Geographical Question Answering Leveraging Neural Language Models for Passage Retrieval

João Miguel de Almeida Vares Coelho

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor(s): Prof. Bruno Emanuel da Graça Martins
Prof. João Miguel da Costa Magalhães

Examination Committee

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Prof. Pável Pereira Calado

November 2021

Acknowledgments

I would like to thank my supervisors, whose guidance was paramount for the realization of this dissertation. My colleagues with whom I had multiple discussions about this work, and my friends who where there for the much needed distractions. Finally, my family who have always supported me throughout this journey.

This research was partially supported through the Fundação para a Ciência e Tecnologia (FCT), namely through the project grant with reference PTDC/CCI-CIF/32607/2017 (MIMU), as well as through the INESC-ID and NOVA LINCS multi-annual funding from the PIDDAC programme, respectively with references UIDB/50021/2020 and UIDP/04516/2020. I also gratefully acknowledge the support of the NVIDIA Corporation, with the donation of the two Titan Xp GPUs used in the experiments.

Resumo

Esta tese de mestrado foca na tarefa de *passage retrieval*, que consiste em identificar as passagens mais relevantes de uma coleção de documentos que possam responder a uma dada questão. Desenvolvimentos recentes nesta tarefa utilizam redes neuronais profundas, mais especificamente métodos baseados em Transformadores, treinados em grandes coleções de dados como o MS-MARCO. Apesar do progresso obtido por estes métodos, poucos estudos focaram especificamente em questões geo-espaciais (i.e., questões sobre localizações, ou questões sobre informação específica de lugares).

Desta forma, este projeto focou no domínio geográfico, explorando o uso de modelos neuronais para recuperação de informação no contexto de questões geo-espaciais, utilizando um subconjunto das instâncias presentes na coleção MS-MARCO, cujas questões e passagens contêm entidades geográficas. O subconjunto foi caraterizado, e uma estratégia de *re-ranking* baseada na distância geográfica foi analisada, tendo sido depois utilizada para amostrar exemplos negativos difíceis para o treino dos modelos.

Modelos seguindo as arquiteturas *bi-encoder* e *cross-encoder* foram treinados utilizando um método de amostragem de negativos baseado na distância geográfica, considerando a intuição de que passagens negativas que contenham entidades espaciais menos distantes das que estão na questão vão, em princípio, ser mais desafiantes para o modelo. Técnicas como expansão de dados e destilação de conhecimento foram empregues no treino dos modelos mais eficientes, baseados em *bi-encoders*, de modo a melhorar os resultados.

As experiências mostram que os modelos treinados seguindo a estratégia descrita neste documento obtêm melhores resultados no subconjunto de perguntas geográficas do MS-MARCO, quando comparados com os modelos base.

Palavras-chave: Recuperação de Informação Geográfica, Resposta a Questões Geográficas, Recuperação de Passagens, Modelos de Linguagem Neuronais, Transformadores

Abstract

This M.Sc. thesis addresses the problem of passage retrieval for question answering systems, which concerns with identifying the top-ranked passages within a collection that may answer a given question. Recent developments on passage retrieval rely on deep neural networks, specifically on methods based on Transformer models, trained on large datasets like MS-MARCO. Despite significant progress in approaches for retrieving (or re-ranking) passages from a document collection according to their relevance to an input query, few studies have specifically looked at geo-spatial queries (i.e., where-questions directly concerning locations, and also questions covering other informational needs relating to places, their types, and affordances).

As such, this project focused on the geographical domain, exploring the use of neural retrieval models in the context of geo-spatial queries, using a subset of the MS-MARCO large benchmark dataset, with questions and passages containing place-names. The subset of data was characterized, and a re-ranking strategy based on geographic distance was analysed, which was further used to sample hard negatives (i.e., irrelevant passages) for model training.

Models following both bi-encoder and cross-encoder architectures were fine-tuned using a negative sampling procedure which leveraged the geographic entities within queries and passages, considering the intuition that negative passages containing spatio-temporal entities that are closer to the ones in the query should, in principle, be more challenging. Other techniques were employed for model training, such as data expansion and knowledge distillation.

Results show that the fine-tuning methods do indeed improve the results on the geographic test set, for both cross-encoders and bi-encoders, when compared to the base models.

Keywords: Geographic Information Retrieval, Geographical Question Answering, Passage Retrieval, Neural Language Models, Transformers

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Question Answering (QA), i.e. the process of computing valid answers to questions formulated in natural language, is gaining increased attention both in industry and academia. Passage retrieval is a crucial component of QA systems, concerning the identification of top-ranked passages containing the answer for a given question, from a target document collection. Recent studies in the area have proposed a variety of passage retrieval methods based on neural language models [Liu et al., 2020], using large datasets such as MS-MARCO [Campos et al., 2016] for model training and evaluation.

Geographic questions (e.g., questions relating to where a particular event has taken place) are often posed to QA systems, motivating the development of tailored approaches. Geographic questions are also featured prominently in the MS-MARCO dataset [Hamzei et al., 2019, 2021], although current models for passage retrieval are not explicitly designed to explore geo-spatial properties when deciding on the matches between questions and passages. Previous studies have addressed the inherent problems of Geographic Question Answering (GeoQA) [Mai et al., 2021] and Geographic Information Retrieval (GIR) [Purves et al., 2018]. For instance, the types of queries GeoQA systems aim to answer are very diverse, and different types of questions need data from different sources (e.g., instead of retrieving answers from document collections, several GeoQA studies have instead relied on structured knowledge bases describing geo-spatial information [Haas and Riezler, 2016, Lawrence and Riezler, 2016, Punjani et al., 2018]). Also, the vagueness of some geographic concepts increases the difficulty of properly answering the questions formulated by users. This dissertation focused mostly factoid questions for which passages can be extracted from a background collection, without considering other external knowledge bases.

The main objectives of this work were to extract the geographic contents of MS-MARCO, a large open-domain passage retrieval dataset, and to explore the usage of neural retrieval models in the context of geo-spatial queries, leveraging techniques that exploit the geographic properties of the queries and passages. A thorough search of the relevant literature showed that this is the first work to address document-based GeoQA at the scale of very large document collections, with the Transformer architecture.

1.1 Methodology

The first step towards the realization of this project was a detailed revision of related work, both on open-domain passage retrieval and the specific case of geographic information retrieval. This showed that while neural architectures such as cross-encoders and bi-encoders achieve state-of-the-art results on this task, few studies have specifically looked at geo-spatial queries.

This way, a subset of MS-MARCO (a large benchmark dataset), containing geo-spatial queries and passages, was extracted and characterized. The construction of this subset leveraged Mordecai, an open-source toponym resolution system to disambiguate place-names into geo-spatial coordinates. After analyzing the relation between the distance and relevancy, a re-ranking strategy based on the geographic distance between places mentioned in the queries, and places within the passages, was evaluated. The results obtained by this strategy led to the hypothesis that the geographic distances could play an important role when sampling negatives for model training.

As such, starting from baseline passage re-ranking models pre-trained on the full MS-MARCO dataset, and considering either bi-encoder or cross-encoder architectures with similar training setups, the models were fine-tuned using a geographically-aware negative sampling procedure. Specifically, the negative passages were selected from those that score highly in terms of BM25 and poorly in terms of geographic proximity (i.e., the passages that are lexically similar to the query, and at the same time geographically distant, are perhaps more challenging for the passage re-ranking models).

Besides model training with a standard cross-entropy loss, a knowledge distillation procedure was also explored, based on a differentiable approximation to Spearman's rank correlation coefficient [Blondel et al., 2020]. The idea was to try approximating the results of fine-tuned cross-encoders with computationally more efficient bi-encoders.

Also, as an attempt to further improve the results, a data expansion technique was used to generate more training examples. Given geographic passages from MS-MARCO that were not associated with any queries, a pre-trained T5 model was used to generate a query for them. With this procedure, nine thousand extra queries were created, to be used during model training.

After analysing the results of the previous experiments, a final bi-encoder model was fine-tuned, combining the aforementioned techniques into a single pipeline. A cross-encoder was used within the geographically-aware negative sampling procedure instead of BM25, so as to sample negative examples following semantic similarity. Those negatives were used for the training of the bi-encoder, where the same cross-encoder also acted as the teacher model for knowledge distillation.

As for the technologies used during this work, Python was used as the main programming language, due to its popularity among the Machine Learning community. More concretely, open-source libraries that deal with PyTorch models such as Huggingface Transformers [Wolf et al., 2020] and SentenceTransformers [Reimers and Gurevych, 2019] were used to fine-tune the models.

1.2 Results and Contributions

Overall, the results showed improvements over both types of baseline models (i.e., bi-encoders and cross-encoders), on the geographical domain, when considering fine-tuning with the proposed approaches. The main contributions of this work can be summarized as follows:

- The geographic subset of MS-MARCO was extracted and geoparsed.
- An expanded set of queries that are relevant to passages within MS-MARCO were generated and geoparsed.
- A fine-tuning strategy which improves both cross-encoders and bi-encoders for the geographic domain was proposed.
- A cross-architecture knowledge distillation method which further improved the bi-encoder models was addressed.

Evaluating on the geographic subset of MS-MARCO, the best fine-tuned bi-encoder achieved a Mean Reciprocal Rank at the 10th passage of 0.4454, and the best fine-tuned cross-encoder achieved 0.5103. This is an improvement over the baseline models, which achieved 0.4019 and 0.4959, respectively.

For reproduction purposes, the code that supports the experiments described in this manuscript, along with the geoparsed datasets (i.e., including the labels for geographic entities) and fine-tuned models, are publicly available in a GitHub repository[1]. Also, an article which covers part of the work here presented, entitled *Improving Neural Models for the Retrieval of Relevant Passages to Geographical Queries*, was accepted for presentation into the 29th International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2021).

1.3 Thesis Outline

The rest of this document is organized as follows: Chapter 2 presents the fundamental concepts necessary for the understanding of this work. Chapter 3 covers related work on passage retrieval and geographic question answering. Chapter 4 presents the data collection that was used, and the methods for extracting its geographic subset. Chapter 5 introduces the retrieval models and the procedures involved in their fine-tuning. Chapter 6 presents the experimental evaluation. Finally, Chapter 7 provides concluding remarks, and discusses directions for future work.

[1]`https://github.com/JMVCoelho/geo-passage-retrieval`

Chapter 2

Fundamental Concepts

This chapter will cover topics necessary for the understanding of this work, starting with textual information representation, followed by deep learning models, neural language models such as BERT, and ranking methods.

2.1 Representing Textual Information

To train Machine Learning models for Natural Language Processing (NLP) tasks, there is a need to represent text as numerical vectors. The Vector Space Model [Salton et al., 1975] is an approach in which words or documents are mapped to a V-dimensional space, where V is the number of tokens in the vocabulary. In an one-hot word encoding style, each dimension would be assigned the value 0, except for the one corresponding to the word being represented, which would be assigned 1. A similar approach can be used to represent documents. For a collection of documents, $D = \{d_1, d_2, ...\}$, the vocabulary V is the set of unique tokens among all documents. Therefore, each document in the collection will be represented by a vector, $d_i = (t_1, t_2, ..., t_{|V|})$. For each document d_i, $t_v = 0$ if $t_v \notin d_i$ or $t_v = 1$ if $t_v \in d_i$.

Instead of having binary values, weights can be assigned to each term. This way, documents are represented by $d_i = (t_1, t_2, ..., t_{|V|})$, but $t_v = w_{v,i}$. One way of computing the weights is by using TF-IDF:

$$w_{v,i} = \mathrm{tf}(v, i) \times \mathrm{idf}(v) \ , \tag{2.1}$$

where $\mathrm{tf}(v, i)$ is the term frequency of t_v on document d_i, and $\mathrm{idf}(v)$ is the inverse document frequency of term t_v, given by:

$$\mathrm{idf}(v) = \log \frac{|D|}{|\{d_{i'} \in D | t_v \in d_{i'}\}|} \ . \tag{2.2}$$

The normalized frequency, $\mathrm{ntf}(v, i)$, can be used instead of $\mathrm{tf}(v, i)$, by considering the maximum frequency of all terms in d_i. It can be computed as follows:

$$\mathrm{ntf}(v, i) = \frac{\mathrm{tf}(v, i)}{\max_{v' \in V} \mathrm{tf}(v', i)} \ . \tag{2.3}$$

The motivation for this metric is that terms that appear very often in a document are important (except for stop-words), but terms that appear in the document and are simultaneously rare among the collection of documents are, in principle, context-related, and should also be more important.

To compute the similarity between two representations, r_1 and r_2, the cosine similarity can be used:

$$\mathrm{sim}_{\cos}(r_1, r_2) = \frac{r_1 \cdot r_2}{||r_1||\;||r_2||} \; . \tag{2.4}$$

It is important to note that the previous methods result in sparse representations, given the high number of words in the vocabulary. Also, all words are considered to be independent. This means that the cosine similarity between two different word representations is always 0, even if they are slightly related. As an alternative, word embeddings can be considered [Smith, 2020]. The objective is to represent words in dense, lower-dimension vectors. Documents can, for instance, be represented by averaging the embeddings of constituting words.

Mikolov et al. [2013a] proposed Word2Vec, a method to generate word embeddings through a simple neural network. The base idea is to train a neural network with a language modelling task and use the learnt weights from the hidden layer as the embeddings. There are two approaches.

The first, CBOW [Mikolov et al., 2013a], tries to predict a word given its its surrounding neighbours (i.e., the word's context) as input. Formally, given the sequence $t_1, t_2, ..., t_N$ and the context size c, the objective of the CBOW approach is to maximize:

$$\frac{1}{N}\sum_{n=1}^{N} \log(\mathrm{P}(t_n | t_{n-c}, ..., t_{n-1}, t_{n+1}, ..., t_{n+c})) \; . \tag{2.5}$$

The second approach, named Skip-gram [Mikolov et al., 2013b], uses the target word as input, with the optimization task being to try to find the word's context. Formally, the objective of the Skip-gram approach is to maximize:

$$\frac{1}{N}\sum_{n=1}^{N} \sum_{-c<=j<=c, j\neq 0} \log(\mathrm{P}(t_{n+j} | t_n)) \; . \tag{2.6}$$

The probability of predicting the output word t_O given t_I can be computed using the softmax function as follows:

$$\mathrm{P}(t_O | t_I) = \frac{\exp\left(r_{t_O}^{\top} r_{t_I}\right)}{\sum_{i=1}^{|V|} \exp\left(r_{t_i}^{\top} r_{t_I}\right)} \; , \tag{2.7}$$

where r_{t_x} is the representation of t_x and $|V|$ is the vocabulary size. Given that $|V|$ can be very large, other methods have been used to compute this probability, such as hierarchical softmax or negative sampling [Mikolov et al., 2013a,b].

The major limitation of word embeddings is that words with multiple meanings will have only one representation. To address this, contextual embeddings can be used, which are context-dependant representations that capture the use of words across multiple scenarios [Liu et al., 2020]. Unlike traditional word embedding techniques, contextual embeddings learn a vector that is a function of an input sequence that contains the target token. As a result, the same word may have different representations if used in different sentences. To further explain how these can be generated, Section 2.2.2 will introduce the Transformer architecture and Section 2.3 will cover the nowadays ubiquitous BERT model.

2.2 Deep Learning

Deep Learning is a subset of Machine Learning, concerned with the use of methods based on large neural networks, leveraging the substantial amounts of available data. This section starts by describing base architectures, namely the Perceptron and the Multi Layer Perceptron. Then, the Transformer is introduced as a model that is better suited to deal with NLP tasks.

2.2.1 Multi Layer Perceptron

An artificial neuron is a mathematical function which conceptualizes a simplified brain cell. Its inputs are weighted separately and fed to a nonlinear function to produce the output. The Perceptron [Rosenblatt, 1958] is an algorithm for supervised learning of binary classifiers, leveraging the aforementioned idea and consisting of a single artificial neuron. Given an n-dimensional input vector $x \in \mathbb{R}^n$, the output of the Perceptron, y, is computed as follows:

$$y = \mathrm{h}(w^\top x + b) \ , \tag{2.8}$$

where $w \in \mathbb{R}^n$ is the learnable weight vector and b is the bias. Originally, the activation function h is the signal function, which returns 1 if the input is positive or -1 otherwise. To train the Perceptron, w can be initialized randomly and updated following the equation:

$$w = w + \mu(\hat{y} - y)x \ , \tag{2.9}$$

where $\hat{y}$ is the true label for the input vector being processed and μ is the learning rate. The main issue with this model is that it can only classify linearly separable sets of vectors. However, that problem can be solved by extending the Perceptron, connecting multiple layers of neurons with non-linear activation functions.

Following the previous idea, a Multi Layer Perceptron, often also referred to as a feed-forward network, comprises one input layer, at least one hidden layer, and one output layer. Each layer l contains one or more neurons, $n^{[l]}$. For a single hidden layer, let $x^{[0]} \in \mathbb{R}^{n^{[0]}}$ be the input vector. The output, y, is given by:

$$y = \mathrm{h}^{[2]}(W^{[2]}\mathrm{h}^{[1]}(W^{[1]}x^{[0]} + b^{[1]}) + b^{[2]}) \ , \tag{2.10}$$

where $\mathrm{h}^{[i]}$,$W^{[i]} \in \mathbb{R}^{n^{[i-1]} \times n^{[i]}}$, and $b^{[i]} \in \mathbb{R}^{n^{[i]}}$ are the activation function, weight matrix and bias vector of the i^{th} layer, respectively. Error functions (e.g., mean squared error or the cross-entropy) are used to compare the expected value and the network output for a given input, and model training consists in minimizing one such error (also named cost or loss) function. Gradient Descent is an optimization technique that in order to minimize a given function, updates parameters by taking steps in the direction of steepest descent in the gradient of the error function:

$$\theta_{t+1} = \theta_t - \mu \nabla E_{\theta_t} \ . \tag{2.11}$$

In the previous expression, t denotes the time-step, E is the error function subject to minimization with respect to parameters θ, and μ is the learning rate.

Variations of gradient descent are applied alongside the back-propagation algorithm to train neural networks such as Multi Layer Perceptrons. After a forward pass, this procedure computes the partial derivatives of the cost function with respect to the different parameters, propagating this information back through the network, resorting to the chain rule to compute the nested derivatives.

Beyond standard gradient descent, one popular first-order gradient-based optimizer is Adam [Kingma and Ba, 2015]. Together with adaptive learning rates for each parameter, this method stores an exponentially decaying average of past squared gradients (i.e., an estimate of the second raw moment of the gradient), v_t, and an exponentially decaying average of past gradients (i.e., an estimate of the first moment of the gradient), m_t. At each time-step t the gradient $g_t = \nabla E_{\theta_t}$ is computed, updating the averages:

$$m_t = \frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} g_i}{1-\beta_1^t} \ , \tag{2.12}$$

$$v_t = \sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} g_i^2}{1-\beta_2^t}} \ , \tag{2.13}$$

and the parameter update rule is given by:

$$\theta_{t+1} = \theta_t - \frac{\mu m_t}{v_t + \epsilon} \ , \tag{2.14}$$

where β_1 and β_2 are hyperparameters to account for how much of the previous gradients should be taken in consideration when updating m_t and v_t. Overall, this approach yields small steps for parameters that are updated very frequently, with larger ones for the others.

2.2.2 Transformer

Proposed by Vaswani et al. [2017], the Transformer neural network architecture follows an encoder-decoder approach to represent inputs and generate outputs, aiming to solve sequence to sequence tasks. The model is particularly suited to deal with NLP tasks, given that these involve processing sequences of word tokens. Figure 2.1 depicts the overall structure. The encoder component is a stack of encoders, and the decoder is a stack of decoders. Each encoder comprises an attention layer and a feed-forward layer. Each decoder has the same layers, but between them there is an encoder-decoder attention layer.

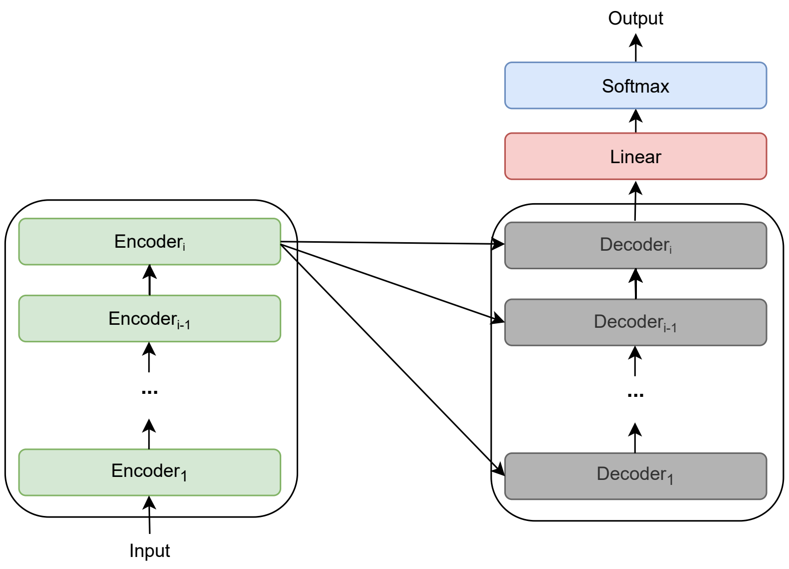


Figure 2.1: General architecture of the Transformer.

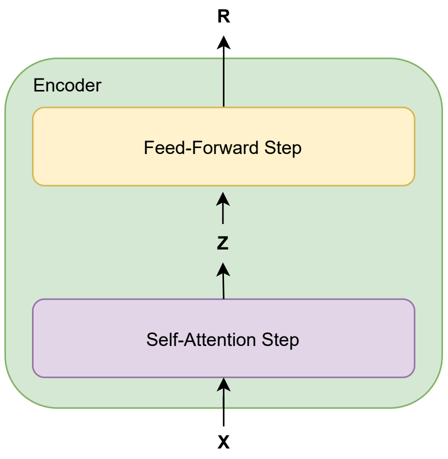


Figure 2.2: Components of a Transformer encoder.

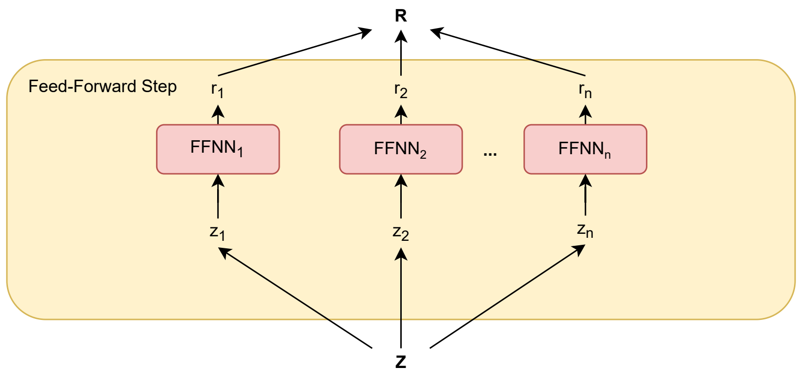
First, the input sentence is processed, where each input token is mapped to an embedding space. Since the tokens flow through the encoder (and decoder) simultaneously, there is no notion of word order. As such, a positional encoding mechanism is used, adding a vector to each token representation. The vectors follow a pattern that serves to approximate the position of each word in the sentence, relying on the cyclic nature of the sine and cosine functions.

After processing, each input token is associated with a vector, x_n. Stacking the n vectors yields a matrix, X, which is the input for the first encoder. The matrix goes through an attention step, and its output is fed to a feed-forward layer, as shown in Figure 2.2.

The objective of the attention step is to help the encoding process by allowing the model to check other words of the sentence. This allows the model to understand which words are relevant to the one that is being processed. One possible approach to achieve this is the scaled dot-product attention:

$$Z = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \,. \tag{2.15}$$

Figure 2.3: Feed-forward step of a Transformer encoder.



In the previous expression, Q, K and V are matrices obtained by multiplying X by three weight-matrices trained during the training process, W^{Q}, W^{K}, and W^{V}. As X is a stack of input vectors, Q, K, and V are stacks of query, key, and value vectors, respectively. The square root of the key vectors dimensionality, d_k, is used as a normalizing factor for more stable gradients during training. Ultimately, after the attention step, every input vector x_n is associated with a vector z_n.

In initial tests, the authors noticed that sometimes each z_n, despite containing information of every other encoding, could be biased towards x_n. To solve this, they proposed a method where the attention layer has multiple attention heads, termed *multi-headed attention*. This means that there will be multiple sets of W^{Q}, W^{K}, and W^{V} matrices. Each set is trained separately and thus, in the end of the training process, each set will map the input to a different subspace. The computations are done using the scaled dot-product attention, once for each attention head. The original Transformer uses 8 attention heads, and thus after computing attention separately, there will be eight different Z matrices. To condense them into a single one, they are concatenated and then multiplied by another weight matrix, W^{O}, which is also trained alongside the model. This way, the final Z matrix captures information from all attention heads.

The feed-forward step (Figure 2.3) comprises one feed-forward neural network which is applied independently to each z_n. The input to the next encoder is the output from the feed-forward step.

For the decoding step, each decoder has an attention layer, followed by an encoder-decoder attention layer and a feed-forward layer. The feed-forward layer works the same way as it does in the encoders, but the attention layer has some differences, as it can only check earlier positions in the output sequence. After the encoders process the input sequence, the output of the last encoder will be transformed into a set of key and value vectors. The encoder-decoder attention layer works similarly to the multi-headed attention, except that it creates the query vectors from the layer bellow it, whereas the key and value vectors are the ones yielded by the transformation on the last encoder's output. After the first decoding step, the output of each step is fed to the first decoder in the next step. This process stops when a symbol indicating completion is reached.

The output for each decoding step is fed to a fully connected neural network, mapping to a vector with higher dimension – a logits vector. The dimensionality of the logits vector is the size of the vocabulary. Each value on the vector is a score to a word on the vocabulary, thus softmaxing the vector yields a probability value for each word. This way, the word with the higher probability is that step's output.

2.3 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a neural language model that achieved state-of-the-art performance on multiple tasks when it was published [Devlin et al., 2019], also having been made publicly available. The available models were trained on books and Wikipedia data on a semi-supervised fashion.

Recall the Transformer, which is comprised by an encoder stack and decoder stack of the same size. BERT consists of an encoder stack, trained on language modeling objectives. BERT$_{\text{LARGE}}$ has 24 encoder layers, while BERT$_{\text{BASE}}$ has 12. Each encoder has an attention step and a feed-forward step.

As for input, the model can process single sentences or pairs of sentences, for a maximum of 512 tokens. For single sentences, a special token [CLS] is prepended to the sentence. Each token is then represented by a regular word embedding. Like the Transformer, positional embeddings are applied. For pairs of sentences, the [CLS] token is also prepended and a [SEP] token is used to separate the sentences. Besides the word and positional embeddings, a segment embedding is added, which is a vector that is learnt to indicate whether a sentence is part of the first or the second input.

To train the encoder stack, the authors proposed two tasks. The first, shown in Figure 2.4, was to find the right word for some masked position of a sentence. As such, they employed a masked language modeling objective, where 15% of the input is masked. Besides, some words are randomly replaced by another, and the model is asked to find the right one for the position. The output associated with the masked/replaced words is fed to a feed-forward network and then softmaxed, yielding probabilities for words over the vocabulary that are used to predict the missing word. The second objective, depicted in Figure 2.5, was to detect whether or not two sentences follow one another. The [SEP] token is used to separate the sentences, and the output associated with the [CLS] token is fed to a classifier. While training the classifier, BERT's layers can either be frozen, or their weights can be tuned alongside the classifier.

For this project, language models such as BERT will be used to generate a representation for either a single-input sentence, or for the concatenation of two sentences with the [SEP] token. To obtain this representations, techniques such as mean pooling (i.e., consider the average of the work token embeddings) can be used, or the original idea of taking the vector associated with the [CLS] token as a representation for the whole sequence can be considered.

Instead of fine-tuning BERT for a specific task, word-level embeddings can also be created from it. From the Transformer architecture, each output cell is associated with an input token. Instead of taking the [CLS] vector to represent the whole sentence, the vector associated with a specific word token can be used as an embedding. Given the attention mechanisms, the representations take context into account, hence being referred to as contextual embeddings.

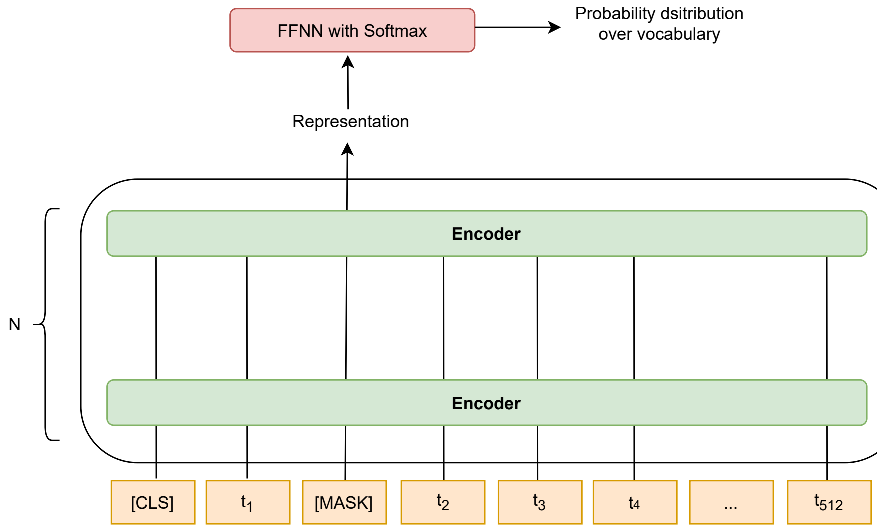


Figure 2.4: Training BERT with a masked language modelling objective.

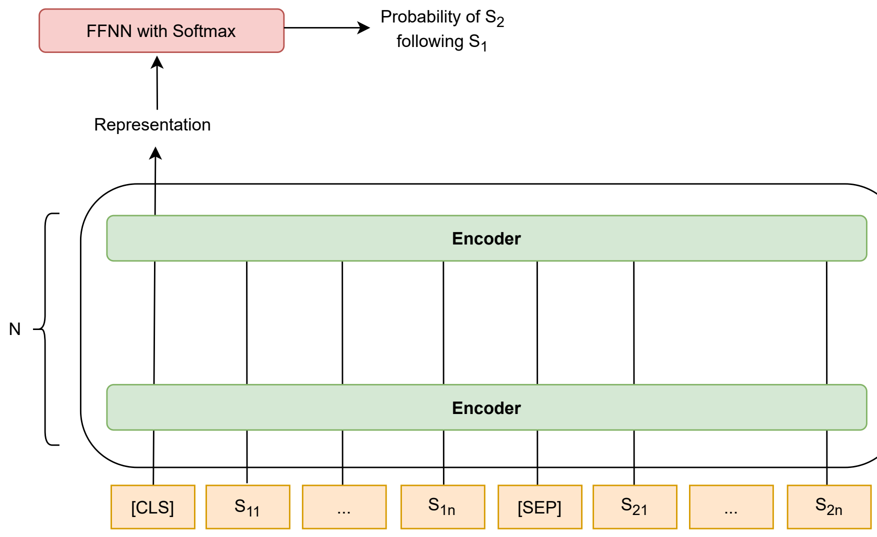


Figure 2.5: Training BERT with a next sentence prediction task.

2.4 Ranking for Information Retrieval

The objective of passage retrieval is to identify the top-ranked passages from a background collection that may answer a given question. Hence, this section will describe ranking functions such as BM25, its extensions, and general ideas related to supervised learning to rank.

2.4.1 BM25

BM25 is a ranking function used to estimate the relevance of documents for a given query, based on the probabilistic relevance framework [Robertson and Zaragoza, 2009]. Given a document d and a query q with n tokens, $(q_1, ..., q_n)$, the relevance score is computed as follows:

$$\mathrm{score}_{\mathrm{BM25}}(q,d) = \sum_{i=1}^{n} \mathrm{idf}(i) \times \frac{\mathrm{tf}(i,d) \times (k_1+1)}{\mathrm{tf}(i,d) + k_1 \times \left(1 - b + b \times \frac{|d|}{\mathrm{avgdl}}\right)} \;, \tag{2.16}$$

where b and k_1 are hyperparameters for tuning, and avgld is the average length of documents in the collection. Section 2.1 already presents how to compute the values for term frequency (tf) and inverse document frequency (idf).

Since textual data may be composed of several fields (e.g., documents from news collections may be divided in headline and body), BM25F [Zaragoza et al., 2004] was proposed as an extension. The authors argue that given the different length of the fields, computing the BM25 scores for each field and combining them linearly raises problems, like not keeping the non-linear relationship between term weights and term frequencies, difficulties in performing the field length normalization, and unstable idf computation. To address these issues, the authors proposed an approach that weights term frequencies accordingly to their field importance. Given the decomposition of a document in a set of fields, F, the scoring method is given by:

$$\widetilde{\mathrm{tf}}(i,d) = \sum_{z \in F} w^z \times \frac{\mathrm{tf}^z(i,d)(k_1+1)}{\mathrm{tf}^z(i,d) + k_1 \times \left(1 - b^z + b^z \frac{|d^z|}{\mathrm{avgdl}^z}\right)} \;, \tag{2.17}$$

where w^z is the weight associated with field z. The remaining terms with superscript z maintain their previous definition, now over the field z instead of the full document. The field-level term frequencies are then used to compute the ranking score:

$$\mathrm{score}_{\mathrm{BM25F}}(q,d) = \sum_{i=1}^{n} \mathrm{idf}(i) \times \widetilde{\mathrm{tf}}(i,d) \;. \tag{2.18}$$

BM25+ [Lv and Zhai, 2011] is another extension, addressing the fact that the term-frequency normalization by document length is not lower-bounded properly, which may over-penalize large documents. The alternative scoring method is given by:

$$\mathrm{score}_{\mathrm{BM25+}}(q,d) = \sum_{i=1}^{n} \mathrm{idf}(i) \times \left(\frac{\mathrm{tf}(i,d) \times (k_1+1)}{\mathrm{tf}(i,d) + k_1 \times \left(1 - b + b \times \frac{|d|}{\mathrm{avgdl}}\right)} + \delta \right) \;. \tag{2.19}$$

The only difference when compared to the original BM25 is the addition of an hyperparameter δ.

2.4.2 Learning To Rank

Learning to Rank (L2R) is a group of techniques that use Machine Learning to build ranking models, e.g. for information retrieval applications. Given a set of n documents, D, and a query q, the goal is to find out which $d \in D$ are relevant to the query, and sort them by their relevance.

Pointwise Learning to Rank formulates the problem as a classification or regression task. Let $\hat{\mathrm{y}}(q,d_i)$ be the graded relevance of d_i with respect to query q and f the learnt function. The objective is to minimize loss functions such as the Mean Squared Error:

$$L_{\text{mse}} = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{f}(q,d_i) - \hat{\mathrm{y}}(q,d_i))^2 \ . \tag{2.20}$$

Another approach, Pairwise Learning to Rank, takes pairs of documents as input, learning whether or not one is more relevant than the other. For example, RankSVM [Joachims, 2002] is a ranking algorithm based on Support Vector Machines, minimizing functions such as the Hinge Loss:

$$L_{\text{hinge}} = \sum_{\hat{\mathrm{y}}(q,d_i)>\hat{\mathrm{y}}(q,d_j)} \max\left(0, 1 - (\mathrm{f}(q,d_i) - \mathrm{f}(q,d_j))\right) \ . \tag{2.21}$$

One last technique is Listwise Learning to Rank, where the objective is to directly optimize ranking metrics such as the Mean Average Precision or the Normalized Discounted Commutative Gain. For instance, once the ranking function f is learnt, let there be a list of documents ordered by it for a given query. The Mean Average Precision (MAP) is computed as follows:

$$\mathrm{MAP} = \frac{\sum_{q\in Q}\mathrm{AP}(q)}{|Q|} \ , \tag{2.22}$$

$$\mathrm{AP(q)} = \frac{\sum_{i=1}^{n}\mathrm{P@}i \times \mathrm{R}(i)}{n_q^*} \ , \tag{2.23}$$

where Q is the set of all queries, n_q^* is the number of ground-truth relevant documents for query q and $\mathrm{R}(i)$ is a function that returns 1 if the document at rank i is relevant to q and 0 otherwise. P@i is the precision at the i^{th} returned document, i.e., the number of relevant documents on the first i results divided by i. Functions such as the MAP are not differentiable, since they depend on the ranked position of the documents. As such, gradient based optimizers cannot be used directly. Multiple methods have been proposed to solve this, under the setting of listwise ranking. For instance, RankGP [Yeh et al., 2012] aims to optimize the MAP resorting to genetic algorithms instead of gradient based optimizers, by using MAP as the fitness function.

2.5 Overview

This chapter reviewed fundamental concepts for the understanding of this work, namely the representation of textual information, starting with the classic approaches and motivating the most recent contextual embeddings. Deep learning was also addressed, by introducing the Multi Layer Perceptron, and the Transformer, an architecture which is more suitable to deal with textual data. Transformer-based neural language models, such as BERT, were also discussed. Finally, ranking functions such as BM25 and general ideas related to supervised lerning to rank were covered.

Chapter 3

Related Work

This chapter will explore previous work on passage retrieval in general, followed by an analysis on specific methods to deal with questions including geographical entities.

3.1 Passage Retrieval

First stage ranking and passage re-ranking are the two main tasks for passage retrieval. The former aims to select the top-k most relevant passages from the whole collection given a query, whereas the latter re-ranks the top-k passages for a query. Recently, Lin et al. [2020] released a survey which covers recent methods for text ranking with neural approaches. As a note, some of the methods to be described were originally proposed for document retrieval, instead of passage retrieval. The main difference relies on the length of documents, which are larger than passages. For representation simplicity, all method explanations will be adapted for passage retrieval.

3.1.1 Ranking with Cross-Encoders

Cross-encoders are encoder stacks that receive concatenated pairs of sentences as input (Figure 3.1). Multiple approaches have been proposed to re-rank passages using cross-encoders, based on pre-trained language models. For instance, Nogueira and Cho [2019] used BERT as a re-ranker, based on this idea. Their objective was, given a query q and a passage p, predict a relevance score, $\mathrm{score}(q,p)$. To achieve this, the query and passage are concatenated, and the relevant tokens are added ([CLS] and [SEP]). The representation of the [CLS] token is fed to a single layer neural network with the cross-entropy loss:

$$L_{\text{crossentropy}} = -\sum_{p\in P_q^+} \log(\mathrm{score}(q,p)) - \sum_{p\in P_q{}^-} \log(1-\mathrm{score}(q,p)) \ , \tag{3.1}$$

where P_q^+ and P_q^- are the sets of positive and negative passages for query q, respectively, provided within the same training batch.

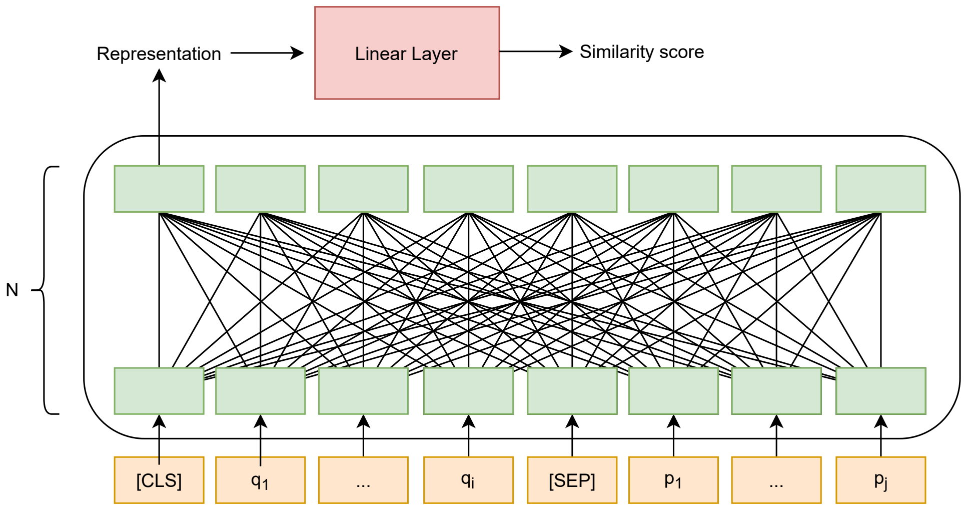


Figure 3.1: Cross-encoder as a stack of N encoders.

Han et al. [2020] followed a similar approach, but used TF-Ranking [Pasumarthi et al., 2019] models instead of the single layer neural network. They tested with point-wise, pair-wise and list-wise learning to rank models, on top of the [CLS] representation, as available within the TF-Ranking library. Also, besides BERT, two more language models were used. The objective of these models is to try to reduce BERT's number of parameters.

RoBERTa [Liu et al., 2019] uses BERT's pre-training masked language model, adopting a dynamic masking strategy which generates a different masking patterns for every sequence. The next-sentence prediction objective was also removed. Moreover, the authors argued that BERT is undertrained and, as such, RoBERTa is trained on a larger dataset and during more time when compared to BERT.

ELECTRA [Clark et al., 2020] adopts instead a slightly different pre-training model that aims to detect replaced tokens in the input sequence. One Transformer (generator) will receive as input a masked sentence and predict the original tokens for the masked ones. A second Transformer (discriminator) will receive the sentence with predictions, and infer whether each token is original or was replaced by the generator.

Scores obtained by cross-encoders using the different language models, i.e. BERT, RoBERTa and ELECTRA, were ensembled on 5 runs each with a list-wise loss. For each run $i \in (1, 2, ..., k)$ and query q, the passages were ranked based on the prediction scores, obtaining the position $\mathrm{Pos}(i, q, p)$ for each passage p. Then, a new score was calculated based on the average reciprocal rank of k runs:

$$\mathrm{score}_{\mathrm{emsemble}}(q, p) = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{\mathrm{Pos}(i, q, p)} \ . \tag{3.2}$$

3.1.2 Ranking with Bi-Encoders

In contrast with cross-encoders, bi-encoders generate two separate representations for a pair of sentences (Figure 3.2). While models based on them can be used for re-ranking, they are mostly used for first stage retrieval, given that the individual representations can be indexed through methods

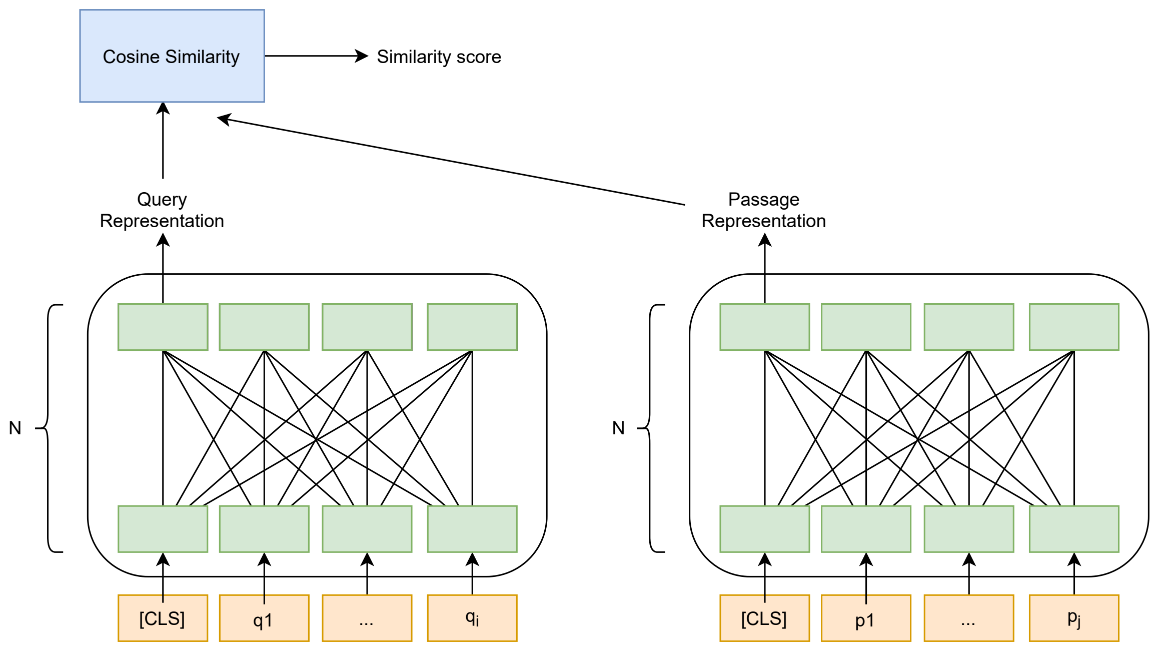


Figure 3.2: Bi-encoder as two stacks of N encoders.

supporting the fast execution of maximum inner product searches, such as FAISS [Johnson et al., 2017].

Reimers and Gurevych [2019] argue that passing concatenations of sentences through large networks is computationally expensive, which makes it unsuitable for semantic similarity tasks over large collections. As such, they proposed SBERT, which leverages bi-encoders to generate representations for queries and passages independently. This way, all passage representations can be pre-computed. The architecture comprises two BERT encoders (with shared weights, as the task in hand is semantic similarity) followed by a pooling layer. Three pooling strategies were tested: Using the [CLS] token representation, Mean Pooling and Max Pooling of the word token embeddings. The objective function depends on the task. For classification, the representations for query, E_q, passage, E_p, and their element-wise difference, $|E_q - E_p|$, are concatenated. This representation is fed to a softmax layer, trained using the cross-entropy loss.

For regression, the cosine similarity between E_q and E_p is computed, using the mean-squared error loss as the objective. A triplet-based objective was also considered, using the triplet-loss with a hinge formulation:

$$L_{triplet} = \max\{0, \mathrm{sim}(q, p^-) - \mathrm{sim}(q, p^+) + \epsilon\} \tag{3.3}$$

where sim is a similarity function, and the objective is to score the positive passage at least ϵ higher than the negative one. The authors choice between the three structures depended on the data and the task, which means that no direct comparison was made between the three.

ColBERT [Khattab and Zaharia, 2020] follows a similar structure, where queries and passages are independently encoded using BERT. The representation for a given query q is represented by E_q. It contains a vector E_{q_i} for each token q_i in q. The same applies for passages, E_p. The query encoder starts by preprending (after [CLS]) a special token [Q], and a query augmentation mechanism is applied by appending [MASK] tokens until the query reaches the maximum number of tokens, as the authors argue that this will allow BERT to produce query-based embeddings at the positions of the masks. BERT is then used to encode the tokenized version of the query, yielding the representations for each token, which are fed to a linear layer to reduce the dimensions. Finally, the vectors are normalized so that the dot product can be used as cosine similarity. Encoding passages works similarly, prepending a [D] token instead of [Q] and filtering out tokens corresponding to punctuation.

This model also proposes a computationally efficient, yet powerful, late interaction mechanism to compute the similarity between a query and a passage, as a sum of maximum similarity (MaxSim) scores:

$$\mathrm{sim}_{colbert}(q,p) = \sum_{i \in |E_q|} \max_{j \in |E_p|} E_{q_i} E_{p_j}^{\top} \ . \tag{3.4}$$

This operation conveys a pruning-friendly behaviour, allowing for top-k retrieval by using fast vector similarity data structures, and applying MaxSim between the query embedding and all passage embeddings across the full collection.

The top-k retrieval process is divided in two stages. First, given a query representation E_q, k' passages will be retrieved for each token in E_q (i.e., queries are scored against the individual token representations), producing $|E_q| \times k'$ results. Then, from those, the unique ones will be re-ranked, exhaustively scoring each passage with respect to the whole query.

During training, the BERT encoders are fine-tuned, while the linear layer's parameters and [Q]/[D] markers embeddings are trained from scratch. Using triplets (q, p^+, p^-), ColBERT produces a score for each passage individually, with pairwise softmax cross-entropy as the loss function over the scores of p^+ and p^-:

$$L = -\log\left(\frac{\exp\left(\mathrm{sim}_{colbert}(q,p^+)\right)}{\exp\left(\mathrm{sim}_{colbert}(q,p^+)\right) + \exp\left(\mathrm{sim}_{colbert}(q,p^-)\right)}\right) \ . \tag{3.5}$$

Wrzalik and Krechel [2020] proposed CoRT, to serve as a complementary ranker for term-based retrievers such as BM25. The queries and passages are encoded using ALBERT [Lan et al., 2020], which is a language model that aims to reduce BERT's memory complexity by using the same weights in all the encoder layers. The representations are then mapped to the desired dimension by a linear layer with a tanh activation. The angular cosine similarity is used to compute relevance scores between queries and passages:

$$\mathrm{sim}_{angcos}(q,p) = 1 - \frac{\arccos(\mathrm{sim}_{\cos}(q,p))}{\pi} \ . \tag{3.6}$$

The model is trained using a triplet loss (Equation 3.3). To generate the training triples, negatives are sampled from BM25 rankings. As the model is a complement to term-based retrievers, a *zipping* procedure is proposed. Let $\mathrm{r}_{\mathrm{CoRT}} = [a, b, c, d, ...]$ be the ranked passage list returned by CoRT, and $\mathrm{r}_{\mathrm{BM25}} = [e, c, f, a, ...]$ the one returned by BM25. The result of *zipping* is $[a, e, b, c, f, d, ...]$, by interleaving the items from the two lists.

Ding et al. [2020] identified some issues on the training of bi-encoders, such as the large number of unlabeled positives, which leads to an high probability of false negatives, and the discrepancy between the training and inference steps, since during training the model optimized probabilities for positive passages in a small candidate set for each query, while during inference positive passages for each query are identified from a collection with millions of candidates. Noting these limitations, they proposed RocketQA, which was built on top of a bi-encoder using neural language models for building representations, presenting three optimizations for the training.

First, cross-batch negatives works when training on multiple GPUs. Each GPU will be training on a mini-batch with B instances, where each query is paired with a positive passage. Instead of sampling additional negatives, each query can be further paired with $B-1$ passages, corresponding to the positives of other queries. By sharing the representations among all GPUs, $n(B-1)$ passages can be used as negatives, where n is the number of GPUs.

Second, denoised negative sampling was used, since the authors argued that cross-batch negatives can increase the number of easy negatives, while hard negatives are more important to train a bi-encoder. As such, a cross-encoder measuring similarity between queries and passages is trained over the original data. The passages predicted as positive with high scores are discarded, and hard negatives are sampled from the remaining top-k. The motivation behind this was that cross-encoders are more powerful when it comes to capturing two-way semantic similarity, when compared to a bi-encoder, but they are not efficient in inference due to the large number of candidates.

Finally, they augmented the data using the cross-encoder to annotate unlabeled questions. Overall, the motivation was to achieve a trade-off between the temporal efficiency of bi-encoders and the higher performance of cross-encoders. Employing these optimizations on the training of a bi-encoder, RocketQA's results are currently state-of-the art on the MS-MARCO [Campos et al., 2016] dataset.

Thakur et al. [2020] followed a similar approach to extend the previously mentioned SBERT model, proposing AugSBERT. A pre-trained cross-encoder is used to label sampled unlabeled pairs, which are then merged with the already labeled pairs. The new pairs to be labeled with the cross-encoder are sampled using different techniques, such as (i) random sampling, (ii) kernel density estimation (KDE), which aims to get a similar label distribution to the set of already labeled pairs, (iii) BM25 sampling, where the top-k passages returned by BM25 for each query are selected, and (iv) semantic search sampling which uses a trained SBERT model on the already labeled set to retrieve the top-k most similar sentences. AugSBERT outperformed the previous SBERT in all tested tasks. The authors noted that the sampling strategy was crucial for the improvement, with BM25 and KDE sampling producing the best results, while random sampling decreases the performance when compared to the original SBERT.

3.1.3 Expansion Models

Expansion models aim to expand the representations with new terms so that vocabulary mismatch (i.e., the difference between query terms and those used in the document) is attenuated. More specifically, Nogueira et al. [2019] proposed a document expansion model, doc2query, which trains a Transformer to predict possible queries that an input document might answer. The documents are expanded by appending 10 sampled predicted queries to them. The expanded documents can then be indexed for retrieval. The example by the authors used BM25 to extract the top-1000 passages for each query, followed by re-ranking with BERT. In subsequent work, Nogueira [2019] further revised this method, replacing the Transformer as the expansion model by T5 [Raffel et al., 2020], i.e., a multitask model that addresses problems as sequence-to-sequence tasks, and which has achieved very strong results on a variety of NLP tasks.

T5's architecture is based on the Transformer, previously explained on Section 2.2.1, with small differences such as a different positional encoder. Each task is converted to a text-to-text format, i.e., a textual prefix is added to the actual query. For instance, "translate English to German: *query*" can be used as a prefix to a machine translation task, where the model will output the translated query. Classification or ordinal regression tasks can also be considered, with the output being the string representation of the target value. As for training, it uses a masked language model similar to BERT. Tokens are replaced by a consecutive span of corrupted tokens, where consecutive words are replaced by one single token. The objective is to output the original sentence, i.e., correctly identify the corrupted tokens. Other approaches were considered, such as next-word prediction and deshuffling. The authors also noted that for multiple downstream tasks, fine tuning all pre-trained parameters outperformed freezing layers.

Besides expansion, Nogueira et al. [2020] also proposed another usage for T5, adapting it directly to the task of document ranking, by performing sequence to sequence classification. For this task, the input prompt is "Query: q Document: d Relevant:". The model is fine-tuned to classify the sequences as *true* or *false*, depending on whether document d is relevant to query q. Documents are ranked based on the probabilities assigned to the *true* token, which are obtained by applying softmax to *true* and *false* token logits.

3.1.4 Kernel Models

Xiong et al. [2017] proposed K-NRM which is a kernel based neural ranking model. Given the word-wise representation of queries and passages (e.g., through regular word embeddings), a matrix M, is created where each element $M_{i,j}$ is the cosine similarity of the i^{th} query term, q_i and the j^{th} passage term, p_j:

$$M_{i,j} = \text{sim}_{cos}(r_{q_i}, r_{p_j}) \ . \tag{3.7}$$

In the previous expression, r_{q_i} is the representation of the i^{th} query term, and r_{d_j} is the representation of the j^{th} passage term.

Query-passage ranking features $\phi(M)$ are then extracted from the matrix using n radial basis function kernels, K, which compute the distribution of pair similarities around them:

$$\phi(M) = \sum_{i=1}^{n} \log K(M_i) \ , \tag{3.8}$$

$$K(M_i) = \{K_1(M_i), ..., K_n(M_i)\} \ , \tag{3.9}$$

$$K_k(M_i) = \sum_j \exp\left(-\frac{(M_{ij} - \mu_k)}{2\sigma_k^2}\right) . \tag{3.10}$$

Features $\phi(M)$ are fed to a linear layer with a tanh activation. For training of the parameters, a triplet loss is used (Equation 3.3).

More recently, Hofstätter et al. [2019] proposed a Transformer-Kernel (TK) approach, based on BERT-based ranking models and K-NRM. This approach aims to balance efficiency, effectiveness, and interpretability. Initial representations of queries and passages pass independently through Transformer layers, producing contextual embeddings. Query and passage encoders have shared learnable weights. TK then creates an interaction matrix M similar to K-NRM (Equation 3.7). The same RBF Kernel feature extraction is used, but unlike K-NRM, this does not enforce exact term matching, as contextualized representations do not produce exact matches. The pooling process in TK is extended by generating passage length (p_{len}) normalized features:

$$\phi_{\text{len}}(M) = \sum_{i=1}^{n} \frac{K(M_i)}{p_{\text{len}}} . \tag{3.11}$$

Both the log-normalized $\phi(M)$ and passage length normalized $\phi_{len}(M)$ features are fed to a linear layer to produce scalars. The final score is a weighted sum of both.

It is important to notice that this architecture shares a problem with the original Transformer, which is the quadratic memory complexity with respect to the input sequence length, due to the self attention mechanism (Equation 2.15). The Conformer-Kernel (CK) [Mitra et al., 2020] tackles this issue by using a Separable Self Attention (SSA) mechanism:

$$\text{SSA}(Q, K, V) = \text{softmax}(Q) \times \text{softmax}(K^\top)V . \tag{3.12}$$

A grouped convolution (i.e., using multiple different convolution filters over the same input, concatenating the results) is also applied before the separable self-attention. Query-term independence is a property that assumes that passages can be scored independently with respect to each query term, accumulating the scores. This assumption is present in CK, achieved by only generating contextual representations for the passages, and by applying the K-NRM's Kernel-Pooling based aggregation to each query term independently. So that explicit explicit term matching can be enforced, the term-passage score is computed as follows:

$$\text{score}(t, p) = w_1 \text{BN}(\text{score}_{latent}(t, p)) + w_2 \text{BN}(\text{score}_{explicit}(t, p)) + b , \tag{3.13}$$

where w_1, w_2 and b are learnable parameters, $\text{score}_{latent}(t, p)$ is computed using CK, and BN is the BatchNorm operation, given by:

$$\text{BN}(x) = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x]}} . \tag{3.14}$$

In the previous expression, $\mathbb{E}$ is the expected value and Var the variance. A model based on BM25 is used to compute $\mathrm{score}_{explicit}(t, p)$:

$$\mathrm{score}_{explicit}(t, p) = \mathrm{idf}(t) \frac{\mathrm{BS}(\mathrm{tf}(t, p))}{\mathrm{BS}(\mathrm{tf}(t, p)) + \mathrm{ReLU}(w \ \mathrm{BS}(|p|) + b) + \epsilon} \ , \tag{3.15}$$

where, w and b are learnable parameters and BS is the batch scale operation:

$$\mathrm{BS}(x) = \frac{x}{\mathbb{E}[x] + \epsilon} \ . \tag{3.16}$$

3.2 Geographical Retrieval Methods

The methods for passage retrieval discussed in the previous sections are designed for general domain questions. As such, some methods to deal with the geographical, mostly focused on their properties and how to represent them, are now considered.

Previous studies have also addressed the difficulties and inherent problems of Geographic Information Retrieval (GIR) [Purves et al., 2018] and Geographic Question Answering (GeoQA) [Mai et al., 2021]. The types of questions GeoQA systems aim to answer are very diverse, and different types of questions need data from different sources (e.g., instead of retrieving answers from document collections, several GeoQA studies have instead relied on structured knowledge bases describing geo-spatial information [Haas and Riezler, 2016, Lawrence and Riezler, 2016, Punjani et al., 2018]). Also, the vagueness of some geographic concepts increases the difficulty of properly answering the questions formulated by users.

Some systems have combined textual representation models with structured geographic information, so as to try to overcome some of the issues posed by geo-spatial questions. This includes seminal GIR research based on heuristics [Purves et al., 2018, Mandl et al., 2008, Cardoso et al., 2005], combining BM25 ranking together with geo-spatial criteria derived from gazetteers or general knowledge bases, and also more recent GIR/GeoQA methods based on machine learning and, more recently, neural networks.

For example, Contractor et al. [2020] developed a spatial-reasoner which aims to answer questions where a geographical entity is the answer. This approach works by scoring a list of candidate entities against a query. To achieve that, it uses a distance aware query encoder, i.e., contextual representations of query tokens are extended with information regarding a candidate entity, c. Each query token representation is appended with an one-hot encoding representing IOB tags (Inside, Outside and Beginning labels, identifying spatial tokens). Then, the spatial tokens (the ones labeled with B and I) are also concatenated with the Manhattan Distance from c to the location mention lm_k, while the remaining tokens are concatenated with 0 (d_i^c). This whole concatenation is then fed to a bi-directional GRU to generate the encodings.

Another element of the architecture is a distance-reasoning layer, that aims to learn a model that can infer both whether a location mentioned is needed to be considered for answering, and how it needs to be used for answering. For each lm in a query, the model outputs a distance weight w_k that captures contributions between lm_k and c, under the constraints present in the query (e.g. "near" or "far", among others). This is achieved by blocks of feed-forward layers with ReLU activation applied at each position of the representation. The output of the final layer, o, is used to compute the distance-weight vector w for a question:

$$w = \tanh(o \odot \mathrm{B}) \ , \tag{3.17}$$

where B is a binary vector assigning 1 to the positions originally assigned with the *B* tag, and $\odot$ stands for the Hadamard product (element-wise multiplication). Finally, the score score_L for a candidate entity c is given by:

$$\mathrm{score}_L = wd^c \ , \tag{3.18}$$

where d^c is the vector of Manhattan Distances between c and all location mentions used in the query encoder.

Another network that focus only on textual properties is also used, through a bi-encoder, combining entity embeddings and question representations to generate a relevance score, score_T. The two scores are combined in an overall score, according to the following expression:

$$\mathrm{score} = \alpha\, \mathrm{sigmoid}(w_T\, \mathrm{score}_T)\tanh(w_L\, \mathrm{score}_L) + \beta\, \mathrm{sigmoid}(w_T\, \mathrm{score}_T) \ . \tag{3.19}$$

The parameters w_T and w_L are learnable weights, while α and β are the combination weights. The model is trained using max-margin loss.

The work by Li et al. [2020] uses semantic encoding to capture the information in geospatial questions, i.e., the words in the questions are mapped to a sequence of predefined semantic elements, such as *PlaceName*, *Activity*, *PlaceType*, among others, also including relations such as *SpatialRelationship* and *LogicRelationship*. The authors proposed a neural tagging approach, which outperformed rule-based tagging pipelines on their evaluation. The method starts by generating embeddings for the sentences, followed by a sequence labelling model to capture inter-word dependencies and sentence structure. The architecture with the best results uses BERT, fine tuning all layers during training. To further enrich this representation, a semantic graph (GSG) was proposed, since semantic encoding by itself does not portrait logical or semantical dependencies between terms. The nodes of the graph are the aforementioned tagged semantic elements, and the edges are labeled with semantic relations between source and target. As a practical example for the usage of these representations, the authors proposed a method to translate them into GeoSPARQL queries, which are used to query RDF-encoded knowledge bases in the form of (subject, predicate, object) triples.

Xu et al. [2020] also proposed a parsing method to identify the syntactic and semantic structure of geographic questions, to extract the intent of those questions and the descriptions and criteria of the intent. First, part-of-speech tagging is used, focusing on *WH*-words (e.g. "what" or "when", among others), helping to differentiate each part of the sentence in semantic types. A pre-trained named entity recognition model is applied to identify toponyms and place types. Relations between the semantic types are captured using a constituency parser. Verbs are classified as activities or situations using embeddings from neural language models. The result of this parsing pipeline is combined with the types of question words (i.e., the previously identified *WH*-words) to divide the geographic question in its intent and criteria, resorting to depth first searches over the parsing tree and/or heuristics for any ambiguous cases. The authors use this process to analyse multiple corpora, identifying the semantic type distributions.

Hamzei et al. [2021] investigated human answers to where-questions, presenting templates to characterize and replicate their structure. They follow machine learning approaches for pattern learning, introducing an encoding of questions and answers based on the type, scale and prominence (TSP) of their toponyms. The authors define type as a reference to a group of places with similar characteristics, scale as the hierarchical organization concerning size and relations between places, and prominence as a measure of how well-known a place is. These three items are used to characterize descriptions and capture relationships among places, i.e., relate toponyms in the questions to those in their answers.

The encoding process starts by modeling the questions and respective answer as sequence of toponyms. Then, the toponyms are encoded into type, scale and prominence. To achieve this, the toponyms are identified by a pre-trained Named Entity Recogniton model. Gazeteers (i.e., index of toponyms) are used to extract attributes for the identified toponyms, which are then used for the TSP encoding. A TSP encoding of a question and its answer can be seen as a generic form. As such, the authors compared questions to answers through type, scale and place distributions, deriving patterns through rule mining algorithms. The generic forms are used in prediction models, that given a question's generic form, try to predict the answer generic form.

The authors evaluated both the rule mining and the predictive systems on MS-MARCO [Campos et al., 2016]. They identified that the scale in answers is, in general, one level coarser than in the questions. More specifically, questions often feature city-level toponyms, while answers contain country-level ones. The extracted rules were very representative, for instance, the most frequent rule can be applied to 1277 question-answer pairs.

Martins and Calado [2010] proposed a learning to rank approach for geographical information retrieval. The data used consisted on queries and a collection of documents divided in headline and body.

Given a query-document pair, textual features were extracted, including term frequencies, inverse document frequencies, document length, TF-IDF scores, and BM25 scores. These features were computed for the headline only and for the concatenation of the headline and the document body.

Geographical features were also considered. For this, a tool to annotate the name places and scopes with their coordinates and bounding boxes was used. Multiple features were then computed, including the area of the geographic scope of the query and the document, the area and degree of overlap between the geographic scopes of the query and the document, and the normalized distance between the centroid point for the geographic scopes of the query and the document. Averaged features were also considered, combining geographical and textual features. The features were used to train a SVM^{map} model [Yue et al., 2007], a listwise L2R approach which aims to optimize the MAP (Equation 2.22).

As for results, the authors noted that using both textual and geographical features yielded the best results in terms of MAP. Using only the geographical features resulted in a poor performance, while using only textual features was a competitive baseline.

Wang et al. [2021a] worked on the subject of temporal question answering, proposing a tailored approach for document re-ranking where the queries contain temporal entities. While it does not address geographic content, the work focuses on distance between the temporal entities, which could be directly adapted to the geographical domain. The proposed method starts by retrieving the top-100 documents for each query using BM25. Then, those documents are re-ranked by a time aware re-ranking module. This module starts by extracting the time scope of the query, an interval $T(Q) = (t^Q_{start}, t^Q_{end})$, using SUTime. If a query contains more than one interval, only the first one is considered. Then, for each retrieved document, two scores are computed. Besides contrasting the time scope against the temporal information within a document, this approach also takes into account the document publication timestamp, considering the intuition that documents published close to the time period associated with the question have an higher probability of being relevant.

To compute the scores associated with each document's temporal content with respect to a query, a list of temporal intervals is extracted from each document, $T(D) = (T_1(D), T_2(D), ...)$. Each interval consists of a start time and an end time. Hence, two lists can be considered, one for the start times, $T^s(D)$, and one for the end times, $T^e(D)$. Then, two probability density functions are generated through kernel density estimation, one for each list. The probabilities of the query's start and end time given the lists are computed following the estimations, taking the average of both for a score, s_1.

The scores associated with the document's publication date are computed in a similar fashion. The probability of generating a time scope given a timestamp is given by a exponentially decaying function of the distance between the publication date and the time scope. This probability is again considered as a score, s_2.

The two scores (s_1 and s_2) are normalized and averaged for a final temporal score, and then the documents are re-ranked by linearly combining the BM25 scores and the temporal scores. The collection of documents and queries with temporal entities used by the authors are publicly available.

3.3 Overview

This chapter covered related work both on open-domain passage retrieval and geographic information retrieval. For passage retrieval, the cross-encoder and bi-encoder architectures were discussed, focusing on their benefits and disadvantages towards one another. Multiple studies focusing on this two architectures were revised, detailing their training, and usage within a retrieval setup. Regarding geographic retrieval methods, studies that combine the geographic information with the textual representations were addressed. Also, works that aim to characterize the properties of the geographic questions and answers were covered.

Chapter 4

A Dataset for Geographic Question Answering

This chapter will introduce the dataset used in the scope of this work. First, a geoparsing procedure will be covered, detailing the process of obtaining the geographic coordinates for entities within an input sentence. Then, MS-MARCO, a large scale benchmark dataset, will be characterized, and the extraction of its geographic subset through the aforementioned geoparsing procedure will be addressed. The properties of this geographic subset will also be covered.

4.1 Geoparsing Textual Data

Since one of the objectives of this work was to study the impact of geographic distances in the re-ranking of passages to geo-spatial questions, the association of geo-spatial coordinates to place-names within a collection of queries and passages was necessary. As such, a tool to recognize and assign coordinates to geographic entities (i.e., a geoparser) was needed. Recent work on this subject relies on the usage of deep neural networks for directly predicting geo-spatial coordinates from textual representations [Cardoso et al., 2019, Kulkarni et al., 2020], although for the specific purposes of this work, the chosen method should be very efficient, so that it could be used to process large collections, including all the queries and the associated top-1000 passages.

Mordecai [Halterman, 2017], an open source tool which is able to resolve entities to geographic coordinates was chosen for this purpose. This system starts by using a pre-trained named entity recognition model to identify place-names in the input texts. Then, a large coverage gazetteer (i.e., a toponym index) is used to find the potential coordinates of the recognized place-names, by matching the place-name strings against candidate gazetteer entries. Neural networks, trained on annotated English data, are used to infer the correct country and gazetteer entry for each place name, combining heuristics based on prominence (i.e., prefer capital cities and important places) and contextual similarity.

This tool was used to identify place-names, and for mapping place-names to coordinates, in both queries and passages. To parse queries in the training data, and since queries are usually small, their

```
Query:
house for rent in hickory creek texas

Passage:
The average square feet of the homes in Hickory Creek is 1,913 sqft. There are currently 3
homes for lease in Hickory Creek subdivision. The average rent in Hickory Creek is $ 1,767
at an average price of 1 per square foot.
```

Figure 4.1: A query and its relevant passage, from MS-MARCO development set.

```
{'word': 'texas',
 'spans': [{'start': 32, 'end': 37}],
 'country_predicted': 'USA',
 'geo': {'admin1': 'Texas',
        'lat': '31.25044',
        'lon': '-99.25061',
        'country_code3': 'USA',
        'geonameid': '4736286',
        'place_name': 'Texas',
        'feature_class': 'A',
        'feature_code': 'ADM1'}}
```

Figure 4.2: Mordecai output when processing the query *house for rent in hickory creek texas* without concatenation.

```
{{'word': 'hickory creek',
 'spans': [{'start': 18, 'end': 31}],
 'country_predicted': 'USA',
 'geo': {'admin1': 'Texas',
        'lat': '33.12234',
        'lon': '-97.04306',
        'country_code3': 'USA',
        'geonameid': '4829219',
        'place_name': 'Hickory Creek',
        'feature_class': 'P',
        'feature_code': 'PPL'}},
{'word': 'texas',
 'spans': [{'start': 32, 'end': 37}],
 'country_predicted': 'USA',
 'geo': {'admin1': 'Texas',
        'lat': '31.25044',
        'lon': '-99.25061',
        'country_code3': 'USA',
        'geonameid': '4736286',
        'place_name': 'Texas',
        'feature_class': 'A',
        'feature_code': 'ADM1'}}}
```

Figure 4.3: Mordecai output when processing the query *house for rent in hickory creek texas* concatenated with the relevant passage.

concatenation with the relevant passage was used as input, so as to better contextualize the query and minimize errors. However, only the entities contained within the length of the query were kept. A query was considered to be geographic if at least one entity is mapped to coordinates. Passages were parsed with no extra pre-processing.

As an example, consider the query and relevant passage pair in Figure 4.1. When parsing the query alone, i.e., without the aforementioned concatenation with the relevant passage, the output includes only the entity *texas*, as shown in Figure 4.2. Note that the entity *hickory creek* was not recognized by Mordecai in this scenario for the query. However, in the passage, *hickory creek* is the only entity, and Mordecai is able to identify it when parsing the passage alone. This causes a misalignment between the identified entities in the query and the passage. To overcome this issue, when processing the query, it is concatenated with the relevant passage, for contextualization. Figure 4.3 shows the output with concatenation, and both *texas* and *hickory creek* were identified in the query. This process may identify some entities that are only present in the passage, and assign it as a query entity, due to the concatenation. To avoid this, the span of the words, present in the output, is used to consider only query entities within its span. As for the remaining of the output, it provides useful information such as administrative regions, geonames identifiers, and the geographical coordinates.

4.2 The MS-MARCO Dataset

MS-MARCO [Campos et al., 2016] is a large benchmark dataset that can be used for multiple retrieval tasks. For passage retrieval, the available data comprises over one million queries from BING and eight million passages, with human annotations concerning relevance judgements (i.e., most queries are associated to one relevant passage). This dataset was used for all the experiments, due to being the most well-known benchmark for passage retrieval.

Other authors [Hamzei et al., 2021] have studied the geographic contents of this dataset, identifying a total of 12548 geographic question-answer pairs (under their particular definition), and 22307 different place-names mentioned in these queries and passages. However, given the need of associating entities to geographic queries, this work fully processed MS-MARCO with Mordecai, as described in the previous section.

4.2.1 Geographical Subset of MS-MARCO

This dataset provides separate training and development sets, both with relevance judgements. There is also an official test set, but the labels are undisclosed, and as such was not considered for this work. The training set was parsed with Mordecai, identifying 27104 geo-spatial queries. From those, only the 16833 queries with at least one relevant passage were considered. From the development set, 292 queries were identified as geo-spatial (i.e., at least one entity resolved to coordinates by Mordecai). A total of 292 random queries were also sampled from the training set so as to compose a separate development set (i.e., given that only the development set of MS-MARCO is available publicly, the data from this development set was used as the test set to evaluate the models, and a separate development set for tuning hyper-parameters was constructed). The top-1000 passages retrieved with BM25 for each query were also processed by Mordecai. Hence, the geo-spatial subset of MS-MARCO is as follows:

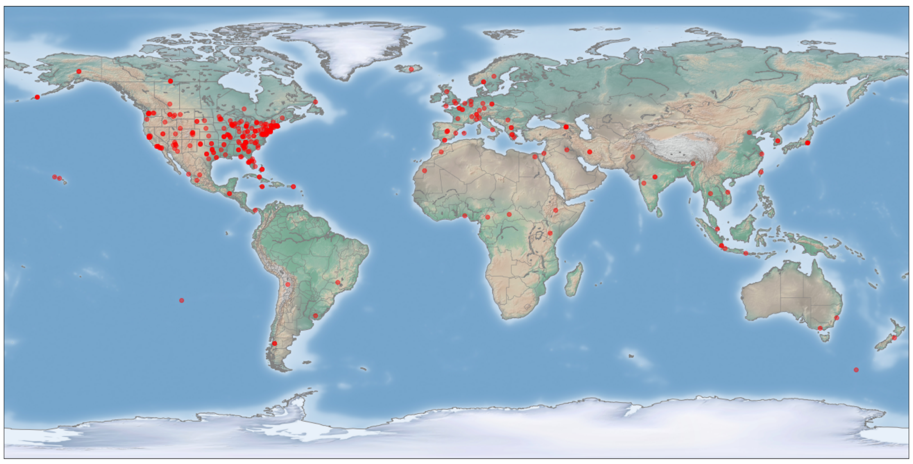


Figure 4.4: Geo-spatial distribution for places within queries of the MS-MARCO development set.

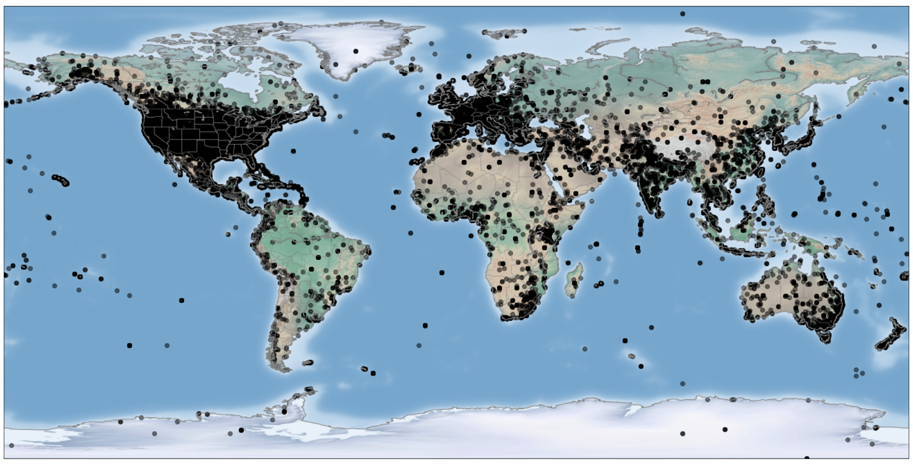


Figure 4.5: Geo-spatial distribution for places within passages associated with queries of the MS-MARCO development set.

- Training set: 16541 queries, together with the top-1000 passages as retrieved by BM25 for each query;
- Validation set: 292 queries, together with the top-1000 passages as retrieved by BM25 for each query;
- Test set: 292 queries, together with the top-1000 passages as retrieved by BM25 for each query.

Every query in each set has at least one relevant passage, and every passage associated with a query was also processed by Mordecai. Ultimately, over one million geographic query-passage pairs were used during model training.

The geographical coordinates of entities identified by Mordecai for queries in the development set of MS-MARCO and their respective passages are depicted in Figures 4.4 and 4.5, respectively. As expected, given the nature of the data, most of the entities are concentrated in North America and Europe, although the data has a global geo-spatial distribution. As for the entities, Figure 4.6 shows that most of the queries within this development set contain only one, the maximum being three. An analysis

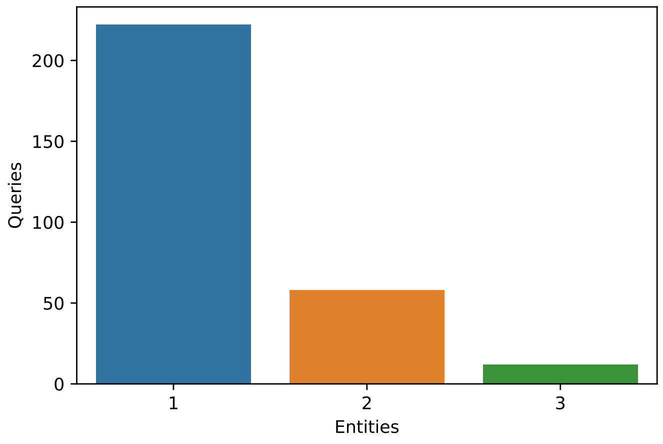


Figure 4.6: Number of entities per query in the geographic subset of MS-MARCO development set.

of the queries showed that most of these queries are factoids, i.e., informal needs about some place, for instance *why did italy attack ethiopia in 1935* and *do us citizens need passports to enter and exit canada*. However, some queries include more complex relations between entities, which would perhaps be more suitable for other retrieval systems based on external knowledge bases, for example *how far is it from colorado springs to la junta*.

4.2.2 Data Expansion

Previous studies have, for instance, used models to augment passages' textual information by concatenating generated queries [Nogueira, 2019]. Others, have used cross-encoders to label unassigned passages in the textual collections [Thakur et al., 2020].

In this work, a T5 [Raffel et al., 2020] model was used to generate relevant queries to MSMARCO passages that have no associated query, to be used during model training. First, 16541 geographical passages (i.e., passages in which Mordecai was able to identify at least one entity) were sampled randomly from the set of passages with no relevant queries. Then, a public T5 model[1] (fine-tuned over MSMARCO, SQUAD, CoQA and RACE for question generation) was used to generate ten queries for each passage. From the ten queries, one with at least one geographic entity was selected. Some passages were paired with queries that did not contain any geographic entities, and as such, were discarded. Ultimately, the extended training set contains the original 16541 queries, and 9000 generated ones.

[1] `https://github.com/amontgomerie/question_generator`

Chapter 5

Neural Models for Geographic Passage Re-ranking

In this chapter, the techniques that were used during the training of the models are discussed. First, the architecture of the models that were used is addressed, namely Transformer-based cross-encoders and bi-encoders. Then, a re-ranking strategy which leverages the geographic distance between entities in queries and passages is introduced. The fine-tuning of both bi-encoder and cross-encoder models is discussed, focusing on hard negative sampling and batching methods. Finally, a cross-architecture knowledge distillation technique is addressed.

5.1 The Bi-Encoder and Cross-Encoder Architectures

As discussed in Section 3.1, neural methods currently achieve state-of-the-art results for passage retrieval. This work follows those baseline architectures, based on the usage of Transformer-based neural language models, either following a bi-encoder or a cross-encoder architecture. For convenience, their general architectures are again depicted in Figure 5.1.

Bi-encoders encode queries and passages independently. This allows the offline indexing of individual passage representations through methods that support the fast execution of maximum inner product searches [Johnson et al., 2017]. Conversely, cross-encoders generate a representation for the concatenation of a query and a passage, directly modelling the interactions between these two components. These representations can be obtained through mean pooling of word token embeddings, or by considering only the [CLS] token from a model like BERT [Devlin et al., 2019] or RoBERTa [Liu et al., 2019]. The representation can then be used to predict a relevance score for the passage to the query, for example by using a feed-forward layer with a sigmoid activation.

Usually, due to their superior computational performance, bi-encoders are used for full retrieval, i.e., to identify the top-k relevant passages from a large background collection. Cross-encoders are mostly used for re-ranking a set of top-k passages, retrieved initially through an efficient first-stage method, since they provide more accurate relevance estimates.

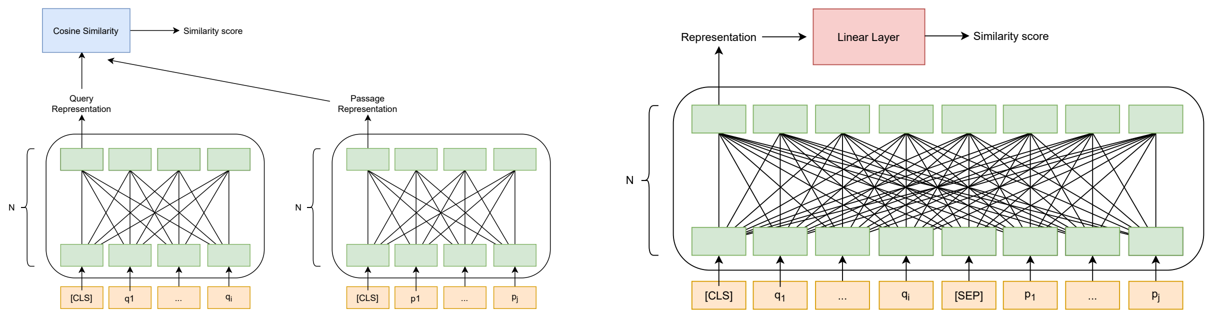


Figure 5.1: Architecture for bi-encoder (left) and cross-encoder (right) retrieval models.

5.2 Model Fine-tuning

This section starts by introducing a re-ranking strategy based on the geographical distance between queries and passages. Then, a negative sampling strategy leveraging said distance is addressed, to be used for fine-tuning of both bi-encoders and cross-encoders.

5.2.1 Geographical Re-ranking

Following the intuition that passages mentioning places that are geographically close to the places mentioned within a query should, in principle, be more relevant, a re-ranking method which leverages the distance between the set of toponyms in a query, T_q, and the set of toponyms in a passage, T_p, is considered:

$$\mathrm{distance}(T_q, T_p) = \min_{t_q \in T_q, t_p \in T_p} \mathrm{h}(t_q, t_p) \ . \tag{5.1}$$

In the previous equation, t_q, t_p are the toponyms in the sets T_q and T_p, and $\mathrm{h}(\cdot)$ is the haversine distance, given by:

$$\mathrm{h}(x, y) = 2r \arcsin \sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_x \cos\phi_y \sin^2\left(\frac{\Delta\lambda}{2}\right)} \ , \tag{5.2}$$

where r is Earth's radius, λ_x, ϕ_x and λ_y, ϕ_y are the geographical latitude and longitude of points x and y, and $\Delta\lambda$ and $\Delta\phi$ are their absolute differences.

As a motivation for the usage of this distance when ranking passages with respect to a query, consider Figure 5.2, which shows the distribution of the distance between queries from the geo-spatial validation set of MS-MARCO (outliers were removed for visualization purposes), and their relevant passage, top-25 BM25 passage, and top-50 BM25 passage. This shows that the relevant passage is more likely to have a smaller distance to the query, when compared to other passage examples.

Given the previous analysis, a re-ranking setup leveraging this notion of distance was considered. First, the top-1000 passages for each query are re-ranked by distance. Then, if distance is tied, the BM25 scores can be considered. Results for this re-ranking, are shown in Table 6.1, and discussed later in Chapter 6. In brief, this strategy slightly surpassed BM25 retrieval, which motivated its usage for hard negative sampling.

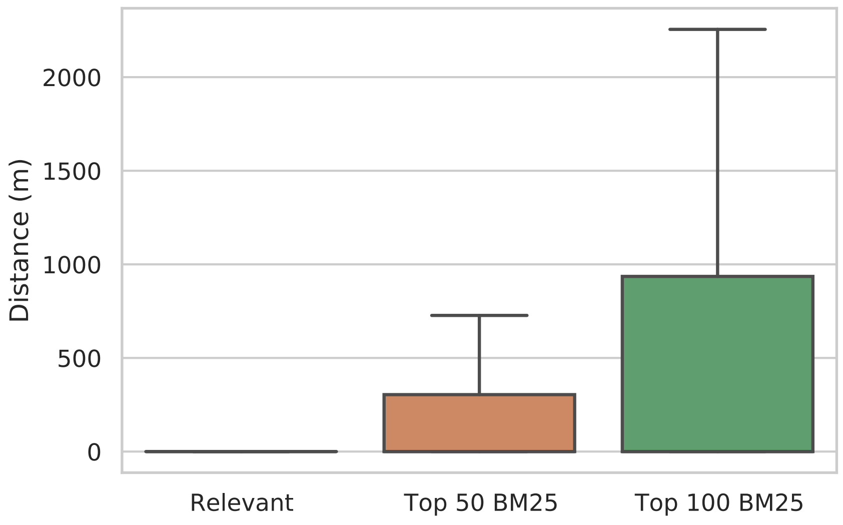


Figure 5.2: Distribution of distance between queries from MS-MARCO geographic subset and corresponding relevant passages, top-25 BM25 passages, and top-100 BM25 passages.

5.2.2 Hard Negative Sampling and Batching

When building a batch for training, choosing the negative passages for a given query is a crucial process. If the passages are sampled randomly, the similarity between the passages and query is likely to be very low (i.e., easy negatives), not contributing to the learning process. Conversely, choosing the passages that despite being negatives are the most similar to the query (hardest negatives), will make the distance between the embeddings very small, which may lead to collapsed models [Wu et al., 2017].

Several authors have discussed the problem of performing negative sampling. For example, as discussed in Section 3.1.2, RocketQA [Ding et al., 2020] uses a cross-encoder to predict relevancy, sampling negative passages from the denoised top-k negatives with a higher score.

In this work, the use of a geographically-aware negative sampling procedure was considered. For this, the previously introduced notion of distance, which corresponds to the minimum haversine distance between the geo-spatial entities in a query and a passage, is used (Equation 5.1).

First, the top-25 passages for each query in the training set are obtained, either using BM25 for lexical similarity, or using a cross-encoder, for semantic similarity. Then, to sample N hard negatives for a query, the N passages in the top-25 list, with highest distance, are chosen. This way, passages that are lexically or semantically similar to the query, yet geographically distant, are considered as hard negatives. This procedure can also be seen as a denoising mechanism, since a passage is less likely to be a false negative to a geographical query if it has an high geographical distance. An initial set of tests was used to fine-tune the value of 25, and it should be noted that, for each query, only a smaller set of sampled passages ends up being used (e.g., a total of 10 negative passages) in association to each query.

When building batches, triples are first created by associating a query to its positive passage and a negative passage, sampled as described above. Then, to maximize the effective training data, batch-wise negative pairing is applied, by using the positive and negative passages of a given query as negatives for the others. This way, for a batch of N triples, $N \times N \times 2$ pairs can be extracted.

To avoid repeated pairs, there are no duplicate queries within a batch. However, following ideas related to those addressed in other recent work [Hofstätter et al., 2021], similar queries are grouped together in the same batch, since hard negative passages sampled for a given query are probably also challenging for similar queries. To divide the queries in groups of N, a corpus with all queries is first considered. Then, one query is sampled randomly. BM25 is used to compare all queries in the corpus to the sampled one. The top $N-1$ queries are extracted, building the first group. The N queries from the first group are removed from the corpus, and the process is repeated until all queries are grouped. The aforementioned procedure aims to balance easy-negatives and hard-negatives for the batch-wise paring, since the similarity of the queries in the first groups will, in principle, be higher than the similarity in the final groups.

5.2.3 Cross-Encoders

The cross-encoder that was considered for fine-tuning is built on top of ELECTRA [Clark et al., 2020], and it was pre-trained on the full MS-MARCO dataset. The model is provided with the SentenceTransformers library [Reimers and Gurevych, 2020]. Specifically, the model named `ms-marco-electra-base` was used, which was the best MS-MARCO model from this library at the time of the first experiments. Batches were built as described in the previous subsection, considering 4 different queries per batch. This yields a total of 32 pairs being compared per batch (i.e., each of the 4 queries is matched to its positive passage, to its hard negative passage, and to the passages from the other 3 queries). Gradients were accumulated for 10 steps, and the standard binary cross-entropy was used as the loss function:

$$L_{BCE} = -\sum_{p \in P_q^+} \log(\mathrm{score}(q,p)) - \sum_{p \in P_q{}^-} \log(1 - \mathrm{score}(q,p)) \ . \tag{5.3}$$

In the previous equation, P_q^+, P_q^- are the sets of positive and negative passages for query q, respectively, provided within the same training batch. To obtain the $\mathrm{score}(q,p)$, i.e, an estimate of the relevancy of passage p to query q, the representation of the [CLS] token is fed to a linear layer with a sigmoid activation function.

5.2.4 Bi-Encoders

The model to be fine-tuned as a bi-encoder for passage re-ranking is based on a distilled version of RoBERTa [Liu et al., 2019]. It is also provided through the SentenceTransformers library, and pre-trained on the whole MS-MARCO dataset [Reimers and Gurevych, 2020]. Specifically, the model named `msmarco-distilroberta-base-v2` was used, which was the best MS-MARCO bi-encoder within the SentenceTransformers library, at the time of the first experiments.

The general setup of the previous section was kept the same when training bi-encoders. The only difference is on computing the relevance of passage p for query q, $\mathrm{score}(q,p)$. Contrary to cross-encoders, the transformer model in bi-encoders receives a single sentence as input. As such, representations are generated for queries and passages independently, through mean pooling of the token embeddings. Within a batch, there are 4 queries and 8 passages (i.e., one positive and one hard negative passage selected per query). Hence, a similarity matrix $M_{4,8}$ can be built, where the value for $M_{i,j}$ is given by the cosine similarity between the representations of query i and passage j.

An inspection to the cosine similarity values allowed to note that their ranges and magnitudes were such that they did not provide a good separation between positive and negative passages. Thus, the values are normalized by first applying the softmax operation over the rows of the similarity matrix independently. Then, the columns of the matrix are also processed through a similar approach. Both values are summed, and the results are then divided by two, so that the final scores range from 0 to 1. The scores are provided to a standard binary cross-entropy loss, during model training.

5.3 Knowledge Distillation

As previously stated, cross-encoders produce better relevance estimates due to processing the concatenation of a query and a passage, hence being able to better capture the interactions between the two sentences. However, bi-encoders provide a computationally more efficient method for retrieval, since the representations are computed independently for queries and passages (i.e., for cross-encoders, all query-passage pairs need to be processed by the neural model).

As such, a knowledge distillation method is now addressed, in order to try to approximate the results achieved by bi-encoders to those of cross-encoders. Previous studies have addressed distillation processes that go from larger models to smaller versions, for example by using the outputs of the large model as targets [Jiao et al., 2020, Sanh et al., 2019]. This is, the larger version of the model can be used to label the examples that the smaller model will use for training. However, when distilling from a cross-encoder to a bi-encoder, trying to fit the cross-encoder's outputs directly is not optimal, since the range and magnitude of the cross-encoder scores (sigmoid of logits) differ from those of the bi-encoder (cosine similarity of vector embeddings). Approaches for cross-architecture distillation have been attempted, for example by optimizing a value that corresponds to the margin between scores [Hofstätter et al., 2020].

The distillation process used for this work is based on the Spearman rank correlation coefficient, which considers the ranked lists of scores instead of the scores themselves. Rank-based metrics are not differentiable, which means that it is not possible to use gradient-based optimizers for model training directly in this scenario. Nonetheless, methods for fully differentiable soft sorting and ranking have already been explored in the literature, which allows for a differentiable implementation of the Spearman rank correlation coefficient [Blondel et al., 2020]. Let X, Y be lists of ranks, m_X, m_Y their mean values, and x_i, y_i their i^{th} elements. The Spearman rank correlation coefficient can be computed as follows:

$$r(X,Y) = \frac{\frac{1}{n}\sum_{i=1}^{n}\left((x_i - m_X)(y_i - m_Y)\right)}{\sqrt{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - m_X)^2\right)\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - m_Y)^2\right)}} \,. \tag{5.4}$$

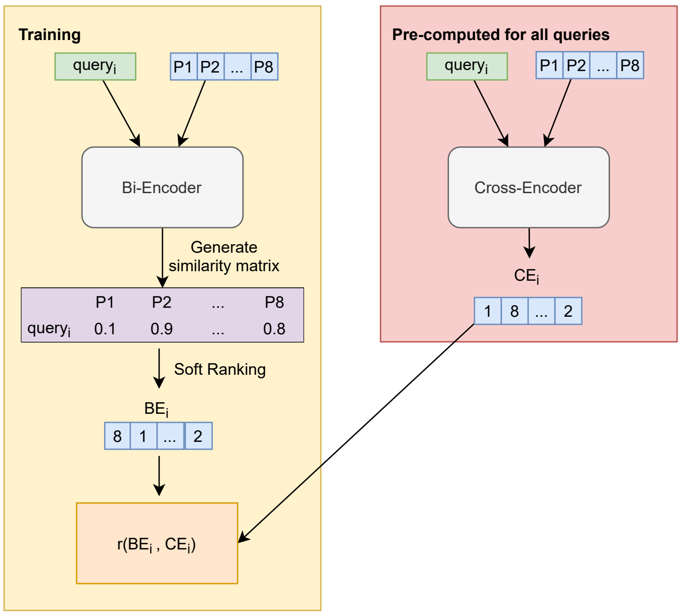


Figure 5.3: Example of computing the Spearman rank correlation coefficient for a single query.

The cross-encoder scores for each pair that is used during training are pre-computed. During training, considering the same setup of the previous subsections, there are 4 lists of scores per batch (i.e., 4 different queries), containing the scores for each of the 8 passages. The Spearman rank correlation coefficient (r) is computed between the ranked lists of cross-encoder scores (CE) and the scores of the model under training (BE). Figure 5.3 shows an example of computing this score for a single pair of lists during training. Then, the average of the 4 pairs of lists is then considered and, to use this value as a loss, a transformation is applied:

$$L_{SRC} = \frac{-\left(\sum_{i=1}^{4} r(\mathrm{BE}_i, \mathrm{CE}_i)\right) + 1}{2} . \tag{5.5}$$

This loss is combined with the binary cross-entropy:

$$L = L_{SRC} + L_{BCE} . \tag{5.6}$$

5.4 Overview

In this chapter, the techniques that were used for model fine-tuning were introduced. First, a notion of distance between queries and passages was covered, and its usage for passage re-ranking was motivated. Then, the training setups were addressed, for both cross-encoder and bi-encoder architectures, focusing on negative sampling and batching methods. Finally, a cross-architecture distillation method leveraging a fully-differentiable approximation to the Spearman's rank correlation coefficient was proposed.

Chapter 6

Experimental Evaluation

In this chapter, the experimental setups and results will be discussed. First, the evaluation metrics and evaluation tasks are introduced. Then, all the experiments that were conducted on MS-MARCO are detailed. Based on those results, a training pipeline for a bi-encoder is proposed and evaluated, which includes the usage of extended data and a cross-encoder for negative sampling and knowledge distillation.

6.1 Experimental Setup

For the evaluation, both the full MS-MARCO development set, and the geo-spatial subset (i.e., our geo-spatial test set, described in Section 4.2.1) were considered. This way, besides evaluating geographic queries, the impact of fine-tuning on geographic data over other types of queries can be studied. The evaluation task was top-1000 re-ranking, i.e., given the top-1000 ranked passages for a query, use the models to re-rank them. The top-1000 passages to be re-ranked were obtained by Pyserini [Lin et al., 2021] (i.e., a Python interface for Anserini), using a BM25 (Equation 2.16) approach that is tuned for MS-MARCO. The hyperparameters b and k_1 were set to 0.68 and 0.82, respectively.

The official evaluation measure for the MS-MARCO passage retrieval task is the Mean Reciprocal Rank at the 10$^{\text{th}}$ passage (MRR@10), given by:

$$\text{MRR@10} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}(i,10)}\ . \tag{6.1}$$

In the previous equation, Q is the set of queries, and rank(i,n) is a function that, given the top-n results for the i^{th} query, returns the position (rank) of the first relevant passage (if no passage is relevant, the MRR is 0). The Recall at k^{th} position (R@k) was also considered:

$$\text{R@k} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{|\text{Rel}_\text{i} \cap \text{Top}_\text{k,i}|}{|\text{Rel}_i|}\ , \tag{6.2}$$

where Rel$_i$ and Top$_{k,i}$ are the relevant and the top k retrieved passages for the i^{th} query, respectively.

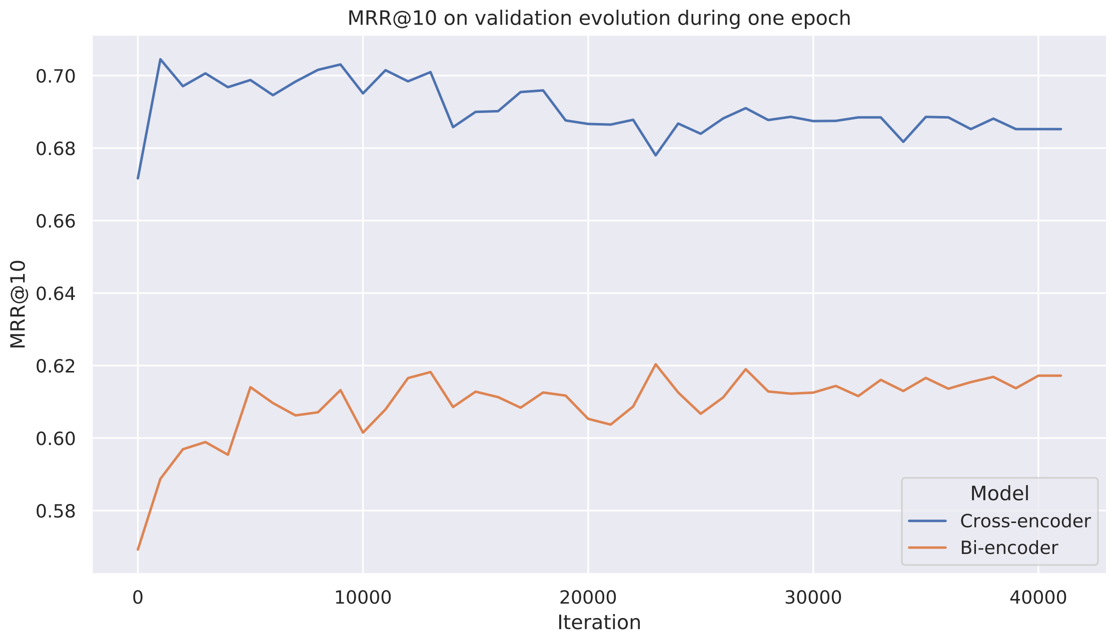


Figure 6.1: MRR@10 evolution during training for both cross-encoder and bi-encoders, considering the validation task of top-25 re-ranking.

6.2 Experimental Results

This section will cover aspects regarding the experimental results. First, the behaviour of the different model architectures during training will be analysed. Then, the evaluation on the geographic subset is conducted, followed by the evaluation on the full MS-MARCO development set. Finally, a bi-encoder trained following a pipeline that leverages the information learnt from the results is also evaluated.

Training Behaviour

Before addressing the results themselves, an analysis on the behaviour of the models during training is addressed. Recall the geographic validation set, which consisted on 292 geographical queries and their top-1000 passages from BM25. This was used to monitor the evolution of both cross-encoders and bi-encoders during training, by evaluating the models every 1000 iterations. Instead of top-1000 re-ranking, the validation task was top-25 re-ranking, since frequent validation steps were taken, and top-1000 would not be suitable in terms of time. Figure 6.1 shows the evolution of the MRR@10 on this task, during one epoch, for one cross-encoder and one bi-encoder. Other models trained in the scope of this work follow the same general behaviour, hence only one is shown for each.

As for the bi-encoders, they seem to benefit from training with a large number of examples (over 40.000 iterations at 32 pairs per iteration, means over 1 million query passage pairs). However, cross-encoders seem to overfit the training data, as the performance on validation slowly converges to an intermediate value. This is in accordance with previous studies that claim that cross-encoders do not need much data for fine-tuning [Nogueira and Cho, 2019]. In this specific case, the base cross-encoder

Table 6.1: MRR@10 and R@{1,5,10,100, 500, 1000} for the geo-spatial subset of MS-MARCO, using the different models.

	MRR@10	R@1	R@5	R@10	R@100	R@500	R@1000
BM25	0.2560	0.1433	0.3893	0.5040	0.7797	0.8990	0.9366
BM25 + Geo. Distance Re-Rankings	0.2633	0.1518	0.4115	0.5143	0.7380	0.8887	0.9366
Base Bi-Encoder	0.4019	0.2631	0.5776	0.6792	0.8921	0.9366	0.9366
Base Bi-Encoder + Geo. Distance Re-Rankings	0.3841	0.2546	0.5451	0.6313	0.7962	0.8990	0.9366
Fine-tuned Bi-Encoder	0.4208	0.2878	0.5982	0.6832	0.8973	0.9366	0.9366
Fine-tuned Bi-Encoder (Extended Data)	0.4253	0.2878	0.6102	0.6889	0.9144	0.9366	0.9366
Base Cross-Encoder	0.4959	0.3333	0.6964	0.8002	0.9281	0.9366	0.9366
Base Cross-Encoder + Geo. Distance Re-Rankings	0.4652	0.3316	0.6142	0.7123	0.8116	0.8990	0.9366
Fine-tuned Cross-Encoder	0.5103	0.3607	0.6861	0.7968	0.9247	0.9366	0.9366
Fine-tuned Cross-Encoder (Extended Data)	0.5054	0.3539	0.6929	0.8025	0.9246	0.9366	0.9366
Distilled Bi-Encoder	0.4291	0.2997	0.5776	0.6809	0.9075	0.9366	0.9366

had already been pre-trained on the whole MS-MARCO collection, which may justify the early validation peak at only one thousand iterations (32 thousand query passage pairs). This led to the hypothesis that bi-encoder may benefit more from the extended training data than the cross-encoders. Given these behaviors, the models considered for evaluation, in the case of bi-encoders, are the final ones, whereas for cross-encoders is the best model at validation.

Geographical Queries

The main objective of this work was to improve ranking models for geographical passages. As such, Table 6.1 shows the results achieved by the fine-tuned models, using the geographically-aware negative sampling with BM25, along those obtained from the base models, for the queries in the geo-spatial subset of MS-MARCO. The table also shows the result of a lexical BM25 baseline, together with a re-ranking based on geographic distance.

While re-ranking based on geographic distance slightly improves the BM25 results, it worsens them when re-ranking the top-1000 passages sorted by the base neural models (i.e., re-ranking passages according to distance, and use the scores from the neural models in case of ties). However, by applying this distance to re-rank passages when sampling hard negatives for fine-tuning the models, the results improved overall. This suggests that while a naive use of geo-spatial distance may fail to improve over strong neural baselines, the information may be used to improve model training.

For cross-encoders, the fine-tuned version achieves a better MRR@10 when compared to the base model. Looking at the multiple recall cuts, the R@1 value also improves, while the R@{5, 10, 100} values slightly decrease. As for bi-encoders, the results on this subset also improved when compared to the base model, but this time all the recall cuts were superior. The distilled bi-encoder achieved the best bi-encoder MRR@10 result, although the performance is still far from that of the cross-encoder model. The cross-encoder behavior of improving R@1 and decreasing other cuts is also transferred in this setting to in the distilled bi-encoder.

To identify the benefits of using more data, the results of training both a bi-encoder and a cross-encoder with the original plus the generated data are also shown. The fine-tuning using extended data followed the same setup, but a weighted cross-entropy was used, where original queries had 10 times more weight than generated ones. This value of 10 was tuned on other tests over a validation set. While cross-encoders do not seem to benefit from more training data, bi-encoders show a small improvement.

Table 6.2: MRR@10 and R@{1,5,10,100, 500, 1000} for the full MS-MARCO development set, using the different models.

	MRR@10	R@1	R@5	R@10	R@100	R@500	R@1000
BM25	0.1874	0.1008	0.2944	0.3916	0.6701	0.8116	0.8573
Base Bi-Encoder	0.2839	0.1667	0.4304	0.5416	0.7957	0.8535	0.8573
Fine-tuned Bi-Encoder	0.2679	0.1519	0.4124	0.5243	0.7840	0.8526	0.8573
Base Cross-Encoder	0.3714	0.2384	0.5385	0.6431	0.8289	0.8565	0.8573
Fine-tuned Cross-Encoder	0.3776	0.2504	0.5330	0.6375	0.8270	0.8564	0.8573
Hybrid with Cross-Encoder	0.3719	0.2395	0.5381	0.6429	0.8288	0.8565	0.8573
Distilled Bi-Encoder	0.2746	0.1692	0.3985	0.5097	0.7862	0.8526	0.8573
Hybrid with Distilled Bi-Encoder	0.2857	0.1696	0.4304	0.5416	0.7963	0.8535	0.8573

As a note, the base bi-encoder and cross-encoder were the best provided by SentenceTransformer when the experiments started. However, during the experimental period, new models have been proposed, which outperform the previous ones. While a robust analysis was not possible, some preliminary experiments on fine-tuning the most recent bi-encoders show that their performance on the geographical subset of MS-MARCO can be slightly improved using the strategy here described.

General Queries

So as to study the impact of the fine-tuning strategy on other types of questions, the models were also evaluated on the full MS-MARCO development set. Table 6.2 shows the results achieved by the fine-tuned models, using the geographically-aware negative sampling with BM25, along those for the base models, and for a lexical BM25 baseline (retrieved using Pyserini), for all queries in the full MS-MARCO development set. The fine-tuned cross-encoder managed to achieve a superior MRR@10 and R@1, again slightly decreasing for other cuts. However, the bi-encoders seem to overfit to the specific geographic context, as the performance of the fine-tuned models decreases in this set when compared to the base models.

The results for hybrid approaches are also provided, considering the base cross/bi-encoder when ranking non-geographic queries, and the best cross/bi-encoder otherwise. With this, we are able to achieve results that slightly surpass the base models, since the fine-tuned models are better at ranking passages for geographic queries. It is worth mentioning that the hybrid cross-encoder performed worse on the full development set when compared to the fine-tuned cross-encoder, which means that the latter is also better at some non-geographical queries than the original model. This may be due to the fact that geographical queries do not necessarily mention place-names explicitly, despite implicitly including geo-spatial criteria. The fine-tuned model may be performing better on these cases, which were not included in the geographic subset because such queries do not contain any entities for Mordecai to disambiguate.

It is worth noting that the results for the base models slightly differ from the ones reported by their authors. This has two reasons. First, the top-1000 passages for each query used in this work were extracted by Pyserini, which has a tuned BM25 retrieval method specific for MS-MARCO. This top-1000 is better in terms of recall at 1000 than the top-1000 BM25 passages provided by the MS-MARCO authors. Second, for bi-encoders, the authors evaluate the task of full-ranking (i.e., extract the top-1000 from the whole collection), while this work covers top-1000 re-ranking.

Table 6.3: MRR@10 for the full MS-MARCO development set, grouping queries by their answer type.

		Cross-Encoders		Bi-Encoders	
Query Type	Number of Queries	Base	Fine-Tuned	Base	Fine-tuned
LOCATION	498	0.4550	0.4738	0.3538	0.3693
NUMERIC	1665	0.3690	0.3823	0.3022	0.2915
PERSON	461	0.4370	0.4365	0.2988	0.2932
DESCRIPTION	3725	0.3599	0.3607	0.2720	0.2568
ENTITY	631	0.3318	0.3461	0.2395	0.2471

To further understand the performance of the models for different types of queries, Table 6.3 shows the results achieved by the base and fine-tuned models for all queries in the full MS-MARCO development set, where the queries are grouped by their original types within the dataset (e.g., a PERSON query is a query for which the answer is a person). As expected, LOCATION queries are the ones where the fine-tuned models achieve the highest results, also showing the highest variation from the base models to the fine-tuned ones. For cross-encoders, the scores improve for all query types, except for a slight decrease in PERSON queries. For bi-encoders, the overfitting to the geo-spatial context is again noticeable, since the fine-tuned model only improves the scores for LOCATION and ENTITY queries.

Final Bi-Encoder Training Pipeline

As previously stated, the properties of bi-encoders make them computationally more efficient than cross-encoders. As such, a final objective was to try to apply the previous techniques into the training of a final bi-encoder model. The previous results can be summarized as follows:

- Both architectures benefit from the fine-tuning on the geographic domain;
- Bi-encoders can be improved by using a cross-encoder as the teacher model;
- Bi-encoders seem to benefit from more training data.

As such, the final training pipeline for a bi-encoder aggregated those strategies, as shown in Figure 6.2. Recall the best fine-tuned cross-encoder achieved during the previous experiments, which was obtained by sampling negatives using the geographic distance on top of the top-25 BM25 passages for each query on the original geographical train set of MS-MARCO. Following previous studies that have leveraged cross-encoders during the selection of negative examples [Ding et al., 2020, Thakur et al., 2020], the best fine-tuned cross-encoder was used to re-rank the candidate negative passages from BM25, prior to being selected by their geographical distance. These negatives were then used to fine-tune a bi-encoder. The cross-encoder was also used as the teacher model for knowledge distillation. For the training of the bi-encoder, the extended data was considered.

The results of this experiment are shown in Table 6.4. In the first group, the base bi-encoder and fine-tuned bi-encoder are shown again for comparison. The fine-tuned cross-encoder, which was the one used for sampling and distillation, is also included. Results are shown for using the cross-encoder for distillation only, sampling only, and the combination of both. When used separately, the sampling

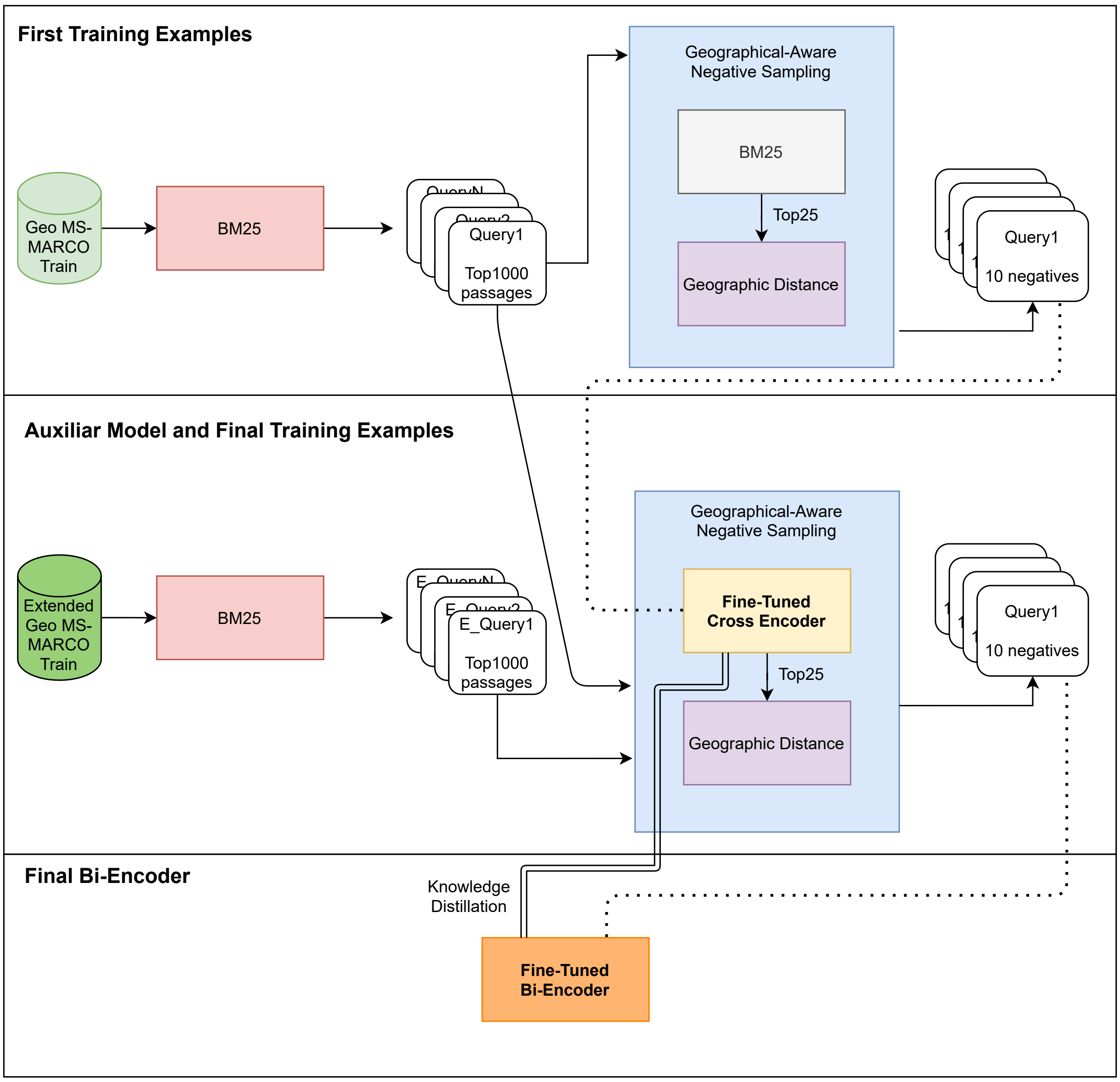


Figure 6.2: Final training pipeline for a bi-encoder.

Table 6.4: MRR@10 and R@{1,5,10,100, 500, 1000} for the geo-spatial subset of MS-MARCO, using the final bi-encoder, with previous results for comparison.

	MRR@10	R@1	R@5	R@10	R@100	R@500	R@1000
Base Bi-Encoder	0.4019	0.2631	0.5776	0.6792	0.8921	0.9366	0.9366
Fine-tuned Bi-Encoder	0.4208	0.2878	0.5982	0.6832	0.8973	0.9366	0.9366
Fine-tuned Cross-Encoder	0.5103	0.3607	0.6861	0.7968	0.9247	0.9366	0.9366
Distilled Bi-Encoder	0.4291	0.2997	0.5776	0.6809	0.9075	0.9366	0.9366
CE Sampling Bi-Encoder	0.4377	0.2973	0.6005	0.7055	0.9145	0.9366	0.9366
Distilled + CE Sampling Bi-Encoder	0.4454	0.3094	0.5936	0.7118	0.9229	0.9366	0.9366

technique achieves better results than the knowledge distillation method. However, the combination of both achieves the strongest bi-encoder on the geographical subset of MS-MARCO.

Table 6.5: Individual Reciprocal Rank (RR) scores for the positive passages associated to the 20 test queries with the highest difference in the scores obtained by the base and the fine-tuned models (averaged scores for cross-encoders and bi-encoders).

	BM25 Baselines		Cross-Encoders		Bi-Encoders	
Query	Text Alone	Geo. Re-Ranking	Base	Fine-Tuned	Base	Fine-Tuned
hot air balloon festival in maryland	1.0000	0.0000	0.3333	1.0000	0.0000	1.0000
benefits management fairport, ny	0.1429	0.1667	0.5000	1.0000	0.2500	1.0000
how much money will americans spend for easter	0.0000	0.1429	0.2000	1.0000	0.2000	0.5000
where is way st. binghamton	0.0000	0.0000	0.5000	1.0000	0.5000	1.0000
what happened in europe as a result of the cooling in climate that occurred in the early fourteenth century	0.5000	0.0000	0.5000	1.0000	0.5000	1.0000
what is the zip code for helena mt	0.1250	0.1250	0.2500	1.0000	0.3333	0.5000
wenatchee washington population	0.1111	0.1111	0.1667	1.0000	1.0000	1.0000
what do partnerships file tax in michigan	1.0000	1.0000	1.0000	1.0000	0.1667	1.0000
which county is greenwood indiana	0.5000	0.5000	0.1600	1.0000	0.5000	0.5000
honolulu chinese new year celebration	1.0000	1.0000	1.0000	1.0000	0.2000	1.0000
population of waukesha wisconsin	0.0000	0.0000	1.0000	1.0000	0.2000	1.0000
what town in kansas is home to boot hill	0.5000	0.5000	0.2500	0.3333	0.3333	1.0000
captain of israel's host	0.0000	0.0000	0.2500	0.5000	0.5000	1.0000
what year was the masstricht treaty	0.1667	0.2500	0.5000	0.5000	0.2500	1.0000
what is the population of perryville missouri	0.0000	0.0000	0.1429	0.3333	0.5000	1.0000
how far is it from chantilly va to baltimore	1.0000	1.0000	0.3333	1.0000	1.0000	1.0000
what is the current time in lagos nigeria	0.0000	0.0000	1.0000	1.0000	0.3333	1.0000
average gas costs in kentucky	0.0000	0.0000	0.5000	0.5000	0.3333	1.0000
driving distance littleton co to ft. collins co	0.0000	0.0000	0.5000	0.5000	0.3333	1.0000
what are the best plants for connecticut gardens	0.0000	0.1667	1.0000	1.0000	0.3333	1.0000

6.3 Qualitative Analysis

In an attempt to understand for which types of queries the fine-tuned models outperform the base ones, Table 6.5 presents the individual reciprocal rank scores for 20 example queries from the MS-MARCO development set, showing the results obtained with (a) the BM25 baseline, (b) BM25 complemented with the geo-spatial re-ranking heuristic, (c) cross-encoder models, and (d) bi-encoder models. The queries correspond to those with the highest difference in the reciprocal rank between the base models (i.e., average between the base cross-encoder and dual encoder) and the fine-tuned models. In all the example queries, the fine-tuned models performed equally or better than the base models, despite the fact that some of the queries express a relation between two locations (e.g., our models performed better on the query *how far is it from chantilly va to baltimore*, even though the proposed approach does not model spatial relations), and despite some errors in the text geoparsing step (e.g., in the case of the query *how much money will americans spend for easter*, the word *easter* was incorrectly recognized as a reference to Easter Island on both the query and on the relevant passage, and only the fine-tuned models could place the correct passage on the top result for this query).

To further understand the results, Table 6.6 presents 3 example queries and their relevant passages. For both the base and the fine-tuned cross-encoders, SHAP was used to assign shapely values to tokens, so as to depict which ones are contributing more for the ranking results. In the examples, both models are taking the geographic entities into consideration, which is probably why the base models already achieve strong results in the geographic subset. However, the fine-tuned model concentrates the attention in those entities, while the base model distributes attention to other tokens. Similar behavior

Table 6.6: Tokens that contribute to classification for the cross-encoders. The higher the shade of red, the higher the contribution.

	Query	Relevant Passage
Base Cross-Encoder	average winter temperature in kent co. delaware	Kent County has a moderate but distinct four-season climate. Average annual temperature: 55 Fahrenheit. January low average temperature: 26.1 Fahrenheit. July high average temperature: 87.2 Fahrenheit. The average annual rainfall is: 44.6 inches. The average annual snowfall is: 14.9 inches.
Fine-Tuned Cross-Encoder	average winter temperature in kent co. delaware	Kent County has a moderate but distinct four-season climate. Average annual temperature: 55 Fahrenheit. January low average temperature: 26.1 Fahrenheit. July high average temperature: 87.2 Fahrenheit. The average annual rainfall is: 44.6 inches. The average annual snowfall is: 14.9 inches.
Base Cross-Encoder	what county is lumberton, nc	Lumberton is a city in Robeson County, North Carolina, United States. The population has grown to 21,542 in the 2010 census from 20,795 in the 2000 census. It is the county seat of Robeson County, the largest county in the state. Lumberton, located in southern North Carolina's Inner Banks region, is located on the Lumber River.
Fine-Tuned Cross-Encoder	what county is lumberton, nc	Lumberton is a city in Robeson County, North Carolina, United States. The population has grown to 21,542 in the 2010 census from 20,795 in the 2000 census. It is the county seat of Robeson County, the largest county in the state. Lumberton, located in southern North Carolina's Inner Banks region, is located on the Lumber River
Base Cross-Encoder	what is prime rate in canada	What is the Prime Rate? In Canada, the prime rate is a guideline interest rate used by banks on loans for their most creditworthy, best, or prime clients. The prime rate rises and falls with the ebb and flow of the Canadian economy, influenced significantly by the overnight rate, which is set by the Bank of Canada.
Fine-Tuned Cross-Encoder	what is prime rate in canada	What is the Prime Rate? In canada, the prime rate is a guideline interest rate used by banks on loans for their most creditworthy, best, or prime clients. The prime rate rises and falls with the ebb and flow of the Canadian economy, influenced significantly by the overnight rate, which is set by the Bank of Canada.

Table 6.7: Tokens that contribute to classification for the bi-encoders. The higher the shade of red, the higher the contribution.

	Query	Relevant Passage
Base Bi-Encoder	what county is lumberton, nc	Lumberton is a city in Robeson County, North Carolina, United States. The population has grown to 21,542 in the 2010 census from 20,795 in the 2000 census. It is the county seat of Robeson County, the largest county in the state. Lumberton, located in southern North Carolina's Inner Banks region, is located on the Lumber River.
Final Bi-Encoder	what county is lumberton, nc	Lumberton is a city in Robeson County, North Carolina, United States. The population has grown to 21,542 in the 2010 census from 20,795 in the 2000 census. It is the county seat of Robeson County, the largest county in the state. Lumberton, located in southern North Carolina's Inner Banks region, is located on the Lumber River.

is present in the bi-encoder models. Table 6.7 shows one of the examples used for the cross-encoder, where the fine-tuned bi-encoder also gives more attention to the relevant toponym when compared to the base model.

Table 6.8: MRR@10 and R@{1,5,10,100, 500, 1000} for the geographic MS-MARCO development set, for the base models and the best fine-tuned versions.

	MRR@10	R@1	R@5	R@10	R@100	R@500	R@1000
Base Bi-Encoder	0.4019	0.2631	0.5776	0.6792	0.8921	0.9366	0.9366
Final Bi-Encoder	0.4454	0.3094	0.5936	0.7118	0.9229	0.9366	0.9366
Base Cross-Encoder	0.4959	0.3333	0.6964	0.8002	0.9281	0.9366	0.9366
Fine-tuned Cross-Encoder	0.5103	0.3607	0.6861	0.7968	0.9247	0.9366	0.9366

6.4 Overview

This chapter presented the experimental setup and results. The fine-tuned models were evaluated on both the geographic subset and full development set of MS-MARCO. Techniques to further improve the performance, such as knowledge distillation and data expansion were also evaluated. Then, those techniques were used in one final training pipeline to fine-tune a bi-encoder.

The results are summarized in Table 6.8, which shows the results for the base cross-encoder and bi-encoder models, and for the best versions achieved in this work. The best cross-encoder was achieved by fine-tuning the base model using the geographically-aware negative sampling procedure, while the best (i.e., the final) bi-encoder was the one which leveraged the aforementioned training pipeline.

Chapter 7

Conclusions and Future Work

This dissertation presented a fine-tuning setup for geographical passage retrieval, leveraging the geographic distance between entities in queries and passages. This chapter overviews the main contributions, and addresses directions for future work in this task.

7.1 Contributions

Geoparsed Data

In this work, the geographic subset of the MS-MARCO passage retrieval dataset was extracted, leveraging Mordecai, an open-source tool to geoparse textual information. Entities within queries and passages were identified, and a geographical subset containing training, validation and test sets was built. A characterization of this subset was conducted, by analysing its queries and entities.

Fine-tuning Methods

A fine-tuning setup for both cross-encoders and bi-encoders, for the task of geo-spatial passage re-ranking (i.e., retrieve relevant passages to questions involving place names) was proposed. The model fine-tuning setup focuses on batch construction through a geographically-aware hard negative sampling procedure. This procedure involves the usage of a notion of distance between queries and passages, given their geographical entities. The usage of the distance was motivated by examining the relation between the distance from a query and a passage, and the relevance of said passage to the query. Other techniques were explored, namely a cross-architecture knowledge distillation method based on the Spearman rank correlation coefficient, and the usage of extended training data.

Results

Experiments showed that both fine-tuned bi-encoders and cross-encoders benefit from the fine-tuning for the geographic context. Also, for cross-encoders, the geographical fine-tuning did not compromise the ability of the model of re-ranking passages to other types of questions. After evaluating all the employed techniques, a final bi-encoder was trained following a pipeline which leveraged data augmentation, and fine-tuned a cross-encoder for negative sampling and knowledge distillation. This model achieved the highest results of all bi-encoders on the geographic subset of MS-MARCO.

Availability

The code that supports the experiments described in this manuscript, along with the geoparsed datasets (i.e., including the labels for geographic entities) and fine-tuned models, were made publicly available.

7.2 Future Work

Besides MS-MARCO, the same experiments can be replicated on other benchmark datasets, for instance NaturalQuestions, which has around 300 thousand queries, along with a collection of over 5 million passages. Data from GeoCLEF can be used to evaluate the models on pure geographical data. However, the GeoCLEF task is document retrieval, instead of passage retrieval. As such, given the size mismatch, other techniques need to be taken into account, for instance the usage of the Longformer architecture [Beltagy et al., 2020].

Also, other types of data augmentation can be considered. For instance, instead of generating queries, hard negative examples for a given query could be generated by entity replacement on the query's relevant passage. Preliminary experiments have been conducted on this subject, by replacing some given geographical entity with another from the same administrative region. Further work is needed, since results showed that the substitute entities generated by the tested method were noisy.

As for the architectures, other efficient and well-performing retrieval strategies, different from the standard bi-encoders, can also be considered. This includes approaches such as SparTerm [Bai et al., 2020] or ColBERT [Khattab and Zaharia, 2020].

Regarding model training, other techniques can be introduced. For instance, PAIR [Ren et al., 2021] leverages a combination of query-centric and passage-centric similarity (i.e., instead of considering only the similarity between queries and passages, the similarity between positive and negative passages within a batch is also used to guide the loss). ADORE and SMILE [Zhan et al., 2021] are two training techniques that focus on employing random negatives to stabilize the training, and using dynamic hard negatives (i.e., use the bi-encoder under training to sample negatives at any given training step). Also, instead of the standard binary cross-entropy, triplet-based losses can be considered, following a dynamic margin which changes its value based on the geographic distance between the passages and the query. Preliminary experiments on this subject were conducted, but further testing is needed.

Finally, while this work focused only on the geographical domain, the same negative sampling procedure can be adapted for different domains. For example, temporal questions (e.g., queries focusing on when a given event has taken place) also involve an inherent distance between temporal entities [Wang et al., 2021b]. By using parsers that identify temporal entities within text (e.g., SUTime resolves them to calendar dates), the work here described can be replicated for this domain.

Bibliography

Y. Bai, X. Li, G. Wang, C. Zhang, L. Shang, J. Xu, Z. Wang, F. Wang, and Q. Liu. Sparterm: Learning term-based sparse representation for fast text retrieval. *ArXiv*, abs/2010.00768, 2020.

I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The Long-Document Transformer. *ArXiv*, abs/2004.05150, 2020.

M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. Fast differentiable sorting and ranking. In *Proceedings of the International Conference on Machine Learning*, 2020.

D. F. Campos, T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, and B. Mitra. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation co-located with the Annual Conference on Neural Information Processing Systems*, 2016.

A. Cardoso, B. Martins, and J. Estima. Using recurrent neural networks for toponym resolution in text. In *Proceedings of the EPIA Conference on Artificial Intelligence*, 2019.

N. Cardoso, B. Martins, M. Chaves, L. Andrade, and M. J. Silva. The XLDB group at GeoCLEF 2005. In *Proceedings of the International Conference on Cross-Language Evalution Forum*, 2005.

K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *Proceedings of the International Conference on Learning Representations*, 2020.

D. Contractor, S. Goel, Mausam, and P. Singla. Joint Spatio-Textual Reasoning for Answering Tourism Questions. *ArXiv*, abs/2009.13613, 2020.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Y. Q. Y. Ding, J. Liu, K. Liu, R. Ren, X. Zhao, D. Dong, H. Wu, and H. Wang. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *ArXiv*, abs/2010.08191, 2020.

C. Haas and S. Riezler. A corpus and semantic parser for multilingual natural language querying of openstreetmap. 2016.

A. Halterman. Mordecai: Full Text Geoparsing and Event Geocoding. *The Journal of Open Source Software*, 2(9), 2017.

E. Hamzei, S. Winter, and M. Tomko. Initial Analysis of Simple Where-Questions and Human-Generated Answers. In *Proceedings of the International Conference on Spatial Information Theory*, 2019.

E. Hamzei, S. Winter, and M. Tomko. Templates of generic geographic information for answering where-questions. *International Journal of Geographical Information Science*, 2021.

S. Han, X. Wang, M. Bendersky, and M. Najork. Learning-to-Rank with BERT in TF-Ranking. *ArXiv*, abs/2004.08476, 2020.

S. Hofstätter, S. Althammer, M. Schröder, M. Sertkan, and A. Hanbury. Improving efficient neural ranking models with cross-architecture knowledge distillation. *ArXiv*, abs/2010.02666, 2020.

S. Hofstätter, S. Lin, J. Yang, J. Lin, and A. Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling. *ArXiv*, abs/2104.06967, 2021.

S. Hofstätter, M. Zlabinger, and A. Hanbury. TU Wien @ TREC Deep Learning '19 – Simple Contextualization for Re-ranking. In *Proceedings of the Text REtrieval Conference*, 2019.

X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. Tinybert: Distilling BERT for natural language understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020.

T. Joachims. Optimizing Search Engines using Clickthrough Data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *ArXiv*, abs/1702.08734, 2017.

O. Khattab and M. Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

S. Kulkarni, S. Jain, M. J. Hosseini, J. Baldridge, E. Ie, and L. Zhang. Spatial language representation with multi-level geocoding. *ArXiv*, arXiv:2008.09236, 2020.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A lite BERT for Self-supervised Learning of Language Representations. In *Proceedings of the International Conference on Learning Representations*, 2020.

C. Lawrence and S. Riezler. Nlmaps: A natural language interface to query openstreetmap. 2016.

H. Li, E. Hamzei, I. Majic, H. Hua, J. Renz, M. Tomko, M. Vasardani, S. Winter, and T. Baldwin. Neural Geospatial Question Answering. *Journal of Spatial Information Science*, Under Review, 2020.

J. Lin, R. Nogueira, and A. Yates. Pretrained Transformers for Text Ranking: BERT and Beyond. *ArXiv*, abs/2010.06467, 2020.

J. Lin, X. Ma, S. Lin, J. Yang, R. Pradeep, and R. Nogueira. Pyserini: An easy-to-use python toolkit to support replicable IR research with sparse and dense representations. *ArXiv*, abs/2102.10073, 2021.

Q. Liu, M. J. Kusner, and P. Blunsom. A Survey on Contextual Embeddings. *ArXiv*, abs/2003.07278, 2020.

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. *ArXiv*, abs/1907.11692, 2019.

Y. Lv and C. Zhai. Lower-Bounding Term Frequency Normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 2011.

G. Mai, K. Janowicz, R. Zhu, L. Cai, and N. Lao. Geographic question answering: Challenges, uniqueness, classification, and future directions. *ArXiv*, abs/2105.09392, 2021.

T. Mandl, P. Carvalho, G. M. D. Nunzio, F. C. Gey, R. R. Larson, D. Santos, and C. Womser-Hacker. Geoclef 2008: The CLEF 2008 cross-language geographic information retrieval track overview. In *Proceedings of the Workshop of the Cross-Language Evaluation Forum*, 2008.

B. Martins and P. Calado. Learning to Rank for Geographic Information Retrieval. In *Proceedings of the Workshop on Geographic Information Retrieval*, 2010.

T. Mikolov, G. Corrado, K. Chen, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2013b.

B. Mitra, S. Hofstätter, H. Zamani, and N. Craswell. Conformer-Kernel with Query Term Independence for Document Retrieval. *ArXiv*, abs/2007.10434, 2020.

R. Nogueira. From doc2query to docTTTTTquery. Technical report, University of Waterloo, 2019.

R. Nogueira and K. Cho. Passage Re-ranking with BERT. *ArXiv*, abs/1901.04085, 2019.

R. Nogueira, W. Yang, J. Lin, and K. Cho. Document Expansion by Query Prediction. *ArXiv*, abs/1904.08375, 2019.

R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020.

R. K. Pasumarthi, S. Bruch, X. Wang, C. Li, M. Bendersky, M. Najork, J. Pfeifer, N. Golbandi, R. Anil, and S. Wolf. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

D. Punjani, K. Singh, A. Both, M. Koubarakis, I. Angelidis, K. Bereta, T. Beris, D. Bilidas, T. Ioannidis, N. Karalis, et al. Template-based question answering over linked geospatial data. In *Proceedings of the ACM Workshop on Geographic Information Retrieval*, 2018.

R. S. Purves, P. D. Clough, C. B. Jones, M. M. Hall, and V. Murdock. Geographic information retrieval: Progress and challenges in spatial search of text. *Foundations and Trends in Information Retrieval*, 12(2-3), 2018.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140), 2020.

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*, 2019.

N. Reimers and I. Gurevych. The curse of dense low-dimensional information retrieval for large index sizes. *ArXiv*, abs/2012.14210, 2020.

R. Ren, S. Lv, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, and J. Wen. PAIR: leveraging passage-centric similarity relation for improving dense passage retrieval. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP*, 2021.

S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 2009.

F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, 65(6), 1958.

G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 1975.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

N. A. Smith. Contextual Word Representations: A Contextual Introduction. *ArXiv*, abs/1902.06006, 2020.

N. Thakur, N. Reimers, J. Daxenberger, and I. Gurevych. Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks. *ArXiv*, abs/2010.08240, 2020.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is All you Need. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2017.

J. Wang, A. Jatowt, M. Färber, and M. Yoshikawa. Improving question answering for event-focused questions in temporal collections of news articles. *Information Retrieval Journal*, 2021a.

J. Wang, A. Jatowt, and M. Yoshikawa. ArchivalQA: A Large-scale Benchmark Dataset for Open Domain Question Answering over Archival News Collections. *ArXiv*, abs/2109.03438, 2021b.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020.

M. Wrzalik and D. Krechel. CoRT: Complementary Rankings from Transformers. *ArXiv*, abs/2010.10252, 2020.

C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling Matters in Deep Embedding Learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. End-to-End Neural Ad-Hoc Ranking with Kernel Pooling. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.

H. Xu, E. Hamzei, E. Nyamsuren, H. Kruiger, S. Winter, M. Tomko, and S. Simon. Extracting interrogative intents and concepts from geo-analytic questions. In *Proceedings of the AGILE conference on Geographic Information Science*, 2020.

J.-Y. Yeh, J. Y. Lin, H.-R. Ke, and W.-P. Yang. Learning to rank for information retrieval using genetic programming. In *Proceedings of ACM SIGIR Workshop on Learning to Rank for Information Retrieval*, 2012.

Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007.

H. Zaragoza, N. Craswell, M. J. Taylor, S. Saria, and S. E. Robertson. Microsoft Cambridge at TREC 13: Web and Hard Tracks. In *Proceedings of the Thirteenth Text REtrieval Conference*, 2004.

J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, and S. Ma. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.