IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 November 2022, accepted 1 January 2023, date of publication 4 January 2023, date of current version 10 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3234072

RESEARCH ARTICLE

Categorical Diversity-Aware Inner Product Search

KOHEI HIRATA[1], DAICHI AMAGATA[1], (Member, IEEE), SUMIO FUJITA[2], AND TAKAHIRO HARA[1], (Senior Member, IEEE)

[1]Graduate School of Information Science and Technology, Osaka University, Osaka 565-0871, Japan
[2]Yahoo Japan Corporation, Tokyo 102-8282, Japan

Corresponding author: Kohei Hirata (hirata.kohei@ist.osaka-u.ac.jp)

This work was supported in part by the Japan Science and Technology Agency (JST) PRESTO under Grant JPMJPR1931, in part by the Japan Society for the Promotion of Science Grant-in-Aid for Scientific Research (A) under Grant 18H04095, and in part by JST CREST under Grant JPMJCR21F2.

ABSTRACT The problem of maximum inner product search (MIPS) is one of the most important components in machine learning systems. However, this problem does not care about diversity, although result diversification can improve user satisfaction. This paper hence considers a new problem, namely the categorical diversity-aware IPS problem, in which users can select preferable categories. Exactly solving this problem needs $O(n)$ time, where n is the number of vectors, and is not efficient for large n. We hence propose an approximation algorithm that has a probabilistic success guarantee and runs in sub-linear time to n. We conduct extensive experiments on real datasets, and the results demonstrate the superior performance of our algorithm to that of a baseline using an existing MIPS technique.

INDEX TERMS Inner product search, category, diversification, high-dimensional data.

I. INTRODUCTION

The problem of maximum inner product search (MIPS) has been extensively studied for a decade [5], [9], [19], [26], [27], because this problem is one of the most important components of modern machine-learning systems. In these systems, objects (e.g., users and items) are usually represented as dense high-dimensional vectors, and the inner product of two vectors (e.g., those of a user and an item) indicates their relevance (a high inner product shows a high relevance). A typical example is recommender systems [3], [5], [16], [20], [22], [30], [31], [32], [33]. Given a user (query) vector, MIPS returns item the most relevant to the user (i.e., the item where the inner product of the user and item vectors is the largest among all items) and this item is considered as a recommendation item.

A. MOTIVATION

Although this problem can search for relevant vectors, considering only relevance may not maximize user satisfaction, since the search result may be skewed.

Example 1: The left part of TABLE 1 shows the k-MIPS result ($k = 5$) of a user in the real dataset MovieLens. We see that 4 out of 5 movies belong to the same category, "adventure". Even if the user wants the system to display a recommendation result with more categories, k-MIPS cannot deal with this case.

To avoid this skew, result diversification is often employed [1], [4]. There are many definitions of diversity, and this paper focuses on categorical diversity, because this is often required in recommendation scenarios [7] of entertainment (e.g., video on demand services) and business (e.g., online recruiting services).

Consider that a given dataset X has multiple categories and each vector $x \in X$ belongs to one category. Past studies of categorical diversification consider *implicit* approaches. For example, Cheng et al. proposed a normalized coverage measurement and tried to obtain a high value of the criterion [7]. In [37], Zheng et al. proposed a learning-based model that exploits a user-item bipartite graph to make more categories reachable from users. These works devised model-level techniques, so it is not guaranteed that the search results contain preferable categories. To summarize, (i) the MIPS problem may yield a skewed result and (ii) existing categorical diversification problems cannot

The associate editor coordinating the review of this manuscript and approving it for publication was Qingli Li.

TABLE 1. Example of difference between k-MIPS and categorical diversity-aware k-IPS result sets for a user in MovieLens ($k = 5$ and $k_{adventure} = 2$, $k_{mystery} = 1$, $k_{fantasy} = 1$, and $k_{animation} = 1$).

k-MIPS result (category)	Categorical diversity-aware k-IPS result (category)
Star Wars: Episode IV (Adventure)	Star Wars: Episode IV (Adventure)
Darwyn Cooke's Batman Beyond (Adventure)	Lord of the Rings: The Return of the King (Adventure)
Lord of the Rings: The Return of the King (Adventure)	Sherlock: The Blind Banker (Mystery)
Star Wars: Episode V (Adventure)	Harry Potter and the Prisoner of Azkaban (Fantasy)
Great Passage (Drama)	Toy Story (Animation)

flexibly control the degree of categorical diversity in the search results.

To remove these drawbacks, an explicit way of categorical diversification is necessary. To this end, we address the following two important points:

- It is desirable for users to be able to specify the categories in which they are interested, because doing so guarantees that the search result contains user-preferable categories.
- In addition, the search results should contain items that are ranked in the top-K, where $K \geq k$ is a user-tolerable ranking (e.g., $K = 100$), to keep highly relevant (and categorically diversified) results. (Notice that $K = k$ is too strict, because the top-k items usually do not cover user-specified categories, as shown in Example 1.)

Based on these two points, we formulate a new variant of the MIPS problem, namely the categorical diversity-aware k-IPS problem. In this problem, users can specify the categories that they want the search result to display, which addresses the first point. Specifically, they can specify k_i ($\leq k$), and this means that k_i vectors belonging to category c_i are required to be included in the search result. Let τ be the inner product of q and x', where q is a given query vector and x' is the top-K vector in the K-MIPS problem for q. Assuming that $k_i \geq 1$, each vector $x \in X_i$, where X_i is a set of vectors belonging to c_i, has to satisfy $x \cdot q \geq \tau$ to be included in the search result. This addresses the second point. The categorical diversity-aware k-IPS problem retrieves those k vectors that satisfy the above requirements.

Example 2: The right part of TABLE 1 shows a categorical diversity-aware k-IPS result for the same user in Example 1 (we set $K = 100$). Assume that this user provides $k_{adventure} = 2$, $k_{mystery} = 1$, $k_{fantasy} = 1$, and $k_{animation} = 1$ (i.e., s/he wants the system to display two adventure movies, one mystery movie, one fantasy movie, and one animation movie). Our problem follows these inputs, and the user can enjoy a diversified result.

B. CHALLENGE

A simple algorithm that solves our problem is to run a K-MIPS algorithm first to obtain τ and then exhaustively search vectors x such that $x \cdot q \geq \tau$ for each category c_i where $k_i \geq 1$. Even a state-of-the-art exact MIPS algorithm [20] requires $O(n)$ time, where $n = |X|$. Therefore, this simple algorithm incurs $O(n)$ time. This time cost is large for search problems, because it corresponds to accessing (at most) all vectors in X. However, this is unavoidable if the exact search result is required.

Many applications are tolerable approximate results and require high efficiency [13], [17], [24], [28], [29], [30]. An approximation algorithm, which runs in time sub-linear to n, is therefore required. The challenge is then to design such an approximation algorithm that returns a search result *with high accuracy* in time sub-linear to n. This is not trivial, and existing approximate MIPS algorithms cannot overcome this challenge. This is because state-of-the-art approximate MIPS algorithms have no theoretical time [13], [29], [36], [38] or have $O(n \log n) > O(n)$ time [17], [28].

C. CONTRIBUTION

We overcome the above challenge and propose a sub-linear time algorithm with a probabilistic success guarantee. We show that our problem can be transformed into the cosine similarity search problem, and this observation enables us to estimate τ. That is, we do not have to run any MIPS algorithm to compute τ. Furthermore, the estimation provides a bounded error by using a constant number of iterations. We carefully design a dataset partitioning technique, and, thanks to this partitioning, we can obtain an accurate search result only from some subsets of X. We theoretically analyze its time and space complexities along with its success probability. Its practical performance is also evaluated by using real datasets, and the results confirm that our algorithm yields high accuracy and is much faster than the baseline algorithm.

We summarize the main contributions of this paper as follows:

- We formalize the categorical diversity-aware k-IPS problem. This is the first work to address this problem.
- We show that exactly solving this problem requires $O(n)$ time.
- We propose a sub-linear time approximation algorithm with a probabilistic success guarantee: for each specified category, if there is at least one vector that satisfies the condition for being in the result, it can be included in the result probabilistically.
- We conduct extensive experiments using real datasets. Our experimental results demonstrate that our algorithm is fast and accurate.

D. ORGANIZATION

The rest of this paper is organized as follows. Section II introduces the problem definition and our baseline algorithm.

Section III reviews related studies. In Sections IV and V, we propose our algorithm and report our experimental results, respectively. Finally, in Section VI, we conclude this paper.

II. PRELIMINARY

We first introduce our formal problem definition, and we then design a baseline algorithm for this problem.

A. PROBLEM DEFINITION

Let X be a set of d-dimensional vectors, where d is high (e.g., $d \geq 100$). We use n to denote $|X|$. The inner product of x and x' is denoted by $x \cdot x'$. For ease of presentation, we first define the problem of k-maximum inner product search (or k-MIPS).

Definition 1 (k-MIPS Problem): Given a set X of vectors, a query vector q, and an output size k, this problem retrieves k vectors $x \in X$ such that $x \cdot q$ is the largest among X (ties are broken arbitrarily).

We here consider categorical diversification. We assume that there are multiple categories, and each $x \in X$ belongs to one category. Let $X_i \subset X$ be a set of vectors belonging to a category $c_i \in C$, where C is a set of categories in X. We have $X = \bigcup X_i$. Consider that the display size is k, e.g., k items can be included in a search result. The k-MIPS problem makes such a result by considering only inner products. This does not guarantee that the result contains items with various categories, so users may not be able to obtain "new notices." For example, in a movie recommendation scenario like Example 1, the result contains only adventure movies the user knows well. From this, it is desirable for users to specify categories as input. More specifically, we consider a scenario where users can specify k_i for each c_i so that $\sum k_i = k$. Note that k_i is the number of items in c_i that will be included in the result. This allows users to flexibly control the categorical diversity of the result. Based on this idea, we define our problem, namely categorical diversity-aware k-IPS.[1]

Definition 2 (Categorical Diversity-Aware k-IPS): Given a set X of vectors, a query vector q, an output size k, an output size k_i for each category c_i ($\sum k_i = k$), and a ranking threshold $K \geq k$, this problem retrieves, for each category c_i, at most k_i vectors $x \in X_i$ such that $x \cdot q \geq \tau$, where τ is the K-th largest inner product in the K-MIPS problem for q. (If X_i contains more than k_i vectors such that $x \cdot q \geq \tau$, any k_i vectors satisfying this can be selected.)

We here put two important notes:

1) Although the above problem considers the output size for each category (i.e., k_i), specifying this when $k_i = 0$ is not user-friendly. We therefore assume that $k_i = 0$ by default, and users specify k_i only when $k_i \geq 1$.
2) This problem uses τ, the K-th largest inner product for q, to retrieve items relevant to q. Therefore, K should be a reasonable rank (e.g., $K = 100$), since each item in the search result of our problem is ranked in the top-K w.r.t. the K-MIPS problem for q. This is totally different from running k_i-MIPS on X_i for each category c_i such that $k_i \geq 1$. For example, assume that we run 1-MIPS on X_i (i.e., $k_i = 1$) and obtain x as a result. If $x \cdot q < \tau$, x cannot be a result in the categorical diversity-aware k-IPS problem. (Also, x is not relevant to q and is therefore not of interest to the user.)

[1] From Definition 2, it is clear that each vector x in the search result satisfies the threshold τ, but $x \cdot q$ may not be *maximum*. Therefore, our problem is k-IPS, not k-MIPS.

TABLE 2. Summary of notations.

Notation	Description
X	Set of n vectors
x	d-dimensional vector
q	Query vector
$x \cdot q$	Inner product of x and q
C	Set of categories in X
c_i	i-th category
X_i	Set of vectors belonging to c_i
k	Output size
k_i	Output size for c_i
K	Ranking threshold
τ	K-th largest inner product for q

Algorithm 1 BASELINE

Require: $X, q, k, \{k_1, \ldots, k_{|C|}\}$ such that $\sum_m k_i = k$, and K

```
τ ← K-MIPS by a state-of-the-art algorithm
C_q ← ∅
for each c_i such that k_i ≥ 1 do
    C_i ← ∅
    C_q ← C_q ∪ {C_i}
end for
for each x ∈ X such that x · q ≥ τ do
    if x ∈ X_i then
        C_i ← C_i ∪ {x}
    end if
end for
S ← ∅
for each C_i ∈ C_q do
    Add at most k_i vectors sampled from C_i to S
end for
return S
```

TABLE 2 summarizes the notations frequently used in this paper.

B. BASELINE ALGORITHM

One simple algorithm that solves the categorical diversity-aware k-IPS problem is to employ an existing state-of-the-art k-MIPS algorithm. Recall that $x \in X$ must have $x \cdot q \geq \tau$ to be included in the categorical diversity-aware k-IPS result. In addition, to obtain τ, we need to run K-MIPS. From this observation, we design a baseline algorithm.

Algorithm 1 describes the baseline algorithm. It first computes τ (the K-th largest inner product for q) by using a state-of-the-art k-MIPS algorithm (we employ FEXIPRO [20]). Then, it obtains a set C_i of vectors x such that $x \cdot q \geq \tau$ for each category c_i where $k_i \geq 1$. Last, at most k_i vectors are

randomly sampled from C_i, and these vectors are returned as the search result.

Although this algorithm does not lose correctness, it incurs $O(n)$ time, because the exact k-MIPS problem requires $O(n)$ time. Theoretically, this time means accessing (at most) all vectors in X, which is not desirable for the search problem. However, to obtain the correct result (i.e., to compute τ), we cannot avoid running a k-MIPS algorithm. This suggests that such an exact algorithm needs $O(n)$ time. Therefore, to address this inefficiency issue, we need to consider an approximation algorithm. It is usually the case that applications allow approximate results in order to improve the computational cost [13], [17], [24], [28], [29], [30]. We therefore devise an approximation algorithm that runs in sub-linear time to n in Section IV.

III. RELATED WORK

This section reviews existing k-MIPS and result diversification works.

A. EXACT MIPS AND ITS VARIANT

As mentioned in Section II-B, solving the k-MIPS problem exactly needs $O(n)$ time. Therefore, existing works developed heuristic algorithms that filter unnecessary vectors.

In [9], [19], and [26], tree-based data structures are considered to prune a subset of vectors that cannot be included in the k-MIPS answer. It is well known that tree-based data structures face the curse of dimensionality, thereby they do not function well on high-dimensional datasets. (This phenomenon has been empirically observed in [20] and [31].) Therefore, these approaches are not appropriate for recent machine-learning systems that employ dense high-dimensional vectors.

To remove this drawback, literature [20] proposed a linear-scan-based algorithm, FEXIPRO. This algorithm exploits an early termination strategy that stops the linear-scan early whenever unseen vectors cannot be the k-MIPS result. To stop the scan as early as possible, FEXIPRO employs some optimization techniques. We incorporate this algorithm into our baseline algorithm.

B. APPROXIMATE MIPS

To improve the efficiency of k-MIPS, approximation algorithms have also been considered. Approximate k-MIPS algorithms are roughly categorized into three techniques, LSH (locality-sensitive hashing), proximity graph, and quantization.

LSH-based algorithms [17], [25], [27], [28], [35] transform the k-MIPS problem into the nearest neighbor search problem in Euclidean space. Fast and accurate approximate nearest neighbor search (ANNS) techniques have also been extensively studied, and these LSH-based algorithms exploit the ANNS techniques. The main advantage of this LSH-based approach is error guarantee. These algorithms probabilistically guarantee the worst case error of their approximate k-MIPS results. However, to have this guarantee, LSH-based algorithms incur many data accesses, resulting in a large computational cost.

Proximity graph algorithms [23], [29], [38] are based on a greedy algorithm. In the pre-processing (offline) phase, these algorithms build a proximity graph, where each vertex of this proximity graph corresponds to a vector in X. If the inner product of x and x' is high, there is an edge between x and x'. The greedy algorithm assumes that, for a given query vector q, if $x \cdot q$ is high, each vector x' having an edge to x also has a high inner product with q. This algorithm traverses such vertices greedily. It is empirically observed that proximity graph algorithms provide fast computation time and high recall, which is the main advantage of this approach. However, this approach does not provide any theoretical guarantee.

Quantization-based algorithms [10], [13], [36] mainly consider space usage reduction. In this approach, each vector in X is quantized, i.e., transformed into a lower-dimensional vector. To minimize the lose derived from this quantization, state-of-the-art works [10], [13], [36] proposed optimized learning functions. Similar to proximity graph algorithms, this approach also has no theoretical error bound. In addition, this approach still incurs a linear time to n.

We do not consider approximate k-MIPS algorithms as competitors for the following reasons. Proximity graph and quantization-based algorithms do not have any theoretical performance guarantees, so the search result may be arbitrarily wrong. LSH-based algorithms can bound the worst-case error, but even state-of-the-art algorithms [17], [28] need $O(n \log n)$ time, which is slower than FEXIPRO.

C. RESULT DIVERSIFICATION

Recently, diversification has been getting more important, and many works in sub-fields of computer science dedicated to diversification. Also, industries (e.g., Airbnb and Spotify) try to display diversified results in their systems [1], [4]. There are many views about diversification, and the following three concepts are representative [18]: category, novelty, and content.

In category-based diversification, past studies, e.g., [21], [37], and [39], consider that the search results need to maximize the number of categories. Although we follow categorical diversity, we improve this past idea. Assume that we automatically maximize the number of categories in a search result for a given user. If the user is not interested in some categories in the result, this maximization does not make sense. Clearly, it is better for users to select their preferable categories, to maximize their satisfaction. Our problem is designed based on this idea. Note that the past studies cannot deal with our problem, since they proposed *only model-level* techniques and cannot guarantee that the search result contains categories in which a user is interested.

Novelty and content share a similar idea: a search result is diverse if it contains dissimilar items or these items are dissimilar to those checked by the user. To measure the dissimilarity between two vectors, past studies usually

employ distance [2], [11], [12], [15], [34]. This problem setting is totally different from ours, so these works cannot solve our problem.

IV. PROPOSED ALGORITHM

A. MAIN IDEA AND OVERVIEW

To devise a sub-linear time algorithm, we have to avoid (i) computing the exact τ and (ii) accessing unnecessary vectors. We achieve these based on two ideas. First, as a given query is interested only in categories c_i such that $k_i \geq 1$, accessing the vectors in X_j such that $k_j = 0$ is unnecessary, whereas the baseline algorithm may incur this access when computing τ. We hence partition X into disjoint subsets $X_1, \ldots,$ and $X_{|C|}$, in order not to access vectors belonging to unnecessary categories. We furthermore partition each subset X_i into disjoint buckets so that each bucket contains vectors x with high $x \cdot q$ for a given q. It can be seen that we can obtain such x by accessing only some buckets, not X. To enable this, we estimate τ and avoid computing its exact value. By using an estimated τ, we search for x in a bucket such that $x \cdot q \geq \tau$. It is clear that, if we can obtain an accurate τ, the accuracy of the search result becomes high. Below, we elaborate how to implement the above ideas.

1) DATASET PARTITION

Recall that $x \in X$ is a d-dimensional vector, so x can be represented as $x = \langle x[1], \ldots, x[d] \rangle$, where $x[i]$ is the value of the i-th dimension of x. Let $\|x\|$ be the Euclidean norm of x, i.e., $\|x\| = \sqrt{\sum_{i=1}^{d} x[i]^2}$. Furthermore, let M be the largest norm among X. For each $x \in X$, we transform x into a $(d+1)$-dimensional vector $\bar{x}$, and

$$\bar{x} = \langle \frac{x[1]}{M}, \ldots, \frac{x[d]}{M}, \frac{\sqrt{M^2 - \|x\|^2}}{M} \rangle. \tag{1}$$

Similarly, any query vector q is transformed into a $(d+1)$-dimensional vector $\bar{q}$ so that

$$\bar{q} = \langle \frac{q[1]}{\|q\|}, \ldots, \frac{q[d]}{\|q\|}, 0 \rangle. \tag{2}$$

We have

$$\bar{x} \cdot \bar{q} = \frac{x \cdot q}{\|q\| M}. \tag{3}$$

Therefore,

$$x \cdot q \geq \tau \Rightarrow \bar{x} \cdot \bar{q} \geq \frac{\tau}{\|q\| M}. \tag{4}$$

Notice that we have $\|\bar{x}\| = 1$ and $\|\bar{q}\| = 1$, so

$$\bar{x} \cdot \bar{q} = \cos(\bar{x}, \bar{q}). \tag{5}$$

Equations (4) and (5) demonstrate that the inner product search problem can be transformed into the cosine similarity search problem.

From this observation, we can employ LSH for cosine similarity [6]. In a nutshell, this approach partitions X into disjoint subsets, and each subset contains vectors such that any two vectors in the subset have a high cosine similarity (probabilistically). It is important to note that we do not employ LSH for inner product, because inner products can have infinite values whereas cosine similarity is bounded in $[-1, 1]$ (or $[0, 1]$ in our problem). This is an important observation for estimating $\frac{\tau}{\|q\| M}$, which is introduced later. An LSH for cosine similarity, $h(\bar{x})$, provides a binary value (0 or 1) based on:

$$h(\bar{x}) = \begin{cases} 1 & (a \cdot \bar{x} \geq 0) \\ 0 & (a \cdot \bar{x} < 0), \end{cases} \tag{6}$$

where a is a $(d+1)$-dimensional random vector, and the value of its each dimension follows the normal distribution $\mathcal{N}(0, 1)$. By using α hash functions, $h_1(\bar{x}), \ldots, h_\alpha(\bar{x})$, with different random seeds, the hash code of $\bar{x}$, $hc(\bar{x})$, is obtained as:

$$hc(\bar{x}) = \langle h_1(\bar{x}), \ldots, h_\alpha(\bar{x}) \rangle. \tag{7}$$

We use this hash code to partition X_i. Specifically, X_i is partitioned into disjoint buckets based on hash code, and a bucket is a set of vectors having the same hash code. A hash table T_i is a set of the disjoint buckets.

Example 3: FIGURE 1 illustrates our dataset partitioning approach. For simplicity, this figure describes how to partition X_1 as an example.

The above example shows the case where we have a single hash table for X_i. To increase the probability that a given q has a similar hash code to those of vectors with high inner product (or cosine similarity), we build β hash tables with different random seeds. This dataset partitioning is done offline, and Section IV-B presents our offline algorithm that builds hash tables. In Section IV-C, we show that this partitioning approach provides a sub-linear time algorithm and a probabilistic success guarantee.

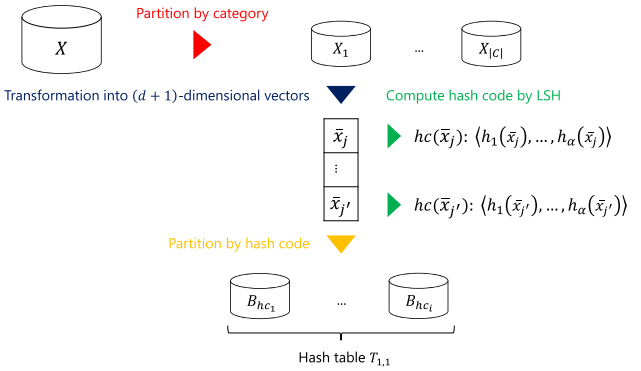


FIGURE 1. Overview of dataset partition. X_i is a set of vectors belonging to category c_i. After transforming each vector $x \in X_i$ into a $(d+1)$-dimensional vector $\bar{x}$, we compute a hash code for $\bar{x}$. X_i is partitioned into disjoint buckets, and each bucket contains vectors having the same hash code.

2) ESTIMATING τ

Let $\tau' = \frac{\tau}{\|q\| M}$. Assume that $k_i \geq 1$ and X_i has k_i vectors x such that $x \cdot q \geq \tau$. Then, Equation (4) demonstrates that there are k_i vectors $\bar{x}$ in the hash table T_i such that $\bar{x} \cdot \bar{q} \geq \tau'$. Let τ'_{est} be an estimated value of τ'. If $\tau'_{est} = \tau'$, we can

obtain the k_i vectors by looking into T_i. Having $\tau'_{est} = \tau'$ without running the MIPS problem is, unfortunately, difficult. However, we can obtain $\tau'_{est} > (1-\epsilon)\tau'$, where $\epsilon < 1$ is a small constant. We below demonstrate this.

Equations (4) and (5) show that we have

$$\frac{\tau}{\|q\|M} = \tau' \in [0, 1]. \tag{8}$$

Now let $\tau'_{est} = (1-\epsilon)^j$, and assume that using τ'_{est} cannot find k_i vectors in T_i such that $\bar{x} \cdot \bar{q} \geq \tau'_{est}$. In this case, it is trivial to see that $\tau'_{est} > \tau'$. Then, we set $\tau'_{est} = (1-\epsilon)^{j+1}$, and assume that using τ'_{est} finds k_i vectors in T_i such that $\bar{x} \cdot \bar{q} \geq \tau'_{est}$. In this case, we have $\tau'_{est} \leq \tau'$. To summarize, there is a non-negative integer j such that

$$(1-\epsilon)^{j+1} \leq \tau' < (1-\epsilon)^j. \tag{9}$$

From Equations (8) and (9), it is trivial that we can have $\tau'_{est} > (1-\epsilon)\tau'$. This approach may include x such that $x \cdot q < \tau$, but the error, i.e., $(1-\epsilon)$, is small. (In our experiments, we set $\epsilon = 0.01$.) Furthermore, to have $\tau'_{est} > (1-\epsilon)\tau'$, we need to set τ'_{est} iteratively (i.e., j is incremented by one multiple times). Since we have $\tau' \in [0, 1]$, the number of iterations is bounded by a constant.[2] This is also a merit of the transformation in Equation (1).

B. OFFLINE PROCESSING

Algorithm 2 describes our offline algorithm. First, for each $x \in X$, it computes its Euclidean norm $\|x\|$, and then obtains the largest norm M (lines 1–2). Next, it transforms each x into $\bar{x}$ (lines 3–10). Last, it builds β hash tables for each category in the manner presented in Section IV-A1, and $\mathcal{T}$ maintains all hash tables (lines 12–20). (Recall that FIGURE 1 also describes the steps in Algorithm 2.) It is important to note that this algorithm is conducted *only once* and the data structures built by this algorithm are general to any queries.

1) TIME COMPLEXITY

Hereinafter, we assume that $d = O(1)$ for simplicity. Computing the norm for every $x \in X$ needs $O(n)$ time, and M is also obtained in $O(n)$ time. The vector transformation needs $O(1)$ time for a given $x \in X$, thus its total time is also $O(n)$. Given a vector $\bar{x} \in \bar{X}$, its hash code is obtained in $O(\alpha)$ time. Since we have β hash tables, hash table construction needs $O(\beta \times \alpha \times n)$ time. Because β is a *fixed* hyper-parameter, we have $O(\alpha\beta n) = O(\alpha n)$. (We set α so that $2^\alpha = \text{polylog}(n)$, which is introduced in Section IV-C1.) Now it is clear that Algorithm 2 needs $O(\alpha n)$ time. In practice, α is a small constant, suggesting that $O(\alpha n) \approx O(n)$.

2) SPACE COMPLEXITY

To maintain the n transformed vectors, trivially $O(n)$ space is required. Because each hash table maintains n transformed vectors and we have β hash tables, the space complexity of the hash tables is $O(\beta n)$. Recall that $\beta = O(1)$, and the total space complexity is $O(n)$.

[2]This is trivial. Considering $(1-\epsilon)^j \approx 0$, we see that j is a constant.

Algorithm 2 OFFLINE-PROCESSING

Require: X, C, β (number of hash tables), and α (number of hash functions for each table)

1: Compute $\|x\|$ for every $x \in X$
2: $M \leftarrow \max\{\|x_1\|, \ldots, \|x_n\|\}$
3: $\bar{X} \leftarrow \varnothing$
4: **for** each category $c_i \in C$ **do**
5: $\quad \bar{X}_i \leftarrow \varnothing$
6: $\quad$ **for** each $x \in X_i$ **do**
7: $\quad\quad$ Transform x into $\bar{x}$
8: $\quad\quad \bar{X}_i \leftarrow \bar{X}_i \cup \{\bar{x}\}$
9: $\quad$ **end for**
10: $\quad \bar{X} \leftarrow \bar{X}_i$
11: **end for**
12: $\mathcal{T} \leftarrow \varnothing$ // $\mathcal{T}$ is a set of hash tables
13: **for** $i = 1$ to $|C|$ **do**
14: $\quad T_i \leftarrow \varnothing$ // T_i will maintain β hash tables for $\bar{X}_i$
15: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{T_i\}$
16: **end for**
17: **for** each $\bar{X}_i \in \bar{X}$ **do**
18: $\quad$ **for** $j = 1$ to β **do**
19: $\quad\quad$ Build a hash table $T_{i,j}$ for $\bar{X}_i$
20: $\quad\quad T_i \leftarrow T_i \cup \{T_{i,j}\}$
21: $\quad$ **end for**
22: **end for**

Algorithm 3 ONLINE-PROCESSING

Require: $X, q, k, \{k_1, \ldots, k_{|C|}\}$ such that $\sum_m k_i = k$, K, and $\mathcal{T}$

1: Transform q into $\bar{q}$
2: Compute β hash codes of $\bar{q}$
3: $S \leftarrow \varnothing$ // S is a result set
4: **for** each c_i such that $k_i \geq 1$ **do**
5: $\quad S_i \leftarrow \text{Retrieval}(\mathcal{T}, \bar{q}, k_i, \epsilon)$
6: $\quad S \leftarrow S \cup \{S_i\}$
7: **end for**
8: **return** S

C. ONLINE PROCESSING

Given a query vector q, k, $\{k_1, \ldots, k_{|C|}\}$, and K, we run our algorithm described in Algorithm 3. It first transforms q into $\bar{q}$ based on Equation (2), and then computes a hash code of $\bar{q}$ for each hash table. After that, for each category c_i such that $k_i \geq 1$, it runs Algorithm 4 that returns S_i, which is a set of vectors $x \in X_i$ such that $\bar{x} \cdot \bar{q} \geq \tau'_{est}$. Finally, Algorithm 3 returns S, which is a merged set of $S_1, \ldots, S_{|C|}$.

Here, we present Algorithm 4 by focusing on a category c_i such that $k_i \geq 1$. Recall that the hash codes are binary vectors, as can be seen from Equation (6). Then, the distance between two hash codes can be obtained by the Hamming distance. Since similar vectors have similar hash codes, buckets with small Hamming distances to the hash codes of a given query vector would contain vectors $\bar{x}$ with high $\cos(\bar{x}, \bar{q})$. Motivated by this, for each hash table, we compute $O(\log n)$ buckets with

Algorithm 4 RETRIEVAL($\mathcal{T}, \bar{q}, k_i, \epsilon$)

1: $X' \leftarrow \varnothing$
2: **for** each $j \in [1, \beta]$ **do**
3: $\mathcal{B}_j \leftarrow O(\log n)$ buckets with the minimum Hamming distance to the hash code of $\bar{q}$ in $T_{i,j}$
4: $X' \leftarrow X' \cup \mathcal{B}_j$
5: **end for**
6: $S_i \leftarrow \varnothing$ // S_i is a result set in this category
7: $\tau'_{est} \leftarrow 1$
8: **while** $\tau'_{est} > \gamma$ **do**
9: **for** each $\bar{x} \in X'$ such that $\bar{x} \cdot \bar{q} \geq \tau'_{est}$ and $x \notin S_i$ **do**
10: $S_i \leftarrow S_i \cup \{x\}$
11: **if** $|S_i| = k_i$ **then**
12: **break**
13: **end if**
14: **end for**
15: **if** $|S_i| = k_i$ **then**
16: **break**
17: **end if**
18: $\tau'_{est} \leftarrow (1-\epsilon)\tau'_{est}$
19: **end while**
20: **return** S_i

the minimum Hamming distance to the hash code of $\bar{q}$, and the vectors in the buckets are maintained by X'. Next, we set $\tau'_{est} = 1$ and search for vectors $\bar{x} \in X'$ such that $\bar{x} \cdot \bar{q} \geq \tau'_{est}$. Such vectors are maintained in S_i. If $|S_i| < k_i$, we update τ'_{est} as in line 18. Algorithm 4 iterates the same operation until $|S_i| = k_i$ or $\tau'_{est} \leq \gamma$, where γ is a hyper-parameter for avoiding redundant iterations.[3]

1) TIME COMPLEXITY

We analyze the time complexity of Algorithm 3 to demonstrate the efficiency of our algorithm. Computing $\bar{q}$ and its hash codes need $O(1)$ and $O(\alpha)$ times, respectively. The main bottleneck of Algorithm 3 is attributed to line 5. Let the time of line 5 be $G(n)$, and the time complexity of Algorithm 3 is $O(\alpha + |C|G(n))$. If α and $G(n))$ are sub-linear to n, we see that $O(\alpha + |C|G(n)) < O(n)$. We below demonstrate that α and $G(n))$ are sub-linear to n.

Assume that $k_i \geq 1$. First, consider the retrieval of $O(\log n)$ buckets with the minimum Hamming distance to the hash code of $\bar{q}$ in $T_{i,j}$. Notice that there are at most 2^α buckets, as the hash codes are α-dimensional binary vectors. By setting α so that $2^\alpha = \text{polylog}(n)$, the number of buckets in a hash table is at most sub-linear to n. (Note that polylog(n) is always smaller than $O(n)$.) This operation therefore needs polylog(n) time. (Recall that the number of hash tables is constant.) In addition, α has to be (much) smaller than $O(n)$, because $\alpha = O(n)$ cannot satisfy the setting of $2^\alpha = \text{polylog}(n)$. We next consider the number of vectors in a bucket. We set α so that this number also follows polylog(n). (In practice, we can obtain such α for large n, and α is often small.[4]) As polylog$(n) \times O(\log n) = \text{polylog}(n)$, $|X'| = \text{polylog}(n)$. Recall that Algorithm 3 scans X' and the number of iterations in Algorithm 4 is constant (see Section IV-A2). We hence have $G(n) = \text{polylog}(n)$, which demonstrates that Algorithm 3 is at most sub-linear to n.

2) SUCCESS PROBABILITY ANALYSIS

Next, we analyze the success probability of our algorithm to understand the high accuracy of algorithm in practice. According to [6], the probability that two vectors $\bar{x}$ and $\bar{x'}$ have $h(\bar{x}) = h(\bar{x'})$, $\Pr[h(\bar{x}) = h(x')]$, is

$$Pr[h(\bar{x}) = h(\bar{x'})] = 1 - \frac{\cos^{-1}(\theta)}{\pi}, \tag{10}$$

where $\theta = \bar{x} \cdot \bar{x'}$. Consider a set P of positive integers p such that $x \in X_i$ is the top p-th vector in the MIPS problem for q. From Section IV-A2, it is trivial to see that $\bar{x} \cdot \bar{q} > (1-\epsilon)\tau'$. For ease of presentation, let $\theta_p = \bar{x}_p \cdot \bar{q}$, where x_p is the top p-th vector in the MIPS problem for q. From Equation (10), the probability that $\bar{q}$ and $\bar{x}_p$ have the same hash code, $\Pr[hc(\bar{q}) = hc(\bar{x}_p)]$, is

$$Pr[hc(\bar{q}) = hc(\bar{x}_p)] = (1 - \frac{\cos^{-1}(\theta_p)}{\pi})^\alpha. \tag{11}$$

Then, we can obtain $\Pr(\theta_p, m)$, namely the probability that the Hamming distance between $hc(\bar{q})$ and $hc(\bar{x}_p)$ is m:

$$Pr(\theta_p, m) = \binom{\alpha}{m}(1 - \frac{\cos^{-1}(\theta_p)}{\pi})^{\alpha-m}(\frac{\cos^{-1}(\theta_p)}{\pi})^m. \tag{12}$$

Given a hash table $T_{i,j}$, we assume that the Hamming distances between the hash codes of the $O(\log n)$ buckets and the query are in $[hd^l_{i,j}, hd^u_{i,j}]$. The probability $\Pr(\theta_p, T_{i,j})$ that $\bar{x}_p$ is in the $O(\log n)$ buckets is:

$$Pr(\theta_p, T_{i,j}) = \sum_{m=hd^l_{i,j}}^{hd^u_{i,j}} Pr(\theta_p, m). \tag{13}$$

Now we see that x_p exists in X', i.e., the $O(\log n) \times \beta$ buckets of the β hash tables $T_{i,1}, \ldots, T_{i,\beta}$, with probability at least

$$Pr(x_p \in X') = 1 - \Pi_{j=1}^{\beta}(1 - Pr(\theta_p, T_{i,j})). \tag{14}$$

That is, for each category c_i such that $k_i \geq 1$, Algorithm 3 can obtain the top p-th vector $x \in X_i$ with probability at least $1 - \Pi_{j=1}^{\beta}(1 - Pr(\theta_p, T_{i,j}))$. Last, let I be a set of positive integers p such that $x \in X_i$ is the top p-th vector in the MIPS problem for q and "$p \leq K$". Assume that $|I| \geq k_i$. The probability that X' contains k_i vectors such that $\bar{x} \cdot \bar{q} \geq \tau'$ is

$$\binom{|I|}{k_i} \Pi_{p \in I'} Pr(x_p \in X'), \tag{15}$$

where I' is a subset of I such that $|I'| = k_i$. Therefore, the probability that X' contains *at least* k_i vectors such

[3] Because $\tau'_{est} > 0$, we need to terminate the iteration at least when $\tau'_{est} \approx 0$.

[4] In our experiments, α is only 6.

that $\bar{x} \cdot \bar{q} \geq \tau'$ is

$$\sum_{k'=k_i}^{|I|} \binom{|I|}{k'} \Pi_{p \in I'} Pr(x_p \in X'). \quad (16)$$

In the next section, we confirm that the above probability is high by investigating the accuracy of the search result returned by Algorithm 3.

V. EMPIRICAL STUDY

We report our empirical results. All experiments were conducted on a Ubuntu 16.04 LTS machine with 128GB RAM and Core i9-9980XE CPU@3.0GHz. We evaluated the baseline algorithm introduced in Section II-B and our algorithm[5] proposed in Section IV. (Recall that this is the first work to address the problem of categorical diversity-aware k-IPS, so no existing works can solve this problem.) These algorithms were implemented in C++, complied by g++ 5.5.0 with -O3 optimization, and single threaded.

A. EXPERIMENTAL SETTING

1) DATASETS

We used the following three real ratings datasets.

- Amazon-Kindle [14]: A rating dataset for books in Amazon, containing 430,530 items.
- Amazon-Movie [14]: A rating dataset for movies in Amazon, containing 200,941 items.
- MovieLens[6]: This is the MovieLens 25M dataset, and the number of items is 59,047.

We used Matrix Factorization [8] to generate query (user) vectors and item vectors in an inner product space. The dimensionality of each vector was 200. Unfortunately, Amazon-Kindle and Amazon-Movie do not have categorical information. We hence added a category for each item in these datasets uniformly at random, and the total number of categories was 25. For MovieLens, we used the original category.[7]

2) DEFAULT PARAMETERS

We set $K = 100$ and $k = 10$ by default. Given a query vector q, it specified random three categories, and $k_i = k/3$. When we investigated the impact of a given parameter, the other parameters were fixed. For our algorithm, we set $\epsilon = 0.01$, $\alpha = 6$, and $\beta = 3$.

3) CRITERIA

We measured the running time and accuracy of the baseline and our algorithms. The accuracy, Acc, is defined below. Given a query q, the accuracy of its search result is:

$$Acc = \frac{t}{\sum \min\{k_i, l_i\}},$$

[5]The code is available at our GitHub repository: https://github.com/peitaw22/Categorical-Diversity-Aware-k-IPS

[6]https://grouplens.org/datasets/movielens/

[7]For items having multiple categories, we chose a random one.

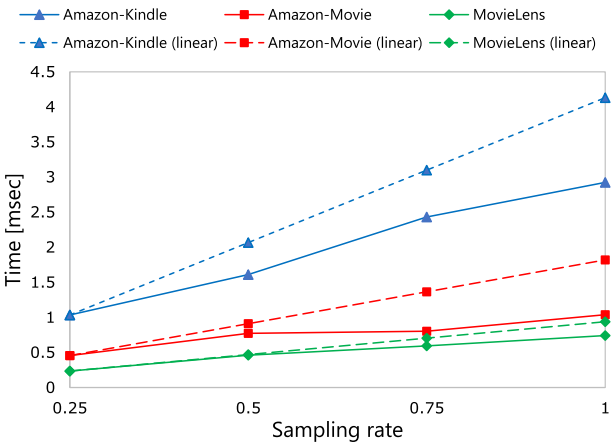


FIGURE 2. Scalability test. Dashed lines show the case of linear scalablity, while solid lines show the performance of our algorithm. The result demonstrates that our algorithm is sub-linear to data size, because its scalablity is better than the linear case.

where t is the number of vectors $x \in X$ in the search result such that $x \cdot q \geq \tau$ and l_i is the number of vectors in X_i satisfying $x \cdot q \geq \tau$. Notice that there are less than k_i vectors in X_i satisfying $x \cdot q \geq \tau$, so $\min\{k_i, l_i\}$ is necessary to have $Acc \in [0, 1]$ for any query. Because our algorithm does not provide false positives, other criteria related to precision cannot be measured.

We used 100 random queries to evaluate the performance of each algorithm. We report the average and median of Acc.

B. EXPERIMENTAL RESULT

1) OFFLINE TIME

We first report empirical offline time of our algorithm (i.e., practical time of Algorithm 2). On Amazon-Kindle, Amazon-Movie, and MovieLens, the offline time was 1.41, 3.05, and 0.41 [sec], respectively. Recall that the offline processing is done only once, and the result confirms that its practical time is short enough.

2) SCALABILITY TEST

We next show that our algorithm scales sub-linearly to n. FIGURE 2 shows the results of our experiments that vary data size by random sampling. Specifically, the solid lines show the results of our algorithm, whereas the doted lines show the cases of linear time. We observe that our algorithm scales better than the linear case, which confirms the consistency with our theoretical analysis in Section IV-C1.

3) IMPACT OF RANKING THRESHOLD K

TABLEs 3–5 show the results of the experiments with varying K. Recall that the baseline is an exact algorithm, so its Acc is always 1. Our algorithm returns an approximate result, but its accuracy is generally high and nearly perfect on the three datasets. We see that, on Amazon-Kindle and MovieLens, the median Acc is 1.00, demonstrating that at least 50% of queries obtained the exact result. In addition, this high accuracy is not affected by K. This is because

TABLE 3. Impact of K on average accuracy, median accuracy, and time [msec] on MovieLens.

K	50			100			150			200		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	9.81	1.00	1.00	12.80	1.00	1.00	14.99	1.00	1.00	16.77
Ours	0.95	1.00	2.93	0.95	1.00	2.92	0.95	1.00	2.93	0.98	1.00	2.92

TABLE 4. Impact of K on average accuracy, median accuracy, and time [msec] on Amazon-Movie.

K	50			100			150			200		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	7.48	1.00	1.00	9.91	1.00	1.00	11.48	1.00	1.00	12.96
Ours	0.88	1.00	1.03	0.87	0.90	1.04	0.92	1.00	1.04	0.94	1.00	1.04

TABLE 5. Impact of K on average accuracy, median accuracy, and time [msec] on MovieLens.

K	50			100			150			200		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	1.34	1.00	1.00	2.06	1.00	1.00	2.68	1.00	1.00	3.15
Ours	0.98	1.00	0.74	0.98	1.00	0.74	0.98	1.00	0.74	0.98	1.00	0.74

TABLE 6. Impact of k on average accuracy, median accuracy, and time [msec] on Amazon-Kindle.

k	5			10			15			20		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	12.95	1.00	1.00	12.80	1.00	1.00	12.73	1.00	1.00	12.99
Ours	0.98	1.00	2.88	0.95	1.00	2.92	0.94	1.00	2.94	0.95	1.00	2.96

TABLE 7. Impact of k on average accuracy, median accuracy, and time [msec] on Amazon-Movie.

k	5			10			15			20		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	9.88	1.00	1.00	9.91	1.00	1.00	9.89	1.00	1.00	9.81
Ours	0.94	1.00	1.01	0.87	0.90	1.04	0.83	0.88	1.05	0.82	0.88	1.06

TABLE 8. Impact of k on average accuracy, median accuracy, and time [msec] on MovieLens.

k	5			10			15			20		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	2.06	1.00	1.00	2.06	1.00	1.00	2.06	1.00	1.00	2.06
Ours	0.98	1.00	0.71	0.98	1.00	0.74	0.97	1.00	0.75	0.96	1.00	0.76

our algorithm employs a threshold estimation approach that works for arbitrary ranking.

In terms of running time, the baseline algorithm needs longer time as K increases. This is a reasonable result, since it runs the k-MIPS algorithm (FEXIPRO [20]) to obtain τ, and it needs a longer time for a larger output size (K in this case). On the other hand, the running time of our algorithm does not change even if K increases. This is attributed to the threshold estimation approach and the fact that the size of each LSH bucket is independent of K. It is important to note that our algorithm is clearly faster than the baseline algorithm. For example, on Amazon-Kindle, Amazon-Movie, and MovieLens, our algorithm is respectively 4.2x, 9.5x, and 2.8x faster than the baseline at the default parameter setting. Since our algorithm yields an accurate result, it provides a good trade-off between time and accuracy.

4) IMPACT OF OUTPUT SIZE k

We next study the impact of output size k, and TABLEs 6–8 show the results. We see that the performance of the baseline algorithm is stable. This is trivial, since its computational bottleneck is to compute τ, which is independent of k.

Similarly, the performance of our algorithm is also not affected by k much. For a fixed ranking threshold K, τ is also fixed. Our algorithm accesses vectors in each corresponding bucket until it finds k (or k_i) vectors. Therefore, as k increases, its running time becomes longer (only slightly). As for its accuracy, our algorithm keeps high Acc, as with the case of the experimental result w.r.t. K.

5) IMPACT OF #CATEGORIES SPECIFIED

Last, we investigate the influence of the number of categories specified in TABLEs 9–11. Again, the performance of the baseline algorithm is stable, since its bottleneck is irrelevant to the number of categories specified. Different from this observation, the running time of our algorithm is affected by the number of categories specified. This is because our algorithm accesses the LSH buckets for each category specified. That is, the running time of our algorithm is (almost) linear to the number of categories specified, as is theoretically confirmed in Section IV-C1. On the other hand,

TABLE 9. Impact of #categories on average accuracy, median accuracy, and time [msec] on Amazon-Kindle.

#categories	2			3			4		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	12.84	1.00	1.00	12.80	1.00	1.00	12.87
Ours	0.93	1.00	1.98	0.95	1.00	2.92	0.97	1.00	3.86

TABLE 10. Impact of #categories on average accuracy, median accuracy, and time [msec] on Amazon-Movie.

#categories	2			3			4		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	9.77	1.00	1.00	9.91	1.00	1.00	9.81
Ours	0.86	0.88	0.71	0.87	0.90	1.04	0.92	1.00	1.33

TABLE 11. Impact of #categories on average accuracy, median accuracy, and time [msec] on MovieLens.

#categories	2			3			4		
	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time	Acc_{avg}	Acc_{med}	Time
Baseline	1.00	1.00	2.06	1.00	1.00	2.06	1.00	1.00	2.07
Ours	0.97	1.00	0.55	0.98	1.00	0.74	0.97	1.00	1.00

the accuracy of our algorithm is almost not affected by the number of categories specified, and our algorithm keeps almost perfect accuracy.

VI. CONCLUSION

Because of the recent widespread application of machine-learning, many objects are represented as dense high-dimensional vectors in inner product space. In such machine-learning systems, the maximum inner product search (MIPS) problem plays an important role. However, the standard MIPS may not yield a preferable result, because it does not care about any diversification. We therefore consider categorical diversity and formulate a new problem, namely the categorical diversity-aware k-IPS problem. In this problem, users can select their preferable categories, and the results displayed must be presented in a user-specified rank. We show that exactly solving this problem incurs $O(n)$ time, where n is the dataset size. From this fact, we proposed a sub-linear time approximation algorithm that has a probabilistic performance guarantee. We conducted extensive experiments using real datasets. Our experimental results demonstrate the high efficiency and accuracy of our algorithm.

Our proposed algorithm is not deterministic w.r.t. approximation guarantee, so developing a deterministic approach is an interesting future work. In addition, although this paper focused on categorical diversity, there are other definitions of diversity, as discussed in Section III-C. To deal with many scenarios, a comprehensive algorithmic framework for different diversity definitions is desirable. Developing such a framework is an open problem.

REFERENCES

[1] M. Abdool, M. Haldar, P. Ramanathan, T. Sax, L. Zhang, A. Manaswala, L. Yang, B. Turnbull, Q. Zhang, and T. Legrand, "Managing diversity in Airbnb search," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 2952–2960.

[2] D. Amagata and T. Hara, "Diversified set monitoring over distributed data streams," in *Proc. 10th ACM Int. Conf. Distrib. Event-based Syst.*, Jun. 2016, pp. 1–12.

[3] D. Amagata and T. Hara, "Reverse maximum inner product search: How to efficiently find users who would like to buy my item?" in *Proc. 15th ACM Conf. Recommender Syst.*, Sep. 2021, pp. 273–281.

[4] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas, "Algorithmic effects on the diversity of consumption on spotify," in *Proc. Web Conf.*, Apr. 2020, pp. 2155–2165.

[5] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet, "Speeding up the xbox recommender system using a Euclidean transformation for inner-product spaces," in *Proc. 8th ACM Conf. Recommender Syst. (RecSys)*, 2014, pp. 257–264.

[6] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. 34th Annu. ACM Symp. Theory Comput. (STOC)*, 2002, pp. 380–388.

[7] P. Cheng, S. Wang, J. Ma, J. Sun, and H. Xiong, "Learning to recommend accurate and diverse items," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 183–192.

[8] W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin, "LIBMF: A library for parallel matrix factorization in shared-memory systems," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2971–2975, 2016.

[9] R. R. Curtin, P. Ram, and A. G. Gray, "Fast exact max-kernel search," in *Proc. SIAM Int. Conf. Data Mining*, May 2013, pp. 1–9.

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

[11] M. Drosou and E. Pitoura, "Multiple radii DisC diversity: Result diversification based on dissimilarity and coverage," *ACM Trans. Database Syst.*, vol. 40, no. 1, pp. 1–43, Mar. 2015.

[12] X. Ge and P. K. Chrysanthis, "Efficient PrefDiv algorithms for effective top-k result diversification," in *Proc. EDBT*, 2020, pp. 335–346.

[13] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar, "Accelerating large-scale inference with anisotropic vector quantization," in *Proc. ICML*, 2020, pp. 3887–3896.

[14] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, Apr. 2016, pp. 507–517.

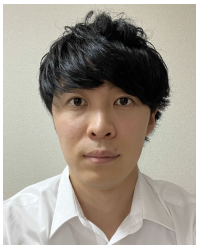
[15] K. Hirata, D. Amagata, S. Fujita, and T. Hara, "Solving diversity-aware maximum inner product search efficiently and effectively," in *Proc. 16th ACM Conf. Recommender Syst.*, Sep. 2022, pp. 198–207.

[16] K. Hirata, D. Amagata, and T. Hara, "Cardinality estimation in inner product space," *IEEE Open J. Comput. Soc.*, vol. 3, pp. 208–216, 2022.

[17] Q. Huang, G. Ma, J. Feng, Q. Fang, and A. K. H. Tung, "Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1561–1570.

[18] M. Kaminskas and D. Bridge, "Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems," *ACM Trans. Interact. Intell. Syst.*, vol. 7, no. 1, pp. 1–42, Mar. 2016.

[19] N. Koenigstein, P. Ram, and Y. Shavitt, "Efficient retrieval of recommendations in a matrix factorization framework," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2012, pp. 535–544.
[20] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis, "FEXIPRO: Fast and exact inner product retrieval in recommender systems," in *Proc. ACM Int. Conf. Manage. Data*, May 2017, pp. 835–850.
[21] L. Li and C.-Y. Chan, "Efficient indexing for diverse query results," *Proc. VLDB Endowment*, vol. 6, no. 9, pp. 745–756, Jul. 2013.
[22] Z. Li, D. Amagata, Y. Zhang, T. Maekawa, T. Hara, K. Yonekawa, and M. Kurokawa, "HML4Rec: Hierarchical meta-learning for cold-start recommendation in flash sale e-commerce," *Knowl.-Based Syst.*, vol. 255, Nov. 2022, Art. no. 109674.
[23] S. Morozov and A. Babenko, "Non-metric similarity graphs for maximum inner product search," in *Proc. NIPS*, 2018, pp. 4721–4730.
[24] H. Nakama, D. Amagata, and T. Hara, "Approximate top-k inner product join with a proximity graph," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 4468–4471.
[25] B. Neyshabur and N. Srebro, "On symmetric and asymmetric LSHs for inner product search," in *Proc. ICML*, 2015, pp. 1926–1934.
[26] P. Ram and A. G. Gray, "Maximum inner-product search using cone trees," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2012, pp. 931–939.
[27] A. Shrivastava and P. Li, "Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS)," in *Proc. NIPS*, 2014, pp. 2321–2329.
[28] Y. Song, Y. Gu, R. Zhang, and G. Yu, "ProMIPS: Efficient high-dimensional C-approximate maximum inner product search with a lightweight index," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, Apr. 2021, pp. 1619–1630.
[29] S. Tan, Z. Xu, W. Zhao, H. Fei, Z. Zhou, and P. Li, "Norm adjusted proximity graph for fast inner product retrieval," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1552–1560.
[30] C. Teflioudi and R. Gemulla, "Exact and approximate maximum inner product search with LEMP," *ACM Trans. Database Syst.*, vol. 42, no. 1, pp. 1–49, Mar. 2016.
[31] C. Teflioudi, R. Gemulla, and O. Mykytiuk, "LEMP: Fast retrieval of large entries in a matrix product," in *Proc. SIGMOD*, 2015, pp. 107–122.
[32] 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 *Proc. Companion Proc. World Wide Web Conf.*, May 2019, pp. 398–403.
[33] 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 J. Ind. Electron. Soc.*, vol. 1, pp. 194–206, 2020.
[34] L. Xia, J. Xu, Y. Lan, J. Guo, and X. Cheng, "Learning maximal marginal relevance model via directly optimizing diversity evaluation measures," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2015, pp. 113–122.
[35] X. Yan, J. Li, X. Dai, H. Chen, and J. Cheng, "Norm-ranging LSH for maximum inner product search," in *Proc. NIPS*, 2018, pp. 2952–2961.
[36] J. Zhang, Q. Liu, D. Lian, Z. Liu, L. Wu, and E. Chen, "Anisotropic additive quantization for fast inner product search," in *Proc. AAAI*, 2022, pp. 4354–4362.
[37] Y. Zheng, C. Gao, L. Chen, D. Jin, and Y. Li, "DGCN: Diversified recommendation with graph convolutional networks," in *Proc. Web Conf.*, Apr. 2021, pp. 401–412.
[38] Z. Zhou, S. Tan, Z. Xu, and P. Li, "Möbius transformation for fast inner product search on graph," in *Proc. NIPS*, 2019, pp. 8216–8227.
[39] Y. Zhu, Y. Lan, J. Guo, X. Cheng, and S. Niu, "Learning for search result diversification," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2014, pp. 293–302.



KOHEI HIRATA is currently pursuing the master's degree with the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, Osaka, Japan. His research interest includes high-dimensional data retrieval.

DAICHI AMAGATA (Member, IEEE) received the B.E., M.Sc., and Ph.D. degrees from Osaka University, Osaka, Japan, in 2012, 2014, and 2015, respectively. He is currently an Assistant Professor with the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University. His research interests include fast algorithms for databases and AI technologies.

SUMIO FUJITA received the D.E.A. degree from the Université de Paris 7, in 1989. He worked at Computer Institute of Japan Ltd., Kanagawa, Japan, from 1985 to 1995. He worked as a Research Associate at The University of Manchester Institute of Science and Technology, from 1993 to 1994, a Chief Researcher at Justsystem Corporation, Tokushima, Japan, from 1995 to 2002, a Research Scientist at Claritech Corporation, Pittsburgh, PA, USA, in 1998, and a Senior Research Scientist at Patolis Corporation, Tokyo, Japan, from 2002 to 2004. He joined Yahoo Japan Corporation, Tokyo, in 2005, participated in the foundation of Yahoo Japan Research, in 2007, and worked as a Senior Chief Researcher. He currently works as a Project Researcher on information retrieval, web mining, and related areas at Yahoo Japan Research. He is a member of ACM, SIGIR, and IPSJ.

TAKAHIRO HARA (Senior Member, IEEE) received the B.E., M.E., and Dr.E. degrees in information systems engineering from Osaka University, Osaka, Japan, in 1995, 1997, and 2000, respectively. Currently, he is a Full Professor at the Department of Multimedia Engineering, Osaka University. His research interests include distributed databases, peer-to-peer systems, mobile networks, and mobile computing systems. He is a Distinguished Scientist of ACM and a member of three other learned societies.