Approximate Nearest Neighbors in External Memory

Anonymous ACL submission

Abstract

This paper introduces a new approach to Approximate Nearest Neighbors (ANN), designed for external memory. The proposed method uses standard ANN packages on a subset of the vectors which we call landmarks. If two vectors are near one another, then they tend to be near many of the same landmarks. This observation can be used to implement ANN with well-understood inverted file techniques.

1 Introduction

Popular packages for Approximate Nearest Neighbors (ANN) such as ANNOY[1] and FAISS[2] (Douze et al., 2024) work well for millions of vectors, but recent advances in LLMs are creating demand for vector databases with billions and billions of vectors. Most of these packages are designed to run in RAM, though some packages are designed for SSD (solid-state drives) (Jayaram Subramanya et al., 2019). This paper introduces a new approach to ANN, designed for external memory. Notation is summarized in Table 1.

We assume we have a vector database, P, that is too large to fit in physical memory. To scale these packages up by a few orders of magnitude, we create a random sample, $S \subset P$, where $P \in \mathbb{R}^{n \times d}$ and $S \in \mathbb{R}^{n_s \times d}$. n is the number of rows in the large database, P, and n_s is the number of rows in the more manageable sample, S. By construction, the sample, S, is designed to be small enough so we can afford to use off-the-shelf solutions such as ANNOY as FAISS, as well as future alternatives not yet invented.

Then, for each row, $p \in P$, we use the off-the-shelf ANN method to find k approximate nearest neighbors of p in S. We will refer to the set of k neighbors of p in S as a neighborhood, $neighborhood(p)$. Note that $|neighborhood(p)| = k$, by construction. Thus, we will refer to k as the size of the neighborhood. The rows of S will be referred to as *landmarks*.

The key observation is that neighborhoods can be used to approximate cosine similarity. That is, if two rows have a large cosine, then they are likely to share many of the same neighbors in S. We will refer to $|neighborhood(p_i) \cap neighborhood(p_j)|$ as *common ground*, $cg(p_i, p_j)$. Thus, the observation above can be expressed as Eqn (1):

$$cos(p_i, p_j) \sim cg(p_i, p_j) \tag{1}$$

We will refer to Eqn (1) as the *common ground approximation*. Section 3 will provide evidence for the common ground approximation, and Section 4 will show how postings can find common ground.

The common ground approximation is convenient because it makes it possible to use well-understood inverted files to do much of the work. In particular, it is well-known how to process inverted files in external memory, and how to compress inverted files. In fact, given well-known methods for compressing invert files (Witten et al., 1999; Yan et al., 2009; Pibiri and Venturini, 2019), it might be possible to store the inverted files in RAM.

2 Approximate Nearest Neighbors (ANN)

There is a large body of work on ANN because ANN has been recognized for a long time as an important practical problem (Indyk and Motwani, 1998), though recently, with the excitement over LLMs, there is even more interest in ANN than ever. ANN has many commercially important applications including chatbots, recommendation systems, product search and paid search (ads).

The ANN problem starts with a vector database, $P \in \mathbb{R}^{n \times d}$. At compile (training) time, the goal is to construct an index to support fast/accurate/cheap queries at inference time. At inference time, ANN

[1] https://pypi.org/project/annoy/1.0.3/
[2] https://pypi.org/project/faiss/

$P \in \mathbb{R}^{n \times d}$	A vector database of n vectors with d hidden dimensions
$q \in \mathbb{R}^d$	A query
$S \in \mathbb{R}^{n_s \times d}$	Landmarks: sample of vectors from P, where $n_s \ll n$; $S \subset P$
p	A vector (row of P); $0 \leq p < n$
l	A landmark (row of S); $0 \leq l < n_s$
$neighborhood(p)$	k nearest neighbors of p in S; each neighbor is a landmark, l
k	Max Common Ground; $\lvert neighborhood(p) \rvert = k$, for all p
$N \in \mathbb{Z}^{n \times k}$	The neighborhood matrix where each row is $neighborhood(p)$
$cg(p_i, p_j)$	Common Ground: $\lvert neighborhood(p_i) \cap neighborhood(p_j) \rvert$
$cos(p_i, p_j) \sim cg(p_i, p_j)$	Common Ground Approximation
$postings(l)$	Postings for landmark l: $\{p \mid l \in neighborhood(p)\}$

Table 1: Notation

takes queries $q \in \mathbb{R}^d$ and a vector database P as input, and retrieve k nearby neighbors of $q \in P$. There are a number of desiderata including:

1. loss (precision, recall, residuals): how close are the k neighbors to the best vectors in P
2. speed: latency, throughput
3. cost: hardware and power (carbon)
4. scale: n (# vectors) and d (# dimensions)

There have been competitions at NeurIPS-2021[3] and NeurIPS-2023[4] with an impressive set of industrial sponsors: Microsoft, Pinecone,[5] AWS and Zilliz,[6] indicating the commercial importance of ANN. The most recent competition consisted of four tracks:

1. filtered: a combination of vector search with traditional SQL-like constraints
2. OOD (Out-of-Domain): the database, P, and the queries, q, come from different distributions
3. sparse: the vectors have many dimensions, d, but relatively few non-zero values
4. streaming: in addition to search queries, there are also insertions and deletions.

The leaderboard for this competition emphasizes throughput (queries per second at 90% recall). This paper is concerned with a slightly different use case. We would like to scale up an API in Semantic Scholar (S2) (Wade, 2022) for recommending papers.[7] This API uses FAISS to recommend papers in Computer Science (CS), about 10% of S2. They would like to scale up this API to recommend papers across all fields, but they found that to be too expensive (personal communication).

3 Common Ground Approximation

The intuition behind the common ground approximation is: "You shall know a word by the company it keeps" (Firth, 1957). That is, it should be possible to infer p from its neighborhood, especially if k and n_s are large enough. That is, if I give you enough nearby landmarks, then you should be able to figure out where you are. Moreover, if I tell you that two vectors are near many of the same landmarks, then it is very likely that the two vectors are near one another.

Figure 1 provides some empirical evidence to support this intuition. The two boxplots were computed over a set of 6894 queries. The evaluation in Section 6 will compare the proposed method with the S2 API in footnote 7. Since the S2 API is limited to Computer Science papers, we chose a query set of SIGIR papers. Figure 1 shows boxplots summarizing search results. For each query, q, we used the proposed method to retrieve 19 candidates, c. For each of the 19 candidates, c, and each of the 6894 queries, q, we computed $cos(q, c)$ and $cg(q, c)$. The two boxplots in Figure 1 show that there is a strong relationship between cg (common ground) and cos over these 19×6894 observations.

The difference between the two plots is k (max common ground). In both cases, k is probably

[3] https://big-ann-benchmarks.com/neurips21.html

[4] https://big-ann-benchmarks.com/neurips23.html

[5] https://www.pinecone.io/

[6] https://zilliz.com/

[7] https://api.semanticscholar.org/recommendations/v1/papers/forpaper/21321bad706a9f9dbb502588b0bb393cf15fa052?from=all-cs&fields=title,externalIds,citationCount

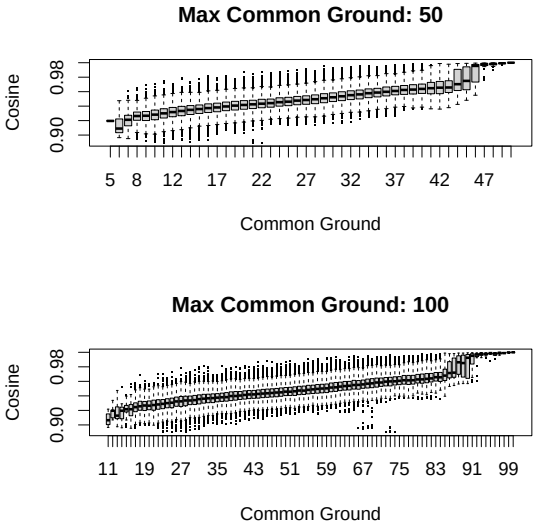

Figure 1: Evidence for the Common Ground Approximation

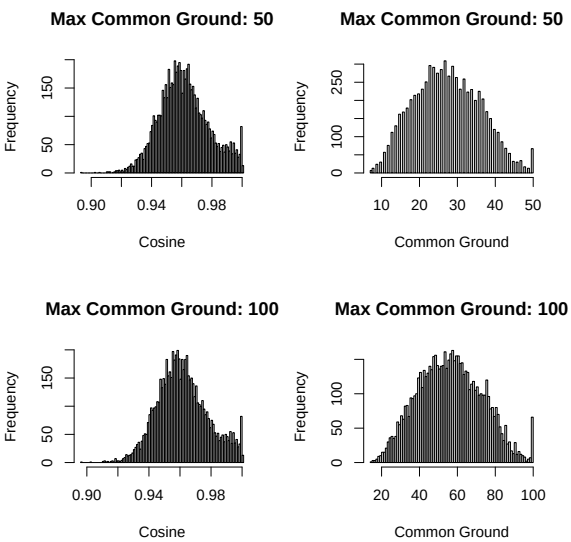

Figure 2: Histograms of Common Ground and Cosines

larger than it needs to be, at least for these vectors, a set of $n = 127,374,634$ with $d = 768$, computed by Semantic Scholar using their Specter2 model[8] (Cohan et al., 2020; Singh et al., 2022). Although the boxplots make it clear that the relationship between cg and cos is far from linear, the correlations are reasonable large, 0.69 for $k = 50$ and 0.70 for $k = 100$. The fact that the two correlations are so close to one another suggests that k is reaching a point of diminishing returns. If we wanted to improve the relationship between cg and cos, it may be more productive to increase the sample size, n_s, than to increase k. In this paper, we will use the same random sample, S, throughout, with $n_s = 1,000,000$. Thus, $n/n_s \approx 127$.

Figure 2 shows histograms of cosines and common ground for the data in Figure 1. These cosines and common ground values are relatively large since all of these pairs were in the top 19 search results. These values would be much lower, of course, if we used a different sample of pairs, such as random pairs.

4 Inverted Files & Common Ground

As mentioned above, we use an off-the-shelf ANN method such as FAISS to compute the neighborhood matrix, $N \in \mathbb{Z}^{n \times k}$, where each row is $neighborhood(p)$ for p between 0 and $n-1$, inclusively. The values of N are landmarks, l, where $0 \leq l < n_s$.

We can then compute $postings(l)$ from N for all landmarks, l, between 0 and n_s. $postings(l) = \{p | l \in neighborhood(p)\}$. If we stored these postings in the most obvious way without compression, the total space would be:

$$\sum_{l=0}^{n_s} |postings(l)| = nk \tag{2}$$

since the inverted files are simply a permutation of the values of N. Thus, the space complexity of the postings are at most nk, though there are obvious opportunities for compression.

On average, $|postings(l)| \approx n/n_s k$. In our case, $n/n_s \approx 127$, and $k = 50$ or $k = 100$.

5 Proposed Method

At query time, we are given a query q. We use the off-the-shelf ANN method to find $neighborhood(q)$, the k nearest landmarks to q. For each of these k landmarks, we fetch $postings(l)$ from external memory with k disk seeks. We then compute common grounds by counting duplicates in these postings.[9]

We can compute the common ground reasonably quickly. The k posting sets are $n/n_s k$ long on average, so their total length is $n/n_s k^2$. We could speed up this computation by reducing k, though, as a practical matter, the k disk seeks are more of a bottleneck than processing the postings.

Since the common ground approximation in Eqn 1 is not perfect, we find that we can improve

[8] https://huggingface.co/allenai/specter2

[9] There is a convenient python function, numpy.unique, that computes common ground when return_counts=True.

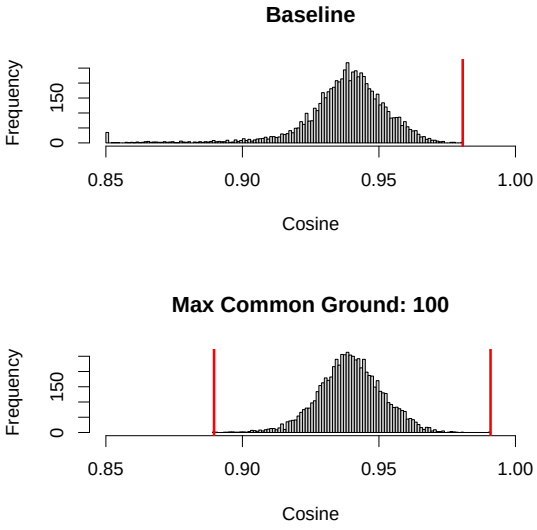

Figure 3: Histograms of 19th Largest Cosine

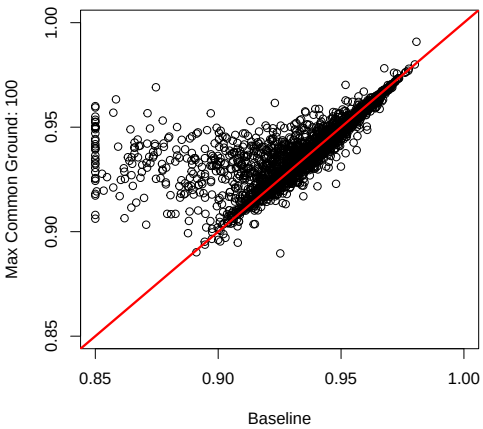

Figure 4: Scatter plot of 19th Largest Cosine

performance by using common ground counts to filter candidates to a short list, and then materialize cosines and rerank the short list. Reranking is the bottleneck in the computation since each cosine will cost a page fault. We can skip reranking when accuracy is less of a priority. In some use cases, such as ANN on a cell phone, we can afford the disk seeks since throughput is less of a priority. This step is also less necessary if we can afford to increase n_s.

6 Evaluation

As mentioned above, the challenge is to scale up the API in Semantic Scholar. It is currently limited to Computer Science papers since it is too expensive to store the 127 million vectors in FAISS. This section will compare the proposed method with the API in Semantic Scholar for the set of SIGIR queries discussed in Section 3.

The main advantage of the proposed system is that it runs in external memory, and therefore, we can afford to scale it up to 100% of the vectors in Semantic Scholar, unlike the baseline method, which is currently running on just 10% of the vectors. As for the quality of the candidates, it is sufficient to show that the quality of candidates from the proposed method are at least as good as candidates from the baseline.

To study quality, we ran both methods on the query set, requesting 19 candidates for each query. We then used the Specter vectors to compute $cos(q, c)$ for the 19 candidates. Figure 3 shows histograms for the smallest (19th largest) cosine for each query. For most queries, the cosines from the baseline system (top) are very close to the cosines from the proposed method (bottom), but the baseline system has a long tail of small values. Actually, if the baseline system produced a value worse than 0.85, we set the value to 0.85, in order to limit the range of the x-axis. The red bars show the largest and smallest values. Theere is no red line for the smallest value of the baseline system, since it is less than 0.85.

Figure 4 shows a scatter plot of the values in Figure 3. There is a point for each query in the test set. Most points fall near the red line (the main diagonal), indicating that the two methods return comparable results in most cases. However, there are a few points well above the red line, indicating that there are a few queries where the proposed method is finding much better candidates.

7 Conclusion

We proposed a simple method for reducing approximate nearest neighbors to well-understood inverted files. This work was primarily motivated by the recommender API in Semantic Scholar. Scaling this API up from 10% of their collection to 100% requires too much RAM. We showed that our proposed alternative retrieves vectors that are comparable to the S2 API in most cases, and sometimes much better. More importantly, by reduced ANN to sampling and inverted files, the proposed method is relatively easy to analyze for completely different architectures such as cell phones, where throughput is less of a priority than memory and power.

References

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613.

Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. 2019. DiskANN: Fast accurate billion-point nearest neighbor search on a single node. *Advances in Neural Information Processing Systems*, 32.

Giulio Ermanno Pibiri and Rossano Venturini. 2019. Techniques for inverted index compression. *ACM Computing Surveys (CSUR)*, 53:1 – 36.

Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. 2022. SciRepEval: A multi-format benchmark for scientific document representations. *ArXiv*, abs/2211.13308.

Alex D Wade. 2022. The semantic scholar academic graph (s2ag). *Companion Proceedings of the Web Conference 2022*.

I. Witten, Alistair Moffat, and T. Bell. 1999. Managing gigabytes (2nd ed.): compressing and indexing documents and images. Morgan Kaufmann.

Hao Yan, Shuai Ding, and Torsten Suel. 2009. Inverted index compression and query processing with optimized document ordering. In *Proceedings of the 18th international conference on World wide web*, pages 401–410.

8 Ethics

It is good for society to make the scientific literature more accessible. There is no sensitive data in this work. All of data we use is freely available through Semantic Scholar. On the other hand, there are many vector databases out there, and some of them could easily be used to do horrible things. The proposed method can be used to make vector databases more effective, for better and for worse.

9 Limitations

We did not discuss many issues such as updates. With the proposed method, it should be easy enough to add and delete vectors, but we have not implemented that. Another of the many issues left for future work is implementing more sophisticated methods for compressing inverted files. While we have implemented enough of the proposed method to create various plots, we have not attempted to build more than a research prototype.

10 Acknowledgements

Removed during review.