
(doi: 10.1145/3459637.3482156)

There may be differences between this version and the published version. You are advised to consult the publisher’s version if you wish to cite from it.

© 2021 Copyright held by Association for Computing Machinery. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in 30th ACM International Conference on Information and Knowledge Management, 01-05 Nov 2021, pp. 3318-3322. ISBN 9781450384469

http://eprints.gla.ac.uk/249263/

Deposited on: 24 August 2021

Enlighten – Research publications by members of the University of Glasgow
http://eprints.gla.ac.uk
On Approximate Nearest Neighbour Selection for Multi-Stage Dense Retrieval

Craig Macdonald
University of Glasgow, UK
craig.macdonald@glasgow.ac.uk

Nicola Tonellotto
University of Pisa, Italy
Nicola.Tonellotto@unipi.it

ABSTRACT
Dense retrieval, which describes the use of contextualised language models such as BERT to identify documents from a collection by leveraging approximate nearest neighbour (ANN) techniques, has been increasing in popularity. Two families of approaches have emerged, depending on whether documents and queries are represented by single or multiple embeddings. ColBERT, the exemplar of the latter, uses an ANN index and approximate scores to identify a set of candidate documents for each query embedding, which are then re-ranked using accurate document representations. In this manner, a large number of documents can be retrieved for each query, hindering the efficiency of the approach. In this work, we investigate the use of ANN scores for ranking the candidate documents, in order to decrease the number of candidate documents being fully scored. Experiments conducted on the MSMARCO passage ranking corpus demonstrate that, by cutting off the candidate set by using the approximate scores to only 200 documents, we can still obtain an effective ranking without statistically significant differences in effectiveness, and resulting in a 2× speedup in efficiency.

CCS CONCEPTS
- Information systems → Information retrieval; Information retrieval query processing; Retrieval models and ranking.

KEYWORDS
Approximate Nearest Neighbours; Dense retrieval.

ACM Reference Format:

1 INTRODUCTION
Search engine architectures often follow a cascading architecture [10, 18], in that an initial inexpensive ranking approach (e.g. BM25 applied on a classical inverted index) identifies a candidate set of documents of say k documents, which are then further processed and re-ranked by more expensive ranking approach(es), to identify the final few documents to present to the user. While learning-to-rank with hand-engineered features has typified such a setup, increasingly contextualised language models, such as BERT [2], have been shown to be efficient, it is less effective than NN search on multiple representations [11]. On the other hand, to scale to number of required vectors, multiple representations require approximate nearest neighbour (ANN) data structures and algorithms. Hence, Khattab and Zaharia [9] propose a two-stage dense retrieval cascading approach. The first stage performs the ANN search to retrieve a set of candidate documents, maximising the recall of the retrieved set. The second stage computes accurate scores for the first-stage candidate documents, to return the final top documents. Overall, compared to re-ranking BM25 results, dense retrieval is appealing as it uses the same model end-to-end, therefore improvements to the learned model results in effectiveness improvements in both stages. In the ColBERT dense retrieval approach, the approximate (distance) scores computed in the first stage are ignored in the second stage, as the exact scores for the candidate documents are computed. However, the time spent in the second stage re-ranking is directly proportional to the number of documents retrieved in the first stage. In this paper we investigate how rankings can be instantiated upon the approximate first-stage retrieval component, such that a smaller candidate ranking of documents can be passed through to the second stage. In summary, this work contributes an examination on the effectiveness of ANN ranking strategies, both alone, and in combination with an effective second stage re-ranker.

2 DENSE RETRIEVAL
In this section, we describe a generic architecture for a multi-stage multi-representation dense retrieval system (Section 2.1), as exemplified by the ColBERT approach [9], and discuss how to obtain and use rankings from the first stage of this architecture (Section 2.2).
The document embeddings from all documents in the collection are vector comparisons. Indeed, ANN search is applied to compute, and document embeddings, however, it is not used by ColBERT. Approximate 
[7], can provide the approximate similarity between query tokens as query embedding \( \phi \) i.e., \( (\hat{\phi}_1, \ldots, \hat{\phi}_n) \). Query token embeddings are computed at runtime; queries may also be augmented with additional masked tokens to provide which have a query expansion-like role \[9\].

To support dense retrieval, the general architecture follows a two-stage retrieval cascading architecture. In this setting, a first stage identifies a candidate set of documents using approximate nearest neighbour; in the second stage these documents are re-ranked by a more expensive exact stage. The overall architecture in shown in Figure 1. At the time of writing, ColBERT \[9\] is the only effective dense retrieval system exploiting the multiple representations for queries and documents proposed thus far, exhibiting higher effectiveness than dense retrieval based on single query and document representations such as ANCE \[22\] (see \[11, Table 27 vs. Table 28\]).

### 2.1 Multi-Stage Multi-Representation Dense Retrieval Architecture

**2.1.1 Set Retrieval of Query Embedding Approximate Neighbours.**

The document embeddings from all documents in the collection are pre-computed and stored into an index data structure for vectors supporting similarity searches. This can identify the closest vectors to a given input vector when leveraging cosine or dot product vector comparisons. Indeed, ANN search is applied to compute, for each query embedding \( \hat{\phi}_i \), the set \( \Psi(\hat{\phi}_i, k') \) of the \( k' \) document embeddings most similar to \( \hat{\phi}_i \) according to some approximate distance; then, these document embeddings are mapped back to their corresponding documents \( D(\hat{\phi}_i, k') \):

\[
D(\hat{\phi}_i, k') = \{ d \in D : f_2(d) \cap \Psi(\hat{\phi}_i, k') \neq \emptyset \}
\]

and finally the union \( D(k') \) of these sets is returned:

\[
D(k') = \bigcup_{i=1}^{l(q)} D(\hat{\phi}_i, k')
\]

Indeed, the ANN search can return multiple document embeddings occurring with a single document; the same document can also be retrieved by more than one query embedding, therefore the total returned documents will usually be less than \( n \times k' \). Moreover, some ANN search techniques, e.g., those based on product quantisation \[7\], can provide the approximate similarity between query and document embeddings, however, it is not used by ColBERT.

**2.1.2 Exact MaxSim Scoring.** Once the approximate nearest documents \( D(k') \) have been identified, they are exploited to compute the final list of top \( k \) documents to be returned. To this end, the set of documents \( D(k') \) is re-ranked using the query embeddings and the documents’ multiple embeddings to produce exact scores, in order to determine the final ranking.

Given two embeddings, their similarity is computed by the dot product. Hence, for a query \( q \) and a document \( d \), their final similarity score \( s(q, d) \) is obtained by summing up the maximum similarity between the query embeddings and document embeddings:

\[
s(q, d) = \sum_{i=1}^{l(q)} \max_{j=1, \ldots, |d|} \phi_i^T \psi_j
\]

To achieve this, the exact, uncompressed embeddings are stored in a further index structure. Due to the large numbers of documents in the candidate set, the index structure needs to be stored entirely in memory. The MaxSim operator determines the final ranking of documents in response to the query.

### 2.2 Rankings from the Approximate First Stage

The number of documents in the candidate set produced by ColBERT, denoted \( k \), plays an important role. Indeed, in the ColBERT instantiation, \( k \) is a function of the size of the union of the documents identified by the \( k' \) nearest document embeddings to each query embedding. However, allowing each query embedding the same chance to contribute to the candidate set may be sub-optimal. Indeed, consider a query embedding representing a stopword appearing in the query – retrieving many nearest neighbours to that query embedding is unlikely to retrieve as many relevant documents as a more discriminative query embedding \[4, 21\].

For this reason, in this paper we investigate different ways to instantiate a ranking of the candidate documents. In doing so, we aim to show that it is possible to (a) have a more precise control of the number of documents requiring further scoring in the second stage ranker, and (b) identify a candidate set that is more effective than a candidate set of a similar size identified using \( k' \). In particular, we investigate four methods:

**Kprime:** This is the default method provided by ColBERT, where the size of the candidate set \( k \) is indirectly controlled by \( k' \), as described above. The candidate documents are unordered.

**Count:** As a document is represented by multiple document embeddings, and each document embedding can be retrieved by one or more query embeddings, we score documents by the number of times a corresponding document embedding is retrieved in the ANN stage. The candidate documents are ranked by descending count.

**Sum Sim:** This method scores and ranks documents by summing the approximate similarities for each query embedding retrieving that document. Approximate similarities are provided by the ANN search, but are not used in the ColBERT implementation. Indeed, they can be inaccurate, due to the ANN quantisation.

**Max Sim:** This method is similar to Sum Sim, but scores and ranks documents by taking for each query embedding the maximum approximate similarity score appearing for any of the document embeddings in the document. In essence, this applies Equation (3) at the first stage, but only for the document embeddings that are

\[1\] Indeed, in [21], we show that query embeddings corresponding to tokens with high frequency in the collection can be omitted during nearest neighbours retrieval.
To address our first research question, we examine the effectiveness of approximate rankings obtained from the ANN index, in comparison to a two-stage (“end-to-end”) ranking using the default setting of \( k \) = 1000 = 1000. The top half of Table 1 reports the effectiveness on both the TREC 2019 and MSMARCO Dev query sets, in terms of high precision metrics as well as Recall. For approximate ranking methods, we obtain the top-ranked \( k = 1000 \) documents, which is a commonly used setting for a first-stage ranker in LTR settings \([15, 20]\). Note that Kprime is omitted, as it forms a set, which cannot be evaluated using ranking metrics such as NDCG@10.

On analysing the top half of the table, it is easy to observe that the approximate scores obtained from Count, Approx SumSim and Approx MaxSim are insufficiently accurate to obtain high MRR, NDCG or MAP values. Indeed, all metrics are significantly decreased (paired t-test, Bonferroni multiple testing correction), however Approx MaxSim appears to be the most effective. On the other hand, the approximate ranking methods only result in a \(~14-15\%\) reduction in the Recall of relevant documents on the TREC 2019 query set \((0.77 \rightarrow 0.63)\), despite reducing the number of retrieved documents by \(~86\%\) compared to the default end-to-end configuration. Similarly, for Dev, Recall is reduced by only \(~7-8\%\), with an analogous reduction in the number of documents retrieved. This suggests that applying the approximate ranking methods to identify a smaller candidate set may be a promising direction, which we study further in RQ2.

To analyse further the accuracy of the approximate methods, Figure 2 shows, for two example queries, how the similarity between the rank each document obtained when scored by the full end-to-end ColBERT pipeline \((x\text{-}axes)\) compared to that obtained from Approx MaxSim \((y\text{-}axes)\). Perfect correlation would be exemplified by all points appearing on the dashed \( y = x \) line. For the first query, ‘chemical reactions’ (Figure 2(a)), there is only a moderate correlation \((\rho = 0.57)\) about the ranks of the documents obtained when using exact and approximate scoring methods. However, a cluster of documents in the top \(~300\) ranks can be observed \((\text{see red box})\) – both rankings agree about what documents should belong in the top \(300\), but not their relative ordering. We postulate that this cluster may be related to exact matches, which ColBERT scores highly \([3]\). For the longer query ‘what is wifi vs bluetooth’, Figure 2(b), we observe less overall correlation \((\rho = 0.40)\), but a little more shading near the origin \((\text{see red box})\), which illustrates more agreement about the highest ranks.

Overall, to answer RQ1, we find that ranking using the approximate scores obtained from the FAISS index result in marked reductions in high precision, but reasonable agreement in which documents should be retrieved (as demonstrated in the Recall reductions, and exemplified by the clusters observed in Figure 2). For this reason, in RQ2, we examine the utility of approximate scoring methods for identifying candidate sets in a two-stage pipeline.

### 4.2 RQ2 - Multi Stage Evaluation

Figure 3 plots different retrieval metrics for TREC 2019 and MSMARCO dev query sets, as the number of retrieved documents \((\text{i.e. the candidate set size, } k)\) varies for different strategies, namely Kprime, Count, Approx SumSim and Approx MaxSim. In particular, for Kprime, we vary \( k' \) in \([1, 10]\) with step size 1, \([10, 50]\) with step size 10, and \([100, 1000]\) with step size 100. For the approximate ranking methods, we control \( k \) directly, in \([10, 20, 50], [100, 1000]\) with step size 100 and \([1000, 5000]\) step size 500. Note that the default ColBERT setting of \( k' = 1000 \) results in, on average, \( k \approx 7100 \) documents being retrieved for each query and, for legibility, is not shown on the right hand side of the figures. In the figures, hollow circles denote points with statistically indistinguishable effectiveness compared to the default ColBERT setting \((k' = 1000)\), according to a paired t-test, \( p < 0.05\), applying Bonferroni correction.
Table 1: Effectiveness of first-stage (top half) and end-to-end retrieval (bottom half) when using different approximate ranking methods. * denotes a significant difference in effectiveness compared to the default ColBERT "end-to-end" configuration, according to a paired t-test, \( p < 0.05 \), with Bonferroni multiple testing correction.

<table>
<thead>
<tr>
<th>Approach</th>
<th>TREC 2019</th>
<th>MSMARCO Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( k )</td>
<td>MRR</td>
</tr>
<tr>
<td>End-to-end</td>
<td></td>
<td>7199</td>
</tr>
</tbody>
</table>

| RQ1 - Approximate Only Evaluation |
| Count | 1000 | 0.5346* | 0.3631* | 0.1518* | 0.63* | 1000 | 0.111* | 0.90* |
| Approx SumSim | 1000 | 0.4342* | 0.3068* | 0.1226* | 0.62* | 1000 | 0.084* | 0.90* |
| Approx MaxSim | 1000 | 0.7425* | 0.5384* | 0.2581* | 0.62* | 1000 | 0.196* | 0.89* |

| RQ2 - Approximate First Stage, Exact Second Stage |
| Kprime (\( k' = 20/100 \)) | 210 | 0.8488 | 0.6757 | 0.3104* | 0.38* | 901 | 0.341* | 0.85* |
| Count | 200 | 0.8365 | 0.6581 | 0.3345* | 0.55* | 1000 | 0.342* | 0.90* |
| Approx SumSim | 200 | 0.8372 | 0.6476 | 0.3199* | 0.52* | 1000 | 0.341* | 0.90* |
| Approx MaxSim | 200 | 0.8702 | 0.6842 | 0.3487 | 0.59* | 1000 | 0.342* | 0.89* |

Figure 3: Effectiveness as \( k \) varies. Hollow circles denote statistically significant differences in effectiveness compared to the default ColBERT end-to-end configuration (i.e. \( k' = 1000 \)), according to a paired t-test, \( p < 0.05 \), Bonferroni correction.

On analysing Figure 3, we observe little degradations in effectiveness until the number of retrieved documents is much smaller than the default (c.f. \( k \approx 7100 \)). Indeed, for NDCG@10 on the TREC 2019 query set, there are no marked degradations in effectiveness until \( k \leq 100 \); only if \( k = 10 \), \( 20 \) result in statistically significant decreases in NDCG@10 for Approx MaxSim, and \( k = 10, \ldots, 40 \) for Approx SumSim. For MAP, upto \( k = 800 \) there is a minor loss in effectiveness compared to the default ColBERT setting (horizontal dashed line), but no significant degradations until \( k = 300 \); Kprime exhibits significant decreases at \( k = 500 \). On the larger Dev query set, most queries have only a single relevant document, and hence effectiveness decreases more sharply for \( k < 100 \); significant degradations are observed for \( k \leq 1000 \).

Among the approximate ranking methods, the TREC 2019 query set favours the Approx MaxSim method for NDCG@10, while for Dev, Kprime appears marginally more effective, particularly around \( k = 100 \). Across both query sets, Approx MaxSim appears to be the most effective approximate ranking method, which in essence applies the ColBERT calculation in the earlier first stage.

We report \( k = 200 \) for TREC 2019, along with results using \( k = 1000 \) for Dev, in the bottom half of Table 1. We also report the effectiveness of the Kprime setting that results in \( k \) value nearest to 200. Indeed, on TREC 2019, while recall is reduced by 18% (0.77 → 0.59), there are no significant differences in MRR, NDCG@10 nor MAP using Approx MaxSim. On Dev, the drop in MRR@10 is statistically significant due to the large number of queries – we note that at \( k = 1500 \), there is no drop in effectiveness. Overall, for TREC 2019, \( k = 200 \) results in effective retrieval without significant differences in MAP or NDCG@10. Indeed, on our hardware, this resulted in a ~2× speedup in mean response time (406ms → 202ms).

Hence, in response to RQ2, we find that using the approximate ranking methods allows to control directly the number of retrieved documents, rather than indirectly via \( k' \). Applying the MaxSim operator on the approximate scores obtained from the FAISS ANN results in sufficient recall to ensure high effectiveness. For TREC 2019, using \( k = 200 \) with Approx MaxSim resulted in no significant differences in MAP, MRR or NDCG@10, and improved mean response time by a factor of 2.

5 CONCLUSIONS

ColBERT’s dense retrieval mechanism can be seen as a first-stage candidate set retrieval, followed by an exact scoring of all the candidates. In this work, we showed that an approximate ranking can be instantiated on the candidate set, and that this allows the size of the candidate set to be markedly reduced (from ~7100 documents to 200), without significantly impacting upon the effectiveness of the final ranking, providing a 2× speedup in efficiency.

In this work, we ignored the role of the differing query embeddings used by ColBERT - e.g. retrieving fewer documents for the less important masked query embeddings compared to other query embeddings. We leave this, and examining the impact of varying the exact ANN configuration, to future work.
ACKNOWLEDGEMENTS
Nicola Tonellotto was partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence). Craig Macdonald acknowledges EPSRC grant EP/R018634/1: Closed-Loop Data Science for Complex, Computationally- & Data-Intensive Analytics.

REFERENCES