An Inspection of the Reproducibility and Replicability of TCT-ColBERT

Xiao Wang
University of Glasgow
Glasgow, Scotland, United Kingdom
x.wang.8@research.gla.ac.uk

Sean MacAvaney, Craig Macdonald, Iadh Ounis
University of Glasgow
Glasgow, Scotland, United Kingdom
first.last@glasgow.ac.uk

ABSTRACT

Dense retrieval approaches are of increasing interest because they can better capture contextualised similarity compared to sparse retrieval models such as BM25. Among the most prominent of these approaches is TCT-ColBERT, which trains a light-weight "student" model from a more expensive "teacher" model. In this work, we take a closer look into TCT-ColBERT concerning its reproducibility and replicability. To structure our study, we propose a three-stage perspective on reproducing the training, inference, and evaluation of model-focused papers, each using artefacts produced from different stages in the pipeline. We find that — perhaps as expected — precise reproduction is more challenging when the complete training process is conducted, rather than just inference from a released trained model. Each stage provides the opportunity to perform replication and ablation experiments. We are able to replicate (i.e., produce an effective independent implementation) for model inference and dense indexing/retrieval, but are unable to replicate the training process. We conduct several ablations to cover gaps in the original paper, and make the following observations: (1) the model can function as an inexpensive re-ranker, establishing a new Pareto-optimal result; (2) the index size can be reduced by using lower-precision floating point values, but only if ties in scores are handled appropriately; (3) training needs to be conducted for the entire suggested duration to achieve optimal performance; and (4) student initialisation from the teacher is not necessary.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Dense Retrieval; Knowledge Distillation; Reproducibility

ACM Reference Format:
Xiao Wang and Sean MacAvaney, Craig Macdonald, Iadh Ounis. 2022. An Inspection of the Reproducibility and Replicability of TCT-ColBERT. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531721

SIGIR '22, July 11–15, 2022, Madrid, Spain

1 INTRODUCTION

For decades, the ranking task in information retrieval has been dominated by sparse retrieval models such as BM25, where the relevance score measured relies on a lexical matching between the queries and documents. However, considering only exact term matching while measuring query-document similarity can hinder effectiveness for queries and documents containing polysemous or synonymous terms. To better cope with this, the BERT-based [4] cross-encoder reranking models have been proposed [11, 17, 22]. Such models increase effectiveness by reranking a set of candidate documents obtained from a sparse retrieval model.

Going further, and to address any limit in the recall achieved by a sparse retrieval model, end-to-end dense retrieval approaches are of increasing interest. By encoding queries and documents (independently) into dense vectors, these models can score the similarity of text even in the absence of lexical matches. This setting also allows for efficient *dense* retrieval using approximate nearest neighbour (ANN) search algorithms on specialised indices, such as those provided by the FAISS library [8]. Two families of end-to-end dense retrieval models have become apparent in the literature, based on the nature of representation for the queries and documents: *single representation* models (e.g. DPR [9] and ANCE [30]) encode each query or document into a single dense representation, which allows a simple similarity function, usually the dot-product, to compute relevance scores. In contrast, in a *multiple representation* dense retrieval model (e.g. [10]), each query or document is encoded into many embeddings, one for each token. The final score for a document is obtained by applying a MaxSim operator for each query embedding over the document embeddings. However, while a multiple representation dense retrieval model can be more effective [20], its larger index representation is a significant drawback, as it must be loaded into memory for sufficiently efficient retrieval. For this reason, single representation dense retrieval is considered to be more attractive, and is subject to much ongoing research.

To this end, one possible solution is to distill the learned knowledge from a more effective retrieval model to a single representation dense retrieval model [6, 7, 14, 15]. Indeed, Hofstätter et al. [6] attempted to distill knowledge of a complex cross-encoder into a single-representation model. Separately, Lin et al. developed an approach called TCT-ColBERT [14, 15], which aims to distill knowledge from a multiple-representation model (ColBERT) into a single representation model. Indeed, since multiple-representation and single-representation dense retrieval models capture different knowledge and measure relevance scores in different ways, knowledge distillation allows the smaller model to learn relevant patterns from the larger model. TCT-ColBERT was applied by several groups

in TREC 2021 [2]. For this reason, we consider it to be worthy of a further in-depth reproduction study.

In this paper, we take a closer look at TCT-ColBERT, in terms of reproducibility and replicability. In particular, while ACM describes [24] reproducibility as "different team, same experimental setup", and replicability as "different team, different experimental setup", we interpret this in a more nuanced manner, by carefully considering different levels of reproducibility. Indeed, while there has been reproducibility of TCT-ColBERT at a level of *"do the author-provided TCT-ColBERT encoded queries and indexed documents attain the expected effectiveness?"*, we argue that the main novelty in TCT-ColBERT is its knowledge distillation training regime, and hence for our science to be able to progress, it is important to reproduce and replicate the training regime. Moreover, due to the complex nature and expensive manner in which TCT-ColBERT was trained (using Google TPUs, with batch size 96, for 500k batches), we investigate some particular choices made by the authors to test whether such an expensive training process is necessary.

To structure our reproduction and replication efforts, we perform our study in three stages, aligned with training, inference, and evaluation. First, in what we call *last-metre reproduction*, we use the pre-indexed documents and pre-computed queries released by the paper's authors. This stage establishes whether the nearest-neighbour search from a pre-existing tool produces comparable results to the original paper. Providing further value, the following *last-mile reproduction* stage performs inference, i.e., computing query and document vectors from released model checkpoints. This stage establishes whether the released models are able to produce comparable results to the original paper. Finally, the *complete reproduction* stage performs the whole process, including training. We find that reproduction and replication become more challenging as we advance from last-metre to complete reproduction. Nevertheless, each stage provides us with opportunities to ablate the model, filling in gaps in knowledge left by the original paper.

Among our interesting observations, we find that: (1) the TCT-ColBERT model can function as an inexpensive re-ranker, establishing a new Pareto-optimal result; (2) the index size can be reduced by using lower-precision floating point values, but only if ties in scores are handled appropriately; (3) training needs to be conducted for the entire suggested duration to achieve optimal performance; and (4) student initialisation from the teacher is not necessary.

The structure of the remainder of this paper is described as follows: Section 2 provides some necessary background on single and multiple dense retrieval; Section 3 proposes a framework for discussing a number of aspects when reproducing dense retrieval models; Section 4 provides a summary of TCT-ColBERT. Section 5 describes the aims of our study and highlights the questions we aim to answer, categorised as last-metre, last-mile and complete. We experiment to address these categorised questions respectively in Sections 6, 7, & 8. Concluding remarks follow in Section 9.[1]

2 BACKGROUND

In the following, we describe in detail both multiple- and single-representation dense retrieval models.

[1] Virtual Appendix: https://github.com/Xiao0728/TCT-Repro-VirtualAppendix
Data: http://dx.doi.org/10.5525/gla.researchdata.1276

2.1 Multiple representation Dense Retrieval

We first focus on the working principle of the multiple representation based ColBERT [10] model, which is employed as the teacher model of TCT-ColBERT. In a multiple representation dense retrieval paradigm, all tokens of the query or document[2] are encoded by a BERT-style encoder. More formally, a query q is firstly segmented and appended with additional [MASK] tokens to a fixed length, typically set to 32. Then the input query tokens are encoded into a set of $|q|$ embeddings $\phi_q = \{\phi_{q_1}, \ldots, \phi_{q_{|q|}}\} = \text{Encoder}_\text{Q}(q)$. Similarly a document d can be encoded into a set of $|d|$ embeddings $\phi_d = \{\phi_{d_1}, \ldots, \phi_{d_{|d|}}\} = \text{Encoder}_\text{D}(d)$. For model-specific design of ColBERT encoding process, we refer readers to [10] for full details.

Thus, the relevance score between a query and a document can be calculated as follows:

$$s(q,d)_{MaxSim} = \sum_{i=1}^{|q|} \max_{j=1,\ldots,|d|} \phi_{q_i}^T \phi_{d_j} \tag{1}$$

ColBERT is normally trained on the *triple* files, where each training sample consists of a query q, a positive document d^+ and a negative document d^-. The model is optimised by minimising the pair-wise cross-entropy loss computed over the MaxSim scores of a query between its positive document and negative document (in-batch negatives – discussed further below – can also be used for improved effectiveness [26]).

After the model is trained, the document embeddings are pre-computed and indexed, while also compressed into a FAISS index [8] that supports approximate nearest neighbour search. In the retrieval phase, ColBERT usually performs in a late interaction paradigm as the FAISS compression makes their scores inaccurate for effective ranking. Hence, for each query embedding, it performs the first stage approximate nearest neighbour (ANN) search to produce a set of candidate documents, followed by a second exact scoring stage.

ColBERT has been shown to be effective (more so than the ANCE single-representation model, particularly for complex, multi-aspect information needs [20]) and extensible (e.g. embeddings in the top-ranked documents can be added to the query as a form of pseudo-relevance feedback (PRF) [28]). Furthermore, the per-token embeddings are interpretable [20], and exhibit correlations with IDF [5, 27]. On the other hand, access to the voluminous accurate document embeddings in main memory required for the efficient application of the second stage is a major drawback of ColBERT. While a followup work, ColBERT v2 [26], demonstrated that quantised embeddings could reduce the space consumption, the need for the two-stage scoring stages remains a significant drawback for such multiple representation models, and makes single representation more attractive.

2.2 Single representation Dense Retrieval

In the single representation dense retrieval paradigm, each query or document is encoded into one single embedded representation. A typical implementation separately applies the encoder to documents and queries, and thus such retrieval models are usually called as *Bi-Encoder* or *dual-encoder*. In doing so, a single pre-indexed representation for each document can be indexed into an approximate

[2] In the following, we use the phrases document and passage interchangeably.

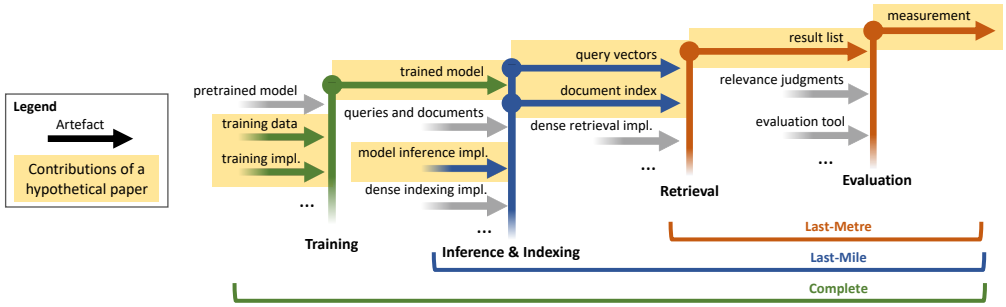

Figure 1: Our conceptual framework for reproducing and replicating various stages of a dense retrieval model. Each arrow represents an artefact (e.g., code, data, model, etc.). Arrows can have dependent artefacts that are shown directly in the figure or hidden to save space (e.g., training data may come from yet another trained model).

nearest neighbour search index (e.g. as provided by the FAISS library), resulting in a considerably smaller index size compared to ColBERT. Further, due to the smaller size of the embedded representations, compression may not be required for nearest neighbour search (e.g. [30]), and hence there is no need for an accurate re-ranking stage.

Similar to multiple representation dense retrieval, a contextual language model, such as BERT, which has considerable pre-trained knowledge, is employed as the query and document encoder, thus the encoded embeddings are capable of capturing contextualised information. The query encoder encodes the query q into a single query embedding $\psi_q = \text{Encoder}_\text{Q}(q)$ and the document d into a single document embedding $\psi_d = \text{Encoder}_\text{D}(d)$. The single representation of the query or document can be obtained in different ways. For instance, most Bi-Encoder dense retrieval models adopt the BERT [CLS] embedding produced by the query or document encoder as the input representation. In TCT-ColBERT, the authors applied a pooling layer to combine the embeddings obtained from the various input tokens, namely `AvgPool` or `MaxPool` (the original work used `AvgPool` as default). Thus, the input query q and document d can be represented as $\psi_q = \texttt{AvgPool}\{\phi_{q_1}, \ldots, \phi_{q_{|q|}}\}$ and $\psi_d = \texttt{AvgPool}\{\phi_{d_1}, \ldots, \phi_{d_{|d|}}\}$, respectively.

To make the query encoder and document encoder produce better representations, such that the query and relevant document embeddings are similar and the query and document embeddings of non-relevant documents are positioned far apart from each other, the BERT-based query encoder and the document encoder are further fine-tuned. Usually, each training instance consists of a *triple* $\langle q, d^+, d^- \rangle$. During training, the model is optimised via the cross-entropy loss between the positive and negative pair scores. After obtaining the trained model, each document is encoded into a single embedding using the trained model and stored within an embedding-based nearest neighbour index. During retrieval, for an input query, the trained model firstly encodes it into a single query representation and a simple dot-product searching function is employed to calculate the relevance score between the query representation and each document representation in the index.

Yet the training of effective single-representation models is challenging: in particular, a model trained for separating relevant and non-relevant (aka. negative) documents using cross-entropy may not be able to create embeddings that easily distinguish among relevant, "could be" relevant and non-relevant documents [25, 30]. As a result, many initial dense retrieval models required significant GPU training, while there has also been significant work towards negative sampling. For example, DPR [9] required 8 GPUs for training; RocketQA [25] is trained on upto 8 x V100 GPUs with cross-batch negative sampling with batch size of 512; ANCE [30] similarly used multiple GPUs, and samples negatives based on an nearest neighbour index from a previous checkpoint. On the other hand, *in-batch negatives* are commonly used to supplement the training triples by using documents from other queries in the batch as additional non-relevant documents. An alternative approach has been to distill knowledge from a more effective model into a single-representation model [7, 14, 15]. One such approach, TCT-ColBERT, is the focus of this paper, for which we conduct the reproduction and replication studies. In the following, we describe a framework for reproduction, replication, and ablation that allows us to clearly delineate our contributions. Later, in Section 4, we describe TCT-ColBERT.

3 A FRAMEWORK FOR MULTI-STAGE REPRODUCTION AND ABLATION

The ACM Reproducibility guidelines [24] place a high importance on produced artefacts; the criterion for a "Results Reproduced" badge are "The main results of the paper have been obtained in a subsequent study by a person or team other than the authors, using, in part, artefacts provided by the author." However, we argue that this criterion is insufficient for much recent work in IR that involves pipelines of dependent artefacts.

To make the problem more concrete, we consider a simple hypothetical paper that proposes an approach for training a dense retrieval model. The process consists of four stages: (1) model training, (2) inference & dense indexing, (3) retrieval from the dense index, and (4) evaluation. Figure 1 shows that each of these steps can make use of some artefacts from the previous step (e.g., a trained model), also introducing new artefacts (e.g., implementations, data, etc.). As an example, the inference & indexing stage uses the trained model from the prior stage, an inference implementation for the model (often distinct from the training implementation), query and document data, and a dense indexing implementation. It produces

a set of vectors representing each query and a dense index of the documents, which are used by the subsequent retrieval step.

We find that many reproduction efforts – such as those documented on the TCT-ColBERT reproduction log[3] – focus almost exclusively on the final retrieval and evaluation steps. Though reproducing results at this stage has some value (and meets the ACM criteria of reproduction), we posit that this barely represents a reproduction of the *contributions* of a typical dense retrieval paper (here, the training approach). All that is involved in the final stages are running an established dense retrieval tool (e.g., FAISS [8]), and using an established evaluation tool (e.g., `trec_eval`) with a standard set of relevance assessments. In some sense, it is only testing the reproducibility of the dense retrieval and evaluation tools, rather than the paper's main contributions. We refer efforts that use post-inference artefacts as *last-metre reproductions* because only the final (often trivial) steps are examined. Another problem with last-metre reproduction is that it offers relatively little room for performing interesting ablations. For instance, a researcher could try a different similarity measure (e.g., cosine similarity rather than dot product), but this is unlikely to be effective because the model was not trained with the alternative measure.

Some reproduction efforts go slightly further. The above reproduction log[3] gives a note about how to perform "on-the-fly" encoding of queries (though it is unclear whether doing so is required to log a successful reproduction). By performing some inference, there are more opportunities for performance to diverge.[4] There exist even more opportunities for ablation when the document inference & indexing procedures are also considered, as we find in Section 7. We refer to efforts that use the final trained model checkpoints as *last-mile reproduction*.[5]

Finally, we consider the *complete reproduction* of a result to be one that covers all the intermediate artefacts central to the contribution of a paper. In most cases, this will include model training, which can be one of the most involved and challenging processes.

Throughout this paper, we consider results to be reproduced or replicated if we can achieve similar metrics to those reported by Lin et al. [14]. An alternative approach would have been to test the rank-biased correlation of the result lists themselves, using a tool like `repro_eval` [1]. However, this is not possible for our study because the original result files for Lin et al. [14] are not available.

Reproduction is an important aspect of science, and to build upon the work of others, we first must be able to reproduce it. As such, last-metre and last-mile reproductions alone are insufficient. However, we certainly do not argue against the release of intermediate artefacts, such as trained models. In fact, we find that the released TCT-ColBERT models are useful for conducting last-mile reproduction, replication, and ablation. They also clearly have benefits outside of reproducibility too, since they allow others to performing model analysis and use existing models as baselines.

Throughout the remainder of this paper, we use this framework to design and motivate our reproduction, replication, and ablation studies over TCT-ColBERT.

[3] https://github.com/castorini/pyserini/blob/master/docs/experiments-tct_colbert-v2.md

[4] Indeed, Lin et al. [14] observed slight differences in performance between their released TensorFlow and PyTorch implementations. [5] We recognise that we mix metric and imperial units in our nomenclature; so it goes in the United Kingdom.

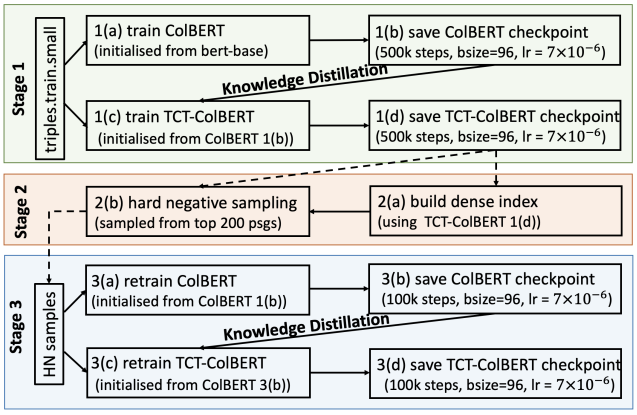

Figure 2: Training steps of a TCT-ColBERT variant.

4 TCT-COLBERT

To achieve the aim of creating an effective single-representation model for dense retrieval, Lin et al. [15] proposed the distillation of ranking knowledge from an effective multiple representation ColBERT model. Indeed, as the late interaction paradigm used by ColBERT is more efficient than the cross-encoder ranking models (e.g. BERT_{CLS} or monoT5), thus, the ColBERT model is selected as the teacher model, while a Bi-Encoder model is employed as the student model. This design allows the teacher and the student models to perform inference in a tightly-coupled way in response to the input query, giving the name of the model Tightly-Coupled Teacher ColBERT model, short for TCT-ColBERT. Thus, by distilling knowledge from ColBERT during training, the student (Bi-Encoder) model can learn to generate better representations.

The authors proposed two versions of TCT-ColBERT models successively [14, 15]. Of these, [15] is as an early effort released as a preprint, which employs ColBERT as the teacher model and distills knowledge in a tightly coupling method to a Bi-Encoder based student model. Later, the same authors introduced additional training ingredients in [14] to further improve its performance. Since the main structures of the two versions are the same, in our work, we focus on reproducing the latest version of TCT-ColBERT. The training steps of a TCT-ColBERT variant are summarised in Figure 2. In the following, we introduce the main ingredients and working principles of TCT-ColBERT.

Knowledge Distillation: TCT-ColBERT employs 'soft labelling' to achieve knowledge distillation from teacher to student, in that the student model is trained to model the score outputs of the teacher, rather than the (hard) labels. Let the probability distributions produced by teacher and student models be defined as follows:

$$P^{q}_{\theta;\,\text{student}}(d, \mathcal{D}_{\mathcal{B}}) = \frac{\exp(\psi_q \cdot \psi_d)}{\sum_{d' \in \mathcal{D}_{\mathcal{B}}} \exp\left(\psi_q \cdot \psi_{d'}\right)} \tag{2}$$

$$P^{q}_{\hat{\theta};\,\text{teacher}}(d, \mathcal{D}_{\mathcal{B}}) = \frac{\exp(\text{MaxSim}(\phi_q, \phi_d)/\tau)}{\sum_{d' \in \mathcal{D}_{\mathcal{B}}} \exp(\text{MaxSim}(\phi_q, \phi_{d'})/\tau)}, \tag{3}$$

where $\mathcal{D}_{\mathcal{B}}$ denotes a batch of documents from the training triples, τ denotes the temperature hyperparameter, $\phi_q = \{\phi_{q_1}, \ldots, \phi_{q_{|q|}}\}$ and $\phi_d = \{\phi_{d_1}, \ldots, \phi_{d_{|d|}}\}$. TCT-ColBERT is trained to force the student model to mimic the probability distribution predicted by the teacher

model, by minimising the KL divergence between the above two distributions, as follows:

$$\mathcal{L} = \sum_{i=1}^{|\mathcal{B}|} \mathrm{KL}(P^{q}_{\hat{\theta};\,\text{teacher}}(d, \mathcal{D}_{\mathcal{B}}) || P^{q}_{\theta;\,\text{student}}(d, \mathcal{D}_{\mathcal{B}})) \quad (4)$$

In-Batching Negative Training: Additionally, the authors used in-batch negatives to train TCT-ColBERT, treating the documents for other queries in the batch as additional non-relevant (negative) documents [7, 9, 15].

Hard Negative Sampling: Another trick used in TCT-ColBERT is hard negative sampling, which can be described as the pink box in Figure 2. In more detail, after obtaining the TCT-ColBERT model trained using the standard MSMARCO:`triples.train.small` training triples, the authors produce the document embeddings for all the documents in the collection using the saved checkpoint. They then perform dense retrieval for each input query using the trained TCT-ColBERT model on the built dense index. Finally, for each query, hard negative instances are sampled from the top 200 returned passages to create hard negative samples (denoted HN). Indeed, as any effectiveness improvements obtained from training with in-batch negatives is limited, existing work [25, 30, 32, 33] have shown that more informative (harder) negative samples allow to train a more effective dense retrieval model. Finally, HN+ denotes distillation from the teacher trained used hard negatives, and using the same hard negatives during distillation.

Student Model Initialisation: As the student model applies an Average Pooling over the embedded representations, the student model is initialised by taking the weights from the teacher model. In this way, the student training does not need to learn from scratch the layer weights inside the model, but instead just how to adjust those weights to best address the new Average Pooling objective. This explains the use of Pooling for the Bi-Encoder model structure, rather than the embeddings of the [CLS] token, which are more conventionally used.

Training Procedure: Figure 2 provides an overview of the training procedure and specific training setup for the TCT-ColBERT model, which falls into three stages. In Stage 1, the teacher then the student models are consecutively trained using the standard triple samples: `triples.train.small`. Then, in Stage 2, TCT-ColBERT employs a hard negative sampling technique to create the HN samples. Finally, in Stage 3, the teacher model and the student models are further fine-tuned following the same steps as Stage 1, but using the HN samples obtained in Stage 2. For both Stage 1 and Stage 3 training, the original work follows the following two training steps: (I) fine-tune the ColBERT teacher model and save the trained ColBERT checkpoint after 500k training steps; (II) freeze the trained teacher model and initialise the parameters of the student model from the trained teacher model, then perform knowledge distillation. In essence, knowledge distillation is only a part of the overall training procedure.

Index Floating Point Precision: From Table 5 in [14], we observe that the original work proposes to reduce the storage overhead of the dense index by using half-precision values, i.e. 16 bits rather than 32 bits. However, any impact on effectiveness is unreported.

Summary: As is clear from the above description and Figure 2, TCT-ColBERT deploys a complex training procedure. Moreover, from the original versions, the motivations or impacts of some of these decisions remain unclear. For this reason, our reproducibility work goes further than the *last-metre* re-evaluation of the output artefacts, by additionally considering questions about the complete TCT-ColBERT training process. In particular, by ablating (varying) certain decisions, we draw more insights about which decisions are necessary for effective retrieval not shown in the original paper. For instance, the TCT-ColBERT [14] is trained using Google TPUs for 500k batches with a batch size of 96 - are these considerable resources necessary? Moreover, the necessity of the student model initialisation remains unclear. By providing a deeper dive into TCT-ColBERT, we aim to increase understanding of this interesting and popular approach, and help others to build upon its key contributions. We describe in more details the questions tackled by this work in the next section.

5 AIMS AND EXPERIMENTAL SETTINGS

The primary aim of this reproducibility paper are to reproduce and replicate the *core results* of TCT-ColBERT [14]. We consider the core results of the paper to be the dense passage retrieval results on the TREC DL 2019 and MSMARCO Dev (small) querysets. We also aim to meaningful and interesting ablations that were missing from the original paper, such as those highlighted in Section 4 above. In particular, we address the following "reproducibility questions", across last-metre, last-mile and complete perspectives:

- Last Metre: Can we reproduce the dense retrieval using released query/doc vectors?
- Last Mile: Can we replicate TCT-ColBERT inference and retrieval using only the released models?
- Last Mile: Can TCT-ColBERT function as a low-latency re-ranker?
- Last Mile: Is using half-precision floats a safe way to reduce the dense index size?
- Complete: Can we reproduce first-stage training?
- Complete: Can we reproduce the results of the first training stage in fewer training samples?
- Complete: What is the impact of initialising the student model to the teacher?
- Complete: Can we reproduce hard negative training?

We address the last-metre, last-mile and complete reproducibility questions respectively in Sections 6, 7 and 8. For all the experiments, we use an RTX 3090 with 24GB GPU memory for indexing/retrieval and an RTX A6000 with 48GB GPU memory for training, which are more readily accessible than Google TPUs.

At the end of each result table, we provide a block indicating which artefacts (e.g., trained models, datasets, evaluation tools) contributed to the particular experiments, along with the baselines and details about what (if any) ablation we perform. Table 1 lists the source of each artefact. Sources can include the URL to the artefact (sometimes including bash-style wildcards for brevity). The `hgf:` prefix indicates models from the HuggingFace model repository. Given the lack of universally-standard naming convention for datasets in IR, we use the identifiers from ir_datasets [18], with an `irds:` prefix. We report performance using the same evaluation measures used by the original paper (nDCG@10 and Recall@1000

Table 1: List of artefact sources.

a	https://rgw.cs.uwaterloo.ca/JIMMYLIN-bucket0/pyserini-indexes/dindex-msmarco-passage-tct_colbert-v2{,-hn,-hnp}-bf-*.tar.gz
b	https://github.com/castorini/pyserini-data/blob/main/encoded-queries/query-embedding-tct_colbert-v2{,-hn,-hnp}-msmarco-passage-dev-subset-*.tar.gz
c	`irds:msmarco-passage/trec-dl-2019`
d	`irds:msmarco-passage/dev/small`
e	https://github.com/microsoft/MSMARCO-Passage-Ranking/blob/master/ms_marco_eval.py
f	https://github.com/usnistgov/trec_eval
g	`hgf:castorini/tct_colbert-v2-msmarco`
h	`hgf:castorini/tct_colbert-v2-hn-msmarco`
i	`hgf:castorini/tct_colbert-v2-hnp-msmarco`
j	https://github.com/terrierteam/pyterrier_dr#inference
k	https://github.com/terrierteam/pyterrier_dr#indexing and https://github.com/terrierteam/pyterrier_dr#retrieval
l	https://github.com/terrierteam/pyterrier_dr#training
m	https://github.com/castorini/tct_colbert/
n	MSMARCO train triples
o	ColBERT trained on MSMARCO triples, provided privately by [14]
p	ColBERT trained on hard negatives, provided privately by [14]
q	Hard negative samples we generate from *g*
r	`hgf:bert-base-uncased`

[sic][6] for *c*; MRR@10 and Recall@1000 for *d*). For ablations, we indicate significant differences compared to the corresponding baseline. We cannot report significance because we do not have the original result files.

6 THE LAST METRE

We start our experiments by exploring the "last metre" of TCT-ColBERT. Section 6.1 explores whether we can reproduce the main results of the paper when we are provided with a built index and pre-computed query vectors from the authors.

6.1 Can we reproduce the dense retrieval using released query/doc vectors?

Given that, at the time of writing, nobody (aside from Lin et al. [14]) had reported the reproduction of TCT-ColBERT (v2) retrieval, we started by following the last-metre reproduction instructions provided by the authors on Pyserini.[7] Note that the reproducibility instructions provided only cover the results on MSMARCO Dev (small), so this is the setting we pursue for this evaluation.

Table 2 presents the results of this reproducibility experiment. We find that our reported results match those reported in the paper in virtually all cases (our RR@10 results differed by a negligible 0.001). However, we note the highly-constrained "last-metre" reproduction setting here: we simply test the reproducibility of the FAISS retrieval, the associated integration in the Pyserini package, and the evaluation tool. Although it is not a given that this inference stage will remain stable over time [13], witnessing reproduction at this stage is far from a complete reproduction of the contributions of the TCT-ColBERT paper. The remainder of our experiments explore the reproduction and conduct ablations beyond the last-metre setting.
Answer: We are able to reproduce the Dev (small) results for TCT-ColBERT using pre-computed query vectors, pre-indexed document vectors, and the authors' retrieval code.

[6] We found that [14] reports recall with a binary relevance cutoff of 1, a departure from the standard practice [2, 3] of applying a cutoff at 2 for binary measures on MSMARCO. To stay consistent with [14], we follow the same practice in this paper.

[7] https://github.com/castorini/pyserini/blob/master/docs/experiments-tct_colbert-v2.md

Table 2: Last-meter reproduction: Dense retrieval from pre-computed documents and vectors.

	dev/small	
Setting	RR@10	R@1k
TCT-ColBERT		
◊ Reported	**0.344**	**0.967**
Ours	**0.344**	**0.967**
TCT-ColBERT HN		
◊ Reported	**0.354**	**0.971**
Ours	**0.354**	**0.971**
TCT-ColBERT HN+		
◊ Reported	**0.359**	**0.970**
Ours	0.358	**0.970**

◊ **Baselines:** Values reported by Lin et al. [14] in Table 3. **Matching Artefacts:** Toolkit: pyserini, Indexed document vectors[a], Pre-computed query vectors[b], Evaluation Data[d], Measurement Tools[ef]

7 THE LAST MILE

Given that we were able to reproduce the results of TCT-ColBERT using pre-computed query and document vectors, the logical next question is whether we can achieve the same inference results when computing the query and document vectors ourselves. We first conduct a reproduction study where we build alternative TCT-ColBERT model inference, dense indexing, and dense retrieval code (Section 7.1). We then use this as a platform to ablate settings that were unexplored in the original paper: the effectiveness of TCT-ColBERT for efficient re-ranking (Section 7.2), and its robustness to more coarse-grained floating point encoding (Section 7.3).

7.1 Can we replicate TCT-ColBERT inference and retrieval using only released models?

To answer this question, we conduct a reproduction study using some of the artefacts produced by the paper (i.e., the released model checkpoints), but with our own vector inference, indexing, and dense retrieval implementations. This is an important step because it allows us to establish (1) whether the released models can generalise to another implementation, (2) whether the query and document inference procedures can be replicated, and (3) whether our implementation could potentially be suitable to replicate the training procedure (i.e., if it could not replicate inference, it is unlikely that it would be able to replicate model training).

For this experiment, we use our own Python implementation for query/document vector inference, and dense indexing/retrieval. Our query and document encoder for TCT-ColBERT uses PyTorch and HuggingFace Transformers [29] rather than those based on TensorFlow in Lin et al. [14]. In one case, we refer to the authors' model inference implementation to resolve a tokenisation discrepancy that was unclear from the original paper.[8] Our dense indexing and retrieval implementations differed substantially from the FAISS-based implementation used by Pyserini; we store the vector data directly to disk as a file that can be memmapped by NumPy and PyTorch. To score, a simple matrix multiplication is conducted. The PyTorch

[8] Specifically, we found that the TCT-ColBERT implementation treats the ColBERT `[Q]` and `[D]` prefixes each as three separate tokens (i.e., '[', 'Q', and ']'), while the original ColBERT treats them as a single token. See https://github.com/castorini/pyserini/blob/master/pyserini/dsearch/_dsearcher.py#L88

Table 3: Last-mile reproduction: Inference and retrieval.

Setting	trec-dl-2019 nDCG@10	trec-dl-2019 R@1k	dev/small RR@10	dev/small R@1k
TCT-ColBERT				
◊ Reported	0.685	0.745	0.344	0.967
Ours	**0.686**	**0.747**	**0.345**	**0.968**
TCT-ColBERT HN				
◊ Reported	0.705	**0.765**	0.354	0.971
Ours	**0.710**	**0.765**	**0.356**	**0.972**
TCT-ColBERT HN+				
◊ Reported	0.719	0.760	**0.359**	**0.970**
Ours	**0.721**	**0.762**	**0.359**	**0.970**

◊ **Baselines:** Values reported by Lin et al. [14] in Table 3. **Matching Artefacts:** Trained models[ghi], Evaluation Data[cd], Measurement Tools[ef] **Different Artefacts:** Toolkit: PyTerrier, Model inference code[j], Dense indexing/retrieval code[k]

implementation allows for scalable acceleration by incrementally moving batches of document vectors from CPU memory to GPU memory, and performing the scoring on GPU. This also allows us to easily reuse the indices for re-ranking (as we explore in Section 7.2).

Table 3 presents the results of our replication experiment. The reported results for TCT-ColBERT v2, HN, and HN+ correspond with the results from rows 6, 8, and 9 from Table 3 in [14]. We find that for all three models, our replication achieves at most a 0.005 difference across all metrics on both TREC DL 2019 and MSMARCO dev (small), and always in favour of our implementation. These results suggest that the released TCT-ColBERT models generalise to an alternative inference implementation (both for documents and queries), that our dense retrieval methods are correct, and that our PyTorch-based inference implementation is potentially suitable for model training.

Answer: We are able to replicate the TCT-ColBERT model inference using our own implementations for query/document vector computation, dense indexing, and dense retrieval.

7.2 Can TCT-ColBERT function as a low-latency re-ranker?

TCT-ColBERT was designed to operate in a dense retrieval setting. However, this is not strictly necessary. MacAvaney et al. [16] observed that one can use pre-computed document vectors for re-ranking (albeit they explored it in the context of learned sparse vectors, rather than learned dense vectors). This is advantageous because scoring only a subset of candidate documents can be done very inexpensively (around 4ms/query for 1000 documents from our tests, under similar memory utilisation as FAISS-based dense retrieval), while the most expensive parts of the process (sparse retrieval @ 9ms/query[9] and query vector computation @ 7ms/query) can be done in parallel. This yields a total latency of only around 13ms/query — an order of magnitude faster than the 107ms/query reported in [14] for full retrieval (which uses a brute force nearest neighbour search on GPU).

With this potential benefit in mind, we conduct an ablation experiment to test whether TCT-ColBERT is effective in a re-ranking setting. We examine three different candidate document sets (each with the top 1000 results): the official re-ranking set from MSMARCO,[10] BM25 results retrieved with PyTerrier [19], and BM25 results over passages expanded using Doc2Query-T5 [23] with PyTerrier. We note that this ablation experiment is different from the "hybrid" setting described in [14]. In particular, rather than using a linear combination of the scores (`bm25 + tct_colbert`), we use a pipelined re-ranking approach (`bm25 » tct_colbert`), which can yield substantially lower latency.

Table 4: Ablation study: TCT-ColBERT as a Re-Ranker (RR).

Setting	trec-dl-2019 nDCG@10	Δ	dev/small RR@10	Δ
TCT-ColBERT				
◊ Dense Retrieval	0.686		0.345	
RR (official)	0.690	+0.004	*0.338	-0.007
RR (BM25)	0.706	+0.020	0.344	-0.001
RR (d2q BM25)	***0.725**	+0.039	***0.349**	+0.004
TCT-ColBERT HN				
◊ Dense Retrieval	0.710		0.356	
RR (official)	0.703	-0.007	*0.345	-0.011
RR (BM25)	0.718	+0.008	*0.352	-0.004
RR (d2q BM25)	***0.732**	+0.022	**0.357**	+0.001
TCT-ColBERT HN+				
◊ Dense Retrieval	0.721		0.359	
RR (official)	0.707	-0.014	*0.346	-0.013
RR (BM25)	**0.734**	+0.013	*0.353	-0.006
RR (d2q BM25)	**0.734**	+0.013	**0.360**	+0.001

◊ **Baselines:** "Ours" from Table 3 **Artefacts:** From Table 3 **Ablation:** Re-ranking (instead of retrieval) * **Significance:** Compared to baseline, paired t-test ($p < 0.05$)

Table 4 provides the results for this ablation experiment. We observe that the official re-ranking results are the least effective, and usually reduce the retrieval effectiveness. However, the BM25 and doc2query rankings often actually improve the effectiveness, particularly on TREC DL 2019, where the model gains between 0.008 and 0.039 nDCG@10. This is an especially encouraging result, given that the approach significantly reduces the query latency (107ms/query down to 13ms/query). Figure 3 shows the re-ranking ablation result plotted alongside various other ranking and re-ranking techniques. We observe that the approach appears to extend the Pareto frontier due to its very low latency, achieving high effectiveness and fast retrieval. We further test the re-ranking approach using ANCE [30], and observe an even more dramatic boost in effectiveness over dense retrieval approaches (in grey), along with a similar low latency.

Answer: We find that TCT-ColBERT also functions as an effective re-ranker of the Doc2Query-T5 expanded passages, and, to the best of our knowledge, results in a new Pareto-optimal result, with a query latency of 13ms/query and a TREC DL 2019 nDCG@10 of 0.734 RR@10.

7.3 Is using half-precision floats a safe way to reduce the dense index size?

Table 5 in [14] suggests reducing the precision of the document representations from 32 bits (FP32) to 16 bits (FP16) in order to reduce the storage overhead. However, the paper does not mention

[9] Our latency measurements use PISA [21], a highly-optimised sparse retrieval implementation.

[10] Lin et al. [15] explored re-ranking the official set in the pre-print [15], but dropped it in the published paper [14] (which featured new models).

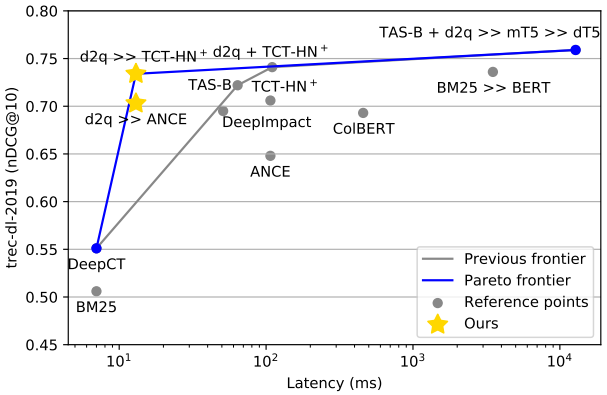

Figure 3: Pareto-optimality of TCT-ColBERT as a re-ranker. A » B denotes a re-ranking of the results of A by B.

whether these reduced-precision values were used in the experiments. The available downloadable indices[a] (used in Section 6.1) suggest that the larger FP32 values were used in Lin et al. [14]. Given that reduced precision can result in more tied ranking scores which can ultimately affect performance [31], we perform an ablation experiment using FP16 indices. For this ablation experiment, we downcast the computed query and document vectors from FP32 to FP16 using our dense retrieval implementation.

Table 5 provides our ablation results. We observe an interesting trend: although the performance on TREC DL 2019 is largely unaffected when using FP16, the RR@10 performance on Dev (small) is consistently reduced (by 0.005). Upon closer inspection, we see that FP16 resulted in considerably more tied scores than FP32 (around 97% of scores are tied when using FP16, compared to under 1% for FP32). We further find that 86% of queries in the Dev (small) set have a document ID above 7M (of around 9M total), while only 3% of queries in the train set do. Since our implementation (and most retrieval and evaluation systems in general [12]) break ties by document ID (ascending), this disadvantages our system. We were able to correct the behaviour simply by shuffling the tied values prior to evaluation (shuffle-ties). We also suspect that this property could be used to "game" the leaderboard,[11] by favouring passages with a higher ID above those with a lower one. Indeed, we observe that the ranking effectiveness increased by between 0.004–0.006 through this exploit (rev-ties) — though we firmly believe that this technique should not be employed to artificially boost one's leaderboard placement.

Answer: We observe that using FP16 results in minimal changes to ranking effectiveness, but it needs to be aligned with responsible handling of ties (shuffling or similar) to avoid unfair comparisons on the MSMARCO leaderboard.

8 COMPLETE REPRODUCTION

Given that our last-mile reproduction experiments were successful, we now explore complete reproduction by training TCT-ColBERT models and Bi-Encoder baseline. We begin by exploring training that does not involve hard negatives (Section 8.1) and then perform ablations that explore the amount of training data required (with the goal of reducing the cost of training, Section 8.2) and test the effect of student model initialisation (Section 8.3). Finally, we test whether we can reproduce the hard negative training process (Section 8.4). The experiments in this section focus on unanswered questions pertaining directly to the methodology of the original paper. This leaves adjacent studies, such as generalisability to other datasets and the stability over initial conditions[12] to future works.

Table 5: Ablation study: Floating point precision.

Setting	trec-dl-2019 nDCG@10	trec-dl-2019 R@1k	dev/small RR@10	dev/small R@1k
TCT-ColBERT				
◇ FP32	0.686	**0.747**	0.345	**0.968**
FP16	**0.691**	0.745	*0.340	**0.968**
FP16 (shuf-ties)	0.686	0.745	0.346	**0.968**
⚠FP16 (rev-ties)	0.683	0.745	***0.351**	**0.968**
TCT-ColBERT HN				
◇ FP32	**0.710**	**0.765**	0.356	**0.972**
FP16	0.709	**0.765**	*0.351	**0.972**
FP16 (shuf-ties)	0.709	**0.765**	0.355	**0.972**
⚠FP16 (rev-ties)	0.702	**0.765**	**0.360**	**0.972**
TCT-ColBERT HN+				
◇ FP32	**0.721**	**0.762**	0.359	**0.970**
FP16	0.720	0.761	*0.354	**0.970**
FP16 (shuf-ties)	0.716	0.761	0.358	**0.970**
⚠FP16 (rev-ties)	0.712	0.761	***0.366**	**0.970**

◇ **Baselines:** "Ours" from Table 3 **Artefacts:** From Table 3 **Ablation:** FP16 scoring (rather than dense FP32) * **Significance:** Compared to baseline, paired t-test ($p < 0.05$) ⚠ **Note:** Lines marked with ⚠ use a newly-discovered exploit in the MSMARCO ranking dataset, and should not be considered valid in terms of leaderboard performance.

8.1 Can we reproduce first-stage training?

We first explore whether we can reproduce the main results of the models that do not make use of hard negatives during the training process – namely the directly-supervised Bi-Encoder model and the plain, distilled TCT-ColBERT model. We conduct two sets of experiments: one using the released training implementation, and another using a training implementation that we wrote independently (with the goal of achieving replication). During this process, we make use of the trained ColBERT teacher model that the authors privately shared with us, both for model initialisation (both models) and as the teacher itself (for the TCT model). In the pursuit of achieving complete reproduction, we also include a version where we first train the teacher model using the released code, before proceeding to TCT training. We reduce all batch sizes to 32 (from the 96 used in the original paper), based on what we could fit into our GPU's memory.

Table 6 presents the results of our first-stage training reproduction experiment. We find that when we use the (TensorFlow-based) training code provided by the authors and their trained teacher, we are able to achieve a performance that is on par with what was reported for the TCT-ColBERT (without hard negatives), but not

[11] To the best of our knowledge, this is a previously unknown and undesirable bias of MSMARCO. Exploiting this property does not bring any new scientific understanding and is counter to the spirit of the MSMARCO coopetition. Nevertheless, we choose to disclose this exploit so appropriate steps can be taken by the leaderboard owners.

[12] An example is stability over random seeds, which is frequently left untested in IR literature, and when it has been studied, hasn't shown to have a strong impact on results [7].

Table 6: Reproduction study: Single-stage model training.

Setting	trec-dl-2019 nDCG@10	trec-dl-2019 R@1k	dev/small RR@10	dev/small R@1k
Bi-Encoder				
◊ Reported	**0.626**	0.658	**0.310**	**0.945**
Ours, their impl	0.614	**0.661**	0.282	0.940
Ours, our impl.	0.607	0.632	0.292	0.928
TCT-ColBERT				
◊ Reported	0.685	**0.745**	0.344	0.967
Ours, their impl.	**0.698**	0.743	**0.348**	**0.969**
+ train teacher	0.545	0.637	0.270	0.924
Ours, our impl.	0.623	0.631	0.312	0.945

◊ **Baselines:** Values reported by Lin et al. [14] in Table 2. **Matching Artefacts:** Training code ("their impl")^{m}, Evaluation Data^{cd}, Measurement Tools^{ef}, Training Data^{n}, Teacher^{o} **Different Artefacts** Toolkit: PyTerrier, Training code^{l}, Model inference code^{j}, Dense indexing/retrieval code^{k}

Table 7: Ablation study: Number of training steps.

Setting	trec-dl-2019 nDCG@10	trec-dl-2019 R@1k	dev/small RR@10	dev/small R@1k
Bi-Encoder				
◊ 500k batches	**0.614**	**0.661**	**0.282**	**0.940**
250k batches	0.564	*0.613	*0.263	*0.931
100k batches	*0.539	0.644	0.280	0.938
50k batches	*0.544	0.669	*0.269	0.936
TCT-ColBERT				
◊ 500k batches	**0.698**	**0.743**	**0.348**	**0.969**
250k batches	0.685	0.737	*0.345	0.967
100k batches	0.685	0.737	0.346	0.967
50k batches	0.680	0.737	*0.344	*0.966

◊ **Baselines:** "Ours, their impl." from Table 6 **Artefacts:** From Table 6 **Ablation:** Reducing the number of training iterations for faster training. * **Significance:** Compared to baseline, paired t-test ($p < 0.05$)

for the Bi-Encoder model. Variations of the approaches were unsuccessful, however. Training the teacher model ourselves resulted in a very poor performance. Further, our own (PyTorch-based) implementation of the training procedure is unable to achieve comparable performances to the originally reported results. This is despite the effectiveness of our model inference implementation (as demonstrated throughout Section 7) and considerable time and effort attempting to diagnose and resolve differences between our implementation and theirs. Given the lack of success completely replicating the training process in the most basic condition, we do not seek any further *replication* efforts of TCT-ColBERT in this paper, and instead focus on reproduction using the available artefacts.
Answer: We are able to successfully reproduce TCT-ColBERT performance (without hard negatives), to a reasonable approximation when using the authors' training code and trained teacher model. However, we are unsuccessful when training the Bi-Encoder model, when training the TCT-ColBERT using our own teacher model, and when writing our own training implementation.

8.2 Can we reproduce first-stage training using fewer training iterations?

Although we are successful in reproducing the first-stage training procedure using the authors' released code in Section 8.1, the process is extremely expensive. On our modern GPU, the training process (for 500k batches) takes roughly 60 GPU hours. Although the process was likely faster for the authors on a TPU, these resources are often not practically available to research teams. However, we postulate that their models are likely trained longer than necessary. After all, the necessary batch size reduction used in Section 8.1 meant that the model encountered only 1/3 the number of training samples than the original model did (since we keep the number of training iterations constant[13]). To test whether the models could be trained effectively using considerably less compute, we conduct an ablation study on both Bi-Encoder and TCT-ColBERT models, using the authors' training implementation, where we check model performances at only 1/2 (250k), 1/5 (100k), and 1/10 (50k) of the number of training iterations.

[13] For a potentially fairer comparison, we considered adjusting the number of training iterations and applying gradient accumulation accordingly. However, the prospect of training a single model taking over a week was unsavoury.

We present the results of our ablation experiment in Table 7. We find that for both models, the full training regime (500k steps) always yielded the highest performance in absolute terms. However, the differences between 500k and fewer steps are not always statistically significant and there are moderate fluctuations – especially for the Bi-Encoder model. These results suggest that an early stopping approach over validation data could be employed (though this process itself would add computational overhead to the training process). Finally, we note that, at least, our Bi-Encoder models may actually be under-trained, given that their highest performance is at the end of their training process.
Answer: Comparable performance can be achieved in some measures at earlier stages in the training, but for best overall performance across measures, the full 500k training steps are necessary.

8.3 What is the impact of initialising the student model to the teacher?

Another setting not explored by the original authors is the initialisation of the student model weights to those of the teacher model. We perform an ablation experiment to fill in a gap in understanding the effect of this decision. Simply, we train versions of the Bi-Encoder and TCT-ColBERT models that we successfully reproduced above, but initialise the weights to those from `bert-base-uncased`, rather than the teacher model. Table 8 presents results for this ablation experiment. We observe that using the teacher's weights to initialise the model has little effect on model performance – both for the Bi-Encoder model and TCT-ColBERT. This suggests that the experimental setup can be simplified by eliminating this step.
Answer: Teacher weight initialisation does not affect performance in a direct supervision or distillation setting.

8.4 Can we reproduce hard negative training?

Finally, we explore whether we can reproduce the process of TCT-ColBERT training using hard negatives. Given that we are unable to reproduce the teacher model training (Section 8.1), we train using a version of the teacher model provided by the authors that was fine-tuned on their hard training triples. However, the authors were unable to provide their hard triples file, so we attempt to recreate it using our implementation (which we verified as effective

Table 8: Ablation study: Model initialisation.

Setting	trec-dl-2019 nDCG@10	trec-dl-2019 R@1k	dev/small RR@10	dev/small R@1k
Bi-Encoder				
◊ Teacher Init	0.614	0.661	**0.282**	0.940
BERT Base^r	**0.620**	**0.667**	**0.282**	**0.942**
TCT-ColBERT				
◊ Teacher Init.	**0.698**	**0.743**	**0.348**	**0.969**
BERT Base^r	0.696	0.740	**0.348**	0.968

◊ **Baselines:** "Ours, their impl." from Table 6 **Artefacts:** From Table 6
Ablation: Change model initialisation from teacher to^r. * **Significance:** Compared to baseline, paired t-test ($p < 0.05$); no sig. differences observed

Table 9: Reproduction/ablation study: Hard negative models.

Setting	trec-dl-2019 nDCG@10	trec-dl-2019 R@1k	dev/small RR@10	dev/small R@1k
TCT-ColBERT HN+				
◊ Reported	**0.721**	**0.762**	**0.359**	**0.970**
Ours	0.712	0.725	0.353	0.962
+ MSMARCO triples	0.705	0.712	0.345	0.964

◊ **Baselines:** Values reported by Lin et al. [14] in Table 3. **Matching Artefacts:** Training code^m, Evaluation Data^{cd}, Measurement Tools^{ef}, Teacher^p **Different Artefacts** Toolkit: PyTerrier, Model inference code^j, Dense indexing/retrieval code^k, Training Data^q **Ablation:** Teacher trained on hard negatives^p, but distilling using MSMARCO triples^n * **Significance:** Comparing "Ours" with "+ MSMARCO triples", paired t-test ($p < 0.05$)

in Section 7.1). As an ablation of this approach, we also train a version instead using the MSMARCO's triples (but with a teacher fine-tuned on hard negatives) to test whether the enhanced teacher alone is sufficient for producing a more effective model. All training is conducted using the original TCT-ColBERT code^m.

Table 9 presents the results of this experiment. We find that when using the hard negatives that we generated ourselves, the model is unable to achieve the performance reported. This observation suggests that, potentially, the same set of training triples is required by both the student and teacher to achieve maximal results. Our ablation using the official MSMARCO triples provides further evidence of this; the stronger teacher provides virtually no change in precision (see Table 6, nDCG@10 of 0.698 vs. 0.705, RR@10 of 0.348 vs. 0.345) and harms recall (0.743 vs 0.705), compared to the teacher trained on official MSMARCO triples. We recognise that further experiments may be warranted to provide stronger evidence of this claim, but given the considerable expense of training these models, we do not pursue this direction further.

Answer: Hard negative training can help, but we find some evidence that when the teacher is trained on hard negatives, the same sequence of hard negatives should be used during model distillation.

9 CONCLUSIONS

In this work, we inspect the reproducibility and replicability of TCT-ColBERT. In particular, we address reproducibility questions from three perspectives to reproduce TCT-ColBERT, namely the last-metre, last mile and complete. Based on the last-metre experiments, we claim that we can replicate the model performance using the released query/doc vectors. We further conduct several ablations that were missing from the original work in the last-mile experiments and find that (a) TCT-ColBERT can function as an inexpensive effective re-ranker; and (b) index size can be reduced with the low-precision floating point but needs to be aligned with responsible handling of ties. Finally, from the complete reproduction experiments, we conclude that (a) TCT-ColBERT does benefit from longer training durations; and (b) performing student model initialisation from the teacher does not necessarily to lead a better TCT-ColBERT model. Overall, while we are able to reproduce and replicate results from last-metre and last-mile perspectives, complete reproduction of the core results remains a challenge – particularly for the most effective "HN+" model variant.

ACKNOWLEDGEMENTS

We thank the authors of TCT-ColBERT for providing their teacher checkpoints, Sasha Petrov for assistance with TensorFlow, and anonymous reviews for helpful feedback.

Xiao Wang acknowledges support by the China Scholarship Council (CSC) from the Ministry of Education of P.R. China. Sean MacAvaney, Craig Macdonald and Iadh Ounis acknowledge EPSRC grant EP/R018634/1: Closed-Loop Data Science for Complex, Computationally- & Data-Intensive Analytics.

REFERENCES

[1] Timo Breuer, Nicola Ferro, Maria Maistro, and Philipp Schaer. 2021. repro_eval: A Python Interface to Reproducibility Measures of System-Oriented IR Experiments. In *Proceedings of ECIR*. 481–486.
[2] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 Deep Learning Track. In *Proceedings of TREC*.
[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the TREC 2019 Deep Learning Track. In *Proceedings of TREC*.
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of ACL*. 4171–4186.
[5] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. A White Box Analysis of ColBERT. In *Proc. ECIR*. 257–263.
[6] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-architecture Knowledge Distillation. *arXiv preprint arXiv:2010.02666* (2020).
[7] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *Proceedings of SIGIR*. 113–122.
[8] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale Similarity Search with GPUs. In *IEEE Transactions on Big Data*. 535–547.
[9] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of EMNLP*. 6769–6781.
[10] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of SIGIR*. 39–48.
[11] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained Transformers for Text Ranking: BERT and Beyond. In *Synthesis Lectures on Human Language Technologies*. 1–325.
[12] Jimmy Lin and Peilin Yang. 2019. The Impact of Score Ties on Repeatability in Document Ranking. In *Proceedings of SIGIR*. 1125–1128.
[13] Jimmy Lin and Qian Zhang. 2020. Reproducibility is a Process, not an Achievement: The Replicability of IR Reproducibility Experiments. In *Proceedings of ECIR*. 43–49.
[14] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-batch Negatives for Knowledge Distillation with Tightly-coupled Teachers for Dense Retrieval. In *RepL4NLP-2021 Workshop*. 163–173.
[15] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy J. Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *ArXiv* abs/2010.11386 (2020).
[16] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with

Contextualization. In *Proceedings of SIGIR*. 1573–1576.
[17] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *Proceedings of SIGIR*. 1101–1104.
[18] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified Data Wrangling with ir_datasets. In *Proceedings of SIGIR*. 2429–2436.
[19] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval. In *Proceedings of CIKM*. 4526–4533.
[20] Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. On Single and Multiple Representations in Dense Passage Retrieval. In *IIR 2021 Workshop*.
[21] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *Proceedings of OSIRRC*. 50–56.
[22] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).
[23] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From Doc2query to DocTTTTTquery. *Online preprint* (2019).
[24] Association of Computing Machinery. 2020. Artifact Review and Badging. https://www.acm.org/publications/policies/artifact-review-and-badging-current.
[25] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of NAACL*. 5835–5847.
[26] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. *arXiv preprint arXiv:2112.01488* (2021).
[27] Nicola Tonellotto and Craig Macdonald. 2021. Query Embedding Pruning for Dense Retrieval. In *Proceedings of CIKM*. 3453–3457.
[28] Xiao Wang, Craig Macdonald, and Nicola Tonellotto. 2021. Pseudo-Relevance Feedback for Multiple Representation Dense Retrieval. In *Proceedings of ICTIR*. 297–306.
[29] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art Natural Language Processing. In *Proceedings of EMNLP*. 38–45.
[30] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *Proceedings of ICLR*.
[31] Ziying Yang, Alistair Moffat, and Andrew Turpin. 2016. How Precise Does Document Scoring Need to Be?. In *Proceedings of AIRS*. 279–291.
[32] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance. In *Proceedings of CIKM*. 2487–2496.
[33] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval. In *Proceedings of WSDM*. 1328–1336.