transfer Learning in Natural Language Processing,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019.

[16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, [U+FFFD] Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation,” *CoRR*, vol. abs/1609.08144, 2016.

[17] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, “Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books.” IEEE International Conference on Computer Vision (ICCV), 2015.

[18] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng, “A Deep Look into Neural Ranking Models for Information Retrieval,” *CoRR*, vol. abs/1903.06902, 2019.

[19] J. Lin, R. Nogueira, and A. Yates, “Pretrained Transformers for Text Ranking: BERT and Beyond,” *CoRR*, vol. abs/2010.06467, 2020.

[20] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic, “Real life information retrieval: a study of user queries on the Web,” *ACM SIGIR Forum*, vol. 32, no. 1, 1998.

[21] F. Scholer, D. Kelly, and B. Carterette, “Information retrieval evaluation using test collections,” *Information Retrieval Journal*, vol. 19, no. 3, 2016.

[22] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, “Towards Unsupervised Dense Information Retrieval with Contrastive Learning,” *CoRR*, vol. abs/2112.09118, 2021.

[23] Y. Luan, J. Eisenstein, K. Toutanova, and M. Collins, “Sparse, Dense, and Attentional Representations for Text Retrieval,” *CoRR*, vol. abs/2005.00181, 2020.

[24] G. Penha and C. Hauff, “Sparse and Dense Approaches for the Full-rank Retrieval of Responses for Dialogues,” *ArXiv*, vol. abs/2204.10558, 2022.

[25] N. J. Belkin and W. B. Croft, “Information filtering and information retrieval: two sides of the same coin?” *Communications of the ACM*, vol. 35, no. 12, 1992.

[26] A. Spoerri, “InfoCrystal, a visual tool for information retrieval,” Ph.D. dissertation, Massachusetts Institute of Technology, 1995.

[27] S. Robertson and H. Zaragoza, “The Probabilistic Relevance Framework: BM25 and Beyond,” *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, 2009.

[28] C. Kamphuis, A. P. de Vries, L. Boytsov, and J. Lin, “Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants,” *Advances in Information Retrieval*, vol. 12036, 2020.

[29] B. Mitra and N. Craswell, “Neural Models for Information Retrieval,” *CoRR*, vol. abs/1705.01509, 2017.

[30] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, “The vocabulary problem in human-system communication,” *Communications of the ACM*, vol. 30, no. 11, 1987.

[31] L. Zhao and J. Callan, “Term necessity prediction,” *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010.

[32] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, “A Deep Relevance Matching Model for Ad-hoc Retrieval,” *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016.

[33] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[34] S. Humeau, K. Shuster, M.-A. Lachaux, and J. Weston, “Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring,” *CoRR*, vol. abs/1905.01969, 2019.

[35] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Y. Wu, S. Edunov, D. Chen, and W. tau Yih, “Dense Passage Retrieval for Open-Domain Question Answering,” *CoRR*, vol. abs/2004.04906, 2020.

[36] R. Nogueira and K. Cho, “Passage Re-ranking with BERT,” *CoRR*, vol. abs/1901.04085, 2019.

[37] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *CoRR*, vol. abs/1702.08734, 2017.

[38] A. Menon, S. Jayasumana, A. S. Rawat, S. Kim, S. Reddi, and S. Kumar, “In defense of dual-encoders for neural ranking,” in *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022, iSSN: 2640-3498.

[39] J. Choi, E. Jung, J. Suh, and W. Rhee, “Improving Bi-encoder Document Ranking Models with Two Rankers and Multi-teacher Distillation,” *CoRR*, vol. abs/2103.06523, 2021.

[40] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019.

[41] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” 2019.

[42] L. Ham, “Using Cross-Encoders as Reranker in Multistage Vector Search,” *Weaviate vector search engine*, 2022.

[43] N. Thakur, N. Reimers, J. Daxenberger, and I. Gurevych, “Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks,” *CoRR*, vol. abs/2010.08240, 2020.

[44] R. Nogueira, W. Yang, K. Cho, and J. Lin, “Multi-Stage Document Ranking with BERT,” *CoRR*, vol. abs/1910.14424, 2019.

[45] W. Kusa and Y. Ghafourian, “DOSSIER at TREC 2021 Clinical Trials Track.” Text REtrieval Conference, 2021.

[46] M. Neumann, D. King, I. Beltagy, and W. Ammar, “ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing,” vol. abs/1902.07669, 2019.

[47] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld, “SPECTER: Document-level Representation Learning using Citation-informed Transformers,” *CoRR*, vol. abs/2004.07180, 2020.

[48] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. B. A. McDermott, “Publicly Available Clinical BERT Embeddings,” 2019.

[49] M. Grootendorst, “KeyBERT: Minimal keyword extraction with BERT.” 2020, version Number: v0.3.0.

[50] B. Milosavljević, D. Boberić, and D. Surla, “Retrieval of bibliographic records using Apache Lucene,” *The Electronic Library*, vol. 28, no. 4, 2010.

[51] B. Koopman and G. Zuccon, “A Test Collection for Matching Patients to Clinical Trials,” *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016.

[52] J. Guo, Y. Cai, Y. Fan, F. Sun, R. Zhang, and X. Cheng, “Semantic Models for the First-stage Retrieval: A Comprehensive Review,” *ACM Transactions on Information Systems*, vol. 40, no. 4, 2022.

[53] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira, “Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations,” *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[54] W. Xiong, X. L. Li, S. Iyer, J. Du, P. Lewis, W. Y. Wang, Y. Mehdad, W.-t. Yih, S. Riedel, D. Kiela, and B. Oğuz, “Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval,” *CoRR*, vol. abs/2009.12756, 2020.

[55] J. Zhan, J. Mao, Y. Liu, M. Zhang, and S. Ma, “RepBERT: Contextualized Text Embeddings for First-Stage Retrieval,” *CoRR*, vol. abs/2006.15498, 2020.

[56] B. Cardoso, “TRIALMATCH: A Transformer Architecture to Match Patients to Clinical Trials,” Master’s thesis, NOVA University Lisbon, 2022.

[57] L. Biester, V. Joopudi, and B. Dandala, “IBM @ TREC Clinical Trials Track 2021.” Text REtrieval Conference, 2021.

[58] J. Lei, X. Chen, N. Zhang, M. Wang, M. Bansal, T. L. Berg, and L. Yu, “LoopITR: Combining Dual and Cross Encoder Architectures for Image-Text Retrieval,” *ArXiv*, vol. abs/2203.05465, 2022.

[59] J. Koontz, M. Oronoz, and A. Pérez, “TREC 2021 Clinical Trials Submission for Universidad del País Vasco.” Text REtrieval Conference, 2021.

[60] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, and S. Ma, “Optimizing Dense Retrieval Model Training with Hard Negatives,” *CoRR*, vol. abs/2104.08051, 2021.

[61] N. Reimers and I. Gurevych, “The Curse of Dense Low-Dimensional Information Retrieval for Large Index Sizes,” *CoRR*, vol. abs/2012.14210, 2020.