OUKA
Osaka University Knowledge Archive

Title	Approximate Top-k Inner Product Join with a Proximity Graph
Author(s)	Nakama, Hayato; Amagata, Daichi; Hara, Takahiro
Citation	Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021. 2021, p. 4468-4471
Version Type	AM
URL	https://hdl.handle.net/11094/92858
rights	© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Note	

Osaka University Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

Osaka University

Approximate Top-k Inner Product Join with a Proximity Graph

Hayato Nakama
Osaka University
Osaka, Japan
nakama.hayato@ist.osaka-u.ac.jp

Daichi Amagata
Osaka University, JST Presto
Osaka, Japan
amagata.daichi@ist.osaka-u.ac.jp

Takahiro Hara
Osaka University
Osaka, Japan
hara@ist.osaka-u.ac.jp

***Abstract*—This paper addresses the problem of top-k inner product join, which, given two sets of high-dimensional vectors and a result size k, outputs k pairs of vectors that have the largest inner product. This problem has important applications, such as recommendation, information extraction, and finding outlier correlation. Unfortunately, computing the exact answer incurs an expensive cost for large high-dimensional datasets. We therefore consider an approximate solution framework that efficiently retrieves k pairs of vectors with large inner products. To exploit this framework and obtain an accurate answer, we extend a state-of-the-art proximity graph for inner product search. We conduct experiments on real datasets, and the results show that our solution is faster and more accurate than baselines with state-of-the-art techniques.**

***Index Terms*—top-k inner product join, high-dimensional vector, approximation algorithm, proximity graph**

1. Introduction

This paper considers the top-k inner product join problem, which is defined as follows: Given two sets $\mathbf{X}$ and $\mathbf{Y}$ of high-dimensional vectors and a result size k, top-k inner product join between $\mathbf{X}$ and $\mathbf{Y}$ retrieves k pairs of vectors $\langle \mathbf{x}, \mathbf{y} \rangle$, where $\mathbf{x} \in \mathbf{X}$ and $\mathbf{y} \in \mathbf{Y}$, with the largest inner product among $\mathbf{X} \times \mathbf{Y}$. This problem has important applications, such as recommendation [1]–[3], information extraction [4], and finding outlier correlations [5]. More specifically, Figure 1 depicts a histogram of inner products of 1 million randomly sampled vector pairs in a Yahoo! dataset[1]. We see that most of them have small inner products, while some vectors have much larger inner products, namely outlier correlations [5]. Our problem is useful for finding such interesting pairs that behave abnormally in correlation analyses.

An intuitive approach for the top-k inner product join problem is to run a k-maximum inner product search (k-MIPS) algorithm for each vector in $\mathbf{Y}$ (or $\mathbf{X}$). Given a query vector $\mathbf{q}$ and k, the k-MIPS problem is to retrieve k vectors such that the inner products of them and $\mathbf{q}$ are the largest. Informally, after iterating k-MIPS on $\mathbf{X}$ by using the vectors in $\mathbf{Y}$ as queries, we can obtain the top-k inner product join result by merging the k-MIPS results. However, solving the k-MIPS problem exactly is computationally expensive, because it needs $O(n)$ or $O(m)$ time, where $n = |\mathbf{X}|$ and $m = |\mathbf{Y}|$, so the above exact solution for the top-k inner product join needs $O(nm)$ time in total. We hence consider an approximate solution to the top-k inner product join problem. Actually, approximate k-MIPS algorithms have also been studied extensively [6]–[13]. There are four main approaches, sampling [7], locality-sensitive hashing [6], [8], quantization [9], and proximity graphs [10]–[13]. Among these, proximity graphs have the best trade-off relationship between speed and accuracy [14]. We therefore consider a solution based on proximity graphs.

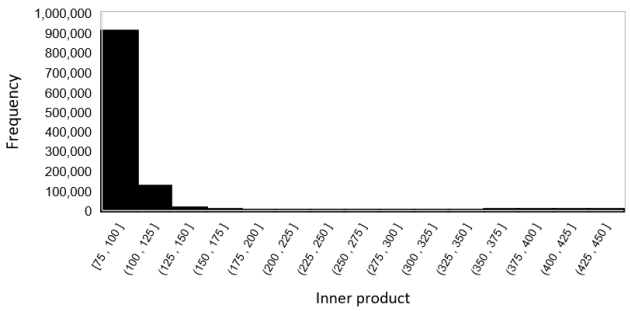


Figure 1. Example of outlier correlation. Most vector pairs have small inner products, whereas some vector pairs have much larger inner products.

Note that simply employing approximate k-MIPS algorithms still has some issues. First, simply iterating approximate k-MIPS is not efficient and is very slow in practice (see TABLE 2). Second, the state-of-the-art proximity graph [12] suffers from an overhead cost incurred by the need to traverse multiple graphs. To overcome theses issues, we make the following contributions:

(1) We propose a general framework for approximate top-k inner product join with a proximity graph. Our framework can employ any proximity graphs and does not necessarily run approximate k-MIPS iteratively. This framework avoids unnecessary inner product computations by considering norm-based access order. Therefore, those vector pairs that cannot be the top-k join result are pruned.

(2) We propose a new proximity graph for inner product

1. https://webscope.sandbox.yahoo.com

spaces. The state-of-the-art [12] considers vector norm and cosine similarity to quickly access vectors such that the inner products of them and a given query are large. (Because the inner product of $\mathbf{x}$ and $\mathbf{y}$ is obtained via $\|\mathbf{x}\|\|\mathbf{y}\|\cos\theta$, where θ is the angle between these vectors, norm and cosine similarity are important.) However, [12] considers norm and cosine similarity *separately*. Specifically, it builds two proximity graphs, one for norm and the other for cosine similarity. It therefore needs to traverse the two proximity graphs, incurring unnecessary computational costs. We take norm and cosine similarity into account *simultaneously* and propose Hybrid-ip-NSW, which, in a more effective manner, builds a *single* proximity graph that supports quick access to vectors which have large inner products with a given query.

(3) We conduct experiments on real datasets with multiple domains. Our experimental results demonstrate that (i) our framework provides much faster join processing than simply running k-MIPS iteratively and (ii) our proximity graph yields faster processing and a more accurate result than the state-of-the-art proximity graphs.

2. Preliminary

Problem definition. Let $\mathbf{x}$ be a d-dimensional real-valued vector in $\mathbb{R}^d$, and we assume that d is high. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, their inner product, $\mathbf{x}\cdot\mathbf{y}$, is obtained as follows: $\mathbf{x}\cdot\mathbf{y}=\sum_{i=1}^{d}\mathbf{x}[i]\times\mathbf{y}[i]$, where $\mathbf{x}[i]$ represents the i-th element of $\mathbf{x}$. Then, our problem is defined as follows:

DEFINITION 1 (TOP-K INNER PRODUCT JOIN). *Given two sets $\mathbf{X}$ and $\mathbf{Y}$ of vectors and a result size k, this problem finds k vector pairs such that their inner products are the largest among all pairs in $\mathbf{X}\times\mathbf{Y}$.*

For vector sets with a large d, this problem incurs $O(nm)$ time (for a fixed d), where $n=|\mathbf{X}|$ and $m=|\mathbf{Y}|$. This paper hence proposes an approximate solution.

Related work. Inner product join has been receiving attention and has important applications, as introduced in Section 1. LEMP is the state-of-the-art exact inner product join algorithm. LEMP partitions the vector sets so that it can select an optimal search algorithm (a cosine similarity search or a linear scan) for each partition. Approximate inner product join has also been considered in [7] and the theory field [5], [15], [16]. However, [7] assumes non-negative vectors, losing generality. Also, [5], [15], [16] do not consider top-k joins and do assume binary vectors, whereas we assume dense real-valued vectors.

In addition to the above algorithms, k-MIPS algorithms can be employed in our problem. Our proposed framework employs a proximity graph, which yields state-of-the-art k-MIPS performance, as a function. Furthermore, we extend ip-NSW+ [12], a state-of-the-art proximity graph for the inner product search problem, to exploit our framework.

3. Our Solution

Main idea. Our idea for solving the top-k inner product join efficiently and accurately is to exploit Cauchy–Schwarz inequality and a proximity graph. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, we have

$$\mathbf{x}\cdot\mathbf{y}\le\|\mathbf{x}\|\|\mathbf{y}\|,$$

where $\|\mathbf{x}\|$ is the Euclidean norm of $\mathbf{x}$. Assume that the vectors in $\mathbf{X}$ and $\mathbf{Y}$ are sorted in descending order of norm: $\|\mathbf{x}_1\|\ge\cdots\ge\|\mathbf{x}_n\|$ and $\|\mathbf{y}_1\|\ge\cdots\ge\|\mathbf{y}_m\|$ without loss of generality. Assume furthermore that we have a threshold τ for the top-k inner product join. If $\|\mathbf{x}_1\|\|\mathbf{y}_i\|<\tau$, $\mathbf{y}_i$ never has pairs that can be the top-k result, from Cauchy–Schwarz inequality. That is, we need to run a k-MIPS only for vectors $\mathbf{y}_j$ such that $\|\mathbf{x}_1\|\|\mathbf{y}_j\|\ge\tau$.

Algorithm 1: FRAMEWORK FOR APPROXIMATE TOP-K INNER PRODUCT JOIN

```
   Input: X (proximity graph), Y, and k
   Output: R (a set of k vector pairs)
1  /* Threshold initialization */
2  τ ← 0, R ← ∅
3  for each y_i ∈ Y where i ∈ [1, k] do
4      q ← y_i, Q ← {x_1}
5      while Q ≠ ∅ do
6          x ← Q[0]
7          Pop Q[0] from Q
8          if This is the first time for q to visit x then
9              γ ← x · q
10             if γ > τ then
11                 Update R and τ
12                 for each x' ∈ x.E do
13                     Q ← Q ∪ {x'}
14 /* Result verification */
15 for each y_j ∈ Y where j ∈ [k + 1, m] do
16     if ||y_j|| ||x_1|| ≤ τ then
17         return R
18     else
19         Execute lines 4–13
```

3.1. Framework

Assume that vectors in $\mathbf{X}$ and $\mathbf{Y}$ are sorted offline, as mentioned above. Our proposed framework assumes that a proximity graph is (or proximity graphs are) specified as an input, and we assume that $\mathbf{X}$ is indexed by a proximity graph. (If $|\mathbf{X}|<|\mathbf{Y}|$, $\mathbf{Y}$ is indexed by a proximity graph.) In a nutshell, a proximity graph of $\mathbf{X}$ consists of nodes and edges, where nodes are the vectors $\mathbf{x}\in\mathbf{X}$ and an edge is created between nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ if they satisfy the requirements of a given graph building algorithm. (In inner product spaces, such requirements are usually high inner products or cosine similarities.) We use $\mathbf{x}.E$ to denote the set of edges held by a node (i.e., vector) $\mathbf{x}$. Note that the graph building is done offline and only once.

Algorithm 1 describes our framework. Lines 2–13 initialize the threshold τ and result set (k vector pairs) R. To

get a tight threshold for pruning many unnecessary vector pairs, we use the observation that vectors with large norms tend to have large inner products [1]. For each $\mathbf{y}_i$ where $i \in [1, k]$, we run a breadth-first-search in the proximity graph of $\mathbf{X}$ from $\mathbf{x}_1$, by setting $\mathbf{y}_i$ as a query. By using the accessed vector pairs in the breadth-first-searches, we initialize τ and R. After this initialization, in lines 15–19, we update R and τ by using the remaining vectors $\mathbf{y}_j \in \mathbf{Y}$, where $j \in [k+1, m]$. Note that this is done iff we have $\|\mathbf{y}_j\|\|\mathbf{x}_1\| > \tau$. This is because, if $\|\mathbf{y}_j\|\|\mathbf{x}_1\| \leq \tau$, all $\mathbf{y}_{j'}$ such that $j' \geq j$ cannot have vector pairs with inner products $> \tau$ from Cauchy–Schwarz inequality. For vectors $\mathbf{y}_j$ such that $\|\mathbf{y}_j\|\|\mathbf{x}_1\| > \tau$, we do the same graph traversal as in the initialization to update the result.

3.2. Hybrid-ip-NSW

The state-of-the-art proximity graph for inner product search, ip-NSW+ [12], considers norm and cosine similarity separately, thereby it needs to traverse the two proximity graphs. (Considering only inner products [17] has been shown not to be effective in [12].) It is intuitively seen that a single graph, which considers both norm and cosine similarity simultaneously, is a better approach. We hence design such a proximity graph.

Recall that vectors are sorted by the norm order. We *incrementally* build a proximity graph based on this order to take into account norm size. (In other words, at initialization the graph has no nodes, and in each iteration a node is added to the intermediate graph.) Given a vector $\mathbf{x}_i \in \mathbf{X}$, in an intermediate proximity graph, we use the greedy algorithm [17] to find $\epsilon \times \mu$ nodes (vectors) that have (approximately) the largest inner product with $\mathbf{x}_i$. Note that $\epsilon \geq 1$ and $\mu \geq 1$, which controls the degree, are hyper-parameters to balance speed and accuracy. We compute nodes with the μ largest cosine similarity with $\mathbf{x}_i$, among the $\epsilon \times \mu$ nodes. Then, we create edges between $\mathbf{x}_i$ and the μ nodes. We do the above operations for each $\mathbf{x}_i \in \mathbf{X}$.

We call the graph built by this algorithm *Hybrid-ip-NSW*. Although a Hybrid-ip-NSW can be built by a simple algorithm[2], it provides a better performance than the state-of-the-art, which is shown in the next section.

4. Experiment

This section reports our experimental results. All experiments were conducted on a machine with 2.5GHz Intel Core i9-9900 CPU and 8GB RAM.

Algorithms. We evaluated the following algorithms.

- *LEMP* [4]: A state-of-the-art exact algorithm.
- *IPJ with ip-NSW*: This algorithm incorporates ip-NSW [17] into our framework proposed in Section 3.1.
- *IPJ with ip-NSW+*: This algorithm incorporates ip-NSW+ [12] into our framework.
- *IPJ with Hybrid-ip-NSW*: This algorithm incorporates Hybrid-ip-NSW into our framework.
- *Iterative Hybrid-ip-NSW*: This algorithm employs Hybrid-ip-NSW and iteratively runs approximate k-MIPS.

All algorithms were single threaded, implemented by C++, and compiled by using g++ 7.4.0 with -O3 flag.

TABLE 1. DATASET STATISTICS

Dataset	$\|\mathbf{X}\|$	$\|\mathbf{Y}\|$	d
Deep1M	1,000,000	10,000	256
ImageNet	2,340,373	80,000	150
WordVector	1,000,000	10,000	300
Yahoo! Music	136,736	100,900	300

TABLE 2. RESULTS OF LEMP AND ITERATIVE HYBRID-IP-NSW ($k = 1000$)

	LEMP	Iterative Hybrid-ip-NSW	
Dataset	Time [sec]	Time [sec]	Recall
Deep1M	43.50	240.55	0.72
ImageNet	126.35	409.30	0.75
WordVector	40.10	267.38	0.81
Yahoo! Music	28.00	41.12	0.88

Datasets. We used the real datasets[3] listed in TABLE 1.

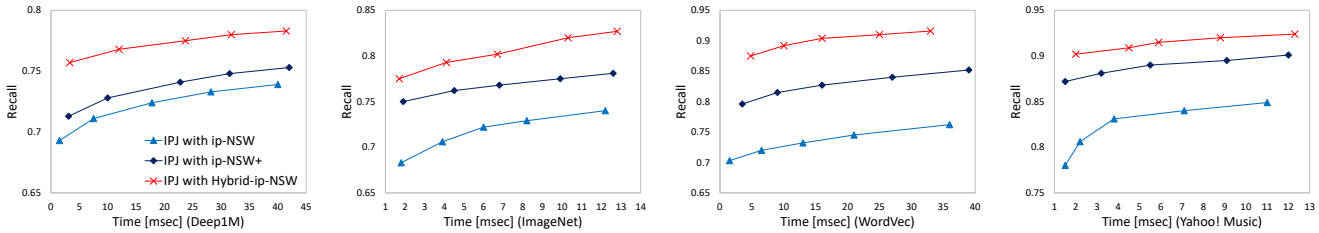
Result. We evaluated the running time and accuracy (recall) of each algorithm. We set $\mu = 50$ and $k = 1000$. TABLE 2 shows the results of LEMP and Iterative Hybrid-ip-NSW, while Figure 3 shows the time-recall curves of IPJ with ip-NSW, IPJ with ip-NSW+, and IPJ with Hybrid-ip-NSW. By comparing the results shown in TABLE 2 with the ones in Figure 3, we see that LEMP and Iterative Hybrid-ip-NSW are much slower than the algorithms that use our framework. (LEMP and Iterative Hybrid-ip-NSW are second-scale, whereas the algorithms that use our framework are microsecond-scale.) This observation suggests that (i) the exact solution (LEMP) is not efficient and (ii) our framework provides much faster processing than the iterative approach.

Now we focus on Figure 3, i.e., IPJ with ip-NSW, IPJ with ip-NSW+, and IPJ with Hybrid-ip-NSW. (The time-recall curves were obtained by tuning ϵ). Notice that the performance differences are directly derived from the effectiveness of the proximity graphs (ip-NSW, ip-NSW+, and Hybrid-ip-NSW), since they use the same framework. Then, it is clear that Hybrid-ip-NSW has the best trade-off relationship between time and recall. (Shorter time with higher recall is better.) As confirmed in [12], we also confirmed that simply considering inner products in a proximity graph (ip-NSW) provides a lower performance than those of the others. In addition, our experimental results demonstrate that Hybrid-ip-NSW, which considers norm and cosine similarity simultaneously, is better than ip-NSW+, which considers norm and cosine similarity separately.

For a different value of k (e.g., $k = 5000$), their performance tendencies are similar and Hybrid-ip-NSW keeps

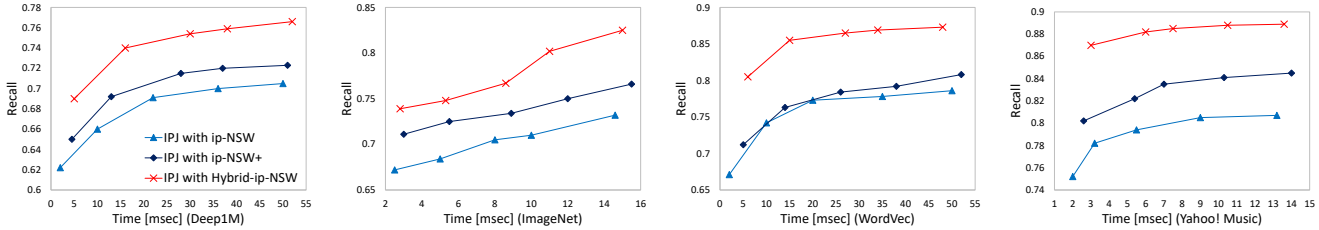
2. This graph building approach follows [17], but how to connect nodes is totally different.

3. http://www.cse.cuhk.edu.hk/systems/hash/gqr/datasets.html



(a) Deep1M (b) ImageNet (c) WordVector (d) Yahoo! Music

Figure 2. Time-recall curve ($k = 1000$)



(a) Deep1M (b) ImageNet (c) WordVector (d) Yahoo! Music

Figure 3. Time-recall curve ($k = 5000$)

outperforming ip-NSW and ip-NSW+, as shown in Figure 3. (The results of LEMP and Iterative Hybrid-ip-NSW are omitted, as they are much slower than the others.)

5. Conclusion

This paper addressed the problem of top-k inner product join and proposed an approximate solution. Specifically, we proposed a general framework for approximate algorithms that employ proximity graphs. To exploit this framework, we extended a state-of-the-art proximity graph. We conducted experiments by using real datasets, and the results show that our framework and proximity graph have the best time-accuracy trade-off.

Acknowledgments. This research is partially supported by JST PRESTO Grant Number JPMJPR1931, JSPS Grant-in-Aid for Scientific Research (A) Grant Number 18H04095, and JST CREST Grant Number JPMJCR21F2.

References

[1] D. Amagata and T. Hara, "Reverse maximum inner product search: How to efficiently find users who would like to buy my item?" in *RecSys*, 2021, pp. 273–281.

[2] H. Wang, D. Amagata, T. Maekawa, T. Hara, H. Niu, K. Yonekawa, and M. Kurokawa, "Preliminary investigation of alleviating user cold-start problem in e-commerce with deep cross-domain recommender system," in *ECNLP*, 2019, pp. 398–403.

[3] H. Wang, D. Amagata, T. Makeawa, T. Hara, N. Hao, K. Yonekawa, and M. Kurokawa, "A dnn-based cross-domain recommender system for alleviating cold-start problem in e-commerce," *IEEE Open Journal of the Industrial Electronics Society*, vol. 1, pp. 194–206, 2020.

[4] C. Teflioudi, R. Gemulla, and O. Mykytiuk, "Lemp: Fast retrieval of large entries in a matrix product," in *SIGMOD*, 2015, pp. 107–122.

[5] M. Karppa, P. Kaski, and J. Kohonen, "A faster subquadratic algorithm for finding outlier correlations," *ACM Transactions on Algorithms*, vol. 14, no. 3, pp. 1–26, 2018.

[6] A. Shrivastava and P. Li, "Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips)," *NIPS*, pp. 2321–2329, 2014.

[7] G. Ballard, T. G. Kolda, A. Pinar, and C. Seshadhri, "Diamond sampling for approximate maximum all-pairs dot-product (mad) search," in *ICDM*, 2015, pp. 11–20.

[8] Q. Huang, G. Ma, J. Feng, Q. Fang, and A. K. Tung, "Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search," in *SIGKDD*, 2018, pp. 1561–1570.

[9] X. Dai, X. Yan, K. K. Ng, J. Liu, and J. Cheng, "Norm-explicit quantization: Improving vector quantization for maximum inner product search," in *AAAI*, 2020, pp. 51–58.

[10] H.-F. Yu, C.-J. Hsieh, Q. Lei, and I. S. Dhillon, "A greedy approach for budgeted maximum inner product search," in *NIPS*, 2017, pp. 5459–5468.

[11] Z. Zhou, S. Tan, Z. Xu, and P. Li, "Möbius transformation for fast inner product search on graph," in *NeurIPS*, 2019, pp. 8218–8229.

[12] J. Liu, X. Yan, X. Dai, Z. Li, J. Cheng, and M.-C. Yang, "Understanding and improving proximity graph based maximum inner product search," in *AAAI*, 2020, pp. 139–146.

[13] S. Morozov and A. Babenko, "Non-metric similarity graphs for maximum inner product search," *NeurIPS*, vol. 31, pp. 4721–4730, 2018.

[14] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement," *IEEE Transactions on Knowledge & Data Engineering*, vol. 32, no. 08, pp. 1475–1488, 2020.

[15] L. Chen, "On the hardness of approximate and exact (bichromatic) maximum inner product," *Theory OF Computing*, vol. 16, no. 4, pp. 1–50, 2020.

[16] T. D. Ahle, R. Pagh, I. Razenshteyn, and F. Silvestri, "On the complexity of inner product similarity join," in *PODS*, 2016, pp. 151–164.

[17] X. Yan, J. Li, X. Dai, H. Chen, and J. Cheng, "Norm-ranging lsh for maximum inner product search," *NIPS*, pp. 2952–2961, 2018.