Improving Efficiency in Dense Passage Retrieval

An Analysis of Employing Smaller Encoder Models

Andrew Harrison
Universiteit van Amsterdam
Amsterdam, Netherlands
andrew.harrison@student.uva.nl

Luuk Kaandorp
Universiteit van Amsterdam
Amsterdam, Netherlands
luuk.kaandorp@student.uva.nl

Shuai Wang
Universiteit van Amsterdam
Amsterdam, Netherlands
shuai.wang@student.uva.nl

ABSTRACT

In this paper, we first reproduce the Dense Passage Retrieval (DPR) findings, subject to computational constraint adjustments. This is then used as a baseline to assess alternate Transformer encoder models beyond the standard BERT models used in the query and passage encoders of DPR's bi-encoder architecture. Dense embeddings have significantly improved passage retrieval, which is crucial to open-domain Question Answering performance, but it still has computational expenses that prevent DPR from being used more broadly. We examine smaller encoder models, both those with alternate pre-training such as ELECTRA, and those using knowledge distillation from BERT; DistilBert and TinyBERT. We assess their retrieval performance on the Natural Question (NQ) dataset using Acc@1, Acc@20, Acc@100, and MRR. We contrast this retrieval performance against the models' efficiency as measured by execution time during fine-tuning, offline phrase index construction and inference time. The use of a final projection layer to larger and smaller embedding sizes is examined, to assess whether efficiency gains can be made with only minor performance loss. Our results find DistilBERT to be a suitable replacement for BERT, with minimal to no performance drop and a significant computational decrease. Whereas ELECTRA and TinyBERT evidence significant performance deterioration, and the use of a projection layer only significantly decreases performance across all models. Our work supports the replacement of BERT with DistilBERT for DPR in compute-bound use cases and justifies future research into this promising avenue of using smaller encoder models in DPR.

CCS CONCEPTS

• **Computing methodologies** → *Neural networks*; **Ranking**.

KEYWORDS

Dense Passage Retrieval, BERT, DistilBERT, TinyBERT, ELECTRA, Efficiency

ACM Reference Format:
Andrew Harrison, Luuk Kaandorp, and Shuai Wang. 2021. Improving Efficiency in Dense Passage Retrieval: An Analysis of Employing Smaller Encoder Models.

1 INTRODUCTION

How can we create systems that allow users to ask a question about anything and get a concise answer back? What if we ask the system *"Who stars in the Netflix show Squid Game?"* or *"When was the treaty of Versailles signed?"* In each scenario, we want a different type of answer. For the first question, we would like a longer passage that includes multiple actors and their roles, while for the second question a specific year is sufficient information. Open-domain Question Answering (QA) is a task that aims to answer a question based on an extensive collection of unstructured documents [24]. As the documents are unstructured, meaning that there is no consistent format that all data adheres to (e.g. Wikipedia dumps or crawled websites), it is qualitatively different from closed-domain QA. For example, when performing closed-domain QA on movies, we set a format that separates the title, duration, actors, director, etc, and pre-processes all the data into that format. This distinction reveals the main challenge within open-domain QA: we need to create accurate representations of queries and documents without any structure to adhere to.

Current state-of-the-art models for open-domain QA propose a pipeline that splits the problem into a two-stage framework: the *document retriever* for retrieving the relevant documents, and the *machine reader* for identifying possible answers from these documents [3]. Models such as ALBERT [15] have proven to be very good readers, with strong scores on the SQuAD2.0 QA challenge. Thus, current state-of-the-art models have no problem in extracting the answer, as long as they are provided with the right document to extract it from. Therefore, this paper will focus on the document retrieval part of the open-domain QA two-stage framework.

In contrast to traditional retrievers based on TF-IDF or BM25, that create extremely high-dimensional sparse embeddings, Dense Passage Retrieval (DPR) is a document retrieval method in which queries and documents are represented as dense vectors that capture a latent semantic encoding, with the final retrieval still then based on similarity measures between the query and the documents [12]. In DPR, two separate models are trained; the *query-encoder* which is used to encode the query and the *context-encoder* which is used to encode the documents. The models are trained jointly on batches of questions with their gold context (i.e. context that contain the answer) and negative contexts (i.e. contexts that do not contain the answer), with the training objective of the joint models being to embed the queries close to the gold contexts and distant from the negative contexts. As shown by Karpukhin et al. [12], DPR achieves state-of-the-art performance on several open-domain QA datasets, such as Natural Questions [14] and SQuAD [20].

Current DPR methods consist of dual-encoder architectures using large pre-trained Transformer encoder models like BERT [5].

Compared to cross-encoder architectures where question and document embeddings must be computed online, the dual-encoder allows the document corpus embeddings to be created offline. This makes the use of dense encodings more computationally tractable in retrieval settings, especially with large document corpora, where they can now be used end-to-end, rather than only being added as a re-ranker second stage to a TD-IDF/BM25 based first stage. Furthermore, the use of approximate nearest neighbour (ANN) techniques in semantic matching, an example being the FAISS (Facebook AI Semantic Search) [11] library, further speeds up the matching between documents and queries with only minor performance loss. Despite all these enhancements, it remains a downside of DPR and dense embeddings more generally, that they are still very computationally expensive. Using these models, the offline computation of the dense embeddings can take approximately 8.8 hours, even when parallelized across 8 GPUs, for a corpus of 21 million passages [12]. This means that DPR is practical of less benefit for many use cases, especially those with a dynamic document corpus, in which documents are changed often, and their content significantly differs, thus requiring ANN retraining.

Beyond offline document embedding and indexing, there is the additional computation needed during the fine-tuning of these large pre-trained Transformer encoder models, that must still be accounted for. Despite their substantial pre-training that has made them suitable for many NLP tasks, with even their zero-shot performance sometimes being reasonable, fine-tuning them on the domain and/or a specific task is still required. Due to the substantial model sizes, this fine-tuning is still computationally expensive. Added to this is the significant inference time requirements, which whilst sped up with offline document embedding and indexing, and ANN for semantic matching, can still be too slow for time-sensitive tasks, or simply too computationally intensive in terms of memory or energy usage for edge devices and mobile phones when inference is done on the device instead of being cloud-based.

To make DPR more accessible and less computationally expensive, we replace the standard BERT encoder models with smaller Transformer-based encoder models. These models have been developed using both distillation techniques, where the knowledge of a full-sized teacher model is distilled down and transferred into a smaller student model, and alternate pre-training tasks that allow smaller models to achieve the same performance level as larger models. While there are further techniques such as quantisation and pruning, these are not examined herein. Using these smaller encoder models, we aim to achieve better computational efficiency while retaining as much document retrieval performance as possible. To be specific, this paper answers the following research questions:

- RQ1 Which pre-trained model provides the best trade-off between efficiency and document retrieval performance in dense passage retrieval?
- RQ2 How does adjusting the embedding size using a final projection layer affect this trade-off?

We reproduce the DPR work as closely as possible, given our computational constraints. Our results show that DistilBERT provides a viable alternative to BERT, with a minor drop in Acc@1 and MRR, but equivalent performance at Acc@20, Acc@100, while being 61% faster to fine-tune during training, 38% faster at building the offline document index, and equivalent at inference time. While ELECTRA and then TinyBERT were faster again, the performance degradation was significant. Finally, adjusting the embedding size using a final projection layer also provided no useful change in the effectiveness versus efficiency trade-off, neither when projecting down nor up the embeddings of larger models, nor projecting up smaller models. Our main contribution is confirming DistilBERT as a viable alternative to BERT within the DPR model architecture, and validating the further assessment of alternate models within the DPR model architecture.

2 RELATED WORK

With an encoder and decoder, and reliance on self-attention, the Transformer architecture [23] replaced the use of Recurrent Neural Networks (RNNs) and opened up a Cambrian explosion of subsequent full and partial Transformer-based models. BERT [5] is an encoder-only model that introduced Masked Language Modelling (MLM) and next sentence prediction (NSP) pre-training tasks. Along with its bi-directionality, it leveraged an enormous number of parameters (110m in BERT base) and substantial pre-training on Wikipedia and the Book corpus to herald in the first golden age of large language models. Its uptake and usage in Information Retrieval were fast [19] with an application to passage re-ranking on MSMARCO [1], leading to a large performance jump and setting off what is termed in passage retrieval the 'BERT revolution' [17].

BERT was designed to function at word level and sentence level. Thus basic BERT applies cross-attention on tokens, and its embeddings are at the token level, but there is no averaging of these embeddings, as instead there is also a special [CLS] token attached to the front of the sentence inputs, which are trained to provide a sentence-level embedding. These embeddings are what is being compared in the dual-encoder architecture of DPR, with one from the query and one from each document. A recent survey [17] first tracks similarity comparisons using dense representations to the introduction of poly-encoders [9] that learnt global, rather than token level self-attention, but there was a limited examination of its performance on retrieval tasks. Subsequently, the dual-encoder design was successfully used in a Siamese architecture with Sentence-BERT [21] and as a cross-encoder distilled down to a bi-encoder in the Distilled Sentence Embedding (DSE) model [2]. With this then came a stampede of explicit dense retrieval papers [17].

The DPR model used as the base of our current research uses a dual-encoder BERT architecture with no additional pre-training tasks. It is distinguished by its simplicity, and use of in-batch positive passages from other queries as negatives for the current query, with a training objective that at batch level compares all pairs of question and positive passage and maximises the inner product. This improved performance over and can be contrasted to the REALM (Retrieval augmented Language Model Pre-training) model [7], which augments the text with sentences from Wikipedia when doing the MLM task during pre-training of the language model. Similarly, the ORQA (Open-Retrieval Question Answering System) model [16] also adjusts pre-training, using an inverse-closure task, along with only fine-tuning the query embedding model, but is also outperformed by DPR.

Some alternate models have sought to use dense encodings but remain usably efficient, such as ColBERT [13] which sits mid-way between bi-encoder and cross-encoder architectures, using a late interaction architecture. However, it compares dense vector representations at the token level and is slower again than standard DPR. Approximate Nearest Neighbour Contrastive Estimation (ANCE) [26] is an alternative dense passage retrieval model, that also uses a bi-encoder design but rather than in-batch negative samples, uses 'hard' negative examples i.e. negative passages that are highly similar to the query. But it is not examined within this work, as we focus only on DPR. Our work thus takes DPR, with its BERT encoders, as the baseline starting point and we examine what alternate encoders might be used in the same overall DPR architecture, that will maintain or minimally reduce performance, while significantly increasing efficiency as evidenced through shorter training times, faster phrase embedding index construction, and quicker inference time performance.

One class of models are positioned within knowledge distillation. Drawing on the metaphor that insects have larvae forms and then adult forms, Hinton et al. [8] argue for a separation between the training stage and deployment stage of machine learning models, with an ensemble of models or a large single cumbersome model distilled into a smaller model for deployment. The generic teacher-student framework within knowledge distillation, takes a full-scale teacher model (in our case BERT), and distils down the knowledge stored in that large neural network and transfers this knowledge across to a smaller student model during training. For a recent survey on knowledge distillation, we refer to work by Gou et al. [6].

DistilBERT [22] is one such distillation model, developed by the industry team behind the HuggingFace NLP library. DistilBERT was specifically developed due to the increased prevalence and usage of large scale pre-trained language models that require fine-tuning on downstream tasks. Recognising the computational budget and inference time constraints of many end-users, the team at HuggingFace pre-trained a smaller general-purpose language model by distilling BERT during pre-training, along with introducing a triplet loss that accounts for cosine-distance, distillation, and language modelling losses.

TinyBERT [10] is a distillation model, developed by Huawei, and again recognises the paradigm shift that has occurred within Natural Language Processing, and subsequently IR, with pre-trained Transformer-based models being fine-tuned on downstream tasks. Focused on edge devices and mobile phones, they seek to reduce the size of the models and speed up inference time. They introduce a Transformer specific distillation method that seeks to capture the embedding output layer, the hidden states, attention matrices of the Transfomer layer, and the prediction layer logits. Furthermore, they use a two-stage distillation learning framework with a *general distillation* from BERT, that initialises weights for the TinyBERT student network, and then a *task-specific distillation* of a fine-tuned BERT that further updates the TinyBERT student network.

ELECTRA [4] is from an alternate class of models that pre-trains from scratch, using a different pre-training task from BERT, rather than distilling from BERT. So rather than learning based on BERT's Mask Language Modelling (MLM), which tries to guess what the masked tokens are, the authors develop replaced token detection, which is a more sample efficient measure that is specifically meant to perform well in low computation situations. A small generator network corrupts the tokens in the text, and then a discriminator network is trained to determine whether tokens are corrupted or not. This pre-training approach is shown to achieve performance comparable to BERT with a smaller model and less training, and can even outperform BERT when an equivalent model size and volume of training to BERT is used.

3 METHODOLOGY

In this work, we focus on improving the retrieval efficiency of open-domain QA by optimizing DPR. Specifically, rather than using the pre-trained BERT model in DPR, we replace it with smaller models and investigate the effectiveness\efficiency trade-off.

3.1 Dense Passage Retrieval

DPR uses two separate dense encoders for questions ($E_Q(q)$) and text passages ($E_P(p)$) to map them into d-dimensional real-valued vectors [12], with $d = 768$ when using the standard BERT encoder models. The system retrieves k passages at run-time whose vectors are the closest (most semantically similar) to the input question vector. Where the similarity is measured using the inner product, as Karpukhin et al. [12] prove that it performs similarly to more complex similarity measures such as cosine and Euclidean distance while being a significantly simpler function. Thus, our similarity measure is defined as follows:

$$sim(q,p) = E_Q(q)^T E_P(p) \tag{1}$$

The training goal of DPR is to learn a better embedding function to then produce a vector space in which relevant pairs of questions and passages have higher similarities than irrelevant ones. Let $D = \{< q_i, p_i^+, p_{i,1}^-, ..., p_{i,n}^- >\}_{i=1}^m$ be the training data that consists of m instances. Each instance contains one question q_i and one relevant passage p_i^+, along with n irrelevant (negative passage p_{ij}^-). We optimize the loss function as the negative log-likelihood of the positive passage [12]:

$$L(q_i, p_i^+, p_{i,1}^-, ..., p_{i,n}^-) = -log \frac{e^{sim(q_i,p_i^+)}}{e^{sim(q_i,p_i^+)} + \sum_{j=1}^n e^{sim(q_i,p_{i,j}^-)}} \tag{2}$$

Negative passages can either be hard negatives or standard negatives. Hard negatives are gathered by using an earlier trained retrieval model to discover negatives and construct a different set of examples in each training iteration. Standard negatives can be collected in three different ways, as explained by Karpukhin et al. [12]. The first is random, where we pick any random passage from the corpus to be the negative for our current question. Secondly, we could use the top passages returned by BM25, which do not contain the answer but most closely match with the question. Thirdly, we can use positive passages for other questions to be the negatives for the current question. Next to these explicit negatives, the joint training architecture of DPR also makes use of implicit in-batch negatives. These in-batch negatives are the positive contexts for the other questions in the batch.

During inference, the passages are stored in a FAISS [11] index that speeds up search and comparisons of dense embedding vectors. It is a highly efficient index that uses k-means clustering and k-NN graph techniques to speed up similarity search. At inference time, we embed the question and return the top-k similar documents from the passage index.

3.2 Replacing Encoder Model

Instead of using the standard BERT model as the encoder, we replace it with smaller models. We hypothesize that the smaller models provide better computational efficiency during training, index building, and inference. Note that the same BERT tokenizer is used across all models, and thus as a constant factor its impact is not in our analysis. Specifically, we employ the following models:

3.2.1 BERT [5]. BERT is the standard encoder model used in DPR. It is a Transformer encoder model consisting of 12 hidden layers, 12 attention heads, and an embedding dimensionality of 768. The model totals 110 million parameters in the base version.

3.2.2 DistilBERT [22]. DistilBERT is a distilled version of BERT that claims to have 40% fewer parameters than BERT base while running 60% faster and preserving 95% of BERT's performance on the GLUE language understanding benchmark. The model consists of 6 hidden layers, 12 attention heads, and an embedding dimensionality of 768. DistilBERT consists of 66 million parameters in total.

3.2.3 TinyBERT [10]. TinyBERT is a model that is trained in a student-teacher fashion, with the original BERT base model being the teacher. It claims to be 7.5 times smaller and 9.4 times faster on inference while retaining competitive performance. The model is based on the BERT architecture and uses 4 hidden layers, 12 attention heads, and an embedding dimensionality of 312. The model totals only 14.5 million parameters for the base version.

3.2.4 ELECTRA [4]. ELECTRA is a BERT model that is trained using the ELECTRA pre-training approach. The new pre-training approach leads to performance that is comparable to larger RoBERTa and XLNet models. Clark et al. [4] claim that the small ELECTRA model achieves the same performance as BERT base, that is why we will be using this model. The model consists of 12 hidden layers, 4 attention heads, and an embedding dimensionality of 128 and a hidden size of 256. Note that the ELECTRA embedding size is not equal to the hidden layer size throughout the model, and this separation is unique to the architectures herein. The model is also unique in the sense that it uses only 4 attention heads per layer, compared to the 12 used by the other models. The final embeddings of the model are of size 256. ELECTRA small consists of 14 million parameters in total.

3.3 Embedding Size

The final embedding size is the length of the question and passage vectors obtained from the encoder. Larger embedding sizes have the capacity to represent more information from passages, but it influences efficiency by increasing the training time, building the index, and at inference time. DPR uses two base-uncased-BERT models as encoders with the embedding dimension equal to 768. In our experiments, we adjust the output embedding size by adding a linear projection layer that is trained with the encoders during fine-tuning of the models. Note that this means this final layer is initialised and trained only on the fine-tuning data for this task, instead of all other layers of the base model which have been fully pre-trained on their original datasets. This projection layer can both decrease and increase the final embedding size, and both directions are explored. Finally, as noted previously the input embedding size of ELECTRA and hidden layer size differ, so in this context, we refer to projecting the final hidden layer size to a larger final embedding size.

3.4 Batch Sizes

The batch sizes used during training deep learning models is important generally in terms of maximising the GPU memory usages and thus speeding up throughput and hence faster training. But also in terms of making a gradient update based on a large number of examples which can improve convergence. However, for DPR there is additional importance as it uses in-batch negatives (positive passages from the other queries) when training the current query-positive phrase pair. Thus, a bigger batch size means more negative examples and thus better contrastive learning [12]. We thus always attempt to maximise the batch size in factors of 2, i.e. {8, 16, 32, 64, 128}, for each model. These batch sizes are shown in Table 1. Note BERT in the original DPR paper is fine-tuned using a batch size of 128, and the authors emphasise the importance of batch size to performance.

4 EXPERIMENTAL SETUP

In the following subsections, we discuss the dataset that is used, the encoder models, evaluation metrics, evaluation method, and hyperparameters. We attempt to reproduce the work by Karpukhin et al. [12] as closely as possible. Our exact implementation can be found on Github[1].

4.1 Natural Questions (NQ) Dataset

In this paper, we focus on the retrieval performance on the Natural Questions (NQ) dataset [14]. The NQ corpus was developed for open-domain question answering. It contains search queries mined from real users with the subsequent answers returned from Wikipedia articles. Specifically, we use the version that is provided by Karpukhin et al. [12] on their GitHub page. This version is already split into usable passages and includes (hard) negative contexts. We use the training dataset for fine-tuning and the development set for evaluation, as the test set is not publicly available.

By default, the BERT tokenizer tokenizes the input to its maximum input length of 512 tokens, using padding as required. However, due to the limited computational resources available during experimentation, we instead attempt to use as short a token length as possible to allow for bigger batch sizes. Therefore, we analyzed the NQ dataset, to see whether it is feasible to use smaller token lengths without adversely impacting performance. Figure 1 shows the density distribution between 0 and 512 tokens, with the red line indicating a cut-off point of 256 tokens. The figure shows that most passages have a length of fewer than 200 tokens. Specifically, 99.9% of the passages contain less than 244 tokens and 99.99% contain less than 364 tokens. This suggests that we can use a maximum token length of 256 without potentially sacrificing much performance.

[1]https://github.com/AndrewHarrison/ir2

Furthermore, all questions contain 34 or fewer tokens, which poses no problems. We thus choose to use a maximum token length of 256.

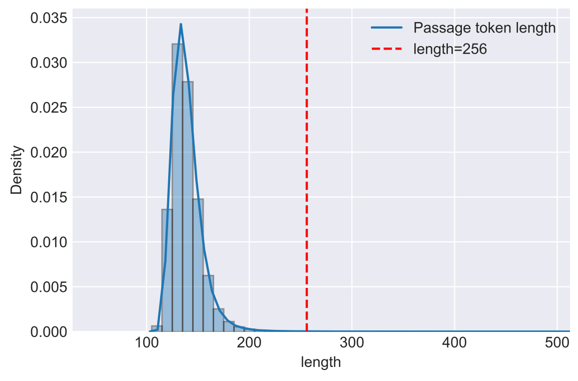

Figure 1: Passage token length distribution.

4.2 Encoders

In our experiments, we utilize different Transformer-based encoder models. The models used are from the HuggingFace library of Wolf et al. [25]. The following pre-trained models are used in our experiments:

- BERT: *bert-base-uncased*
- DistilBERT: *distilbert-base-uncased*
- TinyBERT: *huawei-noah/TinyBERT_General_4L_312D*
- ELECTRA: *google/electra-small-discriminator*

4.3 Evaluation Metrics

In this work, we evaluate the models on two different aspects. The first is the retrieval performance, this shows how well the models can retrieve the correct passage for the given query. The second aspect is the computational efficiency, this shows how efficient the model is during fine-tuning, while building the FAISS index, and at inference time when performing the retrieval task.

4.3.1 Retrieval Performance. We measure retrieval performance with two different metrics. First is top-k retrieval, which is defined as the fraction of questions that retrieved the correct passage in the top-k retrieved documents. We evaluate for $k = 1$ (Acc@1), $k = 20$ (Acc@20), and $k = 100$ (Acc@100). The second evaluation metric is the Mean Reciprocal Rank (MRR), which is a more fine-grained ranking complement. It calculates the reciprocal of the rank at which the first relevant document was retrieved. In our experiments, we take our cut-off point to be at 100. This means that the MRR is calculated for the top 100 retrieved documents and if no relevant document is found its reciprocal rank is set to 0.

4.3.2 Computational Efficiency. Our measure of computational efficiency is execution time. We examine three different aspects of DPR efficiency. The first is fine-tuning time, here we measure the total execution time of training for 40 epochs without the tokenization of the data. This indicates how efficient the encoder is at learning accurate embeddings. The second aspect is the execution time for encoding the passages and creating a FAISS index during evaluation. Here we measure the total time it takes to embed and index the passages during evaluation. This indicates how efficient the encoder is at encoding a large number of passages and building an index. Lastly, we consider the inference time. Here we measure the average time for the FAISS index to retrieve the top-100 passages for a given query. Note that this does not include the embedding of the question. This indicates how efficient the index is at retrieving the relevant passages for a question. Moreover, we also report on the standard deviation of the inference time.

Table 1: Model hyperparameters.

Model	Embedding Sizes	Batch Size
BERT	{1536, *768*, 704, 640, 512, 384}	16
DistilBERT	{1536, *768*, 704, 640, 512, 384}	64
TinyBERT	{1536, 768, 512, 384, *312*}	128
ELECTRA	{1536, 768, 512, 384, *256*}	64

4.4 Evaluation Method

Due to limited computational capacity and time-based usage constraints of 48 hours, we have not been able to evaluate on the full set of Wikipedia passages used by Karpukhin et al. [12]. As described in their paper, computing the dense embeddings and building a FAISS index can be very time-consuming, especially on a single GPU. Therefore, we have decided to evaluate only on the passages in the development set instead of the full Wikipedia passage dump. This means that our FAISS index consists of the answer passage, which is defined as the first positive passage, and all of the hard negative passages, for each of the questions in the development set. This results in a total of 935,159 passages. This will also likely lead to a slight inflation in performance, with higher inflation expected the larger the top-k retrieval metric used.

4.5 Hyperparameters

For the experiments, we used the model hyperparameters as described in Table 1. For each of the models, multiple embedding sizes are tested. The embedding size in *italics* denotes the standard embedding size of the models. Since we are also evaluating the model's efficiency, we wanted to use batch sizes that fully utilize the memory available on the GPU. We were limited to a single Nvidia Titan RTX 24GB GPU. We stayed within the powers of 2 regime (i.e. 2, 4, 8, 16, etc.) for the batch sizes and picked the maximum batch size that the GPU could handle.

All models are trained for 40 epochs with a learning rate of 10-5 using the AdamW optimizer [18], a dropout rate of 0.1, and a linear scheduler with 100 warm-up steps. These hyperparameters are in line with those used by Karpukhin et al. [12].

During evaluation, we use an HNSW-Flat FAISS index on the CPU with the inner product as the similarity measure. We store 512 neighbours per node and use a construction time search depth of 200 with a search depth of 128. This is again according to what is used by Karpukhin et al. [12]. For embedding the questions and passages during evaluation, we use a batch size of 512 which is constant for all models.

5 RESULTS

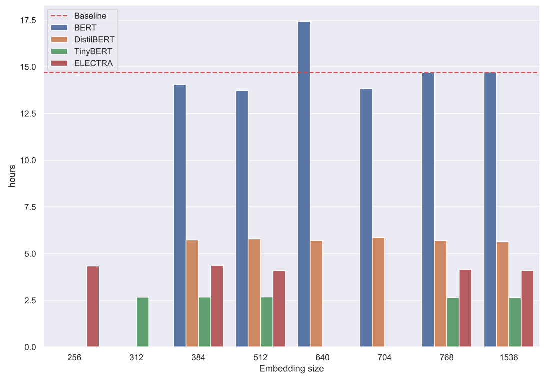

Figure 2: Total training time compared to BERT encoder with embedding size = 768.

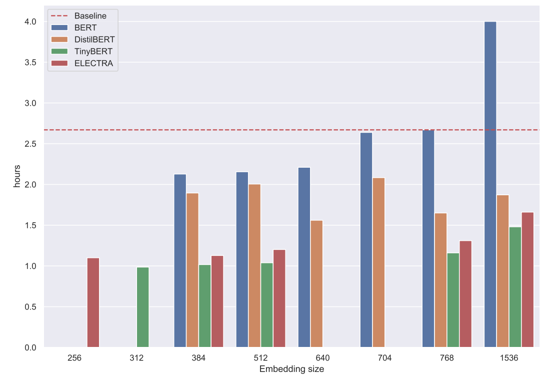

Figure 3: Passage encoding + indexing time compared to BERT encoder with embedding size = 768.

Before turning to the retrieval performance evaluation metrics, we first examine the different models' efficiency across training time, encoding and indexing time, and inference time. The results in Figure 2 show the training time in hours for the different models across a range of different embedding sizes. Note that the base embedding sizes, i.e. without projection, are 256 for ELECTRA, 312 for TinyBert, and 768 for BERT and DistilBERT. The BERT model with base embedding size is used as the baseline (red dotted line). We can see that the training times for DPR using BERT as the encoder model is an order of magnitude larger than the other models. With TinyBERT having the fastest training, at half the time

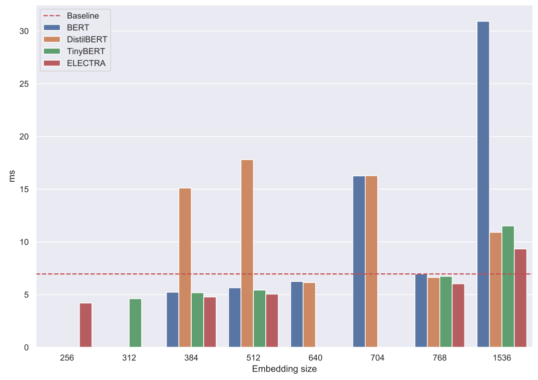

Figure 4: Inference time average compared to BERT encoder with embedding size = 768.

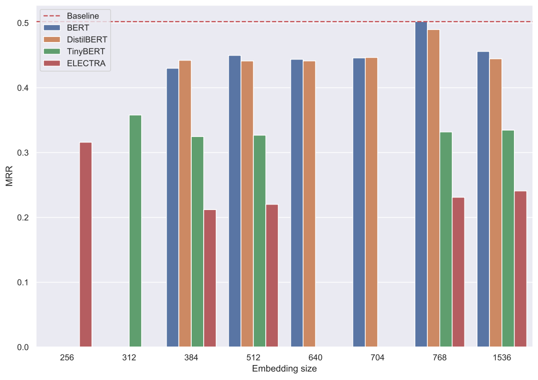

Figure 5: Mean reciprocal rank (MRR) compared to BERT encoder with embedding size = 768.

of DistilBERT, with ELECTRA being in-between TinyBERT and DistilBERT. While projecting the model embeddings up or down requires an additional linear layer attached after the final hidden layer trained during fine-tuning, its impact on the total training time is minimal, given its relatively small parameter count.

The BERT 640 embedding size is the first of several anomalous results we identified as being caused by different compute nodes in the high-performance computing cluster used. This is addressed further in our discussion, but for assurance on our core result, we ran the non-training (i.e. indexing and inference) efficiency evaluations of BERT and DistilBERT at embedding size 768, with their max batch sizes, 5 times each to ensure the difference was not simply noise. In the subsequent figures and tables, we report the median result for each of these two, while all other results are based on a single evaluation run.

In Figure 3 we see the time taken in hours to encode the passages of the development set (935,139 passages), using different

Table 2: Retrieval efficiency for four encoders with different embedding sizes. The Baseline is BERT encoder with embedding size of 768. Metrics are Total training time(TTT), Passage encoding and indexing time(PEI), Inference time average(ITA) and standard deviation. The standard model embedding sizes are shown in *italics*.

Encoder	ES	TTT(h)	PEI(h)	ITA(ms)	ITA(S.D.)
BERT	1536	14:43	4:00	30.9	10.2
	768	14:42	2:40	6.9	1.8
	704	13:50	2:38	16.3	5.1
	640	17:26	2:12	6.3	1.6
	512	13:44	2:09	5.7	1.4
	384	14:03	2:07	5.2	1.3
DistilBERT	1536	5:38	1:52	10.9	3.5
	768	5:42	1:39	6.7	1.8
	704	5:52	2:05	16.3	5.7
	640	5:42	1:33	6.1	1.7
	512	5:47	2:00	17.8	6.2
	384	5:44	1:53	15.1	5.1
TinyBERT	1536	**2:38**	1:28	11.5	3.6
	768	**2:38**	1:09	6.7	1.8
	512	2:40	1:02	5.4	1.3
	384	2:40	1:01	5.2	1.3
	312	2:40	**0:59**	4.6	1.1
ELECTRA	1536	4:05	1:39	9.3	2.5
	768	4:09	1:18	6.0	1.5
	512	4:05	1:12	5.0	1.2
	384	4:22	1:07	4.8	1.1
	256	4:20	1:06	**4.2**	**0.9**

embedding sizes for each model. Again there is a lot of noise in the data, with assurance in 768 BERT and DistilBERT due to using the median of 5 runs. We can see that DistilBERT is over an hour faster than BERT. There is also a pattern of increasing time taken with larger embedding sizes, due to the time taken to encode embeddings and build the index when using larger vectors. TinyBERT is again the fastest, with ELECTRA sitting between TinyBERT and DistilBERT.

The last efficiency metric is inference time, as shown in Figure 4, and this shows the average inference time per question for the model across all of the questions in the development set, shown in milliseconds. The DistilBERT and BERT results are again noisy. However, we can still see the pattern of increased inference time as embedding size increases, with small to no difference between models. ELECTRA has the shortest inference time of all the models. And there is no real difference between the median BERT and DistilBERT models at 768 embedding size. For an overview of the exact efficiency results, see Table 2.

Turning to the retrieval performance metrics, we see in Figure 5 which shows the MRR across the different model embedding sizes, that there is a clear distinction between DistilBERT and the other alternate models. DistilBERT reaches almost the same performance as BERT, while TinyBERT and ELECTRA show a significant drop-off in performance. Note that the difference between TinyBERT and ELECTRA is not as large in their base embedding size, indicating ELECTRA's performance suffers more from the upwards projection of the embeddings. This is also the case for BERT and DistilBERT, which both suffer clear performance drops when either projected up or down in their embedding sizes. With the base embedding sizes for BERT and DistilBERT representing their best respective performances. At this base size, there is only a mininal drop-off in DistilBERT performance versus the performance of BERT. The full performance metrics are shown in Table 3 where we see that the drop in Acc@1 between BERT and DistilBERT is slightly larger. However, DistilBERT shows slightly better performance in Acc@20 and Acc@100. As an aside, regarding the embedding sizes, we can see for the smaller models that while projecting to larger sizes drops performance versus their base size, the larger the embedding size projected to during fine-tuning, the worse than base performance still appears to be slowly improving. This indicates that the fine-tuning of larger projections improves performance over smaller projections, meaning it is capturing some learning.

For reproducibility's sake, we compared our model with a BERT encoder to the original model trained by Karpukhin et al. [12]. Specifically, we use the *facebook/dpr-question_encoder-single-nq-base* model from the Huggingface library. We evaluate both models using our evaluation method, which is using only the 935,159 dev set phrases rather than the full 21m Wikipedia phrases, to make it a fair comparison. The comparison is summarized in Table 4, with *Original DPR* denoting the model that is fine-tuned by Karpukhin et al. [12] and provided in HuggingFace, while *Our BERT* denotes the DPR model with BERT encoder that we have fine-tuned. The results shows a significant gap between the original DPR model and our model with BERT encoder. In our reproduction, we attempted

Table 3: Retrieval Performance for four encoders with different embedding sizes. The baseline is BERT encoder with ES(Embedding Size) of 768, and standard model embedding sizes are shown in *italics*.

Encoder	ES	Acc@1	Acc@20	Acc@100	MRR
BERT	1536	33.6	78.1	90.3	.456
	768	**38.1**	82.3	92.4	**.502**
	704	32.7	77.5	89.4	.446
	640	32.8	77.3	89.4	.444
	512	33.4	76.8	89.0	.450
	384	31.4	75.9	89.0	.430
DistilBERT	1536	32.3	79.0	90.5	.445
	768	36.3	**83.5**	**93.3**	.490
	704	32.9	78.3	90.1	.447
	640	32.0	78.1	89.5	.441
	512	32.0	78.2	90.1	.441
	384	32.3	78.1	89.8	.442
TinyBERT	1536	23.0	65.6	81.3	.335
	768	22.9	65.0	81.3	.332
	512	22.2	64.7	80.6	.327
	384	22.0	64.5	81.0	.325
	312	24.7	68.2	83.8	.358
ELECTRA	1536	16.0	50.2	68.7	.241
	768	14.0	49.0	68.2	.231
	512	14.0	48.0	66.8	.220
	384	13.6	46.8	66.5	.212
	256	22.3	60.2	77.3	.316

Table 4: Reproducibility comparison to original DPR model.

Model	ES	Acc@1	Acc@20	Acc@100	MRR
Original DPR	768	50.4	88.3	94.9	.614
Our BERT	768	38.1	82.3	92.4	.502

to stay as close to the given hyperparameters provided. However, as mentioned before, due to the computational constraints, we had to use smaller batch sizes (16 versus 128) during training. We believe that our lower batch sizes account for the difference in performance, as Karpukhin et al. [12] note that batch size plays a large role in the contrastive learning objective of DPR.

To assess this further, we then reduced BERT's batch size to 8, to observe whether there was a further performance drop. This was in addition to lowered batch sizes for all of the alternate models. Specifically, we made the models comparable across a batch size of 64 for the 3 smaller models, and 16 to compare all models (i.e. including BERT). Details on model performance when using reduced batch sizes are shown in Table 5. We can see that there is indeed a drop in BERT's performance when the batch size is lowered further still to 8, supporting our intuition. Indeed this drop is constant across DistilBERT, and ELECTRA, with only TinyBERT showing comparable performance when dropped from 128 to 64, but then a drop when further reduced to a batch of 16.

6 DISCUSSION & CONCLUSION

Our results show that DistilBERT is a viable alternative to the BERT encoders currently used in DPR, as while there is a small drop in the Acc@1 and MRR for DistilBERT versus BERT when both models are using their maximum batch size of 64 and 16 respectively, the Acc@20 and Acc@100 shows no decline. While the inference time execution time remains approximately the same, the 61% decrease in training time reducing approximately 14:42 hours down to 5:42 hours, and the 38% drop in offline embedding and index building, from 2:40 hours down to 1:39 hours, are significant computational savings. However, when we evaluate both models on a batch size of 16, the performance metrics for DistilBERT are slightly worse and index building time shows only a 10 minute time saving, while the training time is still less than half of BERT. But, we believe that the increased batch size that DistilBERT is able to utilize should also be considered an advantage of the model. Due to its reduction in model parameters, larger batch sizes can fit in the same GPU memory.

One unexpected finding was that ELECTRA achieved the lowest performance, while Clark et al. [4] claimed that the ELECTRA small model we used, achieved the same performance as BERT and outperformed TinyBERT. It may be that ELECTRA, uniquely in our alternate models, keeping 12 hidden layers but using only 4 attention heads in each layer, adversely impacts this retrieval task. We surmise however that both the small input embedding size of 128, and the hidden layer size of 256, being the smallest across all our models, meant that the ELECTRA model was unable to adequately capture the full passage content, to then be useful for

Table 5: Effect of batch size on performance and efficiency for four encoders with optimal embedding size.

Encoder	ES	Batch Size	Acc@1	Acc@20	Acc@100	MRR	PEI(h)
BERT	768	16	**38.1**	**82.3**	**92.4**	**.502**	14:42
	768	8	36.1	80.0	90.9	.478	17:20
DistilBERT	768	64	<u>36.3</u>	<u>83.5</u>	<u>93.3</u>	<u>.490</u>	5:42
	768	16	35.5	81.2	91.9	.476	7:03
TinyBERT	312	128	24.7	68.2	83.8	.358	**2:40**
	312	64	25.4	67.7	82.8	.360	2:45
	312	16	24.0	65.0	80.3	.343	3:22
ELECTRA	256	64	22.3	60.2	77.3	.316	4:20
	256	16	20.7	56.6	73.2	.295	5:50

semantic matching and retrieval in this open-domain QA setting. This lowest performance remained when differing batch sizes were adjusted for, specifically, even when reducing TinyBERT to a batch size of 64 inline with ELECTRA, TinyBerts 312 embedding size still outperformed ELECTRA (note TinyBERT keeps 12 attention heads per layer but reduces to 4 hidden layers). Both TinyBERT and ELECTRA performances were a significant drop from DistilBERT and BERT models, with their 768 embedding size. This is a relevant finding for open-domain oriented retrieval tasks, as even with only moderately sized passages of up to 256 tokens, the embedding sizes in the ranges examined herein are still highly impactful on performance. Thus DistilBERT, maintaining the same embedding size as the BERT model, was the only model that suffered little to no performance drop. Thus, to answer the research question of which pre-trained model provides the best trade-off between efficiency and document retrieval performance, we can convincingly say that this is DistilBERT. DistilBERT's performance is on par with BERT while being significantly more efficient.

For our second research question, we hypothesized that using a final projection layer that projects the embedding size up or down would not significantly impact training times due to the relatively small parameter count of this layer. Our results supported this evidence and showed no significant increase in training times. We also hypothesized that for the large embedding models such as BERT and DistilBERT, compressing the information contained in the embeddings down further would be at some performance cost, but at the benefit of faster embedding and indexing, and perhaps inference. While it did appear to slightly reduce these execution times, the impact on performance was much larger than expected which meant these models were not a viable option. For the smaller embedding size models, we expected that the fine-tuning of an additional upwards projection layer would add capacity and perhaps achieve better model retrieval performance, at a slight efficiency cost. The slight efficiency cost is evidenced, but there is also a marked decrease in the retrieval performance of the smaller model, particularly ELECTRA. We hypothesise that the extra capacity of the projection layer being only fine-tuned and not part of the pre-training results in more noise being added than signal by addition of this this extra layer. The reason why ELECTRA suffers the greatest performance drop when upward projected, is likely due to the ELECTRA (uniquely among our models) already projecting up from input embedding size of 128 to a hidden layer size of 256, with our subsequent up projection again during fine-tuning curtailing performance still further. Thus, answering the question of how embedding sizes affect the trade-off between efficiency and document retrieval performance, we can say that any type of final projection, either up or down, has a deleterious effect on this trade-off.

While our reproduction revealed a significant performance gap between our DPR model with BERT encoder and the model trained by Karpukhin et al. [12], we do not believe this is indicative of lack of validity in their work, nor ours. Our further investigations firmly indicate that the smaller batch sizes resulting from our computational constraints (16 rather than 128) significantly impacted the contrastive learning effects of in-batch negatives. This further confirms the claim made by Karpukhin et al. [12] about the importance of the batch sizes with in-batch negative contrastive training.

The DPR architecture was produced by Facebook, BERT and ELECTRA by Google, TinyBERT by Huawei, and DistilBERT by HuggingFace, all organisations with moderate to significant computing resources and budgets. However, we were far more constrained in terms of computing resources and time. Thus, a shortcoming of our work was the smaller batch sizes necessitated by hardware limitations during our research. Given the importance of larger batch sizes due to DPR's usage of in-batch negatives, the per-step contrastive learning is likely significantly reduced. We witnessed this effect when we compared our DPR model with BERT encoder with the DPR model that was trained by Karpukhin et al. [12]. The second shortcoming of our research is also a result of hardware limitations, namely that we had to use the passages from the development set to construct our passage index during inference. We could not utilize the full 21 million passages from the Wikipedia dump and therefore opted to use the 935,159 passages from the development set. This makes the task significantly easier and therefore yielded better performance scores than those reported in the paper by Karpukhin et al. [12] when using their pre-trained model. Thirdly, our research was conducted on a high performance computing cluster, and whilst compute nodes are restricted to running a single job by the scheduler, there appears to be variance between the nodes peak efficiencies across nodes, and within nodes at different times of the day. Indicating shared Input/Output (I/O) constraints shared across multiple nodes. Therefore, producing all of the results on the exact same hardware in a more highly controlled setting would result in less variation and more reliable results.

As future work, research groups with larger computational resources should validate our results on DistilBERT, when using larger batch sizes, and the full 21 million passage Wikipedia dump for evaluation. Future research should also investigate whether larger ELECTRA models (e.g. embedding and hidden size 1024, 24 hidden layers, each with 16 attention heads), which have been reported as outperforming BERT [4] on the same volume of pre-training, are able to achieve stronger performance on the open-domain QA retrieval task examined here. As they may still be more efficient than BERT in the execution times assessed here while achieving closer performance levels on the MRR and Acc@K metrics herein. Furthermore, if results of ELECTRA remain weak, this could indicate that the specific pre-training task being used, with the discriminator identifying replaced tokens in a sequence, is ill-suited to open-domain retrieval tasks, as compared to the MLM pre-training task. Penultimately, further architectural adjustments to language models performed during pre-training rather than only using a final fine-tuned projection layer, is another avenue to investigate in making dense passage retrieval more efficient. While this will require the significant computational resources of industry players, given Huawei's creation of TinyBERT, and HuggingFace's creation of DistilBERT, both to address their specific computational efficiency requirements, such future work is not unimaginable, and would be a valuable contribution to the research community. Lastly, in the bi-encoder architecture there are 2 encoder models training jointly. An avenue of research would be to explore mixed models i.e. using a smaller architecture on only question encoder or only the passage encoder. This would determine whether the efficiency-effectiveness trade-off could be adjusted through cheaper encoding of questions or passages alone.

In conclusion, our analysis of smaller encoder models has revealed that DistilBERT is a viable alternative to using BERT in Dense Passage Retrieval (DPR), and achieves comparable results to BERT in retrieval performance, while having significantly lower fine-tuning training time, and embedding creation and indexing time. This will allow the use of the DPR model, with DistilBERT, in use cases where corpora are highly dynamic, requiring regular model re-training and offline embedding and index creation, or on edge devices with limited compute, where previously DPR may have been too computationally expensive.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).

[2] Oren Barkan, Noam Razin, Itzik Malkiel, Ori Katz, Avi Caciularu, and Noam Koenigstein. 2020. Scalable attentive sentence pair modeling via distilled sentence embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3235–3242.

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051* (2017).

[4] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[6] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129, 6 (2021), 1789–1819.

[7] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909* (2020).

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[9] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. In *International Conference on Learning Representations*.

[10] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 4163–4174.

[11] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *CoRR* abs/1702.08734 (2017). arXiv:1702.08734 http://arxiv.org/abs/1702.08734

[12] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[13] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 39–48.

[14] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a Benchmark for Question Answering Research. *Transactions of the Association of Computational Linguistics* (2019).

[15] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *CoRR* abs/1909.11942 (2019). arXiv:1909.11942 http://arxiv.org/abs/1909.11942

[16] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6086–6096.

[17] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained transformers for text ranking: Bert and beyond. *arXiv preprint arXiv:2010.06467* (2020).

[18] Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. *CoRR* abs/1711.05101 (2017). arXiv:1711.05101 http://arxiv.org/abs/1711.05101

[19] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[20] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv:1606.05250 [cs.CL]

[21] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[22] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[24] Ellen M. Voorhees and Dawn M. Tice. 2000. The TREC-8 Question Answering Track. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*. European Language Resources Association (ELRA), Athens, Greece. http://www.lrec-conf.org/proceedings/lrec2000/pdf/26.pdf

[25] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

[26] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*.