COMPARING THE REPRESENTATION LEARNING OF AUTOENCODING TRANSFORMER MODELS IN AD HOC INFORMATION RETRIEVAL

by

Jeniffer David

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2020

To my loving father

Table of Contents

Abstract

Information retrieval (IR) saw a recent development in ranking models since the advent of deep learning techniques. Traditionally, classical IR methods, such as BM25, assume query term independence, allowing them to precompute term-document scores which makes them efficient for full-ranking. On the other hand, deep neural ranking models, like BERT, depend on interaction signals between query and document terms for successful retrieval, therefore being restricted to only late stage re-ranking, even though they have superior retrieval performance. Recent work has shown that with offline precomputation of the sentence embeddings, these computation-intensive models can be used for full-ranking and made cost-effective when combined with any indexing structure. However, BERT is not the only advanced language representation model that can be used for information retrieval. Various other models such as RoBERTa, ALBERT, DistilBERT, ELECTRA and many others have surpassed the performance of BERT in several NLP tasks. Since a large number of pre-trained language models have been proposed lately, we believe it is the right time to evaluate their representational learning for ranking. Although these pre-trained models share some fundamental characteristics, their performance varies because of differences in training data, or training procedure. They also differ in their computational requirements. In this work, we evaluate the representational learning of the various autoencoding transformer models extrinsically on the downstream task of Microsoft MAchine Reading Comprehension (MS MARCO) passage retrieval. We observe that BERT and its distilled version DistilBERT are the best performers in terms of ranking, while DistilBERT achieves a good trade-off between effectiveness and computational efficiency in ad hoc document retrieval. We discuss the empirical analysis of these models and provide insights about their performance in tasks like semantic similarity. We believe that our results shed some light on the selection of embeddings for ad hoc retrieval and also serves as a benchmark for future search applications.

Acknowledgements

My sincerest gratitude to my supervisor, Dr. Evangelos Milios for his mentorship and guidance during the course of my thesis.

Special thanks to my colleagues in MALNIS who have inspired and encouraged me in all of my academic endeavours.

Thanks to all my friends and family for believing in me and their continuous support.

Chapter 1

Introduction

In recent times, the information retrieval (IR) community has benefited tremendously from advancements in natural language processing (NLP) techniques. One particular trend is the switch from traditional IR techniques to neural ranking models for ad hoc information retrieval. In ad hoc retrieval task, the user specifies the information he/she seeks in the form of a query to an information system which initiates a search for the most relevant documents within a database in response to the query. The key component of a search system is its ranking algorithm. Different ranking algorithms have different models of relevance. Traditional IR models use term-frequency to compute relevance between each query and document where the issue is lack of semantics [14], whereas Learning-to-rank requires hand-crafted IR features to perform supervised machine learning [43]. Since the re-emergence of neural networks in the machine learning (ML) community, the focus has now shifted to the development of neural ranking models for information retrieval. A neural ranking model is any model that uses shallow or deep neural networks to rank search results for a given query. The main setback of such models is that they are data-hungry and require thousands (sometimes millions) of examples, to train effectively. Fortunately, with the introduction of large-scale IR datasets like MS MARCO [46] and TREC-CAR [13, 12, 11], development of these models flourished rapidly and yielded promising results. Today, deep neural ranking models are topping the IR leaderboards with their state-of-the-art performance mainly because of their ability to handle the complexity of relevance estimation in ranking [21].

Search applications in general consist of large multi-tier architectures for retrieving relevant documents. The first layer may need to filter hundreds or thousands of relevant documents out of a million, while the last layer may only need to retrieve the top-10 of these hundred candidate documents [43]. The layer responsible for the initial ranking needs to be fast and focus primarily on eliminating a vast majority of the non-relevant documents. The layer that does the re-ranking of the initial results must have sophisticated notions of relevance in order to produce more fine-grained results. Many end-to-end systems follow this approach to ad hoc retrieval where TF-IDF based methods like BM25 are used for fast and exact-term matching, while neural networks are limited to re-ranking the top-n documents. The drawback of this approach is that the re-ranker can only be as good as the initial retrieval model. And so, using standard bag-of-words technique in the first layer might miss semantically relevant documents in the initial retrieval. Hence, an initial retrieval algorithm should contain some latent representation of intent that is expressed by the user's query.

Deep neural ranking model architectures are generally representation-focused or interaction-focused [20]. Representation-focused models produce high-level representations of the query and document texts, and use simple evaluation function (like cosine similarity) to produce relevance score [27]. On the other hand, relevance in interaction-focused models is about the detailed relation between the query and document texts. It is a function of the direct interaction between the query and document representations where complex functions (like deep neural networks) are used to produce the relevance score (give an example using BERT attention). Recently, BERT [10], the pre-trained deep bidirectional Transformer, was re-purposed for query-based passage re-ranking [47] beating other deep neural ranking models by a huge margin. These BERT-based models [61, 47] use attention as the interaction function to learn the interaction signals (i.e., [CLS] vector) between input texts. Although the interaction signals by BERT's attention mechanism can be effective for ad hoc retrieval, they are time-consuming during inference in real-world search systems. And so, the representation-based approach of BERT can be used to generate faster inferences while providing rich contextual representations of the query and document texts [40].

Given the successes of BERT in information retrieval and in general machine comprehension tasks [8, 61], we explore the possibilities of using other such powerful language models that are pre-trained on large datasets and made publicly available. One such family of pre-trained language models are called autoencoding transformer models because they use only the encoder part of the original transformer [58]. Some of these models are BERT [10], ALBERT [34], RoBERTa [39], DistilBERT [56], ELECTRA [5]. If the representational learning of BERT can be powerful enough for ad hoc retrieval, then we hypothesize that every model that came after BERT must be as powerful or even more powerful than BERT since they surpass the performance of BERT in benchmark tasks like GLUE [59], RACE [33] and SQuAD [53, 52].

In our work, we use a simple triplet network architecture to produce fixed-size embeddings of query and document text [54] and compute relevance scores for every query-document pair based on cosine similarity of their embeddings. Fine-tuning the models on the ranking task is done prior to inference. For applications like online search, it is required to have the document embeddings pre-computed and stored in an index for faster retrieval. This can be efficiently done by tools like FAISS [30] that come with special in-memory data structure and indexing schemes. Our experiments demonstrate the overall performance of the different autoencoding transformer models in an end-to-end search system for the MS MARCO passage retrieval task[1], which is an ad hoc retrieval task for passage-level documents. We also analyse these models in terms of their inference speed and memory use, answer-type and semantic similarity. Our experiments show that the differences between most pairs of models are statistically significant, with the clear overall winner being BERT. BERT when fine-tuned on the ranking task proves to be the most effective for passage retrieval, but DistilBERT achieves the fastest results while being second-best. Hence we conclude that the embeddings of DistilBERT are the most efficient for passage retrieval task on both CPU and GPU hardwares.

1.1 Contributions

This thesis focuses on fine-tuning various autoencoding transformer models to produce high-quality query/passage embeddings for the passage ranking task in a simple end-to-end retrieval system. We are interested in understanding which model performs empirically better than its contenders for the given architecture. We summarize our contributions as below:

- Evaluate pre-trained language models in the Transformer family of BERT after

[1]`https://github.com/microsoft/MSMARCO-Passage-Ranking`

fine-tuning for representation-focused learning on MS MARCO passage retrieval task.

- Analyse the overall ranking performance on the MS MARCO development set and compare the inference speed and memory consumed for each step by the different models.
- Provide insights on their performance for specific answer types and the search results on semantic similarity.

We conclude that BERT and DistilBERT outperforms all the other models in a representation-learning setup for the ad hoc retrieval task, while there is no statistical significance between BERT and DistilBERT in most cases. But more importantly, we find that DistilBERT strikes the best balance between accuracy and speed/memory in practice. Finally, we hypothesize based on our results that a promising direction for future neural information retrieval research would be knowledge distillation of large pre-trained language models for computational efficiency.

Chapter 2

Related Work

2.1 Background

This section provides some theoretical background on the main focus of this study which is ad hoc document retrieval in IR. A document retrieval task aims at returning the relevant documents to a user's query. The exact answer to the user's query can be extracted from these relevant documents through a question-answering (QA) system. Any intelligent agent that deals with real-world data for answering questions must be able to read and understand text like a human [63]. Therefore, a fundamental ability of any successful IR model is machine reading comprehension (MRC). Several public question-answering (QA) datasets have facilitated research in this area and therefore made deep learning possible. However, these large datasets are mostly synthetic and non-realistic for benchmarking MRC models. With the release of large scale real-world dataset like MS MARCO [46], we can fine-tune large pre-trained language models using actual search queries. Our study analyses the embeddings of different pretrained language models for the purpose of ranking passage-level documents and in doing so, extending the application of these fine-tuned models in real-world search systems.

2.2 Document and Passage Retrieval

In ad hoc document retrieval, the user specifies the information he/she seeks in the form of a short query q where the query is a set of keywords and the task is to produce the best ranking of documents in a corpus based on a relevance function. In general, a document may refer to any piece of text being retrieved, usually a passage, a sentence, sometimes even several sentences or paragraphs. This step is crucial in any QA system because the underlying modules will fail at extracting the right answer if the retrieved documents are not relevant to the question asked by the user [26]. Since the first QA systems like Baseball [18], several studies have presented the challenges in benchmarking QA systems and problems of passage retrieval [57, 41, 60, 15].

Research in this area was catalysed by the Text REtrieval Conference (TREC), co-sponsored by the National Institute of Standards and Technology (NIST) and U.S. Department of Defense. It started in 1992 as part of the TIPSTER Text program attended by about 100 people working in 25 participating groups [22]. The main purpose of this conference was to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies[1].

In the past, the most popular approaches were traditional IR models like TF-IDF that performed exact-term matching between the query and document terms, e.g., BM25 [55], RM3 [29] and QL [42]. These approaches estimate document relevance based on only the number of occurrences of the query terms in the document and so, are only fit for tasks that require specific matching patterns like exact word matching. Information like position and relationship with other terms in the document are not taken into account. Moreover, the concept of relevance can be hard to capture by these simple models because relevance is multidimensional and dynamic in nature [4]. More recently, Learning to rank (L2R) approaches were used to enhance IR technologies because many IR problems are by nature ranking problems [38, 36]. These algorithms are categorized into 3 approaches based on their training objectives: pointwise, pairwise, and listwise. Although these models were the industry favorite for many years and were a commercial success, their drawback was that they required handcrafted features to be trained meaningfully. For a more detailed overview of these traditional IR models, we refer the readers to review the work by Mitra & Craswell [43].

With the advent of deep neural networks, many exciting breakthroughs have happened in fields like speech recognition [23], computer vision [32, 35], and natural language processing (NLP) [17, 3]. Some notable developments in the last few years have shown that deep neural network models have a strong influence on the area of IR as well [6, 27, 25, 49]. The deep neural models can be trained to learn sophisticated notions of relevance that are often difficult to model by shallow or traditional

[1]`https://trec.nist.gov/overview.html`

models. Another survey summarizes the latest research trends on ranking models with deep neural networks describing the different types of model architecture and learning [20]. In general, some of the most effective approaches to improve ranking accuracy developed by the neural IR (Neu-IR) research are: Interaction-focused architectures that learn to capture rich meaningful matching patterns or salient interaction signals between the input texts in a layer-by-layer fashion [19, 60, 25, 47]; Representation-focused architectures focus more on finding high-level representations of the input texts before allowing the query & document representations to interact with each other to compute the relevance [27, 25, 62]; Hybrid architectures combine the advantages of both representation-focused and interaction-focused architectures for feature learning [44]. The key difference is that interaction between the query and document terms happens early on in interaction-focused models and continues to model relevance as a function of interaction whereas a representation-focused model only allows query-document interaction at a much later stage, usually after the maturity of its representations. Although interaction-focused models are better suited for ad hoc retrieval, they are not efficient for online-computation when compared to the representation-focused models. Hence, they are usually employed in a "telescope" setting where the representation-focused models are used in the early search stage and the interaction-focused models for late-stage re-ranking.

In TREC 2019, a new track called Deep Learning Track was introduced with the goal of studying ad hoc ranking in a large data regime with two tasks namely, passage ranking and document ranking. A total of 75 runs submitted by 15 groups using various combinations of deep learning, transfer learning and traditional IR ranking methods showed that deep learning runs significantly outperformed traditional IR runs [7]. Out of all the neural approaches, it was found that the best-performing runs tended to use transfer learning, employing a pretrained language model such as BERT.

2.3 Pretrained Language Models in Ranking

For a few years now, several computer vision tasks have benefited from pretraining the deep neural models on large ImageNet corpus [9]. These models learn general image features which can help them in solving any downstream vision task (e.g.

captioning, detection) through transfer learning [28]. The same idea is used in NLP where language models with millions of parameters are pretrained on large corpus like Wikipedia, and later fine-tuned on downstream tasks (like classification, sentiment analysis, question-answering, named entity recognition, paraphrasing) to achieve state-of-the-art performance using two existing strategies: feature-based [50, 24] and fine-tuning [51, 10].

One such powerful pretrained language model is BERT, which stands for Bidirectional Encoder Representations from Transformers [10]. The main motivation of the authors was to improve the fine-tuning approach of previous pretrained language models (OpenAI GPT [51]), arguing that the unidirectional architecture of standard language models is not suitable for token-level tasks like question-answering where context from both direction is important. In the unidirectional architectures, every token can only attend to tokens before it in the self-attention layers of the Transformer [58], going either from left-to-right or right-to-left. To overcome this problem, they proposed a deep bidirectional Transformer architecture which uses "masked language model" (MLM) pre-training objective that randomly masks 15% words of a sentence and predicts these missing words. Additionally, they used “next sentence prediction” (NSP) task for pre-training on BooksCorpus (800M words) [64] and English Wikipedia (2,500M words) and Google’s compute power. The base version of BERT used in this work has 110 million parameters and 12 Transformer layers, each with 768 hidden dimensions and 12 attention heads. The novel pre-training tasks along with huge computation power unlike any other Neu-IR models are some of BERT’s notable properties useful for ranking. Many researchers have fine-tuned and extended the BERT model on the MS MARCO dataset demonstrating improved performance. Here, feature-based approach in language modelling is synonymous to representation-based approach in IR models, fine-tuning approach is similar to interaction-based rankers.

Currently, many of the submissions on the MS MARCO passage ranking leaderboard[2] are BERT-based models. One of the early implementations used the fine-tuning approach to capture the cross-match attentions between the query and document terms, using BERT as an interaction-based re-ranker [47], shown in Fig. 2.1.

[2]`https://microsoft.github.io/msmarco/`

This secured them a top place in the leaderboard beating other deep neural ranking models at that time. Following that, another work extended the application to ad hoc document retrieval where the documents are longer passages, typically containing several sentences [61]. The popularity of BERT in passage re-ranking prompted the investigation of its successes and failures on the MS MARCO dataset comparing its results to that of BM25 [48]. Many of the fine-tuned BERT models in these work are only used at the re-ranking step during inference [2, 1].

The feature-based approach of BERT was explored in the Sentence-BERT model who showed that the computational overhead of BERT can be greatly minimized for regression tasks like semantic textual similarity (STS) while maintaining its accuracy [54], shown in Fig. 2.1. They used siamese and triplet network architectures to derive semantically meaningful embeddings and used these representations to compute the similarity between sentences. Our work employs the same feature-based technique to derive meaningful embeddings of the query and document to compute the relevance score. We use triplet network structure since the MS MARCO training dataset is available in triples of query, relevant document and non-relevant document, and fine-tune on the ranking task. Another work showed that there is only a small degradation in BERT's performance when term-document scores are pre-computed for passage ranking task by assuming query-term independence during inference [45].

Today there are more than 30 pretrained models provided in the Hugging Face library[3], each of them improving upon the other in one or more NLP tasks and many of them using some form of the original Transformer architecture [58]. A natural question to ask is what happens when BERT is substituted by any of these (preferably newer) models? Can we expect an increase in performance due to increased sophistication of the latest approaches? Are the performances significantly different? Are all of them better than baseline models like BM25 in terms of IR metrics? To answer these questions, we select 4 pretrained models, namely RoBERTa [39], ALBERT [34], DistilBERT [56] and ELECTRA [5], that were introduced after BERT and also belong to the same Transformer family as BERT, that is, Autoencoding Transformers[4]. We demonstrate the ranking performance of these models on the MS MARCO passage ranking task and also analyse them on different answer types and

[3]`https://huggingface.co/transformers/pretrained_models.html`

[4]`https://huggingface.co/transformers/model_summary.html#autoencoding-models`

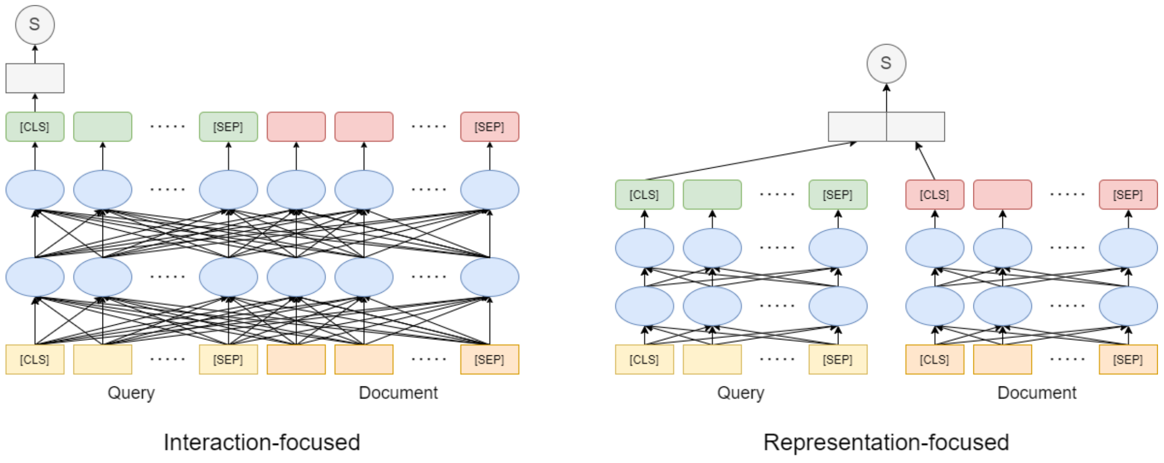

Figure 2.1: The two general neu-IR architectures in the context of BERT. In interaction-focused BERT, the query and document are concatenated first and passed through the layers of BERT for similarity score (S) computation. In representation-focused BERT, the query and document are compared after passing through the BERT layers to compute similarity.

semantic similarity.

Some recent notable works that were developed in parallel to our research have made promising progress in modelling query-document relevance patterns using BERT. One such work augments BERT with more specific search knowledge tailored for those search tasks with limited labeled data by tuning it on a large search log [8]. Other implementations encode queries and passages using BERT and combine it with a learning-to-rank (LTR) model constructed with TF-Ranking in order to further improve ranking performances [21]. Results in these papers are promising because they show that the abilities of BERT in the neu-IR field are numerous and lessons from their experiments can be used to improve existing online-search systems. Another work that supports the conclusions of this work and possibly showing more promise in this direction is using knowledge distillation to transfer the search knowledge within BERT to smaller rankers while being nine times faster [16]. We hope the results of our experiments will throw some light on the usage of these complex pretrained models in real-world search systems.

Chapter 3

Methodology

Ranking for any search problem deals with finding and ordering the most relevant documents for a query entered by the user. Although our work can be generalized to documents of any size, we have only experimented with passage-size documents. Irrespective of its length, a document is considered to be relevant if at least one of its passages semantically matches the query. Hence, for ad hoc information retrieval task, it is enough to compare only the passages of the documents with the query to find the most similar document. A high-level architecture of a simple search system is shown in Figure 3.1.

Our work implements full-ranking of passage-level documents for a given query using deep pre-trained language models by leveraging their ability to generate deep contextualized embeddings of any input text. By modelling relevance based on semantics, we achieve high-recall information retrieval useful for later stages. By pre-computing the embeddings of the documents, we reduce the time taken to compute the embeddings of all documents at real-time for comparing with the query embedding. The results of our first-stage full-ranking can be used for further filtering in later stages. We leave the re-ranking for future research.

3.1 Autoencoding Transformer Models for Passage Ranking

The original Transformer has an encoder and decoder part using a stack of several self-attention and fully connected layers [58]. Self-attention is the mechanism of assigning an attention weight/score to a token corresponding to every other token in the sequence. The output vector of each token is basically a weighted sum of its attention values. In Transformers, this is done using multiple attention heads where each head is responsible for calculating attention at different positions. The overall network architecture of the Transformer encoder is shown in Fig. 3.2.

In our work, we compare the models that rely on only the encoder part of the

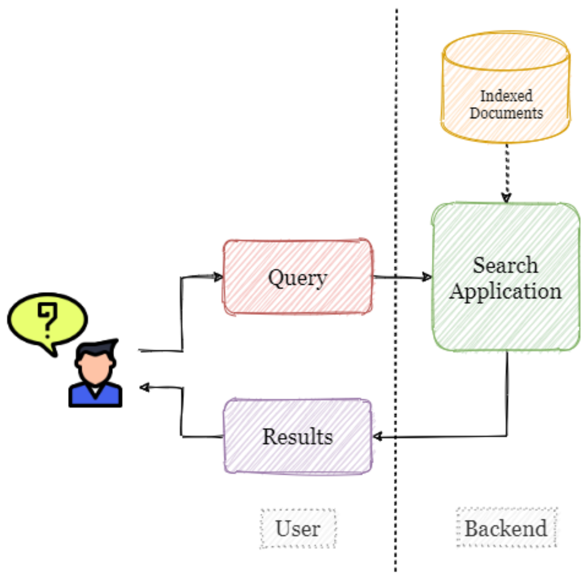

Figure 3.1: High-Level architecture of a typical ad hoc search. When a user enters a query, the search application compares it with an index containing a large but finite collection of document embeddings and stored in the backend. The results of the search, which is usually a ranked-list of documents in decreasing order of relevance to the query, is returned to the user.

original transformer. They do not mask the input tokens and so the attention heads can look at all the tokens during prediction. However, during pretraining, inputs are a corrupted version of the sentence and the task is to reconstruct the original sentence. These models are named the autoencoding transformers as per the Hugging Face library. For timing and computational reasons, we compare only the BERT, RoBERTa, ALBERT, DistilBERT and ELECTRA models, and leave the others for future work. In Fig. 3.3, a breakdown of an online search system shows the application of an autoencoding transformer at the backend for extracting the query embedding. The derivation of the embeddings and the similarity score computation is explained in detail in the sections below.

3.2 Model Architecture

We follow the triplet network architecture of Sentence-Transformer for deriving the query and document embeddings [54]. Each model is initialized with weights pre-trained on one or more generic language modelling tasks (like Masked Language Model, Next Sequence Prediction, Sentence Order Prediction and Replace Token

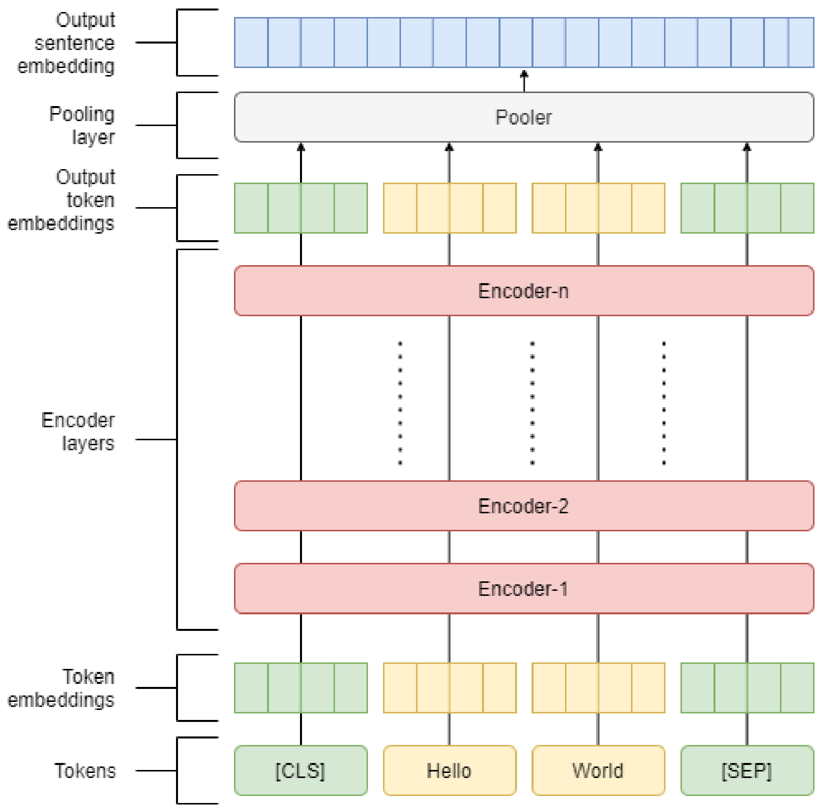

Figure 3.2: Overall network architecture of a Transformer encoder. After tokenizing the input sentence, the tokens are passed through the encoder layers (n>6), followed by a pooling layer to generate the contextualized output sentence embedding.

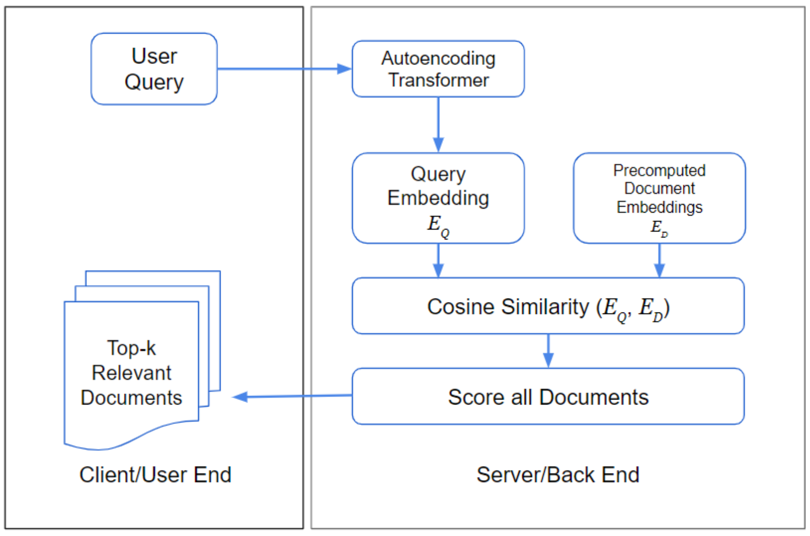

Figure 3.3: Online similarity score computation for ranking and retrieval. The embedding of the query is computed at server time and the embeddings of the documents are precomputed offline. The cosine similarity score of the query and document embeddings is used to retrieve the final top-k relevant documents to the user.

Detection) and later, fine-tuned on the downstream triplet ranking task. We explain the training (i.e. fine-tuning) and the inference (indexing and searching) in detail below for all models.

3.3 Fine-tuning

We fine-tune the autoencoding transformer models on the ranking task. Each Transformer model generates its own token embeddings based on its vocabulary. Each query and passage sequence is fed individually as sentence A and truncated if the number of tokens exceeds the maximum sequence length. The positive passages are those that are marked as relevant by human annotators for the given query while the negative passages are any of the non-relevant passages chosen at random from the collection. For every sequence, a special classification token ([CLS]) is appended at the beginning and another special token ([SEP]) is appended at the end to denote the end of sentence A. The output vector of the last layer of every model is considered to be the contextual token embeddings of that sequence. We derive fixed-size sentence embeddings from these variable length token embeddings using the pooling strategies mentioned in the Sentence-transformer paper [54]. The three pooling strategies are: Using the final hidden state of the [CLS] token, computing the average of all output token embeddings (AVG strategy), and computing the maximum value in each dimension across all output token embeddings (MAX-strategy). Finally, the models are trained on the Triplet objective function to learn meaningful representations of the input text as shown in Fig. 3.4. The models train to rank relevant documents higher than non-relevant documents by learning suitable embeddings that maximize the distance between a query and a non-relevant passage and minimize the distance between the query and the relevant passage.

Triplet Objective Function. Given a query q, a positive passage p, and a negative passage n, our triplet loss fine-tunes the network such that the distance between q and p is smaller than the distance between q and n. We minimize the following objective function:

$$\max\left(cos\left(e_q, e_p\right) - cos\left(e_q, e_n\right) + \epsilon, 0\right) \tag{3.1}$$

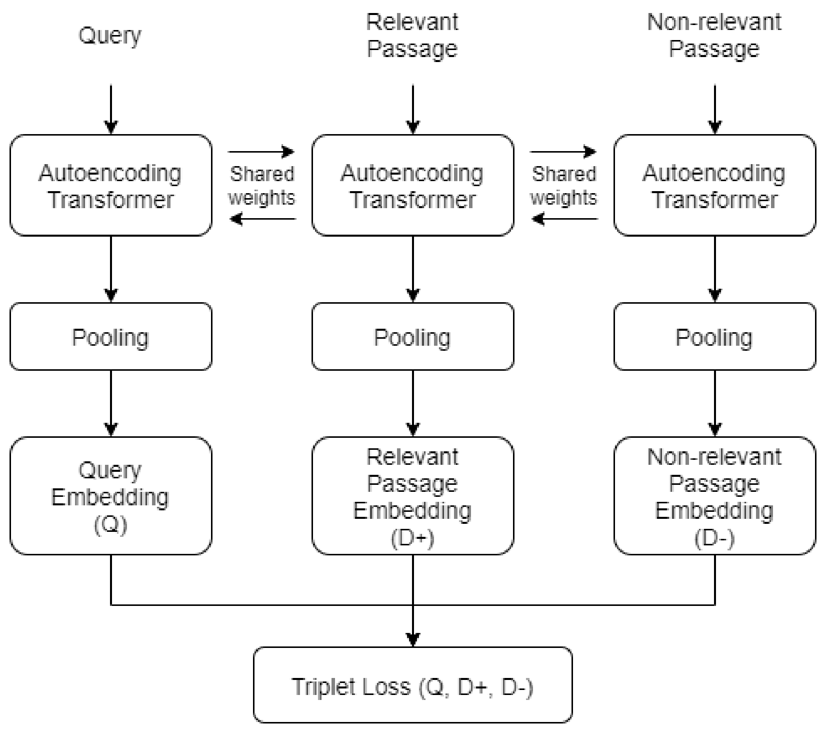

Figure 3.4: Fine-tuning the autoencoding transformer of Figure 3.3. The pretrained language models are fine-tuned on the ranking task using triplet network architecture to produce meaningful embeddings of the given query, relevant and non-relevant documents. The network is trained to minimize the distance between the query embedding (Q) and the relevant document embedding (D+) and maximize the distance between the query and the non-relevant document embedding (D-).

where e_x is the embedding for $x \in \{q, n, p\}$, $cos(*)$ is the cosine-distance metric and ϵ is the margin between positive and negative passages. We set the margin $\epsilon = 1$ to align with the setup of Sentence-Transformer [54].

3.4 Inference

After fine-tuning the Transformer models, we perform inference on the queries in development set. For faster retrieval and to reduce online computation drastically, we first precompute the passage embeddings and store them to a disk, preferably after indexing. Typically at retrieval time, the user enters a query in plain text which is encoded by the transformer model to obtain its embedding in the same latent space as the passage embeddings. The relevance between the query embedding Q and the passage embedding D is given by the $cosine\_distance$ between them. Given a query Q and a passage D, the similarity score $S_{Q,D} \in \mathbb{R}$ is computed after pooling the output token embeddings $E_X \in \mathbb{R}^{|X| \times H}$ where $X \in \{Q, D\}$ and H is the hidden units size of the model:

$$S_{Q,D} = \cos(pooling(E_Q), pooling(E_D))) \tag{3.2}$$

where $pooling(*)$ is the pooling function defined by AVG, MAX or CLS strategy.

3.4.1 Indexing

Before retrieval, we first derive fixed-size embeddings for the passages in our collection. These pre-computed embeddings are stored in a forward index structure and saved to the disk shown in Fig.3.5. We choose the FAISS library for building the index data structure since it is extremely fast and practical for large-scale retrieval [30]. For our experiments, we use the simplest index version that performs brute-force cosine distance search on the vectors. It uses the vector ordinal as the key (first vector is 0, second is 1, etc.) and stores them without any encoding/compression.

3.4.2 Searching

Using the trained ranking model and pre-computed document embeddings, we can perform the search at real-time as shown in Figure 3.6. Using FAISS, we perform a basic k-nearest neighbour search on the index of passage embeddings for each query

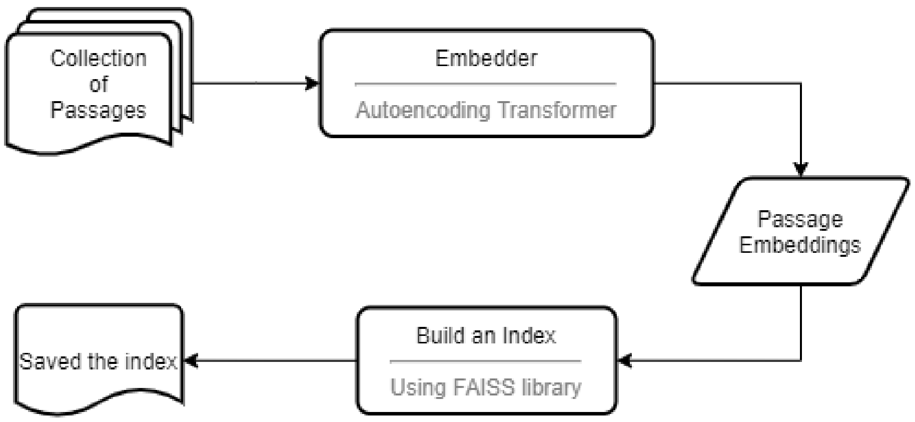

Figure 3.5: Offline precomputation of document embeddings of Figure 3.3. Passages in the collection are encoded using the fine-tuned autoencoding transformer shown in Figure 3.4. These passage embeddings are subject to a fast and GPU-scalable indexing like FAISS [30] which specializes in billion-scale datasets. The index is saved to a persistent storage system to be used at the time of retrieval.

embedding. We set k=1000 to retrieve the top-1000 nearest neighbours of the query vector along with their corresponding cosine distances. The search results are evaluated by standard IR metrics as described below.

3.5 Evaluation

MS MARCO uses MRR@10 as the official evaluation measure. The search results are evaluated on the top-1000 ranking passages retrieved for each query by the models. Other important metrics like Average Precision, R-Precision, R@k used commonly for evaluating ranking performance are also explained below:

Reciprocal Rank (RR) in information retrieval calculates the reciprocal of the rank of the first relevant document in the search result. If the relevant document is retrieved at rank 1, then RR is 1, if at rank 2, then RR is 0.5, if at rank 3 then RR is 0.33 and so on. Mean Reciprocal Rank (MRR) is the average of RR values across multiple queries. For multiple queries Q, the Mean Reciprocal Rank is calculated as follows.

$$\text{MRR} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{\text{rank}_i} \tag{3.3}$$

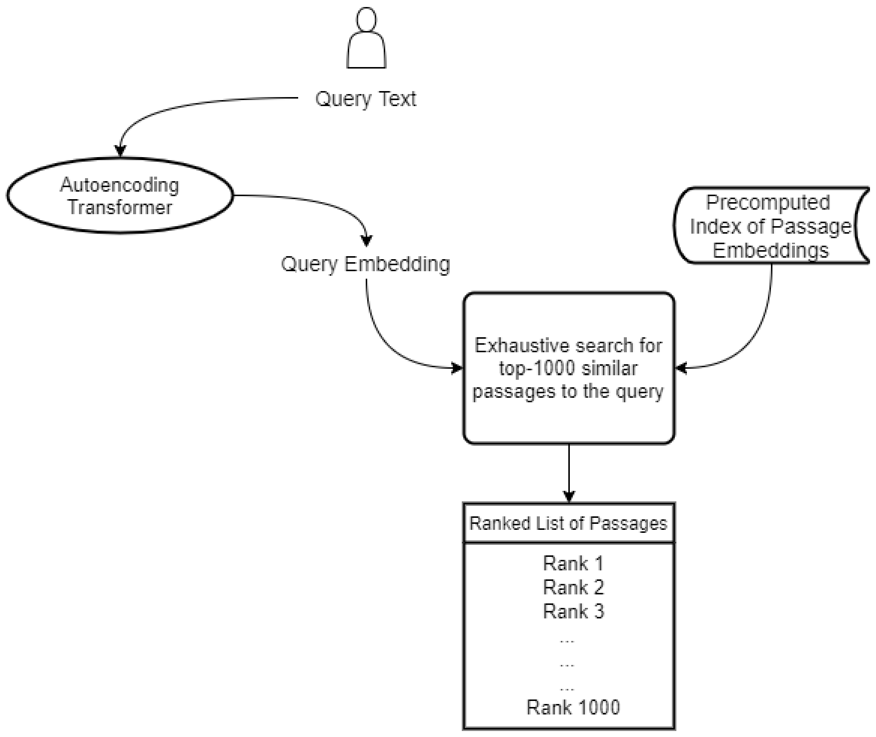

Figure 3.6: At retrieval time, the user's query is embedded using the fine-tuned autoencoding transformer shown in Figure 3.4 to compare with the index of pre-computed passage embeddings shown in Figure 3.5. Then, an exhaustive search for the top-1000 most similar passages is retrieved in the form of ranked list.

where *rank* is the position of the relevant document in the search result. MRR@10 sets a rank threshold at 10 and considers only the top-10 ranked documents for evaluation. Documents at rank lower than 10 are ignored.

Average precision (AP) combines recall and precision for the ranked search results. It It finds the area under the precision-recall curve. Mean Average Precision (MAP) is the mean of the average precision values for a set of n queries. It can be expressed as follows:

$$MAP = \frac{1}{n} \sum_{n} \frac{\sum_r P_n@r}{R_n} \tag{3.4}$$

where r is the rank of each relevant document, R is the total number of relevant documents, and P@r is the precision of the top-r retrieved documents.

Recall@k For a given query, recall@k is simply recall calculated only up to the k-th retrieval. It is the most important metric to evaluate a first-stage retrieval. The higher the recall@k, the better the chance for retrieval by a subsequent re-ranking model. If TP is the number of true positives and FN, the number of false negatives, then recall@k is calculated as follows.

$$Recall@k = \frac{TP@k}{(TP@k) + (FN@k)} \tag{3.5}$$

Chapter 4

Experimental Results

We train the autoencoding transformer models, namely, BERT, RoBERTa, ALBERT, DistilBERT and ELECTRA, on the MS MARCO dataset. Since the MS MARCO training dataset is available in triples of query, positive and negative passages, we use the triplet loss as our training objective. Only the base versions of the pre-trained models (as available in `https://huggingface.co/models`) are fine-tuned on the MS MARCO dataset for fairer comparison. All our models use the Transformer neural architecture which has several (6 or 12) encoder layers stacked on top of each other and perform self-attention on the incoming tokens. The output of the last encoder layer for each token in the input text is pooled to generate fixed-sized embedding of the entire input text (query/passage). More details on the models are available in Appendix B.

4.1 Data

MS MARCO or the MAchine Reading COmprehension dataset features 1,010,916 anonymized user queries sampled from Bing's search logs and 8,841,823 passages extracted from 3,563,535 web documents [46]. Since the questions in MS MARCO correspond to user submitted queries from Bing's query logs, their formulations are often complex, ambiguous, and may even contain typographical and other errors. An example of such a question issued to Bing is: "is a little caffeine ok during pregnancy", shown in Table 4.1. These questions represent human information seeking behaviour and therefore not well-formatted at times. Non-question queries are automatically filtered out by a machine learning based classifier.

Once the queries are filtered, the relevant documents are retrieved for each query using Bing's large-scale web index. At least 10 passages are automatically extracted from these relevant web documents using Bing's state-of-the-art passage retrieval system. A human editor then selects the passages that can be used to answer the

Query	Relevant Passage	Non-Relevant Passage
is a little caffeine ok during pregnancy	We don't know a lot about the effects of caffeine during pregnancy on you and your baby. So it's best to limit the amount you get each day. If you're pregnant, limit caffeine to 200 milligrams each day. This is about the amount in 1.5 8-ounce cups of coffee or one 12-ounce cup of coffee.	It is generally safe for pregnant women to eat chocolate because studies have shown to prove certain benefits of eating chocolate during pregnancy. However, pregnant women should ensure their caffeine intake is below 200 mg per day.

Table 4.1: An example query, relevant and non-relevant passage from the MS MARCO dataset.

query by setting "is_selected" to 1. For queries where no answer is present in the extracted passages, "is_selected" is set to 0 for all passages. The passage annotations by the human editors are non-exhaustive, therefore, there may be passages in the collection that contain the answer to the queries but are annotated as "is_selected": 0. In general, there exists a subset of queries with multiple answer passages and a subset of queries with no answer passages. MS MARCO is a large-scale dataset whose questions are derived from real user search queries and presents itself challenging for benchmarking neural IR models.

4.2 Baseline

We choose our baseline as BM25 because it serves as a strong candidate for initial retrieval when its parameters are tuned. BM25 treats the query as a "bag of words" for ranking documents from the collection using a TF-IDF scoring function [55]. It is based on exact-term matching, where all candidate passages must contain at least one term from the user's query. To get the BM25 retrieval results, we use the Lucene toolkit Anserini[1] with its tuned parameters $k_1 = 3.44,\ b = 0.87$ optimized for average precision (AP).

$$BM25(q,d) = \sum_{t_q \in q} idf\left(t_q\right) \cdot \frac{tf\left(t_q, d\right) \cdot (k_1 + 1)}{tf\left(t_q, d\right) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{avgdl}\right)} \tag{4.1}$$

[1]`https://github.com/castorini/anserini`

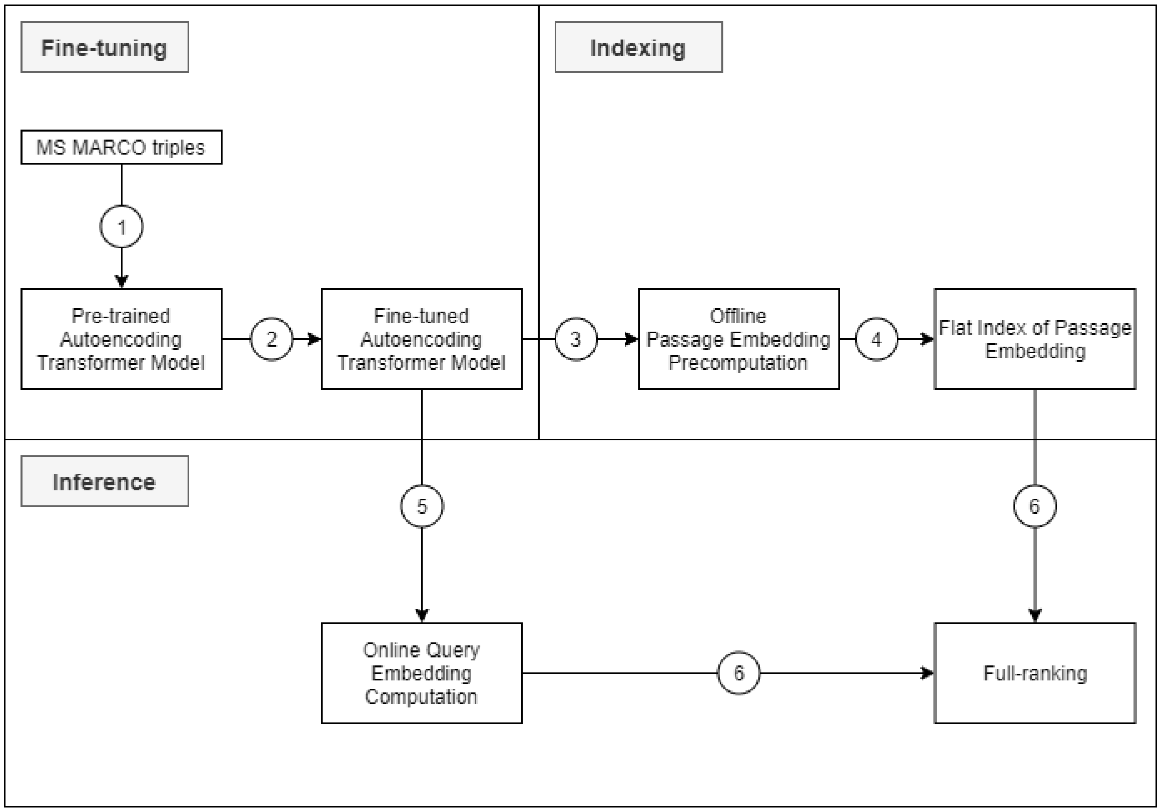
Fine-tuning
Indexing
Inference
MS MARCO triples
1
Pre-trained Autoencoding Transformer Model
2
Fine-tuned Autoencoding Transformer Model
3
Offline Passage Embedding Precomputation
4
Flat Index of Passage Embedding
5
6
Online Query Embedding Computation
6
Full-ranking

Figure 4.1: Experimental setup from training to inference. 1) & 2) The pre-trained autoencoding transformer models are fine-tuned directly on the MS MARCO triples for the ranking task. 3) The passage embeddings are computed and 4) saved to an index. 5) At inference, first, the query embedding is computed online using the fine-tuned model. 6) Finally, the passages in the collection are fully-ranked based on relevance.

where, $avgdl$ is the average length of documents, and k_1 and b are parameters that are usually tuned on a validation dataset.

4.3 Setup

MS MARCO passage retrieval task provides 3 datasets: training, development and evaluation. The original training set "triples.train.full" contains approximately 400 million tuples of a query, relevant and non-relevant passages. To align with the existing work [47], we only employ the 40M triples from "triples.train.small" for training (10% subsample of the full dataset) to keep the training time desirable (approx. 10-15 hours per model) and evaluate our models on the development set since the relevance judgements for the evaluation set are not provided. In our experiments, we stop training after 125K global steps when the models have converged after learning the first 2 million samples in batches of size 16, refer Appendix B. To keep the training data unarbitrary for all models, we sample the data sequentially. The development set contains 6,980 queries with their relevance judgements. On average, each query in the development set has at least one relevant passage.

We start training by loading the pre-trained weights of the respective models and then fine-tuning them on the ranking task with the MS MARCO training data. We choose "base-uncased" versions of BERT, DistilBERT and RoBERTa models, "base-discriminator" for the ELECTRA model and "base-v1" for the ALBERT model to make a fairer comparison. Following the experimental setup of [54], we use three different pooling strategies, namely AVG, MAX and CLS-pooling to generate a fixed-sized embedding for each text (query/relevant/non-relevant passage). We kept the maximum sequence length at 128 tokens per sequence for computational reasons, although the results of increased number of tokens (512) is discussed in Appendix B. We fine-tune all models with a triplet loss objective function (Eq. 3.1) for one epoch with batch-size of 16, Adam optimizer [31] with learning rate $2e^{-5}$, and a linear learning rate warm-up over 10% of the training data following the setup of [54]. A step-by-step flow of the process is shown in Figure 4.1. During inference, each model searches for top-1000 relevant passages from the 8.8 million passages in the collection for each query in the development set.

4.4 Empirical Analysis

We evaluate the performance of our models for the ranking task. We use a batch size of 128 to encode the documents into fixed-size embeddings and normalize them to calculate the inner product between the query and document embedding. The FAISS [30] index used in our experiments is the 'IndexFlatIP' which performs an exhaustive and exact search for top-1000 nearest neighbours. Highly similar documents are ranked at the top and the performance is evaluated on MAP, R-PREC, R@1000, MRR and MRR@10 metrics. The search results of each model are compared for statistical significance as shown in Table 4.3. For the paired student t-test, we divide the development set into 10 groups without replacement. Therefore, each sub-sample

Models	**Pooling**	**MAP**	**R@1000**	**MRR@10**
BM25	-	0.1926	0.8526	0.1839
BERT Reranker [47]	-	-	-	0.3470
BERT	**Avg**	**0.3036**	0.9259	**0.2962**
	CLS	0.2948	0.9211	0.2875
	Max	0.2930	0.9206	0.2863
RoBERTa	**Avg**	0.2783	0.9120	0.2715
	CLS	0.2453	0.8637	0.2387
	Max	0.2686	0.8968	0.2617
ALBERT	**Avg**	0.2703	0.8909	0.2633
	CLS	0.2733	0.8797	0.2664
	Max	0.2785	0.8824	0.2719
DistilBERT	**Avg**	0.2898	0.9222	0.2822
	CLS	0.2939	**0.9266**	0.2870
	Max	0.2985	0.9224	0.2913
ELECTRA	**Avg**	0.2291	0.8131	0.2238
	CLS	0.1922	0.7509	0.1875
	Max	0.2517	0.8290	0.2457

Table 4.2: Models scored on the development set for different ranking measures based on cosine-similarity between passage & query embeddings. Almost all autoencoding transformer models beat the BM25 baseline irrespective of the pooling strategy used. BERT exhibits top scores in Avg. and CLS pooling methods. DistilBERT tops the chart when Max pooling is used. The highest scores for each pooling method across models are in bold. Pairwise statistical significance test results for MRR@10 are shown in Table 4.3. The difference in MRR@10 between BERT and DistilBERT is not statistically significant, whereas all the other pairwise differences with these two models are statistically significant.

has 698 unique query examples. We then compute the MRR@10 score for the passages retrieved for each query sub-sample by the different autoencoding transformer models. Except for the results of some pairs of models (indicated in black in Table 4.3), all other pairs of results are statistically significant. For completeness, we also compare the performance of a BERT Reranker model trained using an interaction-based approach to re-rank the top-1000 passages retrieved by BM25 during inference in Table 4.2 [47]. We included the results of the BERT$_{Base}$ model to show the trade-off in performance for computation time. The BERT Reranker model only performs 1000 inference computation per query compared to the 8.8M inference computations of the representation-focused models.

		BERT			RoBERTa			ALBERT			DistilBERT			ELECTRA		
		Avg	CLS	Max	Avg	CLS	Max	Avg	CLS	Max	Avg	CLS	Max	Avg	CLS	Max
BERT	Avg		N	N	B	B	B	B	B	B	B	N	N	B	B	B
	CLS			N	B	B	B	B	B	B	N	N	N	B	B	B
	Max				B	B	B	B	B	B	N	N	N	B	B	B
RoBERTa	Avg					Avg	N	N	N	N	N	D	D	R	R	R
	CLS						Max	A	A	A	D	D	D	R	R	N
	Max							N	N	N	D	D	D	R	R	R
ALBERT	Avg								N	N	D	D	D	A	A	A
	CLS									N	D	D	D	A	A	A
	Max										N	D	D	A	A	A
DistilBERT	Avg											N	N	D	D	D
	CLS												N	D	D	D
	Max													D	D	D
ELECTRA	Avg														Avg	Max
	CLS															Max
	Max															

Table 4.3: Statistical significance test for the MRR@10 values of different model pairs for 10 sub-samples of the evaluation data. 'N' means the results of the pair in comparison are not statistically significant ($p > 0.05$). The coloured letters indicate that these model pair score differences are statistically significant and the initial letter of the model with the highest MRR@10 value of the pair is indicated in red (B-BERT, R-RoBERTa, A-ALBERT, D-DistilBERT and E-Electra).

4.4.1 Analysis on Inference

We study the performance of all models for different pooling strategies (AVG, MAX, and CLS). The pooling strategy has a significant impact in RoBERTa & ELECTRA models and less impact in BERT, ALBERT and DistilBERT models as shown in Table 4.2. Even though all of these models belong to the same transformer family, BERT & DistilBERT outperform the others in all ranking scores. This is in contrast to our hypothesis wherein we expected the newer and advanced language models to be better at semantic similarity when used this way. We hypothesize that BERT and its distilled version DistilBERT outperform the other models due to the differences in the pre-training tasks. The peak memory usage and required time during inference for each of the transformer models is shown in Fig. 4.2. We believe that when choosing the right embedding model, one must also be aware of its computational cost. We benchmark our results using Huggingface's implementation for benchmarking Transformers[2]. The required time shown here corresponds to time taken for each model (only AVG pooling) to infer on a batch size of 8 and maximum sequence length of 64 tokens on CPU & GPU hardwares. This performance is highly dependent on the hardware and version of software used. In addition to superior ranking performance, DistilBERT is also one of the models that consumes less memory/time. This can be attributed to the lower number of encoder layers (6) compared to the other models. When running our inference on GPU, we find that the time taken per step is comparable to the fast and simple BM25 (which takes ∼0.07s/query[3]). In terms of memory, all computations can be fit on a single CPU/GPU of RAM 12GB. We benchmark our results on a single Tesla V100-SXM2-16GB GPU, CUDA Version 11.0 and Pytorch 0.2.6.

4.4.2 Analysis by Answer-type

Real-world queries contain different types of questions. To understand the performance of these models when faced with different types of queries, we classify them based on the lexical answer type. We use the rule-based answer type classifier[4] to

[2]`https://huggingface.co/transformers/benchmarks.html`

[3]`https://github.com/castorini/anserini/blob/master/docs/experiments-msmarco-passage.md`

[4]`https://github.com/superscriptjs/qtypes`

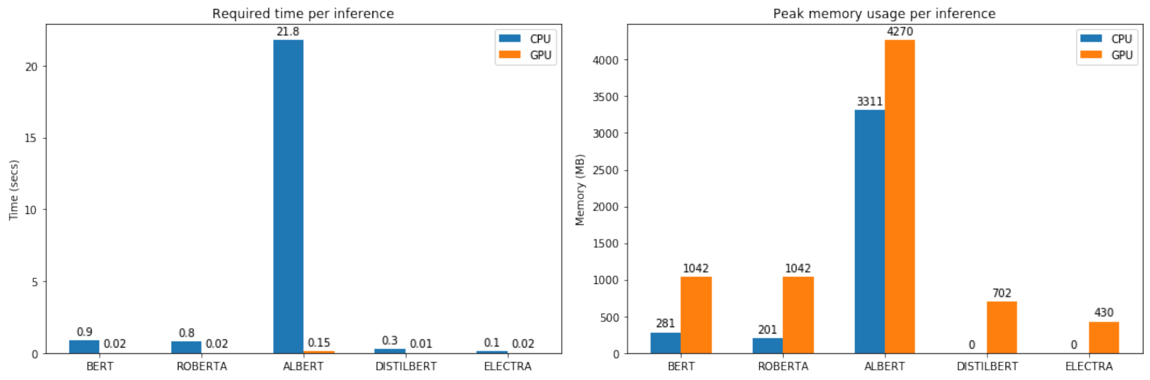

Figure 4.2: Required time and peak memory usage per inference step for each transformer model. Speed is measured in clock seconds while memory is measured in Megabytes when executed on both CPU & GPU hardwares.

Type	DESC	NUM	HUM	LOC	ENTY	ABBR
# of queries	2316	1168	568	563	404	9
BM25	0.19	0.19	0.23	0.25	0.21	0.17
BERT	**0.29**	**0.27**	**0.29**	0.37	0.25	**0.61**
ALBERT	0.25	0.25	0.25	0.34	0.24	0.30
RoBERTa	0.26	0.26	0.27	0.35	**0.27**	0.27
DistilBERT	0.27	0.26	**0.29**	**0.38**	0.25	0.31
ELECTRA	0.21	0.21	0.19	0.28	0.20	0.15

Table 4.4: Average MRR values for different types of answer like description, numerical, human, location, entity and abbreviation for 5028 queries in the evaluation set. The answer types are sorted based on decreasing order of their counts.

identify answer types [37]. The queries are grouped into 6 answer types, namely abbreviation, location, description, human, numerical and entity. Queries that could not be identified by the rule-based classifier are ignored. The MRR across these 6 types for 5028 queries having a valid answer type is shown in Table 4.4. BERT has significantly higher MRR on abbreviation type queries, ELECTRA has the lowest MRR on them. All models record lowest performance on entity type queries and significantly higher on location type queries. DistilBERT has performance on par with BERT failing only at abbreviation type questions.

4.4.3 Analysis on Semantic Similarity

Pre-trained Language Models can attain powerful performance because of their ability to capture deep semantic relationships. Being trained on a language modelling task with large amounts of data, they are expected to produce deep contextualized representations of words/sentences. With limited context, the models have to rely on the learned distribution of the training data to embed the semantics of a sentence. For some queries like "who is aida", the models retrieve lexically matching passages at the least even if the intent is not very clear.

4.5 Limitations of our work

The results of our experiments are surprising because it proves our hypothesis wrong. The newer and incrementally optimized models do not guarantee better performance for retrieval task. Having said that, we also acknowledge the limitations of our experiments. Many of our design choices are justified by less computational costs. Each model comes with a range of checkpoints and varying architectures like tiny, small, large but we only chose the base and uncased (if available) versions for a fairer comparison. Another limitation is the number of training examples, which is only 5% of the triples.train.small subset of examples. We decide to stop training at this point because we notice that all the models have converged on the validation set at 125K global steps (see Appendix B). Since the queries are short texts, the maximum length of sequences in our dataset does not exceed 64 tokens. The number of tokens in passages, however, can exceed 512 tokens which is the highest limit of most models. Instead of truncating the tokens to 512, we set the maximum sequence limit to 128. This again is due to computational reasons as we could not fit certain models in the resources available to us. Nevertheless, for the models that did fit, we observed the performance to be rather degrading as we can see the models struggling to learn better sentence representations. Finally, we keep the set of hyperparameters (number of epochs, optimizer, learning rate) fixed and do not search for the best set for each model for fairer comparison.

Query	Model	Passage
who is adia	**Relevant**	**For the musician, see Adia (musician). For the sovereign wealth fund ADIA...**
	BERT	Ayesha Curry. Ayesha Disa Curry (nee Alexander; born 23 March 1989)...
	RoBERTa	This article is about the song. For the girl's name, see Adia (name)...
	ALBERT	Related Tags. Adia is a song by Sarah McLachlan that originally appeared...
	DistilBERT	Adia (2006) Adia is the story of a 14-year-old Nigerian girl who gets sold...
	ELECTRA	Early Life: Popularly known as Bollywoods dream girl, Hema Malini is an...
what is vulnerable animals	**Relevant**	**Vulnerable definition, capable of or susceptible to being wounded or hurt, as by a weapon...**
	BERT	Officially, threatened species are those listed as Critically Endangered (CR), Endangered...
	RoBERTa	Wild Animal Definition: Animals that, as a matter of common knowledge,...
	ALBERT	by Jennifer Bove. Updated January 30, 2017. An endangered species is a species of wild...
	DistilBERT	An animal is in a good state of welfare if (as indicated by scientific evidence)...
	ELECTRA	Vulnerable Populations. Certain human subjects are categorized as vulnerable...
where is roots	**Relevant**	**A root is the part of a plant that is below ground. Root or roots may also refer to**
	BERT	A root is the part of a plant that is below ground. Root or roots may also refer to
	RoBERTa	roots - the condition of belonging to a particular place or group by virtue...
	ALBERT	The Roots is an American hip hop group, formed in 1987 by Tariq Black Thought...
	DistilBERT	often roots The condition of being settled and of belonging to a particular...
	ELECTRA	roots - the condition of belonging to a particular place or group by virtue...
what is va abbreviation	**Relevant**	**Quick Answer. The abbreviation VA stands for the state of Virginia in the United States...**
	BERT	Quick Answer. The abbreviation VA stands for the state of Virginia in the United States...
	RoBERTa	DVA stands for Department of Veterans Affairs. The meaning of DVA abbreviation...
	ALBERT	abbr. 1. Veterans Administration 2. vicar apostolic 3. also Va. Virginia 4. volt-ampere
	DistilBERT	VA Related Acronyms and Abbreviations. Acronymns Used When Veterans Benefits Are...
	ELECTRA	What does SS stand for? The abbreviation SS S.S. in front of the name of a ship...

Table 4.5: Sample queries for comparison from the MS MARCO dev dataset. Each query is sampled from one of the 6 different answer types. Passage for "Relevant" is the human annotated ground truth provided as reference for evaluation. The passage shown here for each model is the top answer retrieved by the respective models.

Chapter 5

Conclusion

Our work implements an IR system that is capable of fully ranking the documents in a corpus based on its relevance to the user's query using only the embeddings of fine-tuned language models. The complexity and sophistication of deep neural networks provides for high quality representation of the query and document without compromising on inference time. Based on previous work on similar natural language processing tasks, we incorporate the full ranking of the documents through triplet network architecture and use directly the embeddings of the autoencoding transformer models during inference.

We compare the performance of different language models against the overly-used BERT as the ranker. This goal was mainly motivated by the increasing research interests on general-purpose Language Models (LMs) capable of modelling the characteristics of any language when trained vigorously on large amounts of data. Many NLP tasks have seen improvement in their performances when fine-tuned on large pre-trained models. As for the task of document ranking, not much evidence is out there elaborating the usefulness of different language models. And so, we try to empirically compare the performances of the different language model in a full-rank setting to gain some insight about the ideal architecture for the ranking task. Experimental results show that newer language models do not outperform BERT's feature-based ranking performance. The sentence embeddings of BERT, when derived using AVG or CLS pooling techniques, has the highest MRR@10 value for the MS MARCO passage retrieval task. DistilBERT, a distilled version of BERT, strikes the right balance between effectiveness and efficiency in retrieval, since its retrieval performance is on par with BERT's and time taken per inference step is much lower than its non-distilled counterparts.

5.1 Future Work

Several possible extensions can be made to the IR system following the findings of this thesis. If deep neural networks can be used for initial retrieval without sacrificing the inference time, then a single model that satisfies all desiderata of an IR system would save a lot of training time. Having an ensemble approach might not be beneficial in the long run, especially when dealing with highly-complex deep network architectures. Without the advancement of hardware components like GPUs & TPUs that support these operations, we must rely on traditional IR algorithms that do not perform semantically well. Hence, a standalone deep neural model that ticks all checkboxes of a good IR system would benefit the community.

In the near future, we plan to test our hypothesis by doing an ablation study to explain the performances of the models. One incremental development can be made by making use of query expansion techniques previously experimented on several dataset. A major drawback of using short queries is capturing the right intent. A short query may lack much of the context necessary for these deep neural models to produce good quality representations. And hence, the training data might not be sufficient to capture all nuances of the domain and thus, lead to poor modelling of intent. Although recent language models have overcome this drawback to a certain level, it is proven that such models can benefit from introducing some additional context to the queries. However, traditional word-based query expansion is not entirely applicable to make use of BERT's sensitivity to the addition of structure and concepts. An interesting take would be to leverage Active Learning to generate useful intent-matching expansion of queries. The language model would generate potentially correct expansion of the short queries with a user in the loop giving feedback on the right expansion, which would in-turn be fed to the model to retrieve the top-N documents relevant to the enhanced query. There can be either a generative model and a ranker model ensemble or the same model can be optimized for different training objectives. Finally, the performance of DistilBERT in our research and the recent advancements in knowledge distillation (KD) using smallers rankers optimized for IR [16] shows that KD is the best immediate solution for speeding-up the inference without compromising on performance, especially for on-the-edge applications like mobile applications.

Bibliography

[1] Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. Applying BERT to document retrieval with birch. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 19–24, Hong Kong, China, November 2019. Association for Computational Linguistics.

[2] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3490–3496, Hong Kong, China, November 2019. Association for Computational Linguistics.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate.

[4] Pia Borlund. The concept of relevance in ir. *J. Am. Soc. Inf. Sci. Technol.*, 54(10):913–925, August 2003.

[5] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[6] Nick Craswell, W. Bruce Croft, Jiafeng Guo, Bhaskar Mitra, and Maarten de Rijke. Report on the sigir 2016 workshop on neural information retrieval (neu-ir). *SIGIR Forum*, 50(2):96–103, February 2017.

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the trec 2019 deep learning track, 2020.

[8] Zhuyun Dai and Jamie Callan. Deeper text understanding for IR with contextual neural language modeling. *CoRR*, abs/1905.09217, 2019.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[11] Laura Dietz and John Foley. Trec car y3: Complex answer retrieval overview. Text REtrieval Conference (TREC), 2019.

[12] Laura Dietz, Ben Gamari, Jeff Dalton, and Nick Craswell. Trec complex answer retrieval overview. Text REtrieval Conference (TREC), 2018.

[13] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. Trec complex answer retrieval overview. Text REtrieval Conference (TREC), 2017.

[14] Hai Dong, Farookh Hussain, and Elizabeth Chang. A survey in traditional information retrieval models. pages 397 – 402, 03 2008.

[15] Lea Frermann. Extractive NarrativeQA with heuristic pre-training. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 172–182, Hong Kong, China, November 2019. Association for Computational Linguistics.

[16] Luyu Gao, Zhuyun Dai, and Jamie Callan. Understanding bert rankers under distillation. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, ICTIR '20, page 149–152, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Yoav Goldberg. Neural network methods for natural language processing. 10(1):1–309. Publisher: Morgan & Claypool Publishers.

[18] Bert F. Green, Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: An automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '61 (Western), page 219–224, New York, NY, USA, 1961. Association for Computing Machinery.

[19] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 55–64, New York, NY, USA, 2016. Association for Computing Machinery.

[20] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, page 102067, 2019.

[21] Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. Learning-to-rank with bert in tf-ranking. Technical report, Google, 2020.

[22] Donna Harman. Overview of the first trec conference. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, page 36–47, New York, NY, USA, 1993. Association for Computing Machinery.

[23] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[24] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[25] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2042–2050, Cambridge, MA, USA, 2014. MIT Press.

[26] Haiqing Hu. A study on question answering system using integrated retrieval method. 01 2006.

[27] Po-Sen Huang, Xiaodong He, Jianfeng Gao, li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. pages 2333–2338, 10 2013.

[28] Mahbub Hussain, Jordan J. Bird, and Diego R. Faria. A study on cnn transfer learning for image classification. In Ahmad Lotfi, Hamid Bouchachia, Alexander Gegov, Caroline Langensiepen, and Martin McGinnity, editors, *Advances in Computational Intelligence Systems*, pages 191–202, Cham, 2019. Springer International Publishing.

[29] Nasreen Jaleel, James Allan, W. Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark Smucker, and Courtney Wade. Umass at trec 2004: Novelty and hard. 01 2004.

[30] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *ArXiv*, abs/1702.08734, 2017.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[33] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

[34] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[36] Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*, volume 4. 04 2011.

[37] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[38] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3:225–331, 01 2009.

[39] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[40] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. Cedr: Contextualized embeddings for document ranking. SIGIR'19, page 1101–1104, New York, NY, USA, 2019. Association for Computing Machinery.

[41] Sean MacAvaney, Andrew Yates, and Kai Hui. Contextualized pacrr for complex answer retrieval. In *TREC*, 2017.

[42] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, USA, 2008.

[43] Bhaskar Mitra and Nick Craswell. Neural models for information retrieval. *CoRR*, abs/1705.01509, 2017.

[44] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1291–1299, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[45] Bhaskar Mitra, Corby Rosset, David Hawking, Nick Craswell, Fernando Diaz, and Emine Yilmaz. Incorporating query term independence assumption for efficient retrieval and ranking using deep neural networks. 07 2019.

[46] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268, 2016.

[47] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019.

[48] Harshith Padigela, Hamed Zamani, and W. Bruce Croft. Investigating the successes and failures of BERT for passage re-ranking. *CoRR*, abs/1905.01758, 2019.

[49] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2793–2799. AAAI Press, 2016.

[50] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[51] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. "Technical report, OpenAI.", 2018.

[52] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[53] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[54] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP*, 2019.

[55] Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST, January 1995.

[56] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[57] Ricardo Usbeck, Michael Röder, Michael Hoffmann, Felix Conrads, Jonathan Huthmann, Axel-Cyrille Ngonga Ngomo, Christian Demmler, and Christina Unger. Benchmarking question answering systems. *Semantic Web*, 10:1–12, 08 2018.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[59] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[60] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. *CoRR*, abs/1706.06613, 2017.

[61] Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of BERT for ad hoc document retrieval. *CoRR*, abs/1903.10972, 2019.

[62] Hamed Zamani and W. Bruce Croft. Relevance-based word embedding. *CoRR*, abs/1705.03556, 2017.

[63] Xin Zhang, An Yang, Sujian Li, and Yizhong Wang. Machine Reading Comprehension: a Literature Review. *arXiv e-prints*, page arXiv:1907.01686, June 2019.

[64] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 19–27, USA, 2015. IEEE Computer Society.

Appendix A

Normalized Discounted Cumulative Gain (NDCG)

The Normalized Discounted Cumulative Gain (NDCG) is an evaluation metric for IR used to rate the ranking effectiveness of a web search engine. Each document in the search result needs a non-binary graded relevance score to calculate the cumulative gain. It makes two assumptions:

1. Highly relevant documents are more valuable than marginally relevant documents, which again are more valuable than non-relevant documents.

2. Highly relevant documents are only useful if they are ranked at the top of the search list, so that the searcher does not have to pay much effort to find them.

NDCG penalizes highly relevant documents by discounting the graded relevance value logarithimically proportional to the document's ranked position and then normalizes the score. Since the relevance judgements of the passages in the MS-MARCO collection are binary, we set the grade value to 1. The NDCG@k for the base recall $k \in 5, 10, 100, 1000$ values agrees with the other metrics of having BERT and DistilBERT in the lead, although here, DistilBERT has the best performance in 2 out of 3 pooling strategies shown in Table A.1.

	Pooling	**NDCG@5**	**NDCG@10**	**NDCG@100**	**NDCG@1000**
BERT	Avg	**0.3171**	**0.3505**	**0.4081**	**0.4239**
	CLS	0.3074	0.3387	0.3969	0.4143
	Max	0.3059	0.3402	0.3969	0.4137
RoBERTa	Avg	0.2898	0.3225	0.3813	0.3989
	CLS	0.2532	0.2831	0.3406	0.3601
	Max	0.2794	0.3112	0.3677	0.3867
ALBERT	Avg	0.2806	0.3136	0.3690	0.3875
	CLS	0.2843	0.3164	0.3702	0.3881
	Max	0.2902	0.3238	0.3777	0.3943
DistilBERT	Avg	0.3031	0.3357	0.3940	0.4111
	CLS	0.3088	0.3415	0.3981	0.4154
	Max	0.3140	0.3449	0.4023	0.4188
ELECTRA	Avg	0.2372	0.2648	0.3140	0.3358
	CLS	0.1976	0.2226	0.2693	0.2920
	Max	0.2606	0.2890	0.3407	0.3595

Table A.1: The NDCG@k score for 5, 10, 100 and 1000 values of k for each model. The results are mostly consistent with the other metrics, with $BERT_{Avg}$ having the highest score for all 4 metrics.

Appendix B

Analysis on Training

The model configurations used in our experiments is given in detail in Table B.1. The DistilBERT model used here is distilled from the BERT model bert-base-uncased checkpoint. For more details on the model configurations, please refer the corresponding papers.

When trained on the same subset of the dataset, the learning curve shown in Figure B.2 seems to converge at around 100k global steps for all models. Accuracy on the validation dataset at the end of one epoch, which is 125k training steps.

We experimented with maximum sequence lengths of 512 tokens for all models for training and inference. Clearly, increasing the number of tokens significantly impacts the results and also computation complexity. We could not perform inference on RoBERTa since our current hardware resources could not allocate enough memory for the computation.

Version	**Train data (GB)**	**Layer**	**Pretrain Task**	**Params (Mil.)**	**Vocab size**
bert-base-uncased	16	12	static-MLM + NSP	110	30k
roberta-base	160	12	dynamic-MLM + Full Sent. (no NSP)	125	50k
albert-base-v1	16	12	MLM + SOP	11	30k
distilbert-base-uncased	16	6	MLM	66	30k
electra-base-discriminator	16	12	Replaced Token Detection	110	30k

Table B.1: The pretrained model versions that were used in our experiments along with other configuration details like the training data size, number of layers, the language model task they were pre-trained on (MLM=Masked Language Model, NSP=Next Sequence Prediction and SOP=Sentence Order Prediction), number of parameters and the vocabulary size of the autoencoding models. The hidden units size and the number of attention heads are 768 and 12 respectively for all models.

	MAP	**R-Prec**	**R@1000**	**MRR**
BERT	0.2637	0.1617	0.8916	0.2685
RoBERTa	-	-	-	-
ALBERT	0.2492	0.1563	0.8430	0.2538
DistilBERT	**0.2858**	**0.1784**	**0.9100**	**0.2900**
ELECTRA	0.2175	0.1371	0.7933	0.2224

Table B.2: IR scores when the maximum sequence length is 512 tokens at training and inference. Compared to other models except RoBERTa, DistilBERT is the clear winner.

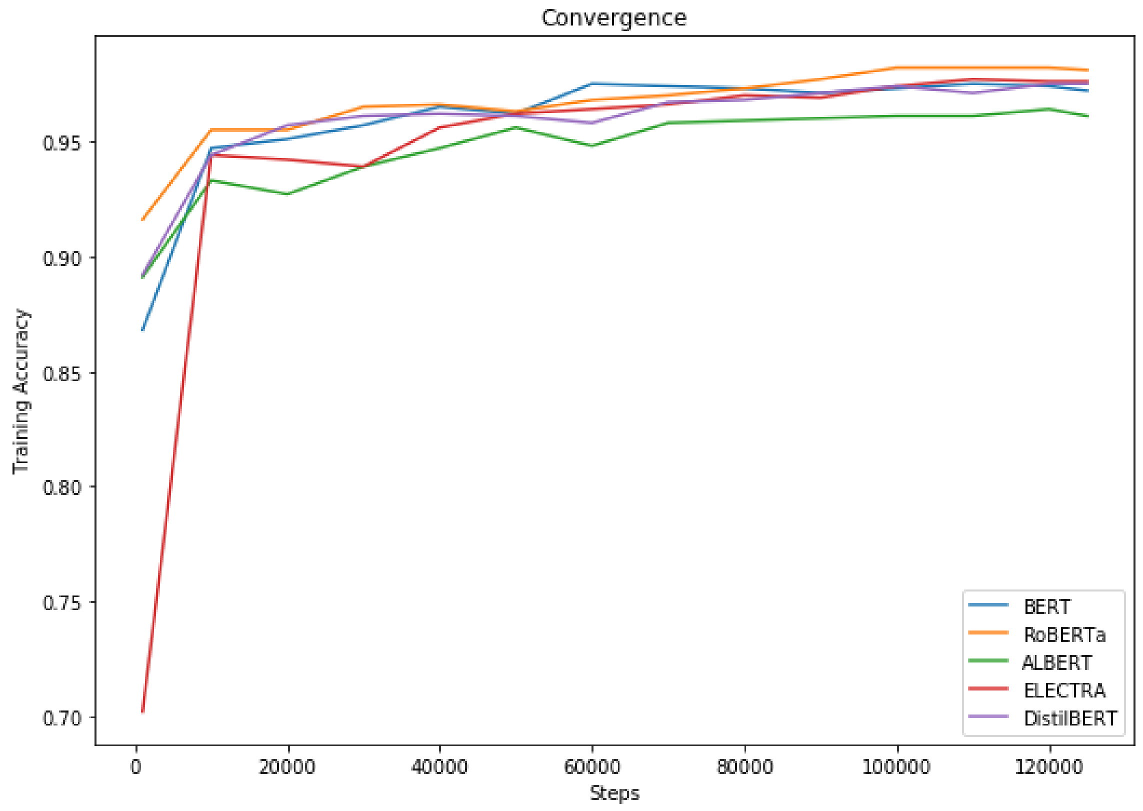

Figure B.1: Amount of time taken for training on 2M examples by each model relative to the fastest training model which is ALBERT.

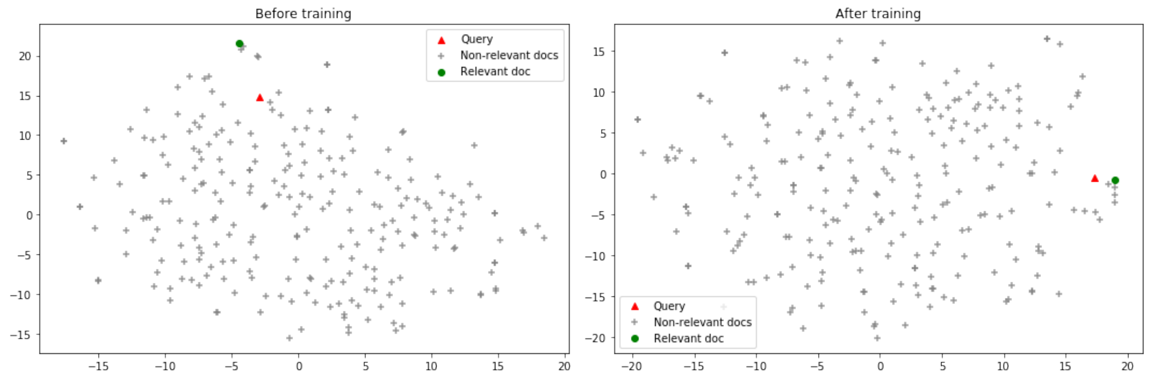

Figure B.2: t-SNE plot of query and passage embeddings before and after training the DistilBERT model.