

面向类别过滤近似最近邻搜索的高效索引

王紫琪¹, 张景哲¹, 胡伟^{1, 2+}

1. 计算机软件新技术全国重点实验室 南京大学, 江苏省 南京市 210023

2. 国家健康医疗大数据研究院 南京大学, 江苏省 南京市 210023

+ 通信作者 E-mail: whu@nju.edu.cn, Phn: +86-189-5199-1205, <http://ws.nju.edu.cn/wiki/Wiki.jsp?page=%E8%83%A1%E4%BC%9F>

摘要: 向量近似最近邻搜索已成为当今智能检索系统中重要的基础功能之一。然而仅能反映原始数据语义的单一向量搜索已无法满足丰富的现实查询需求。类别过滤近似最近邻查询作为一种新的查询方式能够允许用户约束向量的所属类别, 查询结果包含与查询向量同类或为其子类的最相似向量。已有向量索引未能针对该过滤条件优化, 所以无法达到最优查询性能。提出了一种高效解决类别过滤向量查询的近邻图索引结构, 并理论证明了其具有基于近邻图的过滤索引中最优的查询表现。还通过压缩索引的邻居列表在略微降低查询性能的情况下大幅提升了索引构建性能。在多个向量和类别数据集上的实验表明: (1) 具有理论上最优查询性能的索引相比最优对比方法取得了5倍查询性能提升; (2) 压缩邻居列表后索引构建速度提升5~7倍, 相比最优对比方法具有相近或更快的索引构建速度并取得3~5倍查询性能提升。

关键词: 近似最近邻搜索; 近邻图; 过滤向量搜索; 查询优化

文献标志码: A **中图分类号:** TP***

Efficient Index for Category Filtering Approximate Nearest Neighbor Search

WANG Zi-Qi¹, ZHANG Jing-Zhe¹, HU Wei^{1, 2+}

1. State Key Laboratory for Novel Software Technology, Nanjing University, 210023, China

2. National Institute of Healthcare Data Science

Abstract: Approximate nearest neighbor search (ANNS) has emerged as a fundamental component in numerous intelligent retrieval systems. However, with the diverse requirements of real-world semantic queries, relying solely on single-vector queries proves insufficient. As a novel form of filtering-based vector retrieval, category-filtering approximate nearest neighbor search (CFANNS) enables users to retrieve similar vectors whose associated category matches or is a sub-category of the target category. Existing filtering indices are ill-suited for this scenario, failing to achieve optimal query performance. This paper introduces an efficient graph-based index tailored for CFANNS and theoretically proves its optimality in query performance. Additionally, a neighbor list compression technique is proposed, sacrificing minimal query efficiency to significantly enhance index construction speed. Experiments across multiple vector and category datasets demonstrate that: (1)

基金项目: 基金中文完整名称 (*****编号); 基金中文完整名称 (*****编号); 基金中文完整名称 (*****编号)。

This work was supported by the **** Foundation/Program/Project of China (***** No.), the **** Foundation/Program/Project of China (***** No.), and the **** Foundation/Program/Project of China (***** No.).

收稿日期: 2000-00-00 **修回日期:** 2000-00-00

The theoretically optimal index achieves 5x acceleration in query speed compared to the most competitive baseline; (2) After neighbor compression, the index attains 5–7x faster construction than the original one, and 3–5x faster querying than the most competitive baseline, with equivalent or superior construction efficiency.

Key words: approximate nearest neighbor search; proximity graph; filtering vector search; query optimization

随着大模型对数据语义的理解逐渐增强, 基于向量相似度的查询成为继结构化查询后又一个广泛使用的查询方式^[9]。例如, 在检索增强生成(Retrieval-Augmented Generation, RAG)场景中, 知识库中保存了经过向量编码的文本块, 大模型需要将查询文本编码成向量, 并通过向量相似性从中抽取出最相似的 k 个文本作为生成答案的知识源。然而, 精确抽取最相似的向量需要逐个比较查询向量与数据集中向量的相似度^[16]。通常经过编码的向量维度从几百维到几千维不等^[30], 随着数据集规模增大, 精确的最近邻搜索无法满足线上毫秒级别的查询速度需求, 并且大多数应用只需找到大部分邻居向量而不是所有近邻^{[13][16][21][22]}。为了适配线上查询需求, 近似最近邻搜索(Approximate Nearest Neighbor Search, ANNS)允许查找结果中存在少数非最近的邻居, 通过专用索引结构加快查询速度, 这些索引结构又可以分为基于哈希的索引^{[7][18]}, 基于分组的索引^{[10][14]}以及基于图的索引^{[1][8][23][24][25][26][28]}。

简单的向量相似度搜索难以应对日益复杂的现实查询需求^[16]。一类典型的场景是向量结合结构化数据的过滤搜索^{[19][20]}。在该场景下, 数据库中的向量通常附带一个或多个属性, 相应的查询语句包含一个查询向量以及在这些属性上的过滤条件。向量检索引擎需要从数据库中找到满足过滤条件且距离查询向量最近的 k 个向量。本文聚焦于一类特殊过滤查询, 其中数据集中每个向量有且仅有一个表示类别的属性, 并且数据集中的所有类别能够组成树型或有向无环图(Directed Acyclic Graph, DAG)型包含关系。给定一个查询类别与查询向量, 本文的目标是从数据集中找到与查询向量最相似的 k 个向量并且该向量属于查询类

别或属于其子类别, 该问题被称为类别过滤近似最近邻搜索(Category Filtering Approximate Nearest Neighbor Search, CFANNS)。

图1展示了两个类别过滤近似最近邻搜索的应用场景。例1, 知识库中的大量文本文件存放在不同的文件夹中。这些文本文件都经过某个编码模型转换为维度相同的向量。用户查询也被相同的编码模型转换成同维向量。与传统根据文本相似度抽取最相关文本

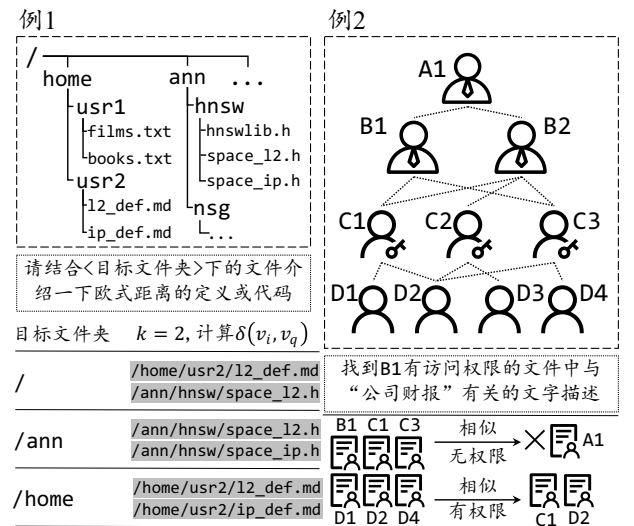


图1 类别过滤近似最近邻搜索的应用举例

Fig.1 Examples of category-filtering approximate nearest neighbor search

的场景区别在于, 用户可能只希望抽取某个文件夹中包含的文本, 因为这些文件夹可能包含相似的主题或用户偏好。例如, 用户希望学习欧式距离的含义来编写高效计算的代码, 则提出的带有类别过滤的问题可以通过显式提供目标文件夹或使用一个智能体决策出一个目标文件夹, 使抽取出来的文本只在目标文件夹之下。假如更希望得到高效的代码实现参考, 则可能倾向于从向量检索 C++库的文件夹“/ann”下抽取相似代码块。假如更希望学习它的形式化定义, 则倾向

于得到“/usr2”文件夹中的文本文件。如果希望同时查找定义和代码,则可以在根目录“/”范围下搜索。另一个具有潜在价值的场景是语料库中的源数据具有树型或有向无环图结构组织的密级(例2)。对于某个查询,用户B1只能让大模型抽取自己有权访问的文件(图1中C1、D2,其访问权限不小于文件密级)作为知识源,以防止文件的越权访问或机密泄露^[32]。

尽管已有的通用过滤向量索引^{[12][33]}和某些标签过滤向量索引能够解决类别过滤向量搜索问题,但它们未针对类别过滤场景的特性做针对性的算法优化或者可能会访问不符合类别关系的向量^[22],因此无法取得最优查询性能。为了解决CFANNS,一种朴素的思路是为每个类别构造一个先知近邻图(Oracle Proximity Graph)^{[6][12]},该近邻图仅在所有该类别及其子类包含的向量上构造。然而构造与类别集合大小相等数量的近邻图(记为 $O(|C|)$,其中 C 为类别集合)需要大量时间和空间开销,因此无法应用到百万向量^[34]和具有大量类别的场景^[35]。

注意到为每个类别建立的先知近邻图之间有很多重复的边,即两个类别的先知近邻图可能有大量点具有相同的邻居集合。基于该发现,针对树型类别关系,本文设计了一种类别增强近邻图索引,将构建这些先知近邻图的时间和空间开销降低并确保每条边仅出现一次。为了解决更广泛的有向无环图型类别过滤问题,本文通过扩展该索引并进一步压缩邻居列表,大幅降低该场景下的索引构建开销。实验表明,在不同类别关系下,索引与已有算法相比,都具有更优的查询性能和相近的构建性能。

本文主要贡献总结如下:

- (1) 本文提出了类别过滤近似最近邻搜索问题并设计了类别增强近邻图来解决树型类别过滤。本文证明其能够通过相互配合的邻居选取与增量构建算法,将 $O(|C|)$ 个先知近邻图无损压缩到 $O(\log|C|)$,并证明在查询时通过动态选择邻居列表,能够取

得与先知近邻图相同的查询性能。

- (2) 本文通过邻居分组将上述索引算法泛化到有向无环图型类别过滤场景。为了进一步降低构建开销,放宽无损压缩的约束,转而采用压缩率更高的邻居列表,并证明其构建时间和空间复杂度与类别集合 C 无关,采用该压缩策略不会对查询性能造成较大影响。
- (3) 在4个含类别的向量数据集上的实验表明,查询性能方面,针对树型类别,相比查询性能最优的方法^[5],索引在压缩前和压缩后分别取得了 $5\times$ 和 $3\sim 5\times$ 的性能提升,且前者是唯一在所有数据集上取得与先知近邻图完全相同的查询性能的方法。针对有向无环图型类别,压缩后相比对比方法仍然取得了 $2\sim 5\times$ 的查询性能提升和稳定的高召回率。构建性能方面,压缩邻居列表优化后能够使索引取得 $5\sim 7\times$ 构建性能提升以及 $0.3\sim 0.8\times$ 更小的索引规模,具有与对比方法相近或更优的构建性能。

1 预备知识

本节定义近似最近邻搜索问题和类别过滤近似最近邻搜索问题。相关工作的详细讨论见附录C。

定义1 (近似最近邻搜索)。给定以 $\delta(\cdot)$ 为距离度量的向量数据集 V ,对于一个定义在同一向量空间中的查询向量 q 以及目标近邻数量 k ,近似最近邻搜索的目标是从 V 中抽取一个满足 $|S| = k$ 的向量子集 S 以最小化 $\sum_{v \in S} \delta(v, q)$ 。

近似最近邻搜索不保证召回查询向量的所有精确近邻,而是建立近似最近邻搜索索引并通过近似或启发式搜索快速召回可能的近邻。通常使用召回率评估近似最近邻搜索索引的查询准确率: $recall = \frac{|S \cap R|}{k}$,其中 R 为查询向量的精确近邻集合。

定义2 (类别过滤近似最近邻搜索)。给定混合数据集 $\mathcal{D} = \{V, C\}$,其中 V 为定义1中的向量数据集, C 为类别集合。映射 $c(\cdot): V \rightarrow C$ 作用在向量 v 上,表示 v 所属的唯一直接类别(即不包含祖先类别)。 $\mathcal{H}$ 为 C 中类

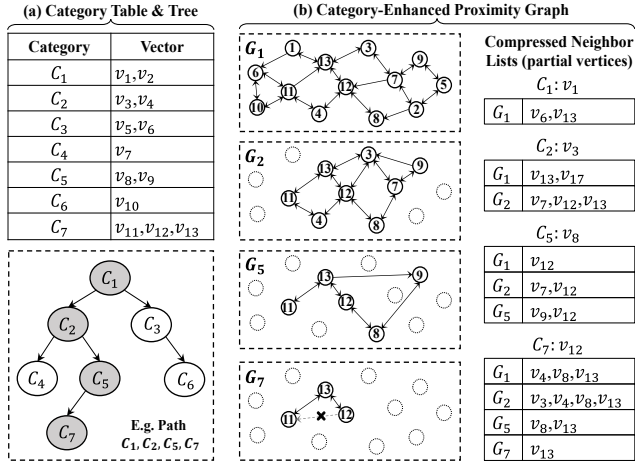


图2 索引结构概览

Fig.2 Index structure overview

别满足的拓扑关系, 为有向树结构 $\mathcal{T}$ 或有向无环图结构 $\mathcal{A}$ 。 C 在 $\mathcal{H}$ 上的偏序关系 $C_1 \leq C_2$, 当且仅当 C_1 为 C_2 的子类或 $C_1 = C_2$, 称类别 C_2 覆盖类别 C_1 。给定查询向量 q , 查询类别 $p \in \mathcal{C}$, 类别过滤近似最近邻搜索的目标是从 $\mathcal{D}$ 中抽取一个满足 $|S| = k$ 的混合数据子集 S 以最小化 $\sum_{(v, c(v)) \in \mathcal{D}} \delta(v, q), c(v) \leq p$ 。

类别过滤近似最近邻搜索答案集合中所有向量的类别需要被查询类别覆盖, 在此过滤条件下结果向量应当尽可能接近查询向量。计算召回率时, 精确近邻集合中的所有向量也需要满足类别的覆盖关系。

2 类别增强近邻图

本节将以树型类别过滤场景为切入点, 介绍增量构建算法及树类别过滤近似最近邻查询。

2.1 索引结构概述

类别增强近邻图的核心思想是将所有类别下的先知近邻图压缩在同一张更稠密的近邻图中, 并且在给定类别上查询时能够准确判断符合子类关系的邻居, 从而动态拟合在先知近邻图上的搜索。

图2展示了在一个二维混合数据集上的例子。图2(a)列举了定义2中的类别映射关系 $c(\cdot)$ 与类层次关系偏序关系 $\mathcal{T}: \leq$ 。例如 $c(v_5) = C_3$, $\{x | x \leq C_2\} = \{C_2, C_4, C_5, C_7\}$ 。图2(b)展示了在部分类别下的增强近邻图结构, 以及部分向量的压缩邻居列表。 G_2 表示类

算法 1. 类别增强近邻图的向量插入算法(Insert)

输入: 待插入向量 v ; 向量所属类别 C_t

输出: 新向量插入后的索引 $\mathcal{G}$

索引参数: 构建时集束搜索宽度 ω_c ; 近邻图最大出度 m ; 类别映射 $c(\cdot)$; 类别关系 $\mathcal{T}: \leq$

```

1   $U \leftarrow \{\};$  // 候选邻居集合
2   $C_R \leftarrow \text{root category of } \mathcal{T};$  // 类别树中的根类别
3  foreach category  $C$  on path  $C_R \rightarrow C_t$  in  $\mathcal{T}$  do
4       $U \leftarrow \{v \mid v \in U \wedge c(v) \leq C\};$ 
5       $ep \leftarrow$  a random vector where  $c(ep) \leq C$ ;
6      if  $|U| < m$  then  $U \leftarrow U \cup \text{CatCA}(v, C, \omega_c, ep);$ 
7       $N_v^{G_C} \leftarrow \text{Prune}(v, U, \frac{m}{2});$ 
8      foreach neighbor  $u \in N_v^{G_C}$  do
9           $N_u^{G_C} \leftarrow N_u^{G_C} \cup \{v\};$ 
10         if  $|N_u^{G_C}| > m$  then  $N_u^{G_C} \leftarrow \text{Prune}(u, N_u^{G_C}, m);$ 
11      $c(v) \leftarrow C_t;$ 
12 return  $\mathcal{G};$ 

```

别 C_2 及其子类(C_4, C_5, C_7)中所有向量组成的近邻图, 以此类推展示了树 $\mathcal{T}$ 上一条路径 C_1, C_2, C_5, C_7 对应的近邻图 G_1, G_2, G_5, G_7 。图中点的分布表示了向量在二维空间中的距离远近, 点上的数字代表向量的标号(ID)。压缩邻居列表记录了每个向量在其所在类别及祖先类别对应的近邻图中的邻居。例如对于 C_2 中的一个向量 $C_2: v_3$, 它有且仅有一个祖先类别 C_1 , 在该类别对应的近邻图 G_1 中 v_7 与 v_{13} 是 v_3 的邻居; 在其同一类别 C_2 对应的近邻图 G_2 中 v_7, v_{12}, v_{13} 为其邻居。另外, G_7 示意图也展示了一个不满足 RNG 性质的例子, 即在插入 v_{12} 时发现了 v_{11} 和 v_{13} 这两个候选邻居。 v_{13} 由于距离更近, 所以先于 v_{12} 被选为邻居, 而检查 v_{12} 时存在 $\delta(v_{11}, v_{12}) > \delta(v_{13}, v_{12})$ 且 $\delta(v_{11}, v_{12}) > \delta(v_{11}, v_{13})$, v_{12} 违背了 RNG 中向量的分布性质, 因此被 RNG 剪枝过程裁剪。

2.2 类别增强近邻图的向量插入

算法1描述了索引的插入过程。索引构建时参数包括构建时的集束搜索宽度 ω_c 、近邻图最大出度 m , 以及向量的类别映射 $c(\cdot)$ 和类别树及其偏序关系 $\mathcal{T}: \leq$ 。算法接受一个插入向量 v 以及该向量的所属类别 C_t , 并首先找出在类别树 $\mathcal{T}$ 上从根节点 C_R 到 C_t 的路径(第

算法 2. 类别感知候选邻居选取过程(CatCA)**输入：**目标向量 v ;目标类别 C_t ;集束搜索宽度 ω ;入口点 ep **输出：**候选邻居集合 U // 集束窗口小顶堆 W ,候选邻居大顶堆 U

```

1   $W \leftarrow \{ep\}, U \leftarrow \{ep\};$ 
2  while  $|W| \neq \emptyset$  do
3     $w \leftarrow \text{pop one vector from } W;$ 
4    foreach unvisited neighbor  $u \in N_w^{C_t}$  do
5      mark  $u$  as visited;
6      if  $|U| < \omega$  or  $\delta(u, v) < \max_{s \in U} \delta(s, v)$  then
7        push  $u$  into  $W, U;$ 
8      if  $|U| > \omega$  then pop  $U$  until  $|U| = \omega;$ 
9  return  $U;$ 

```

3 行)。对于路径上的每个类 C , 首先检查候选邻居集合 U 中属于 C 的子类或同一类的向量。如果符合条件的向量数量小于最大出度 m , 算法通过下节算法 2 的类别感知候选邻居选取得到符合类别覆盖条件的候选邻居(第 6 行), 此时定理 1 保证了候选集合中的向量一定满足集合覆盖条件。如果符合条件向量数量足够, 可以省略重复的候选搜索并且能够得到正确的候选来优化索引构建时间, 这是因为作为上一个类别的子类, 其满足条件的近邻一定在其父类中存在, 否则将与算法 2 的贪心搜索过程矛盾。对于候选邻居中的向量, 算法 1 使用 RNG 剪枝策略(Prune, 第 7 行)选出 $\frac{m}{2}$ 个近邻, 空余的 $\frac{m}{2}$ 槽位为未来的待插入向量保留。将剪枝后的最终邻居加入到向量 v 在 C 覆盖的近邻图 G_C 的压缩邻居列表中(第 7 行), 再将当前插入向量加入到这些邻居的压缩邻居列表以确保连通性。如果列表没有空余槽位, 则再次使用 RNG 剪枝策略调整二阶邻居分布(第 11 行)。算法为从类别树 T 的根节点到 $c(v)$ 的所有类别覆盖近邻图都创建邻居关系, 例如图 1 中 $C_5: v_8$ 有 3 个压缩邻居列表, 分别对应其在 G_1, G_2, G_5 中的邻居。

2.3 类别感知候选邻居选取

算法 2 描述了类别感知候选邻居的选取过程(Category-aware Candidate Acquisition, 简称 CatCA), 采用集束搜索策略。该算法与传统的近邻图上集束搜

索的不同之处在于每一跳邻居选择总是在目标类别 C_t 所覆盖的压缩邻居列表上进行(N^{C_t})。从入口点 ep 出发每次选择距离目标向量最近的点作为下一跳(第 3 行), 然后将尚未探索的且比当前候选邻居 U 中最远的向量更近的邻居加入到候选集 U 和搜索窗口 W (第 6 行)并重新维护结果集合的大小(第 7 行)。直到没有更近的点被发现时, 算法终止(第 2 行)。定理 1 能够保证返回的候选邻居集合 U 中向量均满足查询类别覆盖关系。

定理 1 (集束搜索对 $T: \leq$ 类别关系的忠实性)。算法 2 运行任意时刻, $\forall v \in U, c(v) \leq C_t$ 。(证明见附录 A)

2.4 类别过滤近似近邻搜索

算法 3 类似插入时选取候选邻居的过程, 搜索时使用算法 2 抽取距离查询向量最近的向量。由定理 1,

算法 3. 类别过滤近似近邻搜索过程(CFANNS)**输入：**查询向量 q ; 目标类别 p ; 搜索时集束搜索宽度 ω_s ; 目标邻居数量 k **输出：**查询向量 q 的近似 k 近邻;且类别均为 C_t 或其子类**索引参数：**类别映射 $c(\cdot)$;类别关系 $T: \leq$

```

1   $ep \leftarrow \text{a random vector s.t. } c(ep) \leq C_t;$ 
2   $U \leftarrow \text{CatCA}(q, p, \omega_s, ep);$ 
3  return  $k$  closet vectors to  $q$  in  $U;$ 

```

给定查询类别, 候选邻居中所有向量的类别均与查询类别相同或为查询类别子类, 且由集束搜索的贪心性质, 候选邻居集合中包含 ω_s 个距离查询向量 q 最近的向量。算法选择前 k 个最近的向量作为类别过滤近似近邻搜索的查询结果, 并用参数 ω_s 控制集束宽度。

3 讨论与扩展

本节将证明算法 1 构建的索引是对所有类别先知近邻图的无损压缩, 因此对于任意到来的查询都能达到基于图索引的查询性能上界。将分析并证明索引的空间复杂度、构建时间和查询时间复杂度。最后, 将介绍如何将现有索引扩展到有向无环图场景, 并证明压缩后其构建时间复杂度和空间复杂度与类别数量无关。

定理 2 (无损压缩)。类别增强近邻图对所有类别对应的先知近邻图的无损压缩。(证明见附录 A)

定理 3 (空间复杂度)。索引空间复杂度为 $O(mn\bar{h})$, 其中 $\bar{h}$ 为 $\mathcal{T}$ 中每个子树的平均高度。若 $\mathcal{T}$ 相对平衡, 有 $\bar{h} = \log|\mathcal{C}|$ 。

证明: 向量 v 的压缩邻居列表的个数 $h(c(v)) = |C_R \rightarrow c(v)|$ 满足 $1 \leq h(c(v)) \leq h_{\max}$, 其中 $h_{\max}$ 为 $\mathcal{T}$ 的高度, 每个列表的最大槽位数为 m , 因此平均空间复

杂度为 $O\left(m \sum_{(v, c(v)) \in \mathcal{D}} h(c(v))\right) = O(mn\bar{h})$ 。■

定理 4 (构建时间复杂度)。索引的最坏情况构建时间复杂度为 $O(md\bar{h}n \log n)$ 。

证明: 算法 1 主要由两部分组成: (1) 候选邻居选取以及 (2) RNG 剪枝与二阶邻居调整。对于 (1), 最坏情况下路径 $C_R \rightarrow c(v)$ 上的每个类别都需要调用算法 2, 由定理

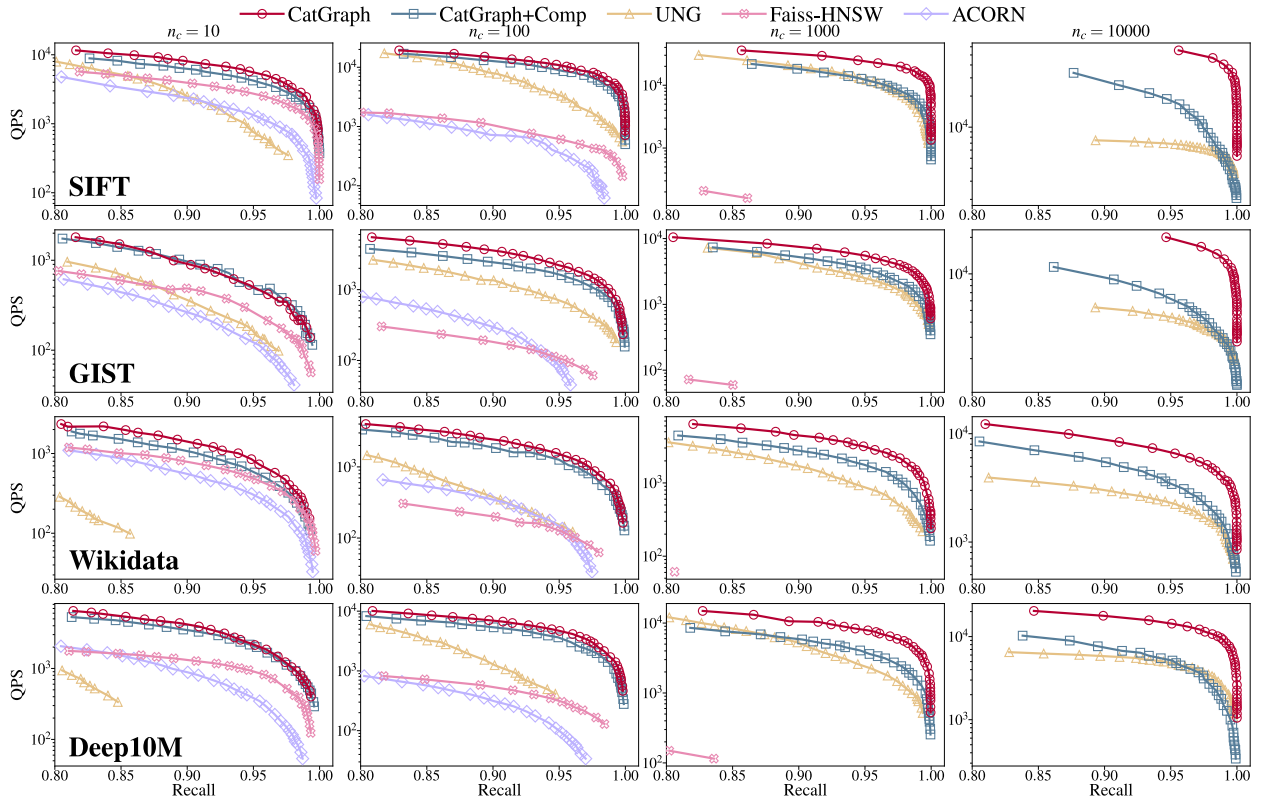


图 3 CFANNS 查询性能: 每秒查询次数比较

Fig.3 Comparison of QPS for CFANNS query performance

1 可知在任意类别上搜索都等价与在该类别对应的先知近邻图上的搜索。由 RNG 单调搜索的性质可知, 高维向量空间近邻图上路径长度为 $O(\log n)^{[23]}$ 。因此候选邻居选取的时间为 $O(md\bar{h} \log n)$ 。对于 (2), 对于其中一个类需要常数时间, 为 $O(\frac{m}{2} \log \omega_c + m^2 \log m)$ 。因此插入 n 个向量的时间复杂度为 $O(md\bar{h}n \log n)$ 。■

定理 5 (搜索时间复杂度)。时间复杂度为 $O(md \log n')$, 其中 n' 为满足类别过滤的向量子集大小。

证明: 算法 3 只需调用算法 2, 由定理 2 和定理 3 可知, 时间复杂度等价于先知近邻图上的遍历, 为

$O(\log n')$ 。■

3.3 索引压缩与有向无环图类别过滤扩展

部分真实世界场景的类层次关系推广为有向无环图(DAG)结构, 即某些类别在 DAG $\mathcal{A}$ 的关系 $\mathcal{A}:\leq$ 下可能有多个直接父类和大量祖先, 这是该场景下相比树型过滤最具有挑战性的特点。为此, 需要重新定义索引的压缩邻居列表以在有限的空间保留更多的类别之内的邻居。本文提出三邻居压缩技术, 即一个向量有三种压缩邻居列表: (1) N^V : 只考虑向量之间相似度的邻居列表; (2) N^M : 保存祖先类别与后继类别中最相

似向量作为近邻的列表; (3) N^C : 保存同类别向量中最相似向量作为近邻的列表。三种邻居列表的类别约束逐渐收缩, 以应对不同类别数量的数据集, 将在第 4.4 节验证这一点。算法 1 对每一类别层次化构建上述三种类别边, 算法 2 按顺序扫描三种类型邻居。三邻居压缩能够使索引在 DAG 场景下取得与类别数量无关的构建时间与空间开销。

4 实验

4.1 实验准备

实验在 4 个真实向量数据集上进行评测, 包括 SIFT^[1]、GIST^[1]、Wikidata^[2]和 Deep10M^[3], 涵盖图像

表 1 索引构建时间(秒)

Table 1 Indexing time (s)

	SIFT				GIST			
	10^1	10^2	10^3	10^4	10^1	10^2	10^3	10^4
ACORN	103	103	103	103	749	749	749	749
HNSW	30	30	30	30	247	247	247	247
UNG	73	26	15	6	481	195	73	14
CatGraph	80	124	195	844	365	562	717	1,517
Comp	123	105	102	95	392	392	355	292
	Wikidata				Deep10M			
	10^1	10^2	10^3	10^4	10^1	10^2	10^3	10^4
ACORN	2,897	2,897	2,897	2,897	1,350	1,350	1,350	1,350
HNSW	1250	1250	1250	1250	392	392	392	392
UNG	1,646	1,211	691	234	417	221	139	96
CatGraph	861	1,568	2,221	5,389	835	1,441	2,448	8,560
Comp	925	1,564	1,611	1,164	1,335	1,443	1,400	1,292

表 2 索引规模(兆字节), 含向量数据集规模

Table 2 Index size (MB), including raw vector size

	SIFT (489)				GIST (3,666)			
	10^1	10^2	10^3	10^4	10^1	10^2	10^3	10^4
ACORN	1,046	1,046	1,046	1,046	4,219	4,219	4,219	4,219
HNSW	626	626	626	626	3,861	3,861	3,861	3,861
UNG	636	684	683	650	3,845	3,844	3,851	3,849
CatGraph	824	1,084	1,473	2,057	4,246	4,532	5,105	5,963
Comp	695	695	695	695	3,960	3,960	3,960	3,960
	Wikidata (15,641)				Deep10M (3,701)			
	10^1	10^2	10^3	10^4	10^1	10^2	10^3	10^4
ACORN	17,854	17,854	17,854	17,854	9,231	9,231	9,231	9,231
HNSW	16,176	16,176	16,176	16,176	5,038	5,038	5,038	5,038
UNG	16,286	16,297	16,358	16,438	5,340	5,365	5,517	5,708
CatGraph	17,228	18,006	19,562	20,859	7,668	9,614	13,505	16,747
Comp	16,449	16,449	16,449	16,449	5,723	5,723	5,723	5,723

和文本模式, 覆盖不同数据规模与距离度量。为每个数据集随机生成 4 种不同类别规模的树型类别层次结构($10 \sim 10000$ 个类别), 向量等概率随机分配类别, 共产生 16 个子数据集。查询集包含 1000 个随机选取的

向量及其类别, 目标近邻数 $k=10$ 。数据集详细信息见附录 B.2。对比方法包括本文提出的类别增强近邻图 CatGraph、使用三邻居压缩的版本 CatGraph+Comp、通用过滤索引 ACORN^[12]、基于 Faiss 的 HNSW 索引^[33] (Faiss-HNSW) 和标签过滤索引 UNG^[5]。共有近邻图构建参数在方法间保持一致, 具体参数见附录 B。

4.2 类别过滤近似最近邻查询性能

使用召回率与每秒查询次数(QPS)评估 4 种索引的查询性能, 单线程结果如图 3 所示。CatGraph 在所有子数据集上均取得最优: 90%召回率下相比 UNG 提升 5 倍, 相比 ACORN 提升 6~10 倍。得益于无损压缩的理论保证, CatGraph 在不同类别数下均能拟合先知

表 3 DAG 类别构建时间(秒)和索引规模(兆字节)

Table 3 Indexing time (s) and index size (MB) for DAG-categorized datasets

		SIFT			GIST		
		20	2000	4000	20	2000	4000
时 间	ACORN	103	103	103	749	749	749
	HNSW	38	38	38	310	310	310
	UNG	68	91	49	569	1,046	123
	Comp	103	162	102	364	399	295
规 模	ACORN	1,046	1,046	1,046	4,219	4,219	4,219
	HNSW	626	626	626	3,861	3,861	3,861
	UNG	737	721	2,857	3,864	3,886	6,060
	Comp	695	695	695	3,960	3,960	3,960

近邻图的查询性能。经三邻居压缩后(CatGraph+Comp)在类别数少时与压缩前相当; 类别数多时因有限邻居列表无法保留所有边而出现预期内的轻微下降, 但仍保持 3~5 倍于 UNG 的优势。UNG 在类别数少($n_c = 10$)时因跨组边破坏 RNG 性质, 召回率上限约 85%(Wikidata、Deep10M); ACORN 在类别数多($n_c = 10000$)时过滤过强, 最高召回率仅 4%(Deep10M)。CatGraph 及压缩变体在所有场景下均稳定取得最优。

4.3 索引构建性能评测

所有索引使用 16 线程构建, 表 1 和 2 展示了构建性能评测。Faiss-HNSW 因仅构建普通 HNSW 而拥有次优的构建时间和最优的索引规模。UNG 的平均构建时间最短、索引规模第二小, 且类别数越多构建越快、规模越大, 但其划分策略缺乏查询性能的理论保证。

CatGraph 的构建时间和索引规模与类别树高呈线性增长关系; 在类别数较少($n_c \leq 10^2$)时与 ACORN 相当或更优, 低属性数数据集上相比 ACORN 取得 1.2~3.4 倍加速。它是唯一具有与先知近邻图相同查询性能理论保证的方法。经三邻居压缩后(Comp), 构建时间和索引规模均与类别数量无关: 大量类别场景($n_c = 10^4$)下相比 CatGraph 取得 5~7 倍构建速度提升和 30%~80% 的索引规模缩减, 相比 ACORN 仍有近 1.5 倍速度优势。CatGraph 拥有最佳查询性能, CatGraph+Comp 在保持较优构建效率的同时仍优于所有对比方法的查询表现。

4.4 有向无环图场景索引性能评测

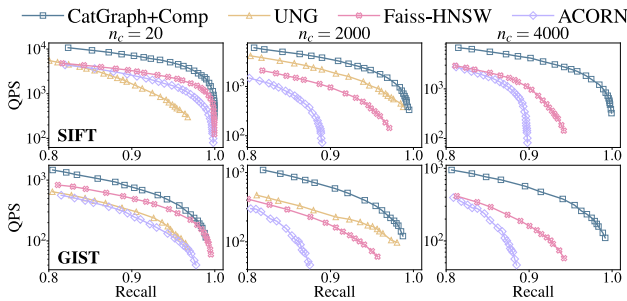


图4 DAG 类别关系 CFANNS 查询性能:每秒查询次数比较

Fig.4 Comparison of QPS for DAG-categorized CFANNS performance

为了进一步评估索引的泛化性, 构造了 3 个具有 DAG 类别关系的数据集并为 SIFT 和 GIST 用同样的方法产生 6 个子数据集。DAG 分别有 20、2,000、4,000 个类别, 相应的平均覆盖类数为 5.55、48.00、493.07。另外, $n_c = 2,000$ 的向量数据集类别采用幂律分布, 即大部分向量(几万到几十万)仅属于少数几个类, 其他大量类别中只包含少量向量(例如 10 个向量或更少)。各方法参数设置与先前实验一致。

表 3 展示了参与比较的 4 种索引的构建时间和索引规模, 与第 4.3 节的结论保持一致。虽然 GIST $n_c = 2000$ 数据集类别数量更多, 经过压缩后的类别增强近邻图仍然取得了比 UNG 更好的构建性能。原因是幂律分布场景下, 需要为少数类别的大部分向量构建大

图(具有更多向量的近邻图), 加上 UNG 较难处理难数据集(局部固有维数较高^[40])上的邻居分布来构建高质量边, 大部分时间将用在构建这些大图上。而 UNG 采用类间并行策略, 可能出现其他构建完小图的线程等待大图的构建而被闲置的情况。而本文构建算法采用点粒度的互斥锁, 保证了 CPU 的高利用(构建时所有 CPU 核利用率均大于 99%)。

图 4 展示了 DAG 类别过滤下的查询性能, CatGraph+Comp 相比 UNG 取得了 2~5 倍性能提升。观察发现 UNG 难以处理具有大量覆盖类别的查询类别($n_c = 4000$, 平均覆盖类数为 493.07)。原因在于 UNG 是以分块为基础的索引并只为每个类别的直接

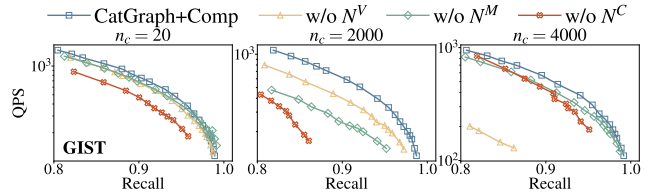


图5 评估压缩索引的 3 种类别邻居的消融实验

Fig.5 Ablation study of three compressed neighbor types 后继类别中的向量建立分块间的跨组边。大量覆盖类别导致 UNG 中大部分邻居均为跨组边而不具有 RNG 性质, 且只为直接后继而不是为所有后继建立跨组边可能会导致与其它间接后继类别中的向量不连通, 从而难以达到可接受的召回率(例如 GIST $n_c = 4096$ 上取得的最高召回率为 69%)。CatGraph 将向量相似度放在首位而非显式划分数据集, 因此对于任意覆盖类别数量都能取得较好的查询性能。

图 5 使用一组消融实验分析了 3 种压缩邻居列表 N^V , N^M , N^C 对查询性能的影响, 可以发现当类别数量较少($n_c = 20$)或数据集采用幂律分布($n_c = 2000$)时, 去掉 N^C 对性能影响最大, 原因是结果中大部分向量属于同一类别, 因此相同类别的邻居关系会带来更大的性能提升。 N^M , N^V 的主要作用是串联不同类别之间的近邻, 分别去掉两种邻居会导致在中等数量($n_c = 2000$)和较多数量($n_c = 4000$)数据集上出现性能下降。

5 总结与展望

本文提出了一种新型过滤向量搜索问题: 类别过滤近似最近邻搜索。首先分析了树型类别过滤场景的特点并设计了一种专用类别过滤向量索引。理论证明了对查询时所有先知近邻图无损压缩的性质以及构建和查询复杂度。对于更泛化的有向无环图型类别过滤, 通过进一步压缩, 在不显著影响查询性能的前提下得到构建性能更好的索引。实验表明在两种场景下均取得了最优查询性能。未来将探索如何将其应用在类别关系可变的动态场景, 以及如何从内存索引迁移到基于磁盘的索引, 使其能够更好地应对亿级向量规模查询并应用到关系数据库中。

参考文献:

- [1] Evaluation of approximate nearest neighbors: large datasets. <http://corpus-texmex.irisa.fr/>
- [2] Multilingual embeddings for Wikipedia. <https://hugging-face.co/datasets/Cohere/wikipedia-2023-11-embed-multi-lingual-v3>
- [3] Benchmarks for billion-scale similarity search. <https://re-search.yandex.com/blog/benchmarks-for-billion-scale-similarity-search>
- [4] ILIAS A, KARIMA E, TEMIS P: Graph-based vector search: an experimental evaluation of the state-of-the-art[C]. SIGMOD 2025.
- [5] CAI Y Z, SHI J Y, CHEN U Z, et al: Navigating labels and vectors: a unified approach to filtered approximate nearest neighbor search[C]. SIGMOD 2025.
- [6] XU Y X, GAO J Y, GOU Y T, et al: iRangeGraph: improving range-dedicated graphs for range-filtering nearest neighbor search[C]. SIGMOD 2025.
- [7] WEI J Q, LEE X D, LIAO Z Y, et al: Subspace collision: an efficient and accurate framework for high-dimensional approximate nearest neighbor search[C]. SIGMOD 2025.
- [8] GOU Y T, GAO J Y, XU Y X, et al: SymphonyQG: towards symphonious integration of quantization and graph for approximate nearest neighbor search[C]. SIGMOD 2025.
- [9] YU R J, HUANG W Z, BAI S H, et al: AquaPipe: a quality-aware pipeline for knowledge retrieval and large language models[C]. SIGMOD 2025.
- [10] XU Q, YANG J, ZHANG F, et al: Tribase: a vector data query engine for reliable and lossless pruning compression using triangle inequalities[C]. SIGMOD 2025.
- [11] WANG Z Y, WANG Q T, CHENG X X, et al: Steiner-hardness: a query hardness measure for graph-based ANN indexes[J]. Proc. VLDB Endow. 17(13): 4668–4682 (2024)
- [12] PATEL L, KRAFT P, GUESTRIN C, et al: ACORN: performant and predicate-agnostic search over vector embeddings and structured data[C]. SIGMOD 2024.
- [13] SU Y Y, SUN Y Q, ZHANG M J, et al: Vexless: a serverless vector data management system using cloud functions[C]. SIGMOD 2024.
- [14] GAO J Y, LONG C: RaBitQ: quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search[C]. SIGMOD 2024.
- [15] ZUO C J, QIAO M, ZHOU W C, et al: SeRF: segment graph for range-filtering approximate nearest neighbor search[C]. SIGMOD 2024.
- [16] PAN J J, WANG J G, LI G L: Survey of vector database management systems[J]. The VLDB Journal 33(5): 1591–1615 (2024).
- [17] ENGELS J, LANDRUM B, YU S D, et al: Approximate nearest neighbor search with window filters[C]. ICML 2024.
- [18] WANG Y F, MA H D, WANG D Z: LIDER: an efficient high-dimensional learned index for large-scale dense passage retrieval[J]. Proc. VLDB Endow. 16(2): 154–166 (2022).
- [19] GOLLAPUDI S, KARIA N, SIVASHANKAR V, et al: Filtered-DiskANN: graph algorithms for approximate nearest neighbor search with filters[C]. WWW 2023: 3406–3416.
- [20] WANG M Z, LV L W, XU X L, et al: An efficient and robust framework for approximate nearest neighbor search with attribute constraint[C]. NeurIPS 2023: 15738–15751.
- [21] GUO R T, LUAN X F, XIANG L, et al: Manu: a cloud native vector database management system[J]. Proc. VLDB Endow. 15(12): 3548–3561 (2022).
- [22] WANG J G, YI X M, GUO R T, et al: Milvus: a purpose-built vector data management system[C]. SIGMOD 2021.
- [23] WANG M Z, XU X L, YUE Q, et al: A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search[J]. Proc. VLDB Endow. 14(11): 1964–1978 (2021).
- [24] SUBRAMANYA S J, DEVVRIT, KADEKODI R, et al: DiskANN: fast accurate billion-point nearest neighbor search on a single node[C]. NeurIPS 2019: 13766–13776.
- [25] MALKOV Y A, YASHUNIN D A: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs[J]. IEEE Trans. Pattern Anal. Mach. Intell. 42(4): 824–836 (2020).
- [26] FU C, XIANG C, WANG C X, et al: Fast approximate nearest neighbor search with the navigating spreading-out graph[J]. Proc. VLDB Endow. 12(5): 461–474 (2019).

- [27] MANOHAR M D, SHEN Z Q, BLELLOCH G, et al: ParlayANN: scalable and deterministic parallel graph-based approximate nearest neighbor search algorithms[C]. SIGPLAN 2024: 270–285.
- [28] FU C, WANG C X, CAI D: High dimensional similarity search with satellite system graph: efficiency, scalability, and unindexed query compatibility[J]. IEEE Trans. Pattern Anal. Mach. Intell. 44(8): 4139–4150 (2022).
- [29] WEI C X, WU B, WANG S, et al: AnalyticDB-V: a hybrid analytical engine towards query fusion for structured and unstructured data[J]. Proc. VLDB Endow. 13(12): 3152–3165 (2020).
- [30] RADFORD A, KIM J W, HALLACY C, et al: Learning transferable visual models from natural language supervision[C]. PmLR 2021: 8748–8763.
- [31] LI J, LIU H F, GUI C H, et al: The design and implementation of a real time visual search system on JD e-commerce platform[C]. Middleware 2018: 9–16.
- [32] ZHANG X Y, WANG Q C, XU C, et al: FedKNN: secure federated k-nearest neighbor search[C]. SIGMOD 2024.
- [33] JOHNSON J, DOUZE M, JÉGOU H: Billion-scale similarity search with GPUs[J]. IEEE Trans. Big Data. 7(3): 535–547 (2019).
- [34] BABENKO A, LEMPITSKY V: Efficient indexing of billion-scale datasets of deep descriptors[C]. CVPR 2016.
- [35] VRANDEČIĆ D: Wikidata: a new platform for collaborative data collection[C]. WWW 2012.
- [36] GAO J Y, LONG C: High-dimensional approximate nearest neighbor search: with reliable and efficient distance comparison operations[C]. SIGMOD 2023.
- [37] TOUSSAINT G T: The relative neighbourhood graph of a finite planar set[J]. Pattern Recognit. 12(4): 261–268 (1980).
- [38] WANG Z Y, WANG P, PALPANAS T, et al: Graph-and tree-based indexes for high-dimensional vector similarity search: analyses, comparisons, and future directions[J]. IEEE Data Eng. Bull. 46(3): 3–21 (2023).
- [39] LAN H, HUANG S X, BAO Z F, et al: Cardinality estimation for similarity search on high-dimensional data objects: the impact of reference objects[J]. Proc. VLDB Endow. 18(3): 544–556 (2024).
- [40] AMSALEG L, CHELLY O, FURON T, et al: Estimating local intrinsic dimensionality[C]. SIGKDD 2015: 29–38.
- [41] 袁培森,沙朝锋,王晓玲,等. 一种基于学习的高维数据 c -近似最近邻查询算法. 软件学报 [J]. 23(8): 2018–2031(2012)
YUAN P S, SHA C F, WANG X L, et al: c -approximate nearest neighbor query algorithm based on learning for high-dimensional data[J]. Journal of Software. 23(8): 2018–2031(2012)
- [42] 冯小康,彭延国,崔江涛,等. 基于最优排序的局部敏感哈希索引[J]. 计算机学报. 43(5): 930–947(2020).
FENG X K, PENG Y G, CUI J T, et al: Locality sensitive hashing index based on optimal linear order[J]. Chinese Journal of Computers. 43(5): 930–947(2020).

附录

附录 A 定理证明

A.1 定理 1 证明

证明:首先证明任意一点 u 的任意压缩邻居列表 N_u^C 仅包含子图 G_C 中的点, 即 $\forall v \in N_u^C, v \in G_C \wedge c(v) \leq C$ 。易证索引 g 为空时成立。插入第一个向量后, 该向量的邻居列表为空, 条件亦成立。假设插入 i 个向量后条件成立, 即 g^i 。对于第 $i+1$ 个向量 v_{i+1} , 对于算法 1 中 $\forall C \in \{C_R \rightarrow c(v_{i+1})\}$ 抽取候选邻居过程, 因为 $c(ep) \leq C$ (算法 1 第 4 行), 由假设可知: (1)路径上的每一跳邻居列表都存在; (2)算法 2 第 4 行从的邻居列表中抽取时每一跳都有 $\forall v \in N_w^G, v \in G_C \wedge c(v) \leq C$, 所以抽取出的候选邻居 $\forall v \in U, c(v) \leq C$ 。于是 $N_{v_{i+1}}^{G_C}$ 作为经过裁边后的最终邻居列表也满足条件(因为裁边不会加入其他类别的邻居)。

A.2 定理 2 证明

证明:只需证明任意类别 p , 类别增强近邻图中任意向量的邻居列表 N^p 和先知近邻图中相同向量的邻居列表相同。假设满足查询类别 p 的向量子集 $D' = \{v \mid v \in \mathcal{V} \wedge c(v) \leq p\}$ 中向量 v 插入到先知近邻图 G_o 与算法 1 构建的属性增强近邻图 G_e 的顺序一致, 不妨设为 $v_1, v_2, \dots, v_{n'}$, 其中 $n' = |D'|$ 。当不插入点时, $G_o = G_e = \emptyset$; 插入点 v_1 时由于 G_o, G_e 中均无边, 所以 $G_o = G_e$ 。假设插入向量 v_i 后 $G_o^i = G_e^i$ 。对于待插入点 v_{i+1} , $i+1 \leq n'$, 在 G_o 中选取的候选邻居路径为 $v_{k_1}, v_{k_2}, \dots, v_{k_m}$ (v_{k_1} 为随机选取)。不妨设算法 1 选取的入口点也为 v_{k_1} , 且设前 $v_{k_1} \rightarrow v_{k_j}$ 跳相同。对于算法 2 搜索路径上的第 $j+1$ 跳 $v'_{k_{j+1}}$, 如果 $v'_{k_{j+1}} \neq v_{k_{j+1}}$, 则对于首先发现 $v'_{k_{j+1}}$ 的一跳 z , 由假设 $G_o^i = G_e^i$ 和定理 1 可知 $v_{k_{j+1}} \in N_z^p$, 所以 $\delta(v_{i+1}, v'_{k_{j+1}}) > \delta(v_{i+1}, v_{k_{j+1}})$, 说明 $v_{k_{j+1}}$ 比 $v'_{k_{j+1}}$ 距离 v_{i+1} 更近却未被加入搜索窗口, 与算法 2 矛盾, 所以 G_e 的路径与 G_o 的路径相同。两个图上的遍历在相同的路径上贪心地选择与目标向量最近的邻居, 因此算法 1 第 6 行选择的候选邻居列表与先知近邻图选择的候选

邻居列表相同。并且算法 1 第 7 行的 Prune 过程只考虑向量距离不考虑属性信息, 因此经过剪枝后的邻居列表也相同, 即 $G_o^{i+1}: N(v_{i+1}) = G_e^{i+1}: N^p(v_{i+1})$ 。对于 $\forall p, \cup_{p \in C} G_o^{i+1}(p): N(v_{i+1}) = G_e^{i+1}: N(v_{i+1})$ 。由归纳假设, 类别增强近邻图无损压缩了任意类别先知近邻图。

附录 B 实验设置细节与附加实验

B.1 实验环境和对比方法

所有实验在一台搭载 Ubuntu 20.04 系统, 配备 Intel Xeon E5-1607 v3 (3.10GHz) CPU, 240GB RAM 的服务器上进行, CPU 启用 AVX2 SIMD 指令集优化。所有方法均使用 C++实现, GCC 9.4.0 编译, 启用 O3 选项优化。

B.2 数据集与参数设置

表 4 树型类别数据集和向量数据集

Table 1 Statics of Tree category and vector dataset

类别数(n_c)	树高	平均覆盖类数
10	5	2.80
100	8	5.17
1,000	14	7.40
10,000	19	9.93

数据集名称	数据集规模	模态	向量维度	距离度量
SIFT ^[1]	1,000,000	图像	128	欧式距离
GIST ^[1]	1,000,000	图像	960	欧式距离
Wikidata ^[2]	4,000,000	文本	1,024	余弦距离
Deep10M ^[3]	10,000,000	图像	96	欧式距离

如表 4, 随机生成了 4 个树型关系类别, 其中平均覆盖类数指树中每个类别所有子类数量+1。在 4 个现实世界向量数据集上评测索引性能, 覆盖不同源数据模态、不同数据集规模、不同向量维度以及不同距离度量方式。之后为每个向量等概率随机分配一个类别, 一共产生 $4 \times 4 = 16$ 个子数据集。查询数据集包含 1000 个从向量数据集中随机抽取的查询向量以及为查询向量随机分配的类别。目标是为每个查询向量抽取 $k=10$ 个近邻。

(1) SIFT^[1]: 采用 SIFT 特征向量编码的一百万个图片, 向量维度为 128。(2) GIST^[1]: 采用 GIST 特征向

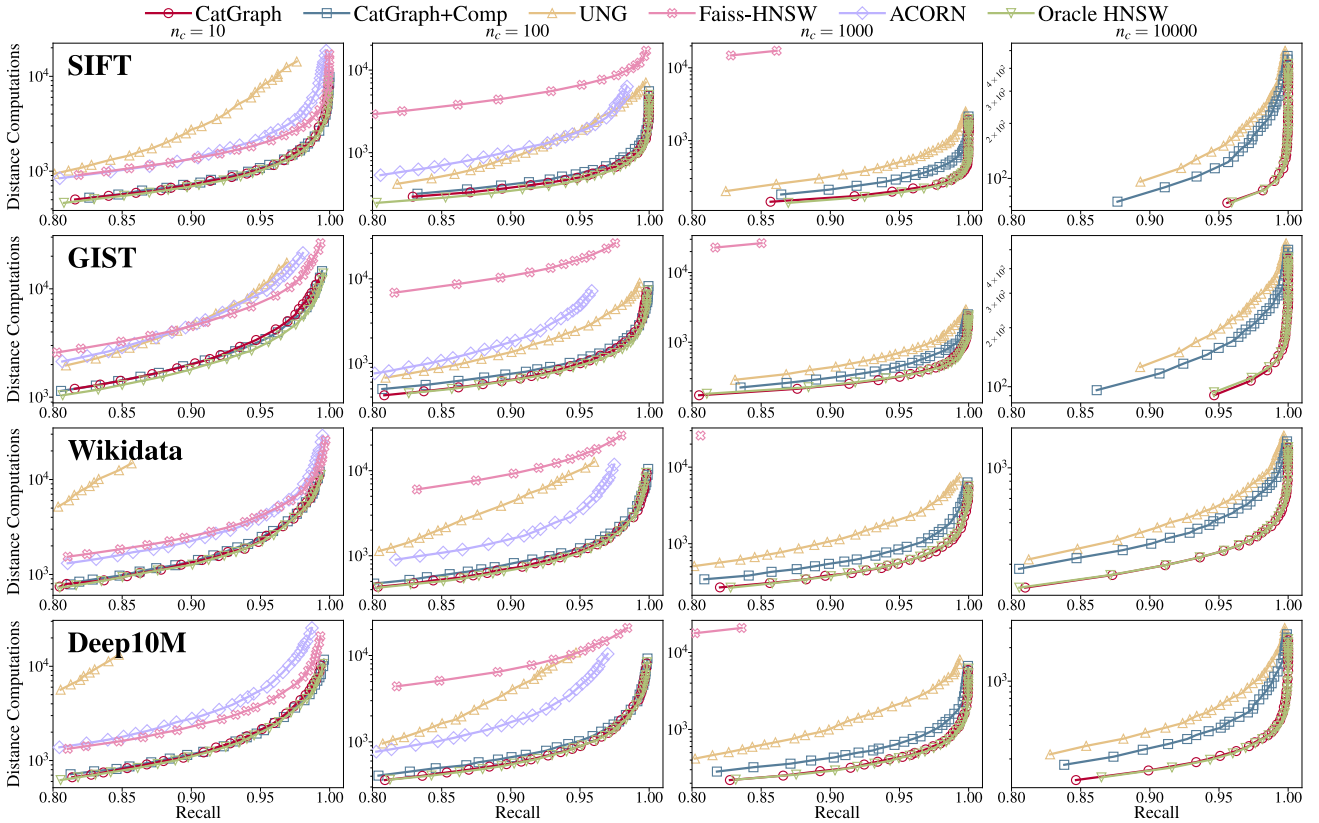


图6 拟合先知近邻图:距离计算次数比较

Fig.6 Comparison of DC for approximation of the Oracle HNSW

量编码的一百万个图片, 向量维度为 960。

(3) Wikidata^[2]: 使用 Cohere Embed v3 模型将截止 2023 年 11 月的 Wikidata 中保存的约四千万英文实体的描述文本进行向量编码得到 1024 维向量, 选取前四百万个向量进行测评。

(4) Deep10M^[3]: 采用 ImageNet 分类数据集预训练的 GoogleNet 将十亿图片特征化, 选取最后一层全连接层的输出结果并进行欧式距离正则化后的 96 维向量, 选取前一千万个向量进行评测。

选取了 3 个基于近邻图的混合向量索引进行对比: 通用过滤索引 ACORN^[12]和使用 Faiss^[33]向量检索库构建的 HNSW 索引(Faiss-HNSW), 以及标签过滤索引 UNG^[5], 三者均为在各自过滤索引类型中表现最佳的索引。

为保持公平性, 共有的近邻图构建参数 (构建时集束搜索宽度 ω_c 以及最大邻居数量 m) 保持一致

(ACORN 和 Faiss-HNSW 除外)。根据先知近邻图在每个数据集上不同参数设置下的最优表现, 为 4 个数据集选取的 ω_c 和 m 分别为: (1) SIFT: $\omega_c = 128$, $m = 16$; (2) GIST: $\omega_c = 256$, $m = 24$; (3) Wikidata: $\omega_c = 256$, $m = 16$; (4) Deep10M: $\omega_c = 128$, $m = 16$ 。

由于 ACORN 无法在以上参数选择下取得最优表现, 因此对所有数据集保持 ω_c 不变, 重新设置 $m=32, m_c=64, \gamma=30$, 其中 m_c, γ 为 ACORN 的额外参数, 其他方法均无额外参数。

B.3 拟合先知近邻图的质量评估

为了进一步分析本文索引能够取得最优查询性能的原因, 图 6 针对查询时各索引的向量距离比较次数 (Distance Computations, DC) 与附录 C 中介绍的先知近邻图搜索时的距离比较次数进行对比。这里选择业界流行的 HNSW 算法为每条查询对应的能够通过过滤条件的向量构建先知近邻图。通过分析得以下结论:

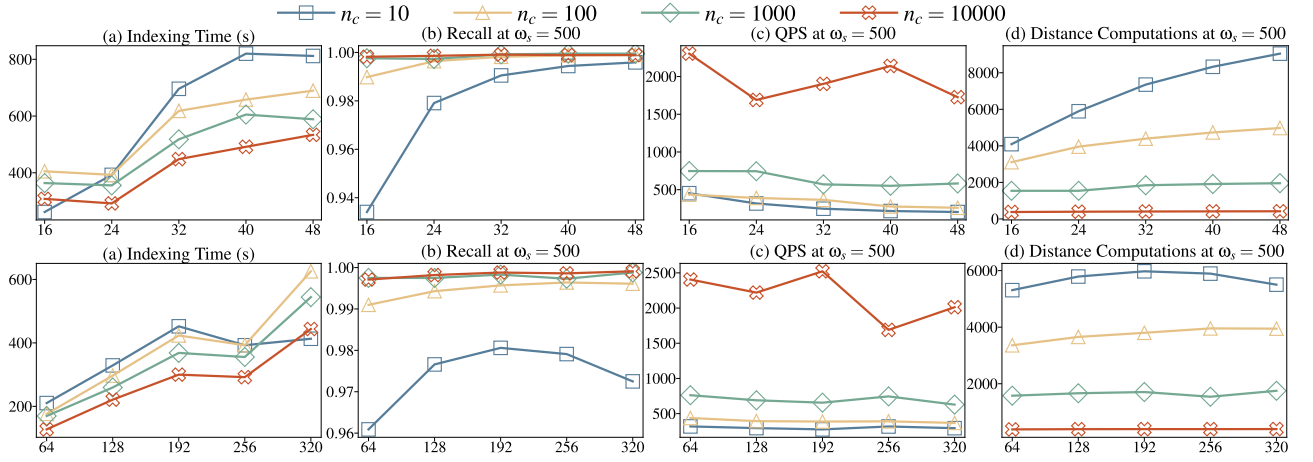


图 7 索引构建参数的影响: m (第 1 行)和 ω_c (第 2 行)

Fig.7 Impact of indexing parameters: m (1st row) and ω_c (2nd row)

(1) 在相同召回率下, CatGraph 取得了与先知近邻图相同的距离比较次数, 印证了定理 2。CatGraph+Comp 的有损压缩不会产生严重的性能损失, 且距离比较次数与先知近邻图相近。它们在相同召回率下均取得了比 UNG 和 ACORN 更少的距离比较次数。

(2) UNG 在平均类内向量数量较多时会产生较多无效的距离比较, 并且在部分数据集上的距离比较次数多于 ACORN(Wikidata、Deep10M $n_c = 100$)。但由于 UNG 构建近邻图比 ACORN 更稀疏并且近邻图搜索过程针对类别层次关系优化使得候选邻居搜索过程比 ACORN 更加高效, 因此即使距离比较次数比 ACORN 更多, 仍然能够取得比 ACORN 更快的查询速度。

(3) ACORN 和 Faiss-HNSW 的距离计算次数易受到类别数量的影响。在少量类别数据集上具有接近先知近邻图的性能(例如 Wikidata $n_c = 10$), 但在多类别数据集上相同的距离比较次数下难以取得高召回率。原因是在少类别数据集下的查询通常过滤能力较低。它们的邻居中保存了大部分符合条件的高质量 RNG 导航边。借助长距离导航能力能够用更少的距离比较抵达向量邻域; 而多类别数据集上的性能损失一方面来自于查询选择性的提升使得导航边被剔除导致

距离比较次数增多, 另一方面来自于向量邻域中大部分邻居不满足过滤条件导致的召回率下降。

B.4 参数敏感性分析

图 7 展示了 GIST 向量数据集上 4 个指标: 索引构建时间(a)、查询召回率(b)、每秒查询次数(c)和平均距离比较次数(d), 在 4 个类别数据集上随 2 个索引构建参数: 图最大出度 m (图 7 第 1 行)和构建时集束搜索宽度 ω_c (图 7 第 2 行)的变化情况。可以观察到:

(1) 随 m 和 ω_c 增大, 索引构建时间呈现上升趋势。 $m = 24, \omega_c = 256$ 处出现了构建速度的持平, 即分别与 $m = 16$ 和 $\omega_c = 192$ 构建时间相近。原因是这两种参数设置下达到了近邻质量和构建时候选邻居选取效率的平衡。

(2) m 越大, 召回率越高, 同时由于距离比较次数增加出现查询速度下降, 因为在同样的搜索宽度下邻居越多就能越早找到更多近邻, 但会带来更多前期无效的距离比较。

(3) ω_c 在 192 和 256 设置下召回率趋于峰值, 并在之后出现了轻微召回率损失($n_c = 10$)。因为过大的集束宽度可能导致更远但不违反 RNG 性质的邻居加入, 能够更快接近查询邻域, 有利于相同 ω_s 下更少的距离比较但会带来轻微的准确度下降。

附录 C 相关工作

已有过滤向量索引根据过滤类型可分为 3 类: 通用过滤索引^{[12][22][34]}、范围过滤索引^{[6][15][17]}、标签过滤索引^{[5][19][20]}。作为通用过滤索引的代表, ACORN^[12]通过扩展邻居列表建立更稠密的近邻图, 并在搜索时通过二阶邻居绕过无法满足过滤条件的向量; iRangeGraph^[6]将数据集通过区间树划分后在搜索阶段动态选择符合范围过滤条件的邻居, 从而实现在该过滤条件下模拟先知近邻图的搜索; UNG^[5]为了解决标签过滤问题, 通过标签集合的子集与超集关系建立有向无环图, 将具有相同标签集的向量归为一组并给每一组构建单独的 Vamana^[24]索引, 然后通过向直接子节点构建跨组边实现每组之间的联通。目前仍然缺少类别过滤的专用索引, 除本文提出的算法外, 可以使用通用过滤索引(ACORN), 或建立标签集和类别间的映射并使用标签过滤索引(UNG)间接回答类别过滤查询。

为了解决带过滤条件的近似最近邻搜索问题, 两种简单的搜索策略被提出: 前过滤(Pre-Filtering)和后过滤(Post-Filtering)^{[22][29]}。前过滤需要首先筛选出满足过滤条件的向量。因为通过筛选的向量上并未建立向量索引, 所以需要逐个与查询向量比较找到近邻。后过滤首先基于在完整数据集上构建的向量索引抽取多于 k 个中间结果, 再过滤掉不满足条件的向量。前过滤在满足条件的向量数量较多(过滤选择性强)时表现较差, 而后过滤在过滤选择性弱时面临大量中间结果被过滤而使得最终结果少于 k 。因此, 为了应对不同过滤选择性, 已有向量检索系统^{[22][29][39]}通过查询优化器根据过滤选择性来动态选择前过滤方法或后过滤方法。然而, 它们继承了前过滤和后过滤的性能瓶颈, 当过滤选择性适中时, 两种方法的性能均无法达到最优^[6]。为此, 专用过滤向量索引被提出。基于图索引的出色性能, 大部分过滤向量索引基于已有图索引做优化^{[5][6][12][15][17]}, 希望能够拟合先知近邻图^{[6][12][15]}的查询

性能。先知近邻图是在所有满足查询过滤条件的向量上构建的近邻图。由于为每个类别构建近邻图需要大量空间和时间^{[12][15]}(例如 $|C| = 100$, 就需要构建 100 个先知近邻图), 因此该方法仅可用来评估理论上专用过滤图索引能够达到的最优查询性能。

向量索引主要分为 3 大类别, 分别为基于哈希的索引^{[7][18][41][42]}, 基于分组的索引^{[10][14]}, 以及基于图的索引^{[1][8][23][24][25][26][28]}。它们的共同目标是通过尽可能少的向量距离计算抽取尽可能多的精确近邻向量, 优化高维向量上距离计算这种耗时的操作^[36]。基于图的索引结构因其较高的准确性和高效的查询性能被多个向量数据库^{[21][22]}和向量检索库^{[27][33]}深度集成。其中, HNSW^[25]、NSG^[26]、NSSG^[28]、Vamana^[24]等图索引结构通过近似构建相关邻居图(Relative Neighbor Graph, RNG)^[37]或单调查询网络(Monotonic Search Network, MSN)^[26]来刻画向量之间的近邻关系。它们都使用了 RNG 剪枝或其变体, 通过多样化邻居分布来建立长距离导航边以避免查询时陷入局部邻域^{[11][38]}。具体而言, RNG 中任意两点 $r, s \in V$ (RNG 的点集) 之间能够连边的充分必要条件是 $\forall t \in V, \delta(r, s) < \delta(r, t) \vee \delta(r, s) < \delta(s, t)$ 。图上的搜索过程是一种通过查询向量引导的集束搜索算法, 搜索时从入口点出发沿搜索路径逐渐接近目标向量空间^{[1][23]}。