Norm Adjusted Proximity Graph for Fast Inner Product Retrieval

Shulong Tan, Zhaozhuo Xu, Weijie Zhao, Hongliang Fei, Zhixin Zhou, Ping Li
Cogntivie Computing Lab
Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA
{tanshulong2011, zhaozhuoxu, zhaoweijie12, feihongliang0, zhixin0825, pingli98}gmail.com

ABSTRACT

Efficient inner product search on embedding vectors is often the vital stage for online ranking services, such as recommendation and information retrieval. Recommendation algorithms, e.g., matrix factorization, typically produce latent vectors to represent users or items. The recommendation services are conducted by retrieving the most relevant item vectors given the user vector, where the relevance is often defined by inner product. Therefore, developing efficient recommender systems often requires solving the so-called *maximum inner product search* (MIPS) problem. In the past decade, there have been many studies on efficient MIPS algorithms. This task is challenging in part because the inner product does not follow the triangle inequality of metric space.

Compared with hash-based or quantization-based MIPS solutions, in recent years graph-based MIPS algorithms have demonstrated their strong empirical advantages in many real-world MIPS tasks. In this paper, we propose a new index graph construction method named *norm adjusted proximity graph* (NAPG), for efficient MIPS. With adjusting factors estimated on sampled data, NAPG is able to select more meaningful data points to connect with when constructing graph-based index for inner product search. Our extensive experiments on a variety of datasets verify that the improved graph-based index strategy provides another strong addition to the pool of efficient MIPS algorithms.

CCS CONCEPTS

• **Information systems** → Information retrieval; Retrieval tasks and goals; Retrieval models and ranking.

KEYWORDS

Recommendation, maximum inner product search, approximate near neighbor search, search on graph

ACM Reference Format:
Shulong Tan, Zhaozhuo Xu, Weijie Zhao, Hongliang Fei, Zhixin Zhou, Ping Li. 2021. Norm Adjusted Proximity Graph for Fast Inner Product Retrieval. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467412

1 INTRODUCTION

With the popularity of representation learning techniques, retrieval of similar vectors in the embedding space becomes a popular operation in various applications [19, 51, 52]. For example, collaborative filtering (CF) [20, 21, 28, 39, 52] projects users and items into a shared latent space, where a user or an item is represented as a vector of latent features. Thereafter a user's previous rating on an item is modeled as a matching function of their latent vectors. In online recommendation services, given a user vector, the system retrieves and returns item vectors with maximum matching scores. Similarly, for the question answering (QA) problem [2], questions, and documents (containing answers) can be embedded into a common semantic space. For each query question, the system will efficiently return answer candidates based on vector comparisons in the embedding space. And then a fine trained ranking model will be used to further score these candidates. Recently, researchers found that with transformer based models and finely designed pre-training, this vector retrieval based approach is even better than the classical term-based BM25, for the answer candidate generation [2, 8, 9].

Inner product is a common and effective searching/ranking measure for similar vector retrieval, such as in recommendation models [5, 30, 31, 51, 52] and retrieval-based question answering [2, 8, 9, 40]. Another popular measure–cosine similarity–can be considered as a special case of inner product on normalized vectors. Retrieval by inner product is often referred as maximum inner product search (MIPS) in literature, which also has extensive applications in multi-class or multi-label classification [10, 25, 45, 50, 55], natural language processing [7, 15, 32], computational advertising for search engines [11], etc. Since the retrieval space is usually very large, say millions or billions, linearly brute force ranking is too slow for online services. Therefore, accelerating the vector retrieval speed would obviously benefit models' application in real-world systems [10, 11, 18]. In practice, we pursue approximate but sub-linear complexity ranking approaches with a proper trade-off between searching efficiency and effectiveness. We note this efficient inner product ranking task as the approximate MIPS problem.

Approximate MIPS is considered as a challenging problem since the inner product does not satisfy the triangle inequality. Traditional *approximate near neighbor (ANN) search* [12, 13, 16, 24, 34, 35, 41, 56] methods were typically designed for metric measures, such as ℓ_2-distance or cosine similarity, thus their efficiency suffers when we directly extend them to MIPS. The early work of [38] proposed tree-based methods to solve the approximate MIPS problem. More recently, a line of works proposed to transform MIPS to the traditional ANN search, by lifting the base data vectors and query vectors asymmetrically to higher dimensions [4, 37, 42–44, 53]. After the transformation, the well-developed ANN search methods can then be applied to solve the MIPS problem. There are other proposals designed for the MIPS task including quantization-based methods [17], Greedy-MIPS [54], and graph-based methods [36, 46, 57].

In particular, ip-NSW [36] tries to solve MIPS by *hierarchical navigable small world (HNSW)* [35], which is the current popular ANN search algorithm. Compared to HNSW, ip-NSW simply replaces the metric measures with inner product. Generally, the framework of graph-based ANN search algorithm [34, 35, 56] can be summarized as two steps: 1) build a proximity graph where each vertex represents a base data vector. Each data vector will connect with a few of its neighbors. 2) given a query vector, perform a greedy search on the proximity graph by comparing the query vector with base data vectors under the searching measures (e.g., cosine similarity or inner product). Then, the most similar candidates are returned as outputs. The key point for these two-step methods is to construct a high-quality index graph, which provides a proper balance between the searching efficiency and effectiveness. To guarantee the searching efficiency, the degree of each vertex is usually restricted to a small number, such as 16. At the same time, we hope the diversity of its neighbors is high, which will ensure the searching effectiveness.

For the graph-based MIPS method ip-NSW, although experiments show that it achieves significant improvement compared to previous MIPS methods, we find that the algorithm of it, which is based on HNSW, is problematic. The key component of HNSW, the edge selection algorithm, which ensures the neighborhood diversity, is specially designed for metric spaces. The rough replacement, from metric measures to inner product, will make the edge selection method fail and lead to inefficient proximity graphs. To fix the issue, [46] proposes an edge selection algorithm specifically for inner product. However, the proposed method IPDG typically works advantageously on top-1 recalls.

In this paper, we design a new proximity graph based MIPS method, Norm Adjusted Proximity Graph (NAPG). We propose to adjust the ℓ_2-norms of data points when performing edge selection to make it work for inner product. The motivation of the proposed method is based on the long tail ℓ_2-norm distributions in real datasets. We observe that most of base vectors have much smaller ℓ_2-norms, comparing those large norm data points, as examples shown in Figure 1. When we perform edge selection, such as by the way of ip-NSW, the data vector x in a small ℓ_2-norm, with its neighbor candidates $\{p_i\}$, can only connect with one vertex p which has the largest inner product with it. Other vertices that also have large inner products with it will not be connected. Since the inner product values between x and $\{p_i\}$ are totally not comparable with

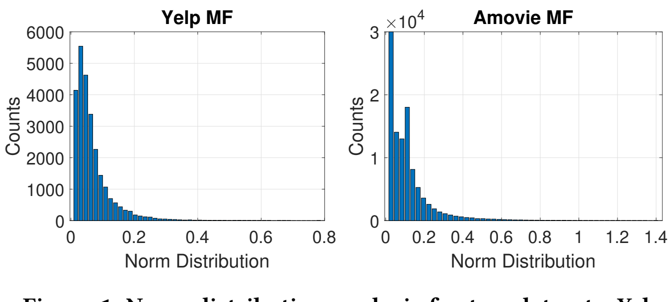

Figure 1: Norm distribution analysis for two datasets: Yelp MF and Amovie MF. In these cases, most of the data points have much smaller norms than large norm data points.

those among $\{p_i\}$. This leads to inefficient graph structures and seriously affects the MIPS performance. By adjusting the ℓ_2-norms of data points in small ℓ_2-norms with a carefully selected factor α, we re-scale the data vector to make $\alpha x^\top p_i$ and $p_i^\top p_j$ have the same expectation. In this way, the edge selection method works again for inner product and more meaningful (and diverse) edges can be constructed. The final index graph has an improved connectivity and leads to performance improvements. The key adjusting factor α is estimated based on small sampled data. Further, we refine the factor into a finer granularity for each norm sub-ranges.

In experiments, we evaluate the proposed method, NAPG, on 6 public datasets. These datasets are in various sizes and extracted by different embedding models. As compared with competitive baselines, NAPG shows significant superiority across different testing cases. Our contributions can be summarized as follows:

- We introduce the norm-adjusted factor for MIPS. This factor bridges the gap between metric measures and inner product that enables an effective adaption for graph-based MIPS methods.
- We propose an estimation method for the global adjusting factor, and refine it into a finer granularity for each norm sub-ranges.
- We perform extensive experiments on 6 real-world datasets to evaluate our proposed method. Results demonstrate its effective and efficient performance for the approximate MIPS problem.

2 THE MIPS PROBLEM: BACKGROUND AND EXISTING SOLUTIONS

Formally, in the space $X \subset \mathbb{R}^d$, given a query/user latent vector set $S_u = \{q_1, \dots, q_n\}$ and an base/item latent vector set $S_i = \{x_1, \dots, x_m\}$, the ranking model defines a continuous symmetric similarity function, $f : X \times X \rightarrow \mathbb{R}$. And the goal of optimal similar vector retrieval is to find

$$p = \arg\max_{x \in S_i} f(x, q), \qquad \text{for } q \in S_u. \tag{1}$$

Here in our paper, we discuss the common case for the vector retrieval by inner product, i.e., $f(x, q) = x^\top q$. The problem in Eq. (1) with respect to inner product is often referred as **maximum inner product search** (**MIPS**), which is closely related with and distinctly different from conventional near neighbor search tasks. This is because the inner product does not satisfy the triangle inequality, unlike the ℓ_2-distance or the cosine similarity.

To see this, we can re-formulate Eq. (1) as a negative ℓ_2-distance problem or a cosine similarity problem:

$$p = \arg\max_{x \in S_i} -\|q - x\|^2 = \arg\max_{x \in S_i} (2x^\top q - \|x\|^2), \tag{2}$$

$$p = \arg\max_{x \in S_i} \frac{x^\top q}{\|x\| \|q\|} = \arg\max_{x \in S_i} \frac{x^\top q}{\|x\|}. \tag{3}$$

Note that the ℓ_2-norm of query q does not affect the ranking.

The search problems defined in (2) and (3) are equivalent to MIPS when all ℓ_2-norms of $x \in S_i$ is a fixed constant. However, the ℓ_2-norms of data points could be significantly different from each other in practice, such as examples in Figure 1, which makes MIPS a distinguishing task.

The brute-force approach to solving MIPS is computationally expensive. Some researchers try to speed up this task by novel indexes and pruning techniques. There are two categories among these techniques: 1) fast **exact MIPS**; and 2) fast **approximate MIPS**, which we will elaborate in details in the next subsections.

In practice, we aim to retrieve not only the most relevant vector, but also top-k ones. Meanwhile, we aim at retrieving these vectors as soon as possible due to the latency restriction. Thus, it becomes:

Let $T_k \subset S_i$ be a set that

$$|T_k| = k, \quad \forall x \in T_k, \quad \forall x' \in S_i \setminus T_k$$
$$f(x,q) \geq f(x',q)$$

In time t, we aim to find k vector candidates C_k that

$$maximize \quad |C_k \cap T_k|, subject\ to \quad C_k \subset S_i.$$

T_k represents the top-k suitable vectors. C_k is the vector candidates that a method returned within the limited time t.

2.1 Existing Solutions for Exact MIPS

Efficient exact MIPS mainly utilize matrix computations and pruning techniques [1, 33, 48, 49]. LEMP transforms the MIPS problem into a number of smaller cosine similarity search problems. This transformation allows LEMP to prune large parts of the search space immediately and to select suitable search algorithms for each of the remaining problems individually [48, 49]. [33] proposes FEXIPRO based on sequential scan. FEXIPRO applies an SVD transformation to the base vector matrix, after which the first several dimensions capture a large percentage of the inner products. This will prune base vectors by only computing their partial inner products with the query vector. [1] shows that the fastest of today's indexes do not always outperform blocked matrix multiply, and thus proposes MAXIMUS, a simple indexing scheme that leverages linear algebra kernels to gain hardware efficiency while also pruning computation. In addition, [1] designs OPTIMUS, a system that can efficiently choose between using an index and blocked matrix multiply. These exact MIPS methods, even though they are much more efficient than the brute-force scan, are nevertheless only suitable for the moderate data size and data dimension.

2.2 Existing Solutions for Approximate MIPS

As the inner product is a non-metric similarity measure that does not comply with the triangle inequality, it makes many well-studied approximate near neighbor (ANN) search techniques ineffective or inefficient. To overcome the variation in ℓ_2-norms, previous works reduced MIPS to an ANN search problem in metric space by pre-processing the base data and queries asymmetrically [4, 37, 42–44]. Recently, researchers found that the method above has limitations in performance due to the long tail distribution of data norms [23, 53]. Based on these observations, [53] designed a new approach, **Range-LSH**. The dataset is first partitioned into small subsets according to the ℓ_2-norm rank and then they apply the transformations of [4, 37, 42–44] in each sub-dataset by a local maximum ℓ_2-norm. Similarly [23] also partitions data by the norm distribution, by proposing an alternative query transformation to avoid the distortion error. The motivation behind our method is also the observation of the long tail distribution in data norms, but we solve it by different methodology, the graph-based index.

Non-reducing MIPS methods have also been proposed and evaluated in MIPS benchmarks. [38] proposed a tree-based approach for MIPS. The authors do not construct a special tree for MIPS, instead, they introduced a new bound for the inner product in the searching phases. This bound determines the conditions whether the sub-trees can be ignored, for speeding up the searching process. Recently, randomized partition tree is also exploited for the MIPS problem [27]. [17] proposes an MIPS method based on product quantization (PQ) and extends the application of the PQ technique. [54] uses an upper bound of the inner product as the approximation of MIPS and designed a greedy search algorithm to find this approximation, called **Greedy-MIPS**. Similarly, [26] provides improved bounds on the inner product under the random projection that matches bounds on the Euclidean distance.

As graph-based indexing approaches have demonstrated its superior efficiency and effectiveness in approximate near neighbor (ANN) search with cosine similarity or ℓ_2-distance [14, 34, 35], the idea of searching by proximity graph has been extended to other similarity measures, such as inner product. [57] and [47] use ℓ_2-graphs to solve the MIPS problem approximately. [36] provides the theoretical basis for conducting MIPS by proximity graph: if the constructed graph contains the Delaunay graph with respect to the inner product, then the greedy search on the graph gives the exact true answer for MIPS. In the implementation, the proposed **ip-NSW** utilizes the algorithm in [35] to construct a Hierarchical Navigable Small World (HNSW) graph for indexing. To adapt the MIPS problem, ip-NSW replaces metric measures in HNSW by the inner product. As mentioned above, it is problematic by simply replacing the ranking measure. To construct better Delaunay graphs with respect to the inner product, [46] proposes a new edge selection algorithm, IPDG. IPDG works surprisingly well on top-1 recall but it is not suitable for top-k (say $k = 10$ or $k = 100$) scenarios.

3 PROXIMITY GRAPH BASED SEARCHING

In this section, we will explain why HNSW and ip-NSW is problematic for solving MIPS and motivate the proposed method after.

3.1 Search on Approximate Delaunay Graph

HNSW has two main components: 1) proximity graph construction (Algorithm 1) and 2) greedy search on graph (Algorithm 2). The graph of HNSW is built in a greedy manner. As represented in Algorithm 1, for each new coming in data point x, it first retrieves a neighbor candidate set $\{p_i\}$ by the greedy search algorithm on the current graph. Then x will be inserted into the graph by connecting to its M finely selected neighbors. Note that the graph is initialized as an empty graph. The first data point is inserted as an isolated node. Then we insert the second node and construct the first edge, and so on. After the graph construction, given a query point, Algorithm 2 performs a greedy search on graph and returns the items which maximize the similarity measure f.

As pointed out in [3], in order to find the exact nearest neighbor by greedy search, the graph must contain *Delaunay graph* as a subgraph. This statement can extend to the MIPS problem [36]. Although the Delaunay Graph has demonstrated its potentials in similarity search, the direct construction of the Delaunay graph in large scale and high dimensional datasets is unfeasible, due to the exponentially growing number of edges. To remedy this issue, practical algorithms usually approximate Delaunay Graphs, such as shown in Algorithm 1 (HNSW/ip-NSW).

Algorithm 1 HNSW/ip-NSW Graph Construction

```
1:  input: Latent vector set $S$, the maximum vertex degree $M$, the
    priority queue size $k$ for searching neighbors and the similarity
    measure $f$.
2:  Initialize graph $G \leftarrow \emptyset$
3:  for each $x$ in $S$ do
4:      Greedy search $k$ vertices $\{p_i\}$ on $G$ by $SoG(x, G, k, f)$ that
        have largest values with $x$ in $f(x, p_i)$, place them in descending
        order.
5:      $C \leftarrow \emptyset$.
6:      for i $\leftarrow$ 1 to $k$ do
7:          flag $\leftarrow$ True
8:          for all $p$ in $C$ do
9:              if $f(x, p_i) < f(p, p_i)$ then
10:                 flag $\leftarrow$ False
11:                 break
12:         if flag = True then
13:             $C \leftarrow C \cup \{p_i\}$
14:             add edge $\{x, p_i\}$ to $G$
15:         if $|C| = M$ then
16:             break
17: output: index graph $G$
```

Algorithm 2 Greedy Search on Graph $SoG(q, G, k, f)$

```
1: Input: the query element $q$, the similarity graph $G = (V, E)$,
   the priority queue size $k$ and the similarity measure $f$.
2: Randomly choose a vertex $p \in V$ as the start point and initialize
   the set $A \leftarrow \{p\}$.
3: Set $p$ as checked and the rest of vertices as unchecked.
4: while $A$ does not converge do
5:     Add unchecked neighbors of vertices in $A$ to $A$.
6:     Set vertices in $A$ as checked.
7:     $A \leftarrow$ top-$k$ elements of $v \in A$ in order of $f(v, q)$.
8: Output: $A$.
```

3.2 Edge Selection

To achieve an efficient and effective approximate Delaunay graph, the key fact of the graph construction is the neighbor/edge selection (shown in Algorithm 1 Line 5-16), which makes sure that the limited selected neighbors are representative (e.g., diverse in directions for the metric cases). This will largely benefit the searching trade-off between efficiency and effectiveness [35]. For example, a schematic illustration for the metric measure is represented in Figure 2(a). Here the similarity measure is negative Euclidean distance. we assume that the current inserting point is x and the returned candidate neighbors are $\{a, b, c\}$. HNSW will first add x's nearest neighbor a to C and connect x to a. Then for other vertices in the neighbor set, HNSW selects c instead of b to connect with for $f(x, c) > f(a, c)$ and $f(x, b) < f(a, b)$—b is more like a while c is more diverse from a in directions. In this way, the navigation of greedy search–especially in the first steps–will be more efficient.

As an heir of HNSW, ip-NSW shares Algorithm 1 and Algorithm 2. The only difference is in the similarity measure. HNSW is designed for metric measures, such as cosine similarity or negative

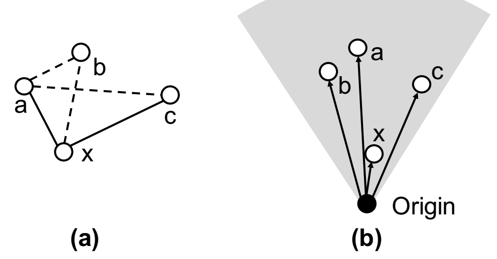

Figure 2: Illustrations of edge selection. (a) The edge selection method for metric spaces. (b) Failures when we extend the same edge selection to inner product.

Euclidean distance, while ip-NSW sets $f(x, q) = x^\top q$. As can be noted, inner product poses challenges to the edge selection method of Algorithm 1 since the inner product does not follow the triangle inequality of metric space. As illustrated by the example shown in Figure 2(b), the small norm data point x retrieves a set of nodes with larger inner product values with it, say $\{a, b, c\}$. As analyzed in [36], $\{a, b, c\}$ are usually nodes with larger norms in the same *Voronoi cell* with x. In Figure 2(b), the gray area represents the corresponding Voronoi cell. The ℓ_2-norms $\{a, b, c\}$ are much larger than that of x. For this case, let's re-check the edge selection method in Algorithm 1. Due to the inner product values between x and $\{a, b, c\}$ are much smaller than those among $\{a, b, c\}$, x will not connect to either b or c for $x^\top b \ll a^\top b$ and $x^\top c \ll a^\top c$ too. So x will only connect to the node, a, which has the largest inner product $x^T a$. Eventually, most of data points will not have (incoming) edges and will not be visited in query searching. Thus the edge selection method in Algorithm 1 will lead to an inefficient index graph and will largely limit the retrieval performance in the MIPS problem, especially for top-k results, where $k \gg 1$, e.g., 100.

We analyzed norm distributions of experimental datasets and found that the case shown in Figure 2(b) is very common. Analysis results of two datasets, Yelp MF and Amovie MF, are shown in Figure 1. Due to the long-tail ℓ_2-norm distribution, only a few data points have large ℓ_2-norms while most of the others have relatively much smaller norms. In the next section, we will introduce a norm adjusted edge selection method—more suitable for constructing graph indices with inner product.

4 NORM ADJUSTED PROXIMITY GRAPH

In this section, we will introduce the newly proposed graph-based MIPS method—Norm Adjusted Proximity Graph (**NAPG**). Let's recall the example shown in Figure 2(b). The norm of x is much smaller than its neighbor candidates—a, b and c, which makes $x^\top p_i$ always smaller than $p^\top p_i$ in Algorithm 1 Line 9. This will lead to an inefficient index graph and hurt the searching performance. The impacts become more apparent in real applications that consider top-10 or top-100 MIPS results.

To address this limitation, we propose a new method to construct proper graphs for MIPS. The basic idea is to make the edge selection work again—representative neighbors can be selected for those nodes with small norms (the vast majority). Specifically, we adjust the norm of the inserted node x by an adjusting factor α and replace the inequality in Algorithm 1 Line 9 by:

$$\alpha x^\top p_i < p^\top p_i. \quad (4)$$

We usually set $\alpha > 1$ and make the scales of $\alpha x^\top p_i$ and $p^\top p_i$ comparable. In this way, not only the node having the largest inner product with x (i.e., a in Figure 2(b)), but also other good candidates can be selected as x's outgoing neighbors. The key question left is how to choose proper values for the adjusting factor α. Next, we introduce a flexible method to estimate α based on sampled data.

4.1 Adjusting Factor Estimation

The function of the factor α is to adjust two sides of the inequality (4) into comparable scales. For each index data point x and its top-n MIPS results $\{p_i\}$ (corresponding to the neighbor candidate set), we wish the expectation of adjusted x's inner product values with $\{p_i\}$ and inner product values among data points $\{p_i\}$ are equal:

$$\mathbb{E}\left[\alpha x^\top p_i\right] = \mathbb{E}\left[p_i^\top p_j\right]. \quad (5)$$

Since α is a constant, Eq. (5) can be rewritten into:

$$\alpha = \mathbb{E}\left[p_i^\top p_j\right] / \mathbb{E}\left[x^\top p_i\right]. \quad (6)$$

The expectations $\mathbb{E}\left[p_i^\top p_j\right]$ and $\mathbb{E}\left[x^\top p_i\right]$ can be estimated by averaging values over a sample. Specifically, we estimate α as follows:

1) Sample a small subset $\{x\}$ from the entire indexing data.
2) For each sampled data point x, retrieve its top-n (e.g., n=100) MIPS results $\{p_i\}$ from the entire data.
3) Calculate the average inner product values between all x and their $\{p_i\}$, $\mathtt{Avg}(x^\top p_i)$.
4) Calculate the average inner product values among all x's $\{p_i\}$, $\mathtt{Avg}(p_i^\top p_j)$.
5) Get the estimated $\alpha = \mathtt{Avg}(p_i^\top p_j)/\mathtt{Avg}(x^\top p_i)$.

After α is estimated, we replace the inequality in Algorithm 1 Line 9 with inequality (4) to construct a better approximate Delaunay graph.

4.2 Norm Range based Adjusting Factors

It can be noted that the global adjusting factor α may not be optimal since the inserting data points have varying norms. A more fine-grained solution is to estimate a local adjusting factor for data points in each norm range. This norm-range idea is similar with ones used in [23] and [53], which reduce the MIPS problem to traditional ANN search problems by pre-processing data in each norm range. Here, we explore the norm-range in a different way, for norm-range based local adjusting factors.

To get local adjusting factors, we will partition data points first by their ℓ_2-norms. Specifically, we split the global norm range (from the smallest to the largest) into N sub-ranges evenly by ℓ_2-norm values. Each sub-range is considered as a strata. We draw samples from the dataset using stratification sampling—each sub-range obtains the same number of samples. α_r is computed on the samples based on Eq. (6) for each sub-range. In the graph construction, we locate the corresponding α_r of the current inserting point based on its norm value. Then α_r is employed as a factor in edge selection.

4.3 Implementations

In the algorithmic level, the proposed NAPG shares the greedy search method shown in Algorithm 2 but exploits a unique graph construction algorithm (i.e., Algorithm 4). Before constructing the NAPG, we will first estimate the adjusting factors for each norm range. The norm range based adjusting factor estimation method is represented in Algorithm 3.

Algorithm 3 Norm Range Based Adjusting Factor Estimation

```
1:  input: Item latent vector set $S$, the number of norm ranges $N$, the sample size $Z$ for each range and the amount $n$ of return results for each sample data.
2:  Partition $S$ into $N$ sub-ranges $S_1, S_2, ..., S_N$ such that $S_r$ holds base vectors whose $\ell_2$-norm ranked in the sub-range $\left[\frac{(r-1)|S|}{N}, \frac{r|S|}{N}\right]$ and record the partition by the map $R$.
3:  for $r \leftarrow 1$ to $N$ do
4:      Randomly sample $Z$ data points $\{x\}$ from $S_r$.
5:      for all $x$ in $\{x\}$ do
6:          Retrieve $n$ data points $\{p_i\}$ from $S$ which have largest inner product values with $x$.
7:      Calculate the average of inner product values between all $x$ and their $\{p_i\}$, $\mathtt{Avg}(x^\top p_i)$.
8:      Calculate the average of inner product values among all $x$'s $\{p_i\}$, $\mathtt{Avg}(p_i^\top p_j)$.
9:      Get the estimated $\alpha_r \leftarrow \frac{\mathtt{Avg}(p_i^\top p_j)}{\mathtt{Avg}(x^\top p_i)}$.
10: output: The norm range map $R$ and the estimated adjusting factors for each sub-range $\{\alpha_1, \alpha_2, \cdots, \alpha_N\}$.
```

Algorithm 4 NAPG Construction

```
1:  input: Item latent vector set $S$, the norm range map $R$, the estimated adjusting factors for each sub-range $\{\alpha_1, \alpha_2, \cdots, \alpha_N\}$, the maximum vertex degree $M$ and the priority queue size $k$.
2:  Initialize graph $G \leftarrow \emptyset$
3:  for all $x$ in $S$ do
4:      Greedy search $k$ vertices $\{p_i\}$ on $G$ by $SoG(x, G, k, f)$ that have largest inner product values with $x$, place them in descending order.
5:      Get the adjusting factor $\alpha_r$ for the range $x$ belongs to.
6:      $C \leftarrow \emptyset$.
7:      for $i \leftarrow 1$ to $k$ do
8:          flag $\leftarrow$ True
9:          for all $p$ in $C$ do
10:             if $\alpha_r x^\top p_i < p_i^\top p$ then
11:                 flag $\leftarrow$ False
12:                 break
13:         if flag = True then
14:             $C \leftarrow C \cup \{p_i\}$
15:             add edge $\{x, p_i\}$ to $G$
16:         if $|C| = M$ then
17:             break
18: output: index graph $G$
```

Based on the estimated factors, the NAPG construction can be implemented as Algorithm 4. Note that the main difference between HNSW/ip-NSW and NAPG lies in the edge selection step. With the norm range based adjusting factors, NAPG will construct a more effective and efficient index graph for MIPS than ip-NSW. As will be shown in experimental results, benefiting from the proper index graph, NAPG works much better than ip-NSW in MIPS.

5 EXPERIMENTS

5.1 Datasets

We evaluate the proposed NAPG algorithm for MIPS on vector datasets from various sources. Firstly, we will evaluate different methods in fast item retrieval for online recommender systems. We exploit two widely used datasets in recommendation: **Yelp**[1] and Amazon Movie (**Amovie**)[2]. For Yelp, we did not further filter it for it was processed before. For Amovie, we filtered the dataset in the way that users with at least 30 interactions are retained. The item amounts in these two datasets are much bigger than those in other recommendation datasets, which are more appropriate for exploring item retrieving efficiency.

Table 1: Dataset Statistics.

Datasets	# Index Vec	# Queries	# Dim
Yelp MF	25815	25677	64
Yelp DMF	25815	25677	64
Amovie MF	104708	7748	64
Amovie DMF	104708	7748	64
Covertype	580012	1000	54
MNIST1m	1000000	1000	784

As shown in Table 1, four different vector datasets were generated from **Yelp** and **Amovie**, which are named **Yelp MF**, **Yelp DMF**, **Amovie MF**, and **Amovie DMF**. In the following, we explain the details on how these four datasets in Table 1 were generated, based on *matrix factorization (MF)* [22, 28, 29] and *deep matrix factorization (DMF)* [52].

Matrix factorization (MF) [22, 28, 29] is the most representative framework for collaborative filtering. We utilize the implementation in [6] to generate the vectors for users and items. Also we apply deep matrix factorization (DMF) [52] to generate alternative vector datasets. Note that, in the original paper [52], the matching function is cosine similarity. Here we set the function as inner product. The loss function is set as the squared loss: $L_{sq} = \sum_{(i,j)\in Y^{+}\cup Y^{-}}(Y_{ij} - \hat{Y}_{ij})^2$, where $\hat{Y}$ is the predicted rating score and Y is the real score. Y^{+} is the positive training data and Y^{-} is the sampled negative data. The number of network layers is 2, as suggested in [52].

For these four datasets, the length for all vectors is set as 64. Note that vectors produced by MF and DMF are quite different. Vectors produced by MF are dense and contain both positive and negative real numbers. While vectors from DMF are sparse and only contain non-negative numbers because of the RELU active function.

In addition to those four datasets, Table 1 also includes two larger vector datasets from other domains: **Covertype**[3] and **MNIST1m**[4]. MNIST1m is a subset of the original data. Different from recommendation datasets, which require a method to produce latent vectors for users and items, we directly use original vectors of these two general datasets. 1000 additional vectors are randomly sampled as queries (corresponding to those user vectors). Statistics of all 6 datasets are listed in Table 1. As can be seen, our experimental datasets vary in dimension, sources and extraction methods, which is sufficient for a fair comparison.

Finally, to produce evaluation labels, we rank item/base vectors for each user/query vector by brute force scanning (in inner product). Top-k items are considered as labels (k = 10 and 100).

5.2 Baselines

As baselines, we choose some representative previous state-of-the-art MIPS algorithms to evaluate the proposed method.

Greedy-MIPS is a state-of-the-art MIPS algorithm proposed in [54]. We used the original implementation[5].

ip-NSW is a MIPS algorithm proposed recently in [36]. As the same with our proposed method, ip-NSW is also a graph-based method. We used the implementation from the authors[6].

Range-LSH is a hash-based method proposed in [53]. As shown in the original paper, Range-LSH outperforms previous hash-based methods for MIPS. So we compare with it as a representative of this line. We used the original implementation[7]. Note that H2-ALSH [23] also utilizes the data partitioning method (and the query transformation). As the experimental results in the original paper, H2-ALSH works well mainly because of the partitioning method but not the query transformation. Therefore, Range-LSH and H2-ALSH are actually similar methods. We pick Range-LSH as a baseline since its implementation is publicly available.

5.3 Experimental Settings

All comparing methods have tunable parameters. NAPG and ip-NSW have three common parameters: M, $k_{construction}$ and k_{search}, which control the degree of each node and the number of search attempts. Besides, NAPG has a unique parameter for the number of norm ranges, N. Greedy-MIPS has a key parameter *budget* and Range-LSH has two parameters: code length and the number of sub-datasets. To get a fair comparison, we vary all parameters over a fine grid for all methods.

There are two popular ways to evaluate ANNS/MIPS algorithms: 1) **Recall vs. Time**; 2) **Recall vs. Computations**. They both evaluate the searching trade-off between efficiency and effectiveness. Recall vs. Time reports the number of queries an algorithm can process per second at each recall level. Recall vs. Computations reports the amount or percentage of pairwise distance/similarity computations that the search algorithm costs at each recall level. We will show both of these perspectives in the following experiments for comprehensive evaluation.

5.4 Recall vs. Time Results

Comparison results via Recall vs. Time will be first represented, as in Figure. 3. All methods can be evaluated by this view. Each row is for each dataset from Yelp MF, Yelp DMF, Amovie MF, Amovie DMF, Covertype and MNIST1m. The two columns are results for top-10 and top-100 labels. As can be seen, the proposed NAPG

[1]https://www.Yelp.com/dataset/challenge
[2]http://jmcauley.ucsd.edu/data/amazon
[3]http://archive.ics.uci.edu/ml/datasets/Covertype
[4]https://leon.bottou.org/papers/loosli-canu-bottou-2006

[5]Greedy-MIPS: https://github.com/rofuyu/exp-gmips-nips17
[6]ip-NSW: https://github.com/stanis-morozov/ip-nsw
[7]Range-LSH: https://github.com/xinyandai/similarity-search

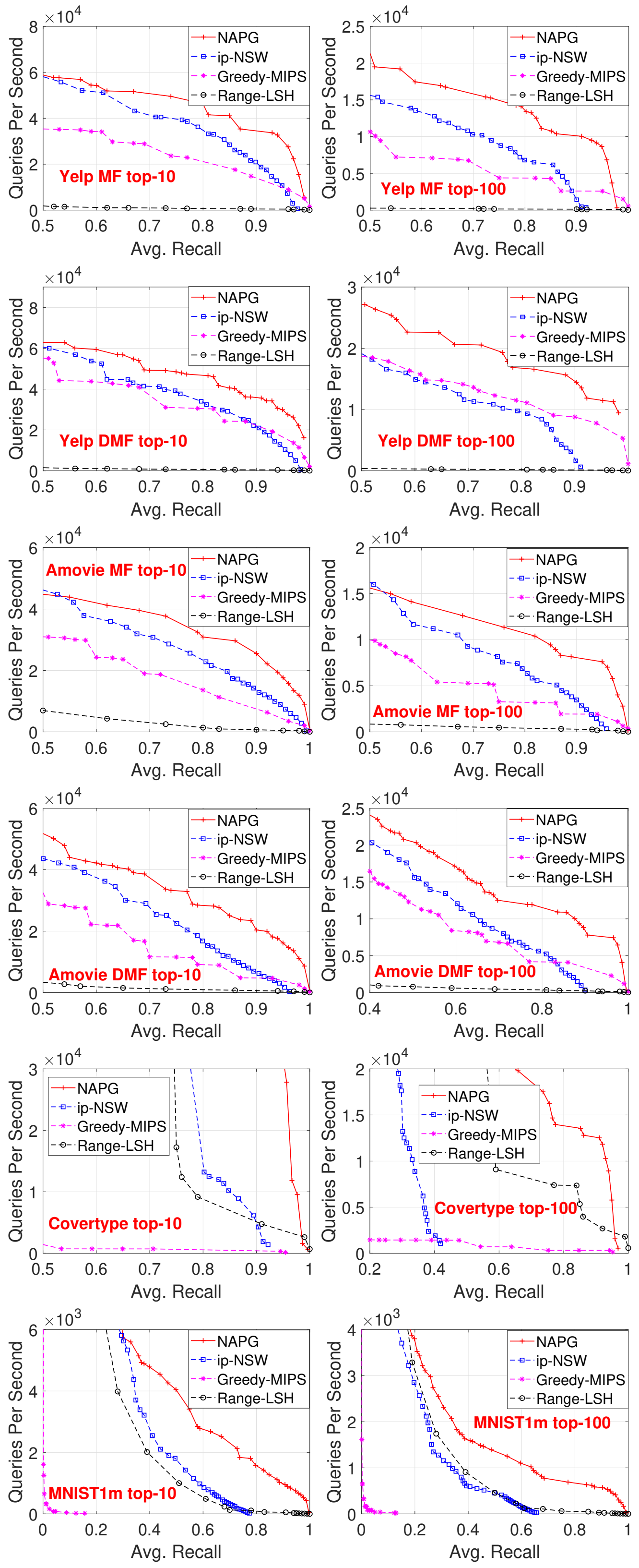

Figure 3: The comparisons via Recall vs. Time. for all methods, based on top-10 and top-100 ground truth labels.

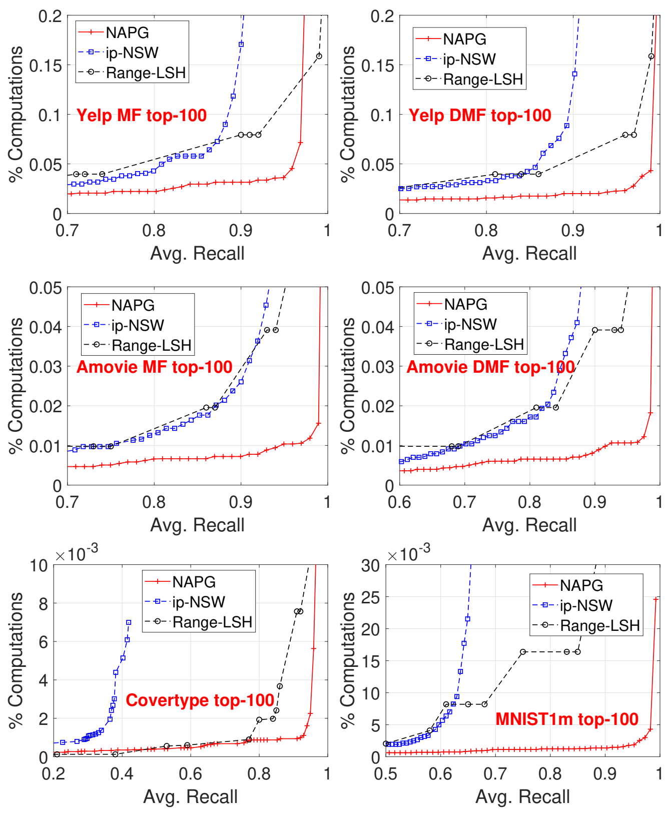

Figure 4: The comparisons via Recall vs. Computations. Results for top-100 labels are shown.

works best in most of the cases, especially in higher recall levels, which are more meaningful in real applications. Besides, NAPG works insistently well on different sizes of labels. Different datasets have extremely different vectors, while NAPG has superiority on all these kinds of vector data. These comparisons prove NAPG's efficiency and robustness.

Conversely, no baselines work insistently well on different sizes of labels. For example, ip-NSW and Greedy-MIPS become worse and worse from top-10 to top-100. Comparing with ip-NSW, NAPG improves the edge selection method by flexibly adjusting data points' norm. It is clear that, on these datasets, NAPG provides much better performance than ip-NSW, especially on top-100 labels. ip-NSW works badly on top-100 labels since it filters out too many meaningful neighbors in edge selection and that hurts the performance heavily. Hash-based method Range-LSH performs badly in this view, comparing with other methods. As can be seen, its performance is not comparable to others. Beyond recommendation datasets, we note that NAPG works consistently well on the two larger datasets, Covertype and MNIST1m. The close baseline ip-NSW works inferior. Especially for the case of Covertype top-100, ip-NSW can only get recalls around 40%.

5.5 Recall vs. Computations Results

The corresponding comparison results via Recall vs. Computations are shown in Figure. 4. From this point of view, Greedy-MIPS cannot be evaluated. So we only represent results for the other three methods. We only represent results for top-100 labels due to the limited space. Similarly, NAPG provides better results comparing baselines (in most of the cases). Quantitatively, to get high recall levels, NAPG only requests a small partition of pair-wise computations. For example, on the Amovie MF dataset and for top-100 labels, to achieve 95% recall, NAPG only requests 1% computations. While for baselines, it is even difficult to achieve 95% recall.

Unexpectedly, Range-LSH works comparably with others in the Recall vs. Computations view. As explained in Section 5.3, Recall vs. Computations does not consider the cost of different index structures. Although Range-LSH works well in this view, its overall time cost is much higher than others as shown in Section 5.4. The possible reason is that the table based index used in Range-LSH is not that efficient in searching. It is clear that graph-based index structures has significant superiority in searching efficiency.

5.6 Parameter Analysis

In this section, we analyze the key parameter N of NAPG, the number of norm ranges. N is a data dependent parameter, dependent on dataset's size and norm distribution. In the experiments above, we consider N as a tunable parameter together with other parameters. Here, to analyze N, we fix $M = 16$ and $k_{construction} = 100$. Results on Yelp MF (top-10) are shown in Figure 5 left. As can be seen, the best N for this dataset is around 3 or 5, which is much better than single norm range. If we split the dataset into even more ranges than this optimal solution, say 10, it will hurt the performance too. Because more ranges will lead to some ranges with extremely small norm data and then will produce extremely large adjusting factors. It will over adjust data and construct too many useless edges.

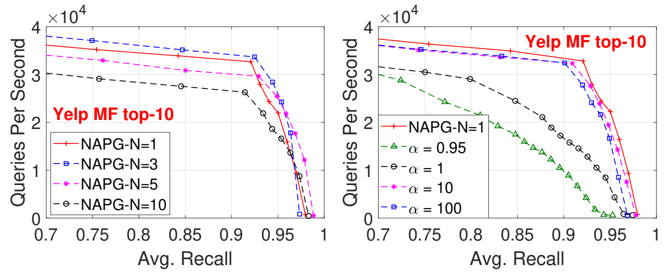

Figure 5: Left: Parameter analysis for the number of norm ranges N in NAPG. Right: Study of factor α estimation. We fix $M = 16$ and $k_{construction} = 100$ for these two experiments.

5.7 Estimated Factor Study

The adjusting factor α is vital for the proposed NAPG. We design a flexible factor estimation method based on sample data (Section 4). Another way is setting α as a tunable parameter (for all index data or for each norm range). In this section, we evaluate our factor estimation method empirically. The results are shown in Figure 5 right. We only show results for Yelp MF on top-10 because of the limit space. In the showing example, we study the case of norm range number $N = 1$ for NAPG vs. tuning α as 0.95, 1, 10, 100. The estimated α of NAPG is 4.33. Note that $\alpha = 1$ is the solution of ip-NSW. If α is extremely large, say 100, the inequality in Algorithm 1 Line 9 will be always false and the edge selection will become invalid. As can been seen, NAPG learns a good estimation of α. It brings almost the best performance among all choices. One tunable parameter less will ease to apply the proposed method, without performance loss. For multiple norm ranges, $N > 1$, the advantage will be more apparent.

6 CONCLUSION

Fast inner product search is an essential task in many application domains such as online recommender systems and retrieval based question answering. In the literature, this task is often referred to as the *maximum inner product search* (MIPS) problem. While solving MIPS exactly can be prohibitively too expensive (or even infeasible) in industrial practice, developing approximate MIPS algorithms is also challenging. The is because the inner product is a non-metric measure, which makes MIPS distinctly different from many traditional approximate near neighbor (ANN) search problems (which typically deal with metric measures such as the cosine similarity). In this paper, we proposed a new graph based searching method, namely *norm adjusted proximity graph* (NAPG) for approximate MIPS. To order to construct a proper index graph for the inner product, NAPG adjusts the norms of data points when inserting them into the graph. A flexible adjusting factor estimation method and a norm range based improvement are introduced. The proposed method is flexible and robust. The empirical evaluations on a range of datasets demonstrate that the proposed NAPG is indeed effective and efficient, compared to common baselines.

REFERENCES

[1] Firas Abuzaid, Geet Sethi, Peter Bailis, and Matei Zaharia. To index or not to index: Optimizing exact maximum inner product search. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1250–1261, Macao, China, 2019.
[2] Amin Ahmad, Noah Constant, Yinfei Yang, and Daniel Cer. ReQA: An evaluation for end-to-end answer retrieval models. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering (MRQA@EMNLP)*, pages 137–146, Hong Kong, China, 2019.
[3] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.
[4] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. Speeding up the xbox recommender system using a Euclidean transformation for inner-product spaces. In *Proceedings of the Eighth ACM Conference on Recommender Systems (RecSys)*, pages 257–264, Foster City, CA, 2014.
[5] Ben Frederickson. Approximate nearest neighbours for recommender systems. https://www.benfrederickson.com/approximate-nearest-neighbours-for-recommender-systems/, 2019.
[6] Ben Frederickson. Fast python collaborative filtering for implicit feedback datasets. https://github.com/benfred/implicit, 2019.
[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
[8] Leonid Boytsov, David Novak, Yury Malkov, and Eric Nyberg. Off the beaten path: Let's replace term-based retrieval with k-nn search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1099–1108, Indianapolis, IN, 2016.
[9] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.
[10] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, pages 191–198, Boston, MA, 2016.
[11] Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. MOBIUS: towards the next generation of query-ad matching in baidu's sponsored search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD) 2019*, pages 2509–2517, Anchorage, AK, 2019.
[12] Jerome H. Friedman, F. Baskett, and L. Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24:1000–1006, 1975.

[13] Jerome H. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

[14] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proc. VLDB Endow.*, 12(5):461–474, 2019.

[15] Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 699–709, Baltimore, MD, 2014.

[16] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB)*, pages 518–529, Edinburgh, Scotland, UK, 1999.

[17] Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. Quantization based fast inner product search. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 482–490, Cadiz, Spain, 2016.

[18] Rob Hall and Josh Attenberg. Fast and accurate maximum inner product recommendations on map-reduce. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 1263–1268, Florence, Italy, 2015.

[19] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2042–2050, Montreal, Canada, 2014.

[20] Jun Hu and Ping Li. Collaborative filtering via additive ordinal regression. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, pages 243–251, Marina Del Rey, CA, 2018.

[21] Jun Hu and Ping Li. Collaborative multi-objective ranking. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1363–1372, Torino, Italy, 2018.

[22] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, pages 263–272, Pisa, Italy, 2008.

[23] Qiang Huang, Guihong Ma, Jianlin Feng, Qiong Fang, and Anthony K. H. Tung. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1561–1570, London, UK, 2018.

[24] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 604–613, Dallas, TX, 1998.

[25] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 935–944, San Francisco, CA, 2016.

[26] Ata Kaban. Improved bounds on the dot product under random projection and random sign projection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 487–496, Sydney, Australia, 2015.

[27] Omid Keivani, Kaushik Sinha, and Parikshit Ram. Improved maximum inner product search with better theoretical guarantee using randomized partition trees. *Machine Learning*, 107:1069–1094, 2018.

[28] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 426–434, Las Vegas, NV, 2008.

[29] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 447–456, Paris, France, 2009.

[30] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[31] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[32] Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. Reasoning in vector space: An exploratory study of question answering. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

[33] Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. Fexipro: fast and exact inner product retrieval in recommender systems. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD)*, pages 835–850, Chicago, IL, 2017.

[34] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.*, 45:61–68, 2014.

[35] Yury A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, 2020.

[36] Stanislav Morozov and Artem Babenko. Non-metric similarity graphs for maximum inner product search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4726–4735, Montreal, Canada, 2018.

[37] Behnam Neyshabur and Nathan Srebro. On symmetric and asymmetric LSHs for inner product search. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1926–1934, Lille, France, 2015.

[38] Parikshit Ram and Alexander G. Gray. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 931–939, Beijing, China, 2012.

[39] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW)*, pages 285–295, Hong Kong, China, 2001.

[40] Min Joon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P. Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*, pages 4430–4441, Florence, Italy, 2019.

[41] Anshumali Shrivastava and Ping Li. Fast near neighbor search in high-dimensional binary data. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pages 474–489, Bristol, UK, 2012.

[42] Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems (NIPS)*, pages 2321–2329, Montreal, Canada, 2014.

[43] Anshumali Shrivastava and Ping Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 981–991, Florence, Italy, 2015.

[44] Anshumali Shrivastava and Ping Li. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 812–821, Amsterdam, The Netherlands, 2015.

[45] Yukihiro Tagami. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 455–464, Halifax, Canada, 2017.

[46] Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. On efficient retrieval of top similarity vectors. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5235–5245, Hong Kong, China, 2019.

[47] Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. Fast item ranking under neural network based measures. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 591–599, Houston, TX, 2020.

[48] Christina Teflioudi and Rainer Gemulla. Exact and approximate maximum inner product search with lemp. *ACM Transactions on Database Systems (TODS)*, 42(1):1–49, 2016.

[49] Christina Teflioudi, Rainer Gemulla, and Olga Mykytiuk. Lemp: Fast retrieval of large entries in a matrix product. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 107–122, Melbourne, Australia, 2015.

[50] Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning*, 81(1):21–35, 2010.

[51] Jun Xu, Xiangnan He, and Hang Li. Deep learning for matching in search and recommendation. *Found. Trends Inf. Retr.*, 14(2-3):102–288, 2020.

[52] Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3203–3209, Melbourne, Australia, 2017.

[53] Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. Norm-ranging LSH for maximum inner product search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2956–2965, Montreal, Canada, 2018.

[54] Hsiang-Fu Yu, Cho-Jui Hsieh, Qi Lei, and Inderjit S Dhillon. A greedy approach for budgeted maximum inner product search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5453–5462, Long Beach, CA, 2017.

[55] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. Large-scale multi-label learning with missing labels. In *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pages 593–601, Beijing, China, 2014.

[56] Weijie Zhao, Shulong Tan, and Ping Li. SONG: approximate nearest neighbor search on GPU. In *Proceedings of the 36th IEEE International Conference on Data Engineering (ICDE)*, pages 1033–1044, Dallas, TX, 2020.

[57] Zhixin Zhou, Shulong Tan, Zhaozhuo Xu, and Ping Li. Möbius transformation for fast inner product search on graph. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8216–8227, Vancouver, Canada, 2019.