Approximate Nearest Neighbor Search and Lightweight Dense Vector Reranking in Multi-Stage Retrieval Architectures

Zhengkai Tu,[1] Wei Yang,[1,2] Zihang Fu,[1] Yuqing Xie,[1,2]
Luchen Tan,[1] Kun Xiong,[1] Ming Li,[1,2] Jimmy Lin[1,2] *
[1] RSVP.ai [2] David R. Cheriton School of Computer Science, University of Waterloo

ABSTRACT

In the context of a multi-stage retrieval architecture, we explore candidate generation based on approximate nearest neighbor (ANN) search and lightweight reranking based on dense vector representations. These results serve as input to slower but more accurate rerankers such as those based on transformers. Our goal is to characterize the effectiveness–efficiency tradeoff space in this context. We find that, on sentence-length segments of text, ANN techniques coupled with dense vector reranking dominate approaches based on inverted indexes, and thus our proposed design should be preferred. For paragraph-length segments, ANN-based and index-based techniques share the Pareto frontier, which means that the choice of alternatives depends on the desired operating point.

ACM Reference Format:
Zhengkai Tu, Wei Yang, Zihang Fu, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2020. Approximate Nearest Neighbor Search and Lightweight Dense Vector Reranking in Multi-Stage Retrieval Architectures. In *Proceedings of the 2020 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '20), September 14–17, 2020, Virtual Event, Norway*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3409256.3409818

1 INTRODUCTION

Multi-stage pipeline architectures are widely adopted for building retrieval systems, both in real-world deployments [16] as well as by researchers [1]. Typically, keyword-based retrieval provides a candidate list of documents that is then reranked by one or more following stages. Later stages often employ computationally expensive approaches—for example, applying inference with deep neural networks—but over smaller subsets of the candidates, thereby striking a balance between effectiveness and efficiency.

The primary goal of candidate generation (and the early stages) is to maximize the recall of retrieved documents while minimizing query latency. In this context, we explore two designs:

- Candidate generation with approximate nearest neighbor (ANN) search. In this work, we explore whether inverted indexes can be replaced by recent developments based on ANN search.
- Lightweight dense vector reranking. We examine whether recent work in representational learning can be deployed as a lightweight reranker to achieve better effectiveness–efficiency tradeoffs. By lightweight, we mean representations that can be pre-computed and approaches that do not depend on costly neural network inference over candidate documents.

The contribution of this work is a thorough exploration of the tradeoff space involving the two components above. While there has been plenty of research on ANN search, its use as a component in a multi-stage pipeline, especially coupled with lightweight vector-based reranking, has not been thoroughly explored. Our experiments reveal several interesting findings that lead to better overall designs for different operating points.

2 RELATED WORK

A brute-force approach to nearest neighbor search quickly becomes impractical as the size of the collection grows. Multi-dimensional indexes (e.g., KD-trees) are not suitable for sparse vectors with very large feature spaces (such as text), but *approximate* solutions based on local sensitive hashing [9] and quantization-based methods [10, 11] have proved workable. Approaches based on hierarchical navigable small world (HNSW) graphs [13] represent the current state of the art in ANN search based on a popular benchmark.[1] However, previous applications of this general approach to document retrieval [2] have yielded mixed results at best.

With the advent of neural networks, continuous representations have emerged as effective alternatives to traditional sparse, bag-of-words representations for text matching. For example, Henderson et al. [10] used deep averaging networks (DAN) to encode questions and answers for conversational answer retrieval; Lee et al. [12] used BERT [6] to generate vector representations for question answering. Conneau et al. [5] demonstrated approaches to learning sentence representations that are generalizable and transferable to many other tasks. The transformer-based Universal Sentence Encoder (USE) [3, 4] represents a further step in this direction.

Given advances in representational learning and ANN search, we feel that dense-vector approaches to document ranking are again worth pursuing. This work adopts USE as the starting point.

3 MULTI-STAGE RANKER DESIGN

We assume a standard multi-stage ranking architecture comprising n stages, S_1 to S_n. The initial stage S_1 is responsible for generating a list of k_1 candidates directly from the document collection, and all subsequent stages perform reranking on the output of the previous stage. Specifically, stage S_p receives a ranked list of k_{p-1} documents from the previous stage, applies internal reranking logic, and passes

*The first two authors contributed equally.

[1] http://ann-benchmarks.com/

ICTIR '20, September 14–17, 2020, Virtual Event, Norway

ACM ISBN 978-1-4503-8067-6/20/09…$15.00
https://doi.org/10.1145/3409256.3409818

along a ranked list of k_p documents to the next stage, with the only constraint that $k_p \leq k_{p-1}$.

In this work, we explicitly consider two designs: a single-stage pipeline with only a candidate generation stage S_1 and a two-stage pipeline with a candidate generation stage S_1 and a lightweight reranking stage S_2. Depending on the task, these outputs may be directly returned to users or they may serve as input to more heavyweight BERT-based reranking stages; see discussions below.

3.1 Candidate Generation

In most multi-stage ranking architectures, the initial candidate generation stage S_1 leverages inverted indexes, typically with bag-of-words queries, to produce documents that are fed to the subsequent rerankers. We compare this approach against an alternative based on approximate nearest neighbor (ANN) search on hierarchical navigable small world (HNSW) graphs, constructed from the sparse BM25 representation of the documents.

At retrieval time, the query is converted to a bag-of-words representation in a similar way as the documents and used to rank the most similar documents based on inner-product distance. This setup provides a fair comparison between the index-based and the ANN-based approaches, since they both operate on equivalent sparse bag-of-words representations, albeit manifesting different effectiveness–efficiency tradeoffs (ANN, of course, being *approximate* in inner-product search). In both cases, S_1 returns k_1 ranked documents.

3.2 Lightweight Reranking

Following the candidate generation stage S_1, we designed a lightweight reranking stage S_2 based on cosine similarity between dense vector representations. For this, we adopt the Universal Sentence Encoder (USE) [3, 4] to represent documents. Under this approach, we can pre-compute and store the vector representations of all documents in the collection, such that reranking can be accomplished by encoding the query, looking up the representations of the candidate documents, and computing the cosine similarity between the query and the stored representations (by brute force). The output of the reranker is the top k_2 documents by score. This approach is lightweight because reranking does *not* require any (potentially costly) neural network inference on candidate documents.

Of course, the effectiveness of this reranking approach depends on the quality of the document representations. Evidence from the literature suggests, in general, that the quality of vector representations degrades, across multiple tasks, as we attempt to encode longer segments of text. Thus, document length is an important experimental variable in our setup that we seek to understand. We note that USE was designed for encoding *sentences*,[2] but we wondered if the approach can be extended to *paragraphs*.

4 EXPERIMENTAL SETUP

4.1 Tasks

Our evaluations considered three different tasks:

Similar question retrieval with Quora Question Pairs (QQP). This dataset contains approx. 400K question pairs from Quora, averaging 11 words per question. Each pair is labeled 1 or 0, with 1 denoting that the sentences are semantically similar. In order to formulate a retrieval task (for example, in a community QA application), we followed exactly the procedure of Gillick et al. [8]. We treated the dataset as a graph where unique sentences are vertices, with edges between each pair of similar sentences. At test time, each sentence in a positive pair forms a query, and all sentences reachable in the graph form the "relevant" set for that query.

Passage retrieval on the MS MARCO passage collection [14], containing 8.8M passages with 57 words on average, which has been used in many recent evaluations at TREC and beyond. We used the development set, comprising 6980 queries, for evaluation.

Retrieval-based factoid QA on Wikipedia articles. We tested on SQuAD 1.1 following exactly the evaluation procedure of Yang et al. [19]. The target collection, comprising 5M Wikipedia articles, was segmented into 30M paragraphs with 66 words on average. The ground truth for this task is given as answer spans.

For convenience, we refer to the unit of retrieval generically as "documents", when in fact they are sentences (QQP) or paragraphs (MS MARCO and Wikipedia). The first two tasks were selected primarily because they represent texts of different lengths, an important experimental variable discussed in Section 3.2.

For QQP, we use Recall@10 as the metric. The average number of relevant labels per query is 3.05 and with a cutoff of 10 we already achieve a high level of effectiveness with low query latencies (see results for more details). For passage retrieval on MS MARCO, we use Recall@100 as the effectiveness metric because it places the end-to-end system in an operating region that is most practical given the state of the art today, which uses a BERT-based reranker for the final stage. As Nogueira et al. [15] reported, feeding 1000 passages to BERT yields latencies of over three seconds per query (using TPUs), which is far too slow for real-world use.

We purposely selected retrieval-based QA as the final task to explore the limits of our designs, where ranked results are fed into a BERT-based model that identifies the answer span, and following the literature, we evaluate effectiveness using the exact match (EM) metric. We expect this task to *not* favor our designs because answer extraction with transformers is slow and inference time dominates any upstream latency savings.

4.2 Detailed Settings

Index-based S_1 stage. For passage retrieval and similar question retrieval, we used Elasticsearch 7.3. For factoid QA, we used the Anserini IR toolkit [18], matching the setup of Yang et al. [19]. In both cases, we use BM25 with the settings of $k_1 = 0.9$ and $b = 0.4$. In our results, we refer to this condition as "BM25 (index)".

ANN-based S_1 stage. We used Lucene and `gensim` to generate BM25 sparse document representations, on which HNSW indexing was performed with `nmslib`[3] using inner product distance. We limit the vocab size to 200k. Following previous work, HNSW indexing parameters were set as $M = 35$ and $\text{ef}_{\text{construction}} = 2500$, similar to Fu et al. [7]; $\text{ef}_{\text{search}}$ was kept at 1000. In our results, we refer to this condition as "BM25 (ANN)".

[2] https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3

[3] https://github.com/nmslib/nmslib

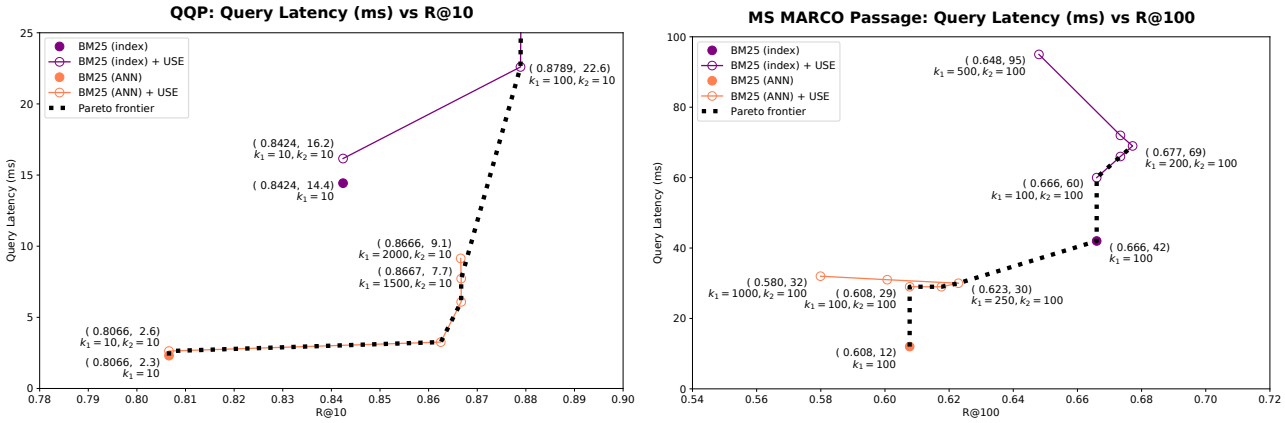

Figure 1: Effectiveness–efficiency tradeoffs for QQP (left) and MS MARCO passage (right). Pareto frontiers shown in black.

USE reranking S_2 stage. We use `USE-multilingual-large` as our encoder (available from TensorFlow Hub) and feed documents as strings directly without any pre-processing. We did not explicitly differentiate between passages and sentences. Queries are processed in exactly the same way.

Hardware configuration. All experiments were run on an otherwise idle machine with an Intel Xeon E5-2620v4 processor and 128GB of RAM. All neural inference used a single Tesla P40 GPU.

5 RESULTS

The effectiveness–efficiency tradeoff space for similar question retrieval on QQP is shown in Figure 1 (left). Operating points of interest are annotated with both evaluation results as well as the parameter settings (k_1 for S_1 and k_2 for S_2). Since our metric is Recall@10, with a single stage S_1 (either index- or ANN-based), $k_1 = 10$ is the only setting that makes sense, which corresponds to the solid orange and purple circles. However, with the addition of the USE reranking stage S_2, it is possible to sweep increasing values of k_1 to obtain a tradeoff curve (while holding $k_2 = 10$ for fair evaluation); these are the orange and purple lines for the index-based and ANN-based first stages, respectively.

We see that the ANN-based S_1 is faster but produces worse results, while the index-based S_1 is slower but produces better results. However, the addition of S_2 USE reranking over ANN S_1 brings about noticeable effectiveness gains with only a modest increase in latency. Most of the latency comes from reranking itself, as the speed of ANN S_1 is relatively insensitive to k_1. While S_2 USE reranking also benefits index-based S_1, increasing k_1 yields longer S_1 retrieval times, which increases overall latency to a greater extent. Furthermore, the upper bound effectiveness is around 0.88 (i.e., what can be achieved with *any* setting). That is, the extension of the purple line rises almost vertically beyond the bounds of the plot, and recall can only be further improved (slightly) at a great cost in terms of latency.

The Pareto frontier in the effectiveness–efficiency tradeoff space is shown as the dotted black line. For each point on the frontier, there exists no other setting that achieves both higher recall *and* lower latency; that is, each point represents that best possible tradeoff for a particular operating point. We see that ANN-based S_1 + USE reranking S_2 dominates the frontier—except for the highest recall, this design is preferred. Further note that, for this task, S_2 outputs are of sufficient quality for human consumption and further reranking with a slow BERT-based model is unlikely to be worthwhile. Thus, this graph characterizes the end-to-end tradeoffs.

For sentences, which form the "sweet spot" for USE encoding, S_2 reranking seems to work well. What if we extend to paragraphs? For MS MARCO passage retrieval, the effectiveness–efficiency tradeoff space is characterized in Figure 1 (right). Due to the nature of the task and longer passages, "interesting" operating points occupy different k_1 and k_2 settings. We see that the relative positions of the single-stage S_1 approaches are similar: that is, ANN search is faster but less accurate than index search. However, the addition of the USE reranker S_2 yields less improvement on MS MARCO than on QQP (i.e., the orange line does not extend to the right as far); this is not unexpected given the recommended use of USE on *sentences*. Consequently, index-based setups more easily overtake ANN-based setups along the Pareto frontier.

We note that each curve has an interesting "kink" in it: starting with S_1, adding S_2 first trades off query latency (slower) for better quality (higher recall), but beyond a point, the output becomes worse in both tradeoff dimensions (lower recall *and* slower). This is a surprising finding that to our knowledge has not previously been reported in the literature. We explain as follows: S_1 (either index- or ANN-based) is focused on exact term matching, while S_2 performs semantic matching. With smaller k_1 values, USE reranking only receives paragraphs that already have high term matching scores, on top of which semantic matching can improve recall by bringing relevant paragraphs to higher ranks. However, beyond a certain point, USE reranking starts receiving paragraphs that may have poor term matching scores, and semantic matching becomes distracted by paragraphs with fewer matching query terms. Without strong term-matching signals, semantic matching alone is insufficient to capture overall relevance.

We further note that the "kink" we observe in the MS MARCO case is indeed also present in the QQP plot as well, but the effect is more subtle. Focusing on the orange line (Figure 1, left), it does bend "leftwards" with greater k_1 values—that is, beyond a certain point (the "apex" marked in the figure), R@10 becomes slightly worse as latency continues to rise. The "kink" similarly occurs with the purple line of QQP, but beyond the bounds of the figure.

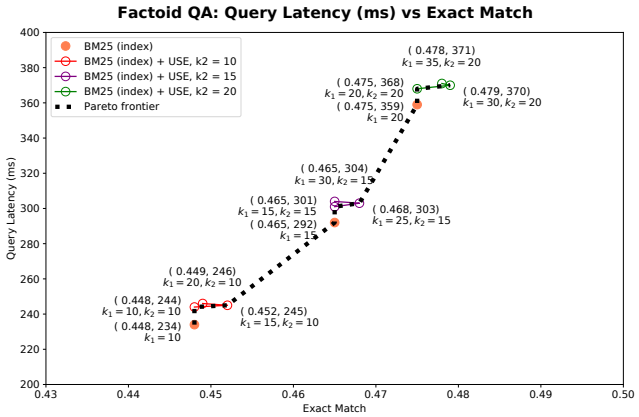

Figure 2: The end-to-end tradeoff space for factoid QA.

Our results thus far show that paragraph-length documents already push S_2 reranking away from the operating region it was designed for, making it much less effective than when encoding sentences. We further push the limits of our design in the QA task by introducing slow BERT-based models for span extraction. The tradeoff space for the factoid QA task on Wikipedia is shown in Figure 2. The starting point for comparisons is the two-stage approach of Xie et al. [17]: keyword search with BM25 followed by a BERT reader to extract the exact answer spans. We have replicated this design, and the comparable condition is shown with orange circles, sweeping different values of k_1. Here we focus on an operating region with latencies that are practical for real-world deployment; higher EM is possible but with latency measured in seconds.

In this context, the index-based approach is preferred to ANN because the slight latency advantages of the latter are negligible given costly BERT inference. Nevertheless, with each single-stage setup there is a comparable setup where we inject USE reranking (i.e., retrieve k_1 paragraphs with BM25, rerank with USE, then feed top k_2 paragraphs to the BERT reader). These are shown as the short colored lines above each solid orange circle. We see that, in each case, we can increase EM slightly at small costs in latency, which provides the developer with an alternative of simply varying k_1 in a single-stage architecture. Thus, even in this scenario that is unfavorable to our designs, USE reranking may still have a role to play in finding the best end-to-end tradeoffs.

6 DISCUSSION

What do we make of these results at a high level? We believe that two conclusions are warranted:

(1) For S_1, ANN is faster but less effective; this holds for our sparse BM25 representations, on both sentences and paragraphs.
(2) For S_2, our USE encoder is effective for reranking sentences; this isn't surprising since sentences are the designed use case. As a result, an ANN + USE combination dominates much of the tradeoff space, and is preferred for all operating points except for the highest recall requirements (with high latency); see Figure 1, left. USE can still be useful for reranking paragraphs, but is not as effective, and thus the ANN + USE combination is preferred for a smaller range of operating points. However, USE can still benefit index-based approaches; see Figure 1, right.

Furthermore, we note the need to couch these conclusions in the final end-to-end pipeline, which may or not may include feeding the output of S_2 to more rankers. For QQP, we argue that an S_3 reranker doesn't make much sense given the already high quality of the output. For factoid QA, the nature of the task makes a final span extraction stage necessary. In the latter case, the high inference cost of transformers greatly reduces the flexibility offered by our designs. However, accelerating inference in such models is the subject of much recent work, which means that span extraction will dominate end-to-end latencies less and less, thus making tradeoffs in the earlier stages (as we explore here) more important.

7 CONCLUSION

The contribution of our paper is a thorough characterization of the tradeoff space along three dimensions: index-based vs. ANN-based candidate generation, the effectiveness of lightweight reranking, and the impact of document length. Our findings provide the system designer guidance on the nature of the effectiveness–efficiency tradeoffs with respect to these dimensions. Selection of the appropriate operating point depends on operational constraints, but how to determine these lies beyond the scope of this paper.

REFERENCES

[1] N. Asadi and J. Lin. 2013. Effectiveness/Efficiency Tradeoffs for Candidate Generation in Multi-Stage Retrieval Architectures. In *SIGIR*.
[2] L. Boytsov, D. Novak, Y. Malkov, and E. Nyberg. 2016. Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search. In *CIKM*. 1099–1108.
[3] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, Guajardo-Cespedes M, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil. 2018. Universal Sentence Encoder for English. In *EMNLP: System Demonstrations*. 169–174.
[4] M. Chidambaram, Y. Yang, D. Cer, S. Yuan, Y. Sung, B. Strope, and R. Kurzweil. 2019. Learning Cross-Lingual Sentence Representations via a Multi-task Dual-Encoder Model. In *RepL4NLP-2019*. 250–259.
[5] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*. 670–680.
[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.
[7] C. Fu, C. Wang, and D. Cai. 2019. Satellite System Graph: Towards the Efficiency Up-Boundary of Graph-Based Approximate Nearest Neighbor Search. (2019). arXiv:1907.06146
[8] D. Gillick, A. Presta, and G. S. Tomar. 2018. End-to-End Retrieval in Continuous Space. *arXiv:1811.08008* (2018).
[9] A. Gionis, P. Indyk, and R. Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *VLDB*. 518–529.
[10] M. Henderson, R. Al-Rfou, B. Strope, Y. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, and R. Kurzweil. 2017. Efficient Natural Language Response Suggestion for Smart Reply. *arXiv:1705.00652* (2017).
[11] H. Jegou, M. Douze, and C. Schmid. 2011. Product Quantization for Nearest Neighbor Search. *PAMI* 33, 1 (2011), 117–128.
[12] K. Lee, M. Chang, and K. Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *ACL*. 6086–6096.
[13] Y. Malko and D. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *PAMI* 42, 4 (2020), 824–836.
[14] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016. MS MARCO: A Human-Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268* (2016).
[15] R. Nogueira, W. Yang, J. Lin, and K. Cho. 2019. Document Expansion by Query Prediction. *arXiv:1904.08375* (2019).
[16] Jan Pedersen. 2010. Query Understanding at Bing. In *SIGIR*.
[17] Y. Xie, W. Yang, L. Tan, K. Xiong, N. Yuan, B. Huai, M. Li, and J. Lin. 2020. Distant Supervision for Multi-Stage Fine-Tuning in Retrieval-Based Question Answering.. In *WWW*. 2934–2940.
[18] P. Yang, H. Fang, and J. Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *JDIQ* 10, 4 (2018), Article 16.
[19] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin. 2019. End-to-End Open-Domain Question Answering with BERTserini. In *NAACL: Demos*. 72–77.