The impact of the semantic matching within interpolation-based re-ranking

Alexandru Nistor[1]

Supervisor(s): Avishek Anand[1]**, Jurek Leonhardt**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Alexandru Nistor
Final project course: CSE3000 Research Project
Thesis committee: Avishek Anand, Jurek Leonhardt, Alan Hanjalic

Abstract

The crucial role of information retrieval (IR) is highlighted by its presence across a wide range of tasks, such as web search and fact-checking, and domains, including finance and healthcare. Effective and efficient IR systems are critical for finding relevant information from vast amounts of data. Traditional sparse retrieval methods such as BM25 are efficient but often fail to capture the context, while more recent dense retrieval models are highly inefficient in terms of resources and latency.

In our research, we evaluate multiple Transformer-based models to understand the impact of the semantic re-ranking phase within interpolation-based re-ranking, using the FAST-FORWARD indexes framework, a retrieve-and-re-rank approach which combines the benefits of both lexical and semantic matching. We focused on identifying specific scenarios in which particular models excel in terms of ranking performance or latency, aiming to provide model recommendations tailored to different settings. Our evaluations reveal that no single model outperforms others across all datasets. We hypothesise that the main factors influencing encoder performance are the datasets used for fine-tuning and the method employed for computing the contextualised vector embedding. However, ablations studies would be beneficial for validating these observations.

1 Introduction

Ad-hoc retrieval is the process of retrieving a ranked list of documents from a large collection, known as a corpus, based on their relevance to a specific query. This method is considered the basis of web search engines, which index the internet and produce ranked lists of websites in response to user queries.

Sparse retrieval is represented by fast and efficient methods such as BM25 [1], based on Term Frequency and Inverse Document Frequency, which are still widely used for document retrieval tasks. However, since they rely solely on exact matches between the terms in queries and documents, it is difficult to capture the similarity between their meanings.

To address this challenge of lexical mismatch, dense retrieval can be employed, which utilises low-dimensional vector representations for text [2]. The embedding models used offer outstanding retrieval performance without the need for additional training, making them a preferred choice in IR [3]. They are able to capture the meaning beyond simple token overlap. This capability allows them to embed semantically similar queries, such as "What is the capital of France?" and "Where is the Eiffel Tower located?", closely within the vector space. For this reason, text embedding has become essential in numerous natural language processing tasks, including text retrieval, text classification, question answering, and dialogue systems [4]. Encoders, which are based on large Transformer [5] architectures, are employed for creating the embeddings. They are often inefficient in terms of latency and require significantly more resources than sparse retrievers.

Hybrid approaches use both sparse and dense retrieval, combining their retrieved sets of documents and then computing the final score as an interpolation of the sparse and dense scores [6]. One challenge with this approach is handling cases in which a document is retrieved by only one method. In such instances, approximations are employed to estimate the missing score. Additionally, its dense retrieval utilises an (approximate) nearest neighbour search, which requires a significant amount of resources (i.e., GPUs).

The retrieve-and-re-rank method [7] mitigates the downsides of sparse, dense and hybrid retrievals. It uses an efficient lexical retriever to obtain a candidate set of documents, followed by a complex neural ranker that re-ranks the results based only on semantic similarity (dense retrieval score). However, these models generally use cross-encoders, which compute relevance scores based on the concatenation of a query and a document. This approach is computationally expensive, thus requiring a low retrieval depth in the lexical retrieval phase for maintaining low latency, which results in worse ranking performance as fewer documents are retrieved.

In our study, we focus on interpolation-based re-ranking, using FAST-FORWARD indexes [3]. This approach leverages dual-encoders within the re-ranking process, exploiting the merits of both lexical and semantic matching via score interpolation. Its pipeline has two stages for document retrieval: the first one uses an efficient lexical (sparse) retriever to collect relevant documents for a given query, and the second one employs an expensive semantic re-ranker, which re-ranks the documents using the interpolation between the lexical and semantic scores. Addressing the limitations of the retrieve-and-rerank [7] method, the framework enables re-ranking at high retrieval depths by using independent query and document encoders. The separation within the indexing phase makes dual-encoders a perfect fit for re-ranking, as document representations, which are the most expensive part, are pre-computed. However, this approach remains slower than sparse retrieval because queries are encoded at runtime, which is a resource-intensive task.

In previous research, most of the dual-encoder architectures used a symmetric design, in which both the query and the document encoders follow the same architecture [8]. Asymmetric approaches were also addressed by Lassance and Clinchant [9], who used separate document and query encoders and Leonhardt et al. [3], who identified runtime text encoding as a bottleneck of the FAST-FORWARD indexes framework. For this reason, the lightweight query encoders were explored with the aim of reducing latency without compromising ranking performance [3]. More recent state-of-the-art models, such as Nomic, use the same encoder for both query and document encoding but add specific prefixes, such as "search query:" or "search passage:", before each text span in a pre-processing phase [10]. Although these encoders are designed for general text embedding, unlike TCT-ColBERT [11], which is optimised for ranking tasks, they yield superior ranking performance.

The neural models used for query and document encoding are usually based on pre-trained large language models,

following various architectures such as GPT, trained in an autoregressive manner, where the model predicts the next token in a sequence given the previous tokens [12]. In contrast, BERT is trained using masked language modelling (MLM), which enables it to understand the context bidirectionally [13]. XLNet combines the strengths of both approaches by adopting a permutation-based training strategy that incorporates bidirectional context while retaining the predictive capabilities of autoregressive models [14].

We recognise that while it is ideal for a text retrieval method to have an outstanding ranking performance and low latency across all tasks and domains, achieving this goal is challenging. Therefore, our first key idea is to analyse specific scenarios in which certain models demonstrate superior performance when employed within the semantic re-ranking phase of the FAST-FORWARD indexes pipeline. This analysis allows us to provide recommendations on which dense encoders are best suited across various settings while considering different trade-offs between ranking accuracy and latency.

In this paper, we focus on symmetric Transformer-based dual-encoders within the re-ranking stage of the FAST-FORWARD indexes framework to address the question: "What is the impact of the re-ranking model?", alongside its sub-questions:

RQ1: What is the ranking performance impact of different models during the semantic re-ranking stage?

RQ2: What is the latency impact of different models during the semantic re-ranking stage?

We conduct comprehensive experiments on established ranking benchmarks and identify that both the ranking and latency performances are influenced by the vector embedding computation approach (mean pooling of token embeddings as opposed to utilizing the `[CLS]` token embedding). We hypothesise that the ranking outcomes are further affected by specific datasets used within the fine-tuning stage. Additionally, we find that latency is significantly dependent on the dataset features, particularly on the average query length.

The remainder of this paper is structured as follows. Section 2 provides a background discussion on the basics of re-ranking in IR, alongside a review of the Transformer architecture [5] and the BERT model [13]. Section 3 delves into the state-of-the-art encoders used for general text embedding. Section 4 outlines the experimental setup employed in this study. Section 5 presents the results, along with a discussion of latency and ranking performance. Section 6 explores the ethical considerations and responsible research practices underlying our study. Section 7 identifies potential directions for future work. Lastly, Section 8 concludes the paper by summarizing the key findings and reiterating our hypotheses.

2 Background

This section delves into the basics of re-ranking in IR, the significant impact of the Transformer [5] architecture within this process, and its influence on subsequent models such as BERT [13].

2.1 Document re-ranking

The FAST-FORWARD indexes pipeline begins with a sparse retriever, which fetches a set of candidate documents for a specific query. They are associated with lexical scores, which are based on term matching only, thus not taking the context into consideration. Subsequently, within the semantic re-ranker, the vector representation of each query is computed using an expensive Transformer-based encoder running only on the CPU.

The pre-computed document embeddings are then fetched from the FAST-FORWARD dense index using their unique identifier. These documents can either be represented by a single comprehensive embedding or by multiple embeddings corresponding to consecutive text chunks known as passages. For the latter, the semantic score is derived by computing the dot product between each passage representation and the query embedding, followed by taking the average or the maximum of these scores. A similar method is applied for documents with a single embedding, which treats this representation as a single passage.

The final score for each document associated with a query is determined by interpolating (cf. Eq. (1)) its sparse ($\phi_S(q,d)$) and dense ($\phi_D(q,d)$) scores, using the α hyperparameter. Ultimately, the documents are sorted based on this interpolated score, thus combining the merits of both lexical and semantic matching.

$$\phi(q,d) = \alpha \cdot \phi_S(q,d) + (1-\alpha) \cdot \phi_D(q,d) \qquad (1)$$

2.2 Transformers

In the realm of language processing, prior to the release of the Transformer [5] architecture, sequence transduction models were dominantly based on recurrent networks [15]. However, this approach is significantly limited by its inability to parallelise computations. This constraint arises because the hidden state at each time step, H_t, is computed as a function of the previous hidden state H_{t-1} and the input for position t. Unlike its predecessors, the Transformer architecture avoids recurrence and relies completely on the attention mechanism for deriving global dependencies between input and output, allowing the weights for important tokens to be increased and ones for less important tokens to be diminished [5]. Similar to other neural sequence transduction models, the architecture proposed by Vaswani et al. follows an encoder-decoder structure based on multi-head self-attention mechanisms, designed to efficiently capture in parallel diverse relationships across the input sequence [5]. Within the decoder, a masking technique is employed, which sets future positions to $-\infty$ as the attention mechanism should not share any information about subsequent tokens when giving a prediction based on the previous tokens. To further support this goal, the output embedding is also shifted back by one position. Many large language models (LLMs) adopt the Transformer architecture, with BERT [13] being a particularly popular choice due to its bi-directional nature.

2.3 BERT

Unlike previous heavily engineered task-specific architectures, BERT (Bidirectional Encoder Representations from

Transformers) is conceptually simple yet highly effective [13]. This model overcomes a major limitation of the standard language models: their unidirectional nature. Typically, prior models, such as OpenAI's GPT [12], process text in one direction, usually from left to right, where each token can only attend to previous tokens within the self-attention layers [5]. However, in question-answering (QA) tasks, it is essential to understand the context, namely the relationship between different parts of the text, to accurately pinpoint the answer. Hence, there is a need to incorporate context from both directions, thereby making BERT an excellent choice for IR tasks.

The input representation of BERT makes it compatible with a variety of downstream tasks as it can accommodate both single text spans as well as text pairs (e.g., `<query, passage>` for passage retrieval). For each input sequence, a special `[CLS]` token is prep-ended to the embedding. The final hidden state corresponding to this token aggregates the overall sequence representation, which is typically utilised for classification tasks (e.g., sentiment analysis). For text pairs, a `[SEP]` token is used to separate the segments. The input representation for each token is formed by summing the WordPiece [16] embedding of the token, the segment identifier (indicating whether it belongs to the first or second text span), and its position within the segment.

Devlin et al. proposed two unsupervised tasks for pre-training the deep bidirectional transformer, namely the "masked language model" (MLM) task, which enables merging the left and right context across all layers of the neural network and the Next Sentence Prediction task, which pre-trains using consecutive text-pairs [13].

Within the MLM task, 15% of the input tokens are masked at random. The model then tries to predict the masked tokens by feeding the final hidden vectors corresponding to the masked tokens to an output softmax function over the vocabulary. However, the authors highlighted the drawback of this approach: a mismatch between the pre-training and fine-tuning stages, as the `[MASK]` token is not present in the latter. In order to mitigate this to some extent, when a position is chosen to be masked, instead of replacing it with the `[MASK]` token every time, a random token is used in 10% of the cases, or it remains unchanged 10% of the time [13].

Additionally, the Next Sentence Prediction task trains the model to determine whether one sentence logically follows another, which is not explicitly addressed within MLM. This training improves the model performance on downstream tasks that require an understanding of the relationships between sentences, such as question answering (QA) and natural language inference (NLI).

3 State of the art models

Recent research in general text embedding presented state-of-the-art models that build upon BERT-based architectures. The models we experimented with, namely BGE [17], GTE [18], E5 [19], Nomic [10] and Arctic-Embed [4], differ primarily in their training datasets and minor architectural details. A summary of the specifications of these models can also be found in Table 3 within Appendix A.

3.1 Architecture details

On top of the Transformer [5] architecture, GTE, E5 and Nomic employ mean pooling over the deeply contextualised token representations. More precisely, they compute the average of the token embedding vectors, serving as the representation of the entire sequence.

The 768-dimensional Arctic-Embed is based on `e5-base-unsupervised`, as its weights were preferred to the detriment of the general purpose ones. However, it was concluded that this design choice had a weak impact on performance but increased the convergence speed, which proved useful during the model development. Additionally, the 384-dimensional Arctic-Embed is based on MiniLMv2 [20], featuring 23M parameters. Inspired by the ablation study made by Li and Li [21], rather than employing mean pooling strategy as GTE [18], the final state of the `[CLS]` token is used as the embedding vector within Arctic-Embed [4], thus, aligning with BGE's architecture [17].

Nussbaum et al. aimed to address the main drawbacks of existing models, namely their limited (input) sequence length, which is typically capped at 512 tokens [10]. To overcome this issue, Nomic is built upon an adapted BERT architecture, `nomic-bert-2048`, that targets a sequence length of 8192 tokens. The proposed modifications include replacing absolute positional embeddings with rotary positional embeddings [22], utilising SwiGLU activation instead of GeLU [23], and implementing Flash Attention [24], an IO-aware attention algorithm.

3.2 Training data

In order to create an embedding model that is highly discriminative, hundreds of millions of training instances are needed [19], which is significantly greater than task-specific datasets such as MS MARCO [25]. Additionally, the data should be extracted from a diverse set of resources to enhance the general nature of the model across different tasks.

Merrick et al. emphasised the reliance on its highly curated datasets used within both the pre-training and fine-tuning phases of Arctic-Embed, outlining that quantity is less important than quality and an excessive amount of low-quality data can lead to lower-quality models [4]. Quality filtering is applied to individual text pieces and text pairs to improve overall data quality. Each piece of the pair undergoes language and text quality filtering, while text pairs are carefully assessed for semantic similarity within the consistency filtering, thus enhancing the relevance of the dataset. Long sequences were truncated to 512 words, as it was observed that, within web-based corpora, answers were usually found at the beginning of the text span.

The pre-training phase utilises 308 million query-document pairs, of which 71% consist of web search documents paired with queries and titles. This methodology draws inspiration from Wang et al.'s insights on the effectiveness of training on large-scale datasets using web-crawled title-document pairs [19]. Meanwhile, the fine-tuning dataset comprises approximately 1 million pairs, which is created by merging the web search data with public datasets, including HotpotQA, NQ, Fever, and StackExchange. Due to

the scarcity of high-quality data and the need to add negative documents, the dataset is augmented using a synthetic mining strategy. This approach involves generating synthetic queries, as it has been observed that LLMs struggle to produce high-quality hard negatives (documents) that match the quality of those in existing corpora [4].

Li et al. [18] highlighted that a significant limitation within prior research was the reliance on in-house data for pre-training, which is also the case of Arctic-Embed. In contrast, GTE uses only open-source data without employing any filtering or cleaning methods, except for deduplication on text pairs. With almost 3 fold increase compared to Arctic-Embed, 800M text pairs were utilised for the unsupervised pre-training stage. These were extracted from a wide range of sources such as web pages, scientific papers, community QA forums, social media, and code repositories. The presence of hyperlinks is also leveraged, as it makes text extraction easier. What enhances the fine-tuning stage's effectiveness are the human-annotated datasets, summing up to around 3M text pairs from a variety of tasks and domains (e.g., web search, open-domain QA, fact-checking).

Xiao et al. released a publicly available dataset for general English embedding used for training the BGE models, encompassing around 200M text pairs employed for the model training [17]. The main source of data is web corpora, for which various structures are extracted, such as `<title, body>`, `<title, passage>`, `<question, answer>`, `<paraphrased titles>` and `<paraphrased answers>`. The data is thoroughly curated, similar to the process used for Arctic-Embed and E5, removing non-textual, duplicated, and malicious content.

E5 is also trained on an in-house dataset, CCPairs, which provides high-quality and diverse text pairs from web sources [19]. Most of the data comes from Reddit and Common Crawl, comprising 270M text pairs for contrastive pre-training. A remarkable approach was used for consistency-based filtering. Initially, a model is trained on 1.3B noisy text pairs, which is then employed to rank each pair against 1M random passages [19]. A text pair is retained only if it ranks within the top two passages. This technique draws inspiration from the memorization patterns observed in neural networks when employed on noisy datasets [26]. These mechanisms tend to learn the clean labels at the beginning and then gradually begin to overfit the noisy ones.

Nomic leverages publicly available data to form pairs, applying a consistency filtering process similar to that used in E5, yielding approximately 235 million pairs [10]. Since the majority of these datasets contain fewer than 2048 tokens, additional datasets that provide longer contexts are utilised to facilitate the learning of long-range dependencies. This includes `<title, article>` pairs from Wikipedia and `<abstract, paper body>` pairs from S2ORC [27].

3.3 Loss function

One of the most popular choices in terms of loss functions for training embedding models is infoNCE (cf. Eq. (2)), which values not only positive pairs but also the relationship between positive and negative pairs. More precisely, the training instances are formed from a query q, a positive (relevant) document d^+, and a set D^- of negative (irrelevant) documents [28]. Cosine similarity is used to compute the distance between the embeddings ($\phi(q, d)$).

$$L_{\text{CL}} = -\log\left(\frac{e^{\phi(q,d^+)}/\tau}{e^{\phi(q,d^+)}/\tau + \sum_{i=1}^{n} e^{\phi(q,d_i^-)}/\tau}\right) \tag{2}$$

The in-batch strategy is employed, which treats all documents in the minibatch that are associated with queries different from the current one as negative instances [29]. Karpukhin et al. highlighted the substantial improvement yield by enhancing the set of negatives with one "hard" negative retrieved by BM25, which has a high lexical score for a given question, despite not containing the answer [29]. These documents are difficult to differentiate as less relevant compared to those explicitly labelled as relevant, thus improving the model's ability to discriminate [4].

Unlike all the other models explored, which use the standard form of infoNCE, GTE features a bi-directional improved version (cf. Eq. (3)), which enhances the negative samples with both in-batched queries and documents [18].

$$L_{\text{ICL}} = -\frac{1}{n}\sum_{i=1}^{n}\log\left(\frac{e^{\phi(q_i,d_i)/\tau}}{Z}\right) \tag{3}$$

$$\begin{aligned} Z = &\sum_{j} e^{\phi(q_i,d_j)/\tau} + \sum_{j\neq i} e^{\phi(q_i,q_j)/\tau} \\ &+ \sum_{j} e^{\phi(q_j,d_i)/\tau} + \sum_{j\neq i} e^{\phi(d_j,d_i)/\tau} \end{aligned} \tag{4}$$

Thus, both query to document contrast (first two terms (cf. Eq. (4))), and its inverse (last two terms (cf. Eq. (4))) are utilised. Additionally, the temperature τ hyperparameter is fixed to 0.01 within GTE.

3.4 Training methods

The training procedure employed is consistent across GTE, E5 and Arctic-Embed models, which undergo two training rounds using different kinds of datasets: a large pre-training and a fine-tuning. The initial phase leverages in-batch negatives of query-document pairs, while the second one improves the performance of the model by introducing "hard" negative documents.

Arctic-Embed employs a tunable negative mining strategy, using positive query-document pairs, accompanied by a set of hard negatives. From these, a refined subset is selected by applying upper and lower thresholds on the relevance level between the query and the documents.

Within GTE's pre-training with in-batch negatives, larger batch sizes are preferred with the aim of reducing the gap between training and inference, thus using a maximum sequence length of 128 for facilitating this objective. In contrast, within Arctic-Embed, Merrick et al. [4] highlighted through an ablation study that better performance can be obtained when using a maximum sequence length of 256 and a batch size similar to those employed by BGE [17] and GTE [18]. During the fine-tuning stage, GTE's max sequence length is increased to 512 to improve the model's ability to

handle longer text. Additionally, a multinomial distribution-based sampling strategy is employed to address the significant size differences in the datasets used for GTE's pre-training.

Unlike the other models, BGE's training recipe features one more step before the contrastive learning, namely the pre-training with plain text, aiming to support the embedding task [17]. The MAE-style approach is employed in which the polluted text is encoded in a low-dimension representation, followed by the recovery of the clean text version utilising a lightweight encoder. Additionally, Xiao et al. outlined that the impact of different downstream tasks can be mutually contradicted [17]. For this reason, instruction-based fine-tuning is used by adding a specific instruction such as "Represent this sentence for searching relevant passages:" before each type of query: $q' \leftarrow I_t + q$.

Even though E5 employs a fine-tuning strategy similar to GTE's, it uses only a subset of the datasets included in this phase, namely MS MARCO, NLI, and NQ. Wang et al. reported the results for both the pre-trained only and fine-tuned version of E5, emphasising that the pre-trained only encoder (`e5-base-pt`) was the first unsupervised model to outperform BM25 on the BEIR benchmark [19]. For that reason, we were interested in finding the impact of both E5 versions within the FAST-FORWARD indexes framework.

Unlike the standard BERT model, used as the base for most of the analysed models, during the training of `nomic-bert-2048`, the base model of Nomic, the MLM task masks 30% of the tokens instead of 15% and the Next Sentence Prediction task is removed. This model uses a sequence length of 2048 during the contrastive pre-training phase for scaling up to a sequence length of 8192 during inference. Throughout fine-tuning, seven hard negatives per pair are employed, as it has been observed that additional hard negatives do not significantly enhance performance [10].

4 Experimental Setup

In this section, we present the experimental setup, delving into the model versions and datasets utilised, as well as details about latency and ranking performance measurement.

4.1 Pipeline

The pipeline used is based on the FAST-FORWARD indexes [3], employing BM25 for the first stage retrieval, which relies on exact term matching between queries and documents for candidate selection. This choice is motivated by the widespread use of BM25 [3; 30; 25; 31], being one of the earliest sparse retrieval models, renowned for effectively handling term frequency saturation, where additional occurrences of a word do not linearly increase the document relevance, and adjusting scores based on document length. Within our pipeline, the sparse retrieval depth (number of documents retrieved per query) is set to $k_s = 1000$. During the semantic re-ranking, Transformer-based encoders are employed. We explored models with dimensions of 384 and 768 to balance the trade-offs between memory usage and ranking performance. Larger dimensionalities were avoided due to large memory footprints, long indexing times, and impracticality in many production environments [4]. Therefore, our experiments feature the 768-dimensional versions of Arctic-Embed, BGE, GTE, E5 (both the pre-trained only and the fine-tuned versions) and Nomic, alongside the smaller 384-dimensional `bge-small`, `arctic-embed-xs` and `e5-small`. These models range from 23M (`arctic-embed-xs`) to 137M (`nomic`) parameters, allowing for flexibility in balancing between efficiency and effectiveness within the semantic re-ranking stage.

4.2 Datasets

We extensively evaluate the ranking performance on various datasets, utilizing large-scale open-evaluation benchmarks. A detailed summary of these datasets' specifications, including the average number of relevant documents per query and the average number of words per query and document, is provided in Table 4 in Appendix A.

The BEIR (Benchmarking-IR) benchmark [30] was employed, providing a robust and heterogeneous evaluation set for IR, aggregating a significant range of retrieval tasks across multiple domains. For our evaluation, we selected all datasets that included a development set because we found that tuning hyperparameters, specifically the α value for interpolating lexical and semantic scores, was essential. Therefore, we were able to analyse the following datasets along with their associated tasks and domains: NFCORPUS (bio-medical IR, bio-medical domain), HOTPOTQA (question answering, open-domain), FIQA (question answering, financial data), QUORA (duplicate question retrieval, open-domain), DBPEDIA-ENTITY (entity retrieval, open-domain), FEVER (fact checking, open-domain) and SCIFACT (fact checking, scientific domain).

The TREC Deep Learning track was utilised for the web search task, evaluating on the MS MARCO corpora [25], which features an extensive scale and real-world nature. This benchmark comprises of questions corresponding to real user queries from Bing's search log, each with a human-generated answer. In our experiments, the TREC-DL-PSG'19 (web-search, open-domain) ranking test set is employed, simulating a question answering scenario in which the system retrieves a short answer for the user's query [31].

4.3 Evaluation details

Latency evaluation. Our latency experiments are performed using a single machine featuring an Intel Core i7-10750H CPU (Comet Lake-H architecture). The latency per query is measured within the semantic re-ranking phase, including operations like encoding queries, retrieving document representations from the FAST-FORWARD indexes, computing similarity scores using the dot product between <`query, document`> pairs, interpolating lexical and dense scores and sorting documents. Both sparse and dense indexes are loaded into memory before experimentation to obtain faster retrieval times. However, as Leonhardt et al. outlined, the primary bottleneck remains encoding queries without GPU acceleration, which relies solely on the encoder capabilities [3]. This limitation allows us to assess the true performance of the models accurately.

The command line interface of the `timeit` Python module is employed, which performs seven runs of each exper-

iment, reporting the average of these measurements. However, as Beazley et at. highlighted, the module relies on `time.perf_counter()` function, which measures wallclock time and can be impacted by many different factors, such as machine load [32]. We have tried to mitigate this risk by ensuring the latency tests were the main active tasks on the machine despite the presence of essential system processes.

Ranking evaluation. The Pyterrier [33] toolkit is used for creating the sparse indexes for the first stage retrieval, while the encoders available on HuggingFace[1] were employed for generating query and document vector embeddings. Moreover, Pyterrier was also utilised to assess ranking performance as it provides a streamlined way of conducting experiments and computing ranking results. In terms of evaluation metrics, we used `nDCG@10` (Normalised Discounted Cumulative Gain), which assesses the quality of an ordered list of the first 10 documents, and `RR@10` (Mean Reciprocal Rank), which evaluates the position of the first relevant document.

Prior to interpolation, we normalise both lexical and semantic scores to address the significant variations in semantic scores across different models. This normalization is employed with the aim of having a standardised approach to tune the α hyperparameter used for score interpolation, which is optimised based on the `nDCG@10` metric for each dataset.

5 Experimental Results and Discussion

In this section, we present the results of our experiments, showing the impact of the semantic re-ranking within the FAST-FORWARD indexes pipeline. Although the model recommendations can be trivially retrieved from the result tables based on the metrics users want to optimise, we aim to present our hypothesis regarding the underlying reasons for the model performance. Each of the following subsections corresponds to one research question.

5.1 What is the ranking performance impact of different models during the semantic re-ranking stage?

5.1.1 Baseline model

Considering that `tct-colbert` was employed within the FAST-FORWARD framework when introduced by Leonhardt et al. [3], and its performance does not surpass that of state-of-the-art models, we considered it an appropriate choice for a baseline model.

5.1.2 Tasks and Domains

As outlined in Section 4.2, each task is represented by one dataset, with the exception of QA and fact-checking tasks.

For the QA task, we observed that there is no model that excels in both HOTPOTQA and FIQA. In contrast, for the fact-checking task, within FEVER and SCIFACT, GTE demonstrates superior performance for both datasets. Considering this variation, alongside the fact that the other tasks are represented by only one dataset, we propose focusing our analysis on individual datasets.

Similarly, each domain is represented by only one dataset, with the exception of the open-domain. The dataset-specific analysis is further justified by the observation that within the open-domain datasets (TREC-DL-PSG'19, HOTPOTQA, QUORA, DBPEDIA-ENTITY, FEVER), no model demonstrates outstanding performance consistently. Therefore, for the remainder of this section, we concentrate on analyzing the results based on individual datasets.

5.1.3 Supervised Fine-tuning Data

First, we believe that the datasets utilised during the supervised fine-tuning stage significantly influence the ranking results on specific datasets. For instance, GTE's superior results in the web-search task within the TREC-DL-PSG'19 evaluation set can be attributed to its inclusion of the MS MARCO dataset in its fine-tuning stage, unlike Arctic-Embed, which relies on its in-house curated web-search datasets for fine-tuning.

Similarly, Nomic and Arctic-Embed, unlike GTE, are fine-tuned on HOTPOTQA, which may contribute to their outstanding `nDCG@10` results on this dataset. The impact of fine-tuning datasets on semantic re-ranking performance is further supported within the NFCORPUS dataset evaluation. Since it is not included in the fine-tuning datasets of any model, as expected, all encoders exhibit similar results on this dataset, with the 368-dimensional versions showing slightly inferior ranking outcomes, as they fail to embed all the information, given the high (232.26) number of words per document within this dataset compared to the average (103.29).

Fine-tuning's impact on the MS MARCO corpus can also be observed in the case of `e5-small`, which outperforms all the other 384-dimensional encoders on the TREC-DL-PSG'19 evaluation set, being the only encoder of its size that includes it in its fine-tuning collection. It is also worth noticing that there are some instances in which the fine-tuned version of E5 performs worse than the pre-trained only model, showing that fine-tuning on a less diverse set can lead to overfitting.

We also encountered datasets such as FEVER, which are part of the fine-tuning datasets for multiple models, including both Arctic-Embed and GTE. For this reason, we explored additional factors influencing their ranking outcomes.

5.1.4 Mean polling and CLS token embedding

Our analysis reveals that GTE outperforms both BGE and Arctic-Embed on all the datasets with an average document word length exceeding 50 words: TREC-DL-PSG'19, NFCORPUS, FIQA, FEVER and SCIFACT. We hypothesise that GTE's superior performance is due to its use of mean-polling across the token representations to form the text embedding, while the other two models rely on the `[CLS]` token embedding, which is usually used for classification tasks [13] and might not always capture the most nuanced details of the input. Furthermore, in scenarios in which the document length is below 50 words, the `[CLS]` token-based models (BGE and Arctic-Embed) yield better results. For example, despite GTE including QUORA (11.44 words per document) in its fine-tuning datasets and BGE incorporating it during pre-training, BGE achieves superior outcomes on this dataset, which we attribute to the usage of the `[CLS]` embedding.

[1] https://huggingface.co/models

		Fast-Forward Indexes: BM25 >>									
		768-dimensional							384-dimensional		
	BM25	tct-colbert	gte-base	bge-base	arctic-m	e5-base	e5-base-pt	nomic	bge-small	arctic-xs	e5-small
TREC-DL-PSG'19	0.4795	0.6924	0.7137	0.6897	0.7042	0.6921	0.5889	0.6977	0.699	0.6924	0.7086
NFCORPUS	0.3223	0.3362	0.3649*	0.3623*	0.3619*	0.355*	0.359*	0.3582*	0.3593*	0.3408	0.3479*
HOTPOTQA	0.5128	0.6363	0.687*	0.7087*	0.7255*	0.6987*	0.6342	0.7307*	0.6873*	0.668*	0.6906*
FIQA	0.2526	0.3139	0.4755*	0.4103*	0.4241*	0.4148*	0.4169*	0.3878*	0.4084*	0.3555*	0.4038*
QUORA	0.7676	0.8464	0.8939*	0.8944*	0.8795*	0.8832*	0.8669*	0.8656*	0.893*	0.8718*	0.8734*
DBPEDIA-ENTITY	0.2744	0.4004	0.4145	0.4101	0.4443*	0.4313*	0.3898	0.4439*	0.4122	0.4071	0.4152*
FEVER	0.4273	0.6887	0.8672*	0.8058*	0.8155*	0.7528*	0.7045*	0.8171*	0.7983*	0.7608*	0.7659*
SCIFACT	0.6722	0.6901	0.7599*	0.7458*	0.7471*	0.7308*	0.7541*	0.7218*	0.7211*	0.7128*	0.7255*

Table 1: Ranking results of the FAST-FORWARD Indexes Framework on BEIR and TREC-DL benchmarks (nDCG@10). A retrieval depth of $k_S = 1000$ was used for the sparse (first stage) retrieval. For each dataset, the best-performing model is underlined. Statistical significant differences ($p < 0.05$) between the baseline model (`tct-colbert`) and the analysed models are reported with $*$.

5.1.5 Reciprocal Rank and Normalised Discounted Cumulative Gain

It is important to note that some datasets, including QUORA, FEVER and SCIFACT, contain few relevant documents per query (< 2) [30]. For these datasets, the position of the first relevant document, namely the reciprocal rank (`RR@k`), might be more important than `nDCG@k`, which takes into consideration the relevance of all first k documents. However, within our experiments for these datasets, the same models excelled in both metrics (see Table 2 in Appendix A).

5.1.6 Sequence length impact

Despite architectural improvements, Nomic is outperformed by models that use a sequence length of 512 tokens. This outcome was expected, as the datasets employed in our experiments had an average maximum document length of 232.26 words (NFCORPUS), making regular models a well-suited choice. Future research could extensively explore Nomic's capabilities within the re-ranking stage by utilizing benchmarks specialised for evaluating long-context scenarios such as LoCo [34].

5.2 What is the latency impact of different models during the semantic re-ranking stage?

Latency is measured only for the semantic re-ranking stage, as the first-stage (sparse) retrieval is constant for all methods [3]. Additionally, due to limited hardware resources and the fact that the indexes are loaded into memory for latency measurement, only FIQA (Figure 1), NFCORPUS (Figure 2) and SCIFACT (Figure 3) are evaluated. Additionally, the Pareto front [35] is displayed for each figure, signifying that no solution can enhance one metric, namely latency or `nDGC@10`, without degrading the other metric.

Firstly, when analysing the BGE, E5 and Arctic-embed models, we observe that the 384-dimensional versions are always faster, but they also lose context, thus achieving a lower `nDGC@10`. The main reason for this reduction in latency is the decreased number of computations required during inference, as each layer of the model performs smaller matrix multiplication operations.

In Figures 2 and 3, we can observe that `arctic-xs` excels in terms of latency across all datasets. However, when

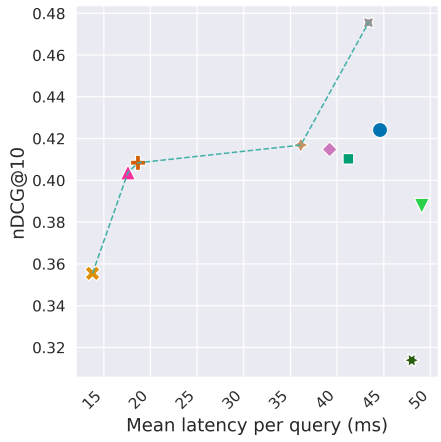

Figure 1: Latency vs. nDGC@10 on FIQA Dataset

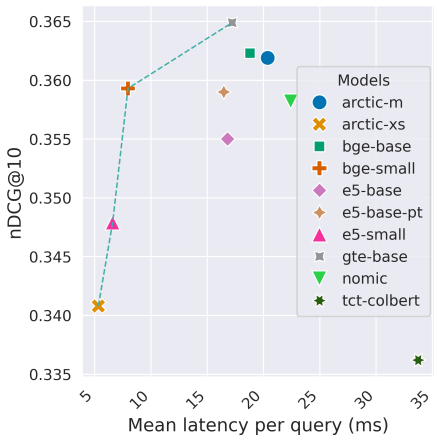

Figure 2: Latency vs. nDGC@10 on NFCORPUS Dataset

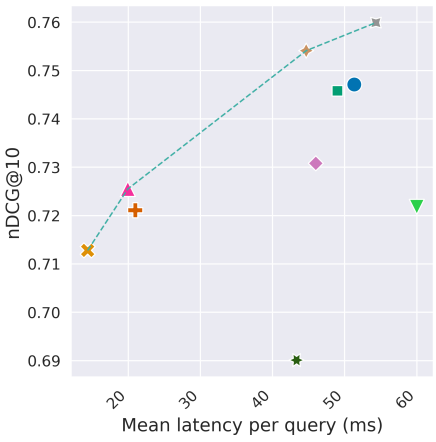

Figure 3: Latency vs. nDGC@10 on SCIFACT Dataset

analysing it against its 768-dimensional version, the drop in ranking performance (nDGC@10) is significant. In contrast, the 384-dimensional BGE and E5 do not follow the same pattern, as their ranking differences are insignificant, except for BGE on SCIFACT and E5 on NFCORPUS.

In our analysis, GTE demonstrates superior performance in terms of nDCG@10 compared to other models. However, when models with an embedding vector of 768 are used, GTE does not demonstrate outstanding latency performance. We hypothesise that this may be due to GTE's use of mean pooling, which averages all token embeddings, in contrast to faster models like Arctic-Embed and BGE, which utilise the [CLS] token embedding.

It is also worth noticing that the latency achieved is also dependent on the datasets. For instance, most models have latency in the range [5, 20] ms for the NFCORPUS dataset, while for SCIFACT, the range is [15, 50] ms. We believe that the difference is mainly due to the average number of words per query, which are 3.30 and 12.37, respectively. The Transformer-based encoders employed are quadratic in the length of the input [36]. For achieving the same vector size, the input is padded with [PAD] tokens, which are effectively ignored by the model's attention mechanism, thus resulting in lower latency for shorter inputs. Additionally, as shown in Figure 4, we observed that query encoding takes the largest part of the semantic re-ranking latency compared to other operations such as retrieving document representations from the FAST-FORWARD (dense) index, computing the scores as dot-products and sorting the results. Besides query encoding and document representation retrieval, these operations remain constant, regardless of the model employed.

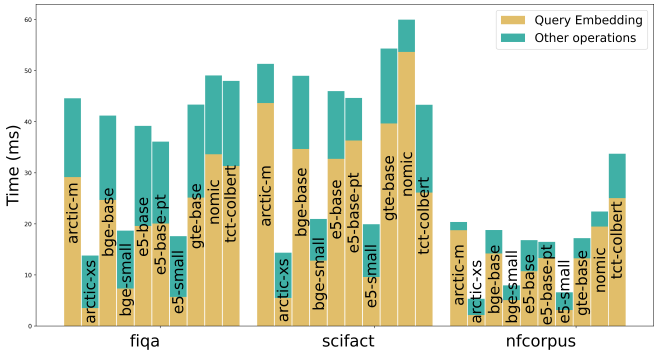

Figure 4: Breakdown of Latency per Query (ms)

6 Responsible Research

In this section, we discuss the broader implications of our research, ensuring that all experiments are reproducible, the data employed is publicly accessible, and our work adheres to ethical standards.

6.1 Reproducibility

Our study is compliant with the Netherlands Code of Conduct for Research Integrity (2018), and the FAIR (Findability, Accessibility, Interoperability, and Reusability) principles for scientific data management [37].

The code used within this research is *findable* and *accessible*, as it is uploaded to a public GitHub repository [2], which is an open-source platform. The datasets selected for model evaluation are part of widely used benchmarks (BEIR and TREC DEEP LEARNING), which are also publicly available [3]. Both the pre-trained models employed in our analysis, which are available on HuggingFace [4], as well as the Pyterrier library [5], used for evaluating the pipeline, are open-source.

The experimental results are both *interoperable* and *reusable* due to the comprehensive documentation provided with the codebase. This is also accompanied by the detailed Experimental Setup presented in Section 4. Additionally, we have stored the ranking results for each model across various α values used within the interpolation of sparse and dense scores. This data supports the selection of the α parameter in our final experimentation. The codebase is designed to prioritise extensibility, thereby facilitating researchers in replicating existing experiments, initiating new ones, and evaluating the impact of future state-of-the-art encoders on the semantic re-ranking of the FAST-FORWARD indexes framework.

We acknowledge that there are also some challenges related to replicability. Considering that we rely on multiple open-source resources, it is possible that underlying changes in these libraries could affect the functionality of our setup. Additionally, future users will benefit from faster indexing times when creating the dense indexes if a graphics card with CUDA cores is available. However, such hardware is not a requirement for conducting experiments within our codebase.

6.2 Ethical impact

The aim of our analysis is to offer guidelines for encoder usage in different scenarios while focusing solely on ranking and latency performance. While the FAST-FORWARD indexes framework can be employed within search engines or other IR tasks, it can also amplify biases, given its reliance on encoders that are based on neural networks. We recommend that future users explore the potential biases these models may introduce before utilizing them.

7 Future work

There are multiple paths future research can take to enhance our understanding of the impact of semantic re-ranking.

One potential direction is to conduct ablation studies on the hypotheses concerning the model's ranking performance, discussed in Section 5.1. For the encoders where we suggest that specific fine-tuning datasets might enhance results, a pre-trained version of the same model can be used. This model would undergo fine-tuning on all datasets on which the actual model is fine-tuned, excluding the one that is thought to have a significant impact in a particular scenario. This approach would allow us to isolate and assess the influence of that specific dataset on the ranking results.

Another line of research that can be followed is the impact of substituting dual-encoders with cross-encoders within the

[2] https://github.com/anistor09/Neural_Ranking_Models
[3] https://ir-datasets.com/
[4] https://huggingface.co/models
[5] https://github.com/terrier-org/pyterrier/

semantic re-ranking stage. Although cross-encoders are computationally more expensive and require a reduced retrieval depth within the sparse retrieval phase for reasonable latency, they might still achieve competitive ranking results at lower depths.

8 Conclusion

Our research evaluated the impact of the semantic re-ranking stage within interpolation-based re-ranking, using the FASTFORWARD indexes framework, by analyzing specific scenarios in which certain models demonstrate superior ranking performance and lower latency.

The results indicate that no single model excels across all datasets. Considering this, GTE [18] is a preferred choice for most datasets, but it is outperformed on HOTPOTQA by Nomic [10], on DBPEDIA-ENTITY by the 768-dimensional Arctic-Embed and on QUORA by the 768-dimensional BGE [17]. We hypothesise that these ranking performances are influenced primarily by the datasets employed during the fine-tuning stage and the vector embedding computation approach (mean pooling of token embeddings as opposed to utilizing the `[CLS]` token embedding) [21].

It can also be concluded that although GTE yields superior results for all datasets in the latency analysis, this model does not excel in efficiency. We believe that the reason for this is, similarly to ranking results, the approach employed for computing the vector embedding alongside the larger matrix multiplications employed at each layer. If latency optimization is crucial, the 384-dimensional Arctic-Embed [4], BGE [17] and E5 [19] are preferred. Additionally, it is important to note that the average query length has the most significant impact on latency for all models, as query encoding is a resource-intensive task and takes place only on the CPU.

In summary, this study offers detailed guidance on selecting the most appropriate models for various IR scenarios, highlighting the outstanding performance that the FASTFORWARD indexes pipeline can achieve. It demonstrates how to balance ranking efficiency and effectiveness without relying on expensive GPU-accelerated re-ranking, thereby making IR techniques based on semantic understanding more accessible and cost-effective for diverse applications.

References

[1] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at trec-3," *NIST Special Publication SP*, vol. 109, p. 109, 1995.

[2] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. [Online]. Available: https://aclanthology.org/2020.emnlp-main.550

[3] J. Leonhardt, H. Müller, K. Rudra, M. Khosla, A. Anand, and A. Anand, "Efficient neural ranking using forward indexes and lightweight encoders," *ACM Transactions on Information Systems*, 2023.

[4] L. Merrick, D. Xu, G. Nuti, and D. Campos, "Arctic-embed: Scalable, efficient, and accurate text embedding models," 2024.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[6] S.-C. Lin, J.-H. Yang, and J. Lin, "In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval," in *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, A. Rogers, I. Calixto, I. Vulić, N. Saphra, N. Kassner, O.-M. Camburu, T. Bansal, and V. Shwartz, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 163–173. [Online]. Available: https://aclanthology.org/2021.repl4nlp-1.17

[7] V. Gupta, M. Chinnakotla, and M. Shrivastava, "Retrieve and re-rank: A simple and effective IR approach to simple question answering over knowledge graphs," in *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, J. Thorne, A. Vlachos, O. Cocarascu, C. Christodoulopoulos, and A. Mittal, Eds. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 22–27. [Online]. Available: https://aclanthology.org/W18-5504

[8] E. Jung, J. Choi, and W. Rhee, "Semi-siamese bi-encoder neural ranking model using lightweight fine-tuning," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 502–511.

[9] C. Lassance and S. Clinchant, "An efficiency study for splade models," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '22. ACM, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1145/3477495.3531833

[10] Z. Nussbaum, J. X. Morris, B. Duderstadt, and A. Mulyar, "Nomic embed: Training a reproducible long context text embedder," *arXiv preprint arXiv:2402.01613*, 2024.

[11] S.-C. Lin, J.-H. Yang, and J. Lin, "In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval," in *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, A. Rogers, I. Calixto, I. Vulić, N. Saphra, N. Kassner, O.-M. Camburu, T. Bansal, and V. Shwartz, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 163–173. [Online]. Available: https://aclanthology.org/2021.repl4nlp-1.17

[12] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[14] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," 2020.

[15] R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview. arxiv 2019," *arXiv preprint arXiv:1912.05911*, 1912.

[16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[17] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, "C-pack: Packaged resources to advance general chinese embedding," 2023.

[18] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, "Towards general text embeddings with multi-stage contrastive learning," *arXiv preprint arXiv:2308.03281*, 2023.

[19] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[20] W. Wang, H. Bao, S. Huang, L. Dong, and F. Wei, "Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers," *arXiv preprint arXiv:2012.15828*, 2020.

[21] X. Li and J. Li, "Angle-optimized text embeddings," *arXiv preprint arXiv:2309.12871*, 2023.

[22] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[23] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[24] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 344–16 359, 2022.

[25] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "Ms marco: A human-generated machine reading comprehension dataset," 2016.

[26] V. Feldman and C. Zhang, "What neural networks memorize and why: Discovering the long tail via influence estimation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2881–2891, 2020.

[27] K. Lo, L. L. Wang, M. Neumann, R. Kinney, and D. S. Weld, "S2orc: The semantic scholar open research corpus," *arXiv preprint arXiv:1911.02782*, 2019.

[28] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *ArXiv*, vol. abs/1807.03748, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49670925

[29] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. [Online]. Available: https://aclanthology.org/2020.emnlp-main.550

[30] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models," *arXiv preprint arXiv:2104.08663*, 2021.

[31] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, E. M. Voorhees, and I. Soboroff, "Trec deep learning track: Reusable test collections in the large data regime," in *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 2021, pp. 2369–2375.

[32] D. Beazley and B. K. Jones, *Python cookbook: Recipes for mastering Python 3*. " O'Reilly Media, Inc.", 2013.

[33] C. Macdonald and N. Tonellotto, "Declarative experimentation ininformation retrieval using pyterrier," in *Proceedings of ICTIR 2020*, 2020.

[34] J. Saad-Falcon, D. Y. Fu, S. Arora, N. Guha, and C. Ré, "Benchmarking and building long-context retrieval models with loco and m2-bert," *arXiv preprint arXiv:2402.07440*, 2024.

[35] A. Kesireddy and F. Medrano, "Elite multi-criteria decision making—pareto front optimization in multi-objective optimization," *Algorithms*, vol. 17, p. 206, 05 2024.

[36] F. D. Keles, P. M. Wijewardena, and C. Hegde, "On the computational complexity of self-attention," in *International Conference on Algorithmic Learning Theory*. PMLR, 2023, pp. 597–619.

[37] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne *et al.*, "The fair guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

A Appendix

		Fast-Forward Indexes: BM25 >>									
		768-dimensional							384-dimensional		
	BM25	tct-colbert	gte-base	bge-base	arctic-m	e5-base	e5-base-pt	nomic	bge-small	arctic-xs	e5-small
TREC-DL-PSG'19	0.7944	0.82	0.7953	0.8463	0.8503	0.8178	0.6975	0.8643	0.8684	0.8585	<u>0.8779</u>
NFCORPUS	0.5344	0.5496	0.5852*	0.5767	0.5786*	0.5666	0.581*	0.5668	<u>0.5853*</u>	0.5507	0.5679
HOTPOTQA	0.6624	0.8007	0.8608*	0.8579*	<u>0.8785*</u>	0.8461*	0.7961	0.8777*	0.8359*	0.8327*	0.8376*
FIQA	0.3103	0.3853	<u>0.5587*</u>	0.4901*	0.5033*	0.4955*	0.4868*	0.4607*	0.4881*	0.427*	0.4721*
QUORA	0.7584	0.8382	0.8877*	<u>0.8888*</u>	0.871*	0.8766*	0.8579*	0.8575*	0.8871*	0.8645*	0.8661*
DBPEDIA-ENTITY	0.5268	0.7199	0.7381	0.7237	<u>0.7827*</u>	0.7671*	0.7229	0.7438	0.7423	0.7212	0.7412
FEVER	0.3839	0.6694	<u>0.8854*</u>	0.8058*	0.8164*	0.7378*	0.6829*	0.8165*	0.799*	0.7512*	0.7555*
SCIFACT	0.6324	0.6567	<u>0.7217*</u>	0.7106*	0.7041*	0.6998*	0.7163*	0.6849*	0.6867	0.6788*	0.6915*

Table 2: Ranking results of the FAST-FORWARD Indexes Framework on BEIR and TREC-DL benchmarks (RR@10). A retrieval depth of $k_S = 1000$ was used for the sparse (first stage) retrieval. For each dataset, the best-performing model is <u>underlined</u>. Statistical significant differences ($p < 0.05$) between the baseline model (`tct-colbert`) and the analysed models are reported with $*$.

	Model Specifications				
	Number of Parameters	Embedding Method	#Pre-training Pairs	#Fine-tuning Pairs	Fine-tuning Datasets
GTE	137M	Mean Pooling	788M	3M	MS MARCO, NLI, NQ, FEVER, QUORA
ARCTIC-EMBED	23M/109M	CLS Token Embedding	308M	1M	NQ, FEVER, HOTPOTQA, STACKEXCHANGE TITLE-BODY
BGE	34M/109M	CLS Token Embedding	200M	-	-
E5	34M/109M	Mean Pooling	270M	<3M	MS MARCO, NLI, NQ
NOMIC	137M	Mean Pooling	235M	-	MS MARCO, NLI, NQ, FEVER, HOTPOTQA

Table 3: Overview of the Transformer-based models employed within the semantic re-ranking of the FAST-FORWARD indexes pipeline. The table provides insight into the number of parameters (for each dimensionality) of each model employed in the analysis, the number of pairs used within pre-training and fine-tuning, and the datasets utilised for fine-tuning. If specifications are absent from this table, it indicates that the corresponding papers did not explicitly state them.

			Dev	Test			Avg. Word Length	
Task	Domain	Dataset	#Query	#Query	#Corpus	Avg. D/Q	Query	Document
Passage-Retrieval	Open-domain (Bing)	MS MARCO (TREC-DL-PSG'19)	6,980	200	8,841,823	58.61	5.96	55.98
Bio-Medical IR	Bio-Medical	NFCORPUS	324	323	3,633	38.2	3.30	232.26
Question Answering (QA)	Open-domain (Wikipedia)	HOTPOTQA	5,447	7,405	5,233,329	2.0	17.61	46.30
Question Answering (QA)	Finance	FIQA-2018	500	648	57,638	2.6	10.77	132.32
Duplicate-Question Retrieval	Open-domain (Wikipedia)	QUORA	5,000	10,000	522,931	1.6	9.53	11.44
Entity-Retrieval	Open-domain (Wikipedia)	DBPEDIA	67	400	4,635,922	38.2	5.39	49.68
Fact Checking	Open-domain (Wikipedia)	FEVER	6,666	6,666	5,416,568	1.2	8.13	84.76
Fact Checking	Scientific	SCIFACT	809	300	5,183	1.1	12.37	213.63

Table 4: Statistics of datasets in the BEIR [30] and TREC Deep Learning track [31] benchmarks, which were used to evaluate the impact of the semantic re-ranking of the FAST-FORWARD indexes. "Avg. D/Q" indicates the average number of relevant documents per query.