软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(3):1084−1103 [doi: 10.13328/j.cnki.jos.007519] [CSTR: 32375.14.jos.007519]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向批量更新的向量索引召回率优化*

王 可[1], 胡思劼[1], 胡卉芪[1], 赵明昊[1], 魏 星[2], 屠要峰[2], 周 烜[1]

[1](华东师范大学 数据科学与工程学院, 上海 200062)

[2](中兴通讯股份有限公司, 广东 深圳 518057)

通信作者: 赵明昊, E-mail: mhzhao@dase.ecnu.edu.cn

摘 要: 近似最近邻搜索 (approximate nearest neighbor search, ANNS) 是支撑向量数据库、推荐系统及大语言模型等上层应用的关键技术. 其中, 分层可导航小世界 (hierarchical navigable small world, HNSW) 图索引通过构建层级化结构, 迅速定位结果至目标区域, 从而以较低的计算成本实现较高的检索召回率. 然而, 现有 HNSW 算法主要面向静态数据检索场景而设计, 而忽略了数据更新对检索性能的影响. 通过对现实数据集的研究发现, 向量数据库中的数据通常以批量方式进行更新, 其相似特性会削弱 HNSW 算法中启发式剪枝的有效性, 并诱发相似向量连接的稀疏化问题, 共同造成查询召回率的显著下降. 针对上述问题, 提出一种基于图结构局部调整的自适应细粒度剪枝策略, 构建了融合识别与修复机制的优化方案. 首先, 在识别阶段, 通过计算区域邻居距离量化局部拓扑密度, 从而精准定位待干预的致密区域. 其次, 在修复阶段, 针对处于致密区域的枢纽节点, 采用双重剪枝的邻居选择策略: 协同应用原生的与修正的启发式剪枝规则, 合并两种规则的结果集以在保证检索精度的同时提升邻居连接的多样性, 有效缓解过度剪枝与连接稀疏化问题. 在多个公开数据集上的实验结果表明, 所提方法对数据更新频繁的场景具备良好的适应性, 在维持查询延迟和吞吐量稳定的前提下, 实现了 1%–4% 的召回率提升.

关键词: 近似最近邻搜索; 向量检索; 图向量索引

中图法分类号: TP311

中文引用格式: 王可, 胡思劼, 胡卉芪, 赵明昊, 魏星, 屠要峰, 周烜. 面向批量更新的向量索引召回率优化. 软件学报, 2026, 37(3): 1084–1103. http://www.jos.org.cn/1000-9825/7519.htm

英文引用格式: Wang K, Hu SJ, Hu HQ, Zhao MH, Wei X, Tu YF, Zhou X. Vector Index Recall Optimization for Batch Updates. Ruan Jian Xue Bao/Journal of Software, 2026, 37(3): 1084–1103 (in Chinese). http://www.jos.org.cn/1000-9825/7519.htm

Vector Index Recall Optimization for Batch Updates

WANG Ke[1], HU Si-Jie[1], HU Hui-Qi[1], ZHAO Ming-Hao[1], WEI Xing[2], TU Yao-Feng[2], ZHOU Xuan[1]

[1](School of Data Science & Engineering, East China Normal University, Shanghai 200062, China)

[2](Zhongxing Telecommunication Equipment Corporation, Shenzhen 518057, China)

Abstract: Approximate nearest neighbor search (ANNS) is a foundational technology supporting applications such as vector databases, recommendation systems, and large language models (LLMs). Among these, the hierarchical navigable small world (HNSW) graph indexing technique constructs a hierarchical structure to quickly locate results within the target region, thus achieving high retrieval recall at low computational cost. However, existing HNSW algorithms are primarily designed for static data retrieval scenarios and fail to account for the impact of data updates on retrieval performance. Through research on real-world datasets, it is found that data in vector databases is typically updated in batches, and their similar characteristics weaken the effectiveness of heuristic pruning in HNSW

* 基金项目: 国家重点研发计划 (2023YFC3341200); 中兴通讯产学研合作基金 (IA20250625030)
王可和胡思劼为共同第一作者.
本文由“向量数据库及 DB4LLM 技术”专题特约编辑高宏教授、李国良教授、张蓉教授推荐.
收稿时间: 2025-05-07; 修改时间: 2025-06-30, 2025-08-14; 采用时间: 2025-08-20; jos 在线出版时间: 2025-09-02
CNKI 网络首发时间: 2026-01-08

algorithms and lead to sparsification issues in the connections among similar vectors, collectively causing a significant decline in retrieval recall. To address these issues, this study proposes an adaptive fine-grained pruning strategy based on local adjustments to the graph structure and constructs a comprehensive optimization scheme that integrates an identification and repair mechanism. First, in the identification phase, the regional neighbor distance is calculated to quantify local topological density, thereby precisely locating the dense regions requiring intervention. Second, in the repair phase, for hub nodes in dense regions, a dual pruning neighbor selection strategy is adopted: native and modified heuristic pruning rules are applied synergistically, and the results of both rules are merged to enhance neighbor connection diversity while maintaining retrieval accuracy, effectively alleviating over-pruning and connection sparsification issues. Experimental results on multiple public datasets show that the proposed method demonstrates good adaptability in scenarios with frequent data updates, achieving a 1%–4% improvement in recall while maintaining stable query latency and throughput.

Key words: approximate nearest neighbor search (ANNS); vector retrieval; graph-based vector index

向量的近似最近邻搜索 (approximate nearest neighbor search, ANNS)[1−6]致力于在大规模高维数据 (即高维向量) 中迅速获取目标向量的最近向量, 是当代人工智能应用, 特别是大模型相关应用中的重要技术. 在当今主流 AI 系统中, 多模态异构数据 (如文本、图像、音视频等非结构化数据)[7]首先经由向量嵌入 (vector embedding)[8−9]技术将其语义抽取表征为高维向量, 向量间的几何距离 (如欧氏距离、余弦相似度等) 表征了向量间的语义关系. 在这种模式下, 在多模态数据中检索语义相近的数据的任务即转化为了向量间的最近匹配. 由于高维空间中最近邻匹配的计算复杂度较高, 近似最近邻 (ANN) 以匹配精度换取计算的高效性, 提升了算法在 AI 应用中的可行性. 向量数据库[10,11]存储与管理向量数据, 集成 ANNS 等向量检索算法, 现已成为前沿大模型应用的重要数据基座. 例如, 在生成式大语言模型 (generative large language model)[12,13], 检索增强技术 (retrieval-augmented generation, RAG)[14−17]通过向量检索从外源数据获取相关语义来增加 LLM 的生成和推理能力, 有效地解决了大模型的幻觉[18]问题, 显著提升大模型生成内容的质量.

为了实现高效的 ANNS, 学术界与工业界已探索出多种索引方法, 主要分为基于树 (tree-based)[19,20]、基于图 (graph-based)[21−23]、基于哈希 (hash-based)[24,25]、基于聚类 (clustering-based)[26]和基于量化 (quantization-based)[27,28]等几类. 其中, 基于图的 ANNS 方法因其在查询效率、召回精度与可扩展性之间取得了卓越的平衡而备受青睐, 该类方法构建并利用向量间的近邻图进行快速的图导航. 作为该领域的代表性算法, 分层可导航小世界 (hierarchical navigable small world, HNSW)[29]通过引入层级化的图结构与可导航小世界网络, 实现了以较低的计算开销达到较高的召回率, 已成为当前应用最为广泛的图索引方法之一.

然而, 尽管现有的 ANNS 算法在通用场景下性能优异, 但在向量数据频繁写入场景中, 其性能面临显著挑战. 这一挑战在许多现实应用中尤为突出, 例如在知识库构建或实时推荐系统中, 新数据常以批量形式集中写入, 其批内向量因其相似的来源或主题, 在空间中呈现高度的局部聚集性. 在这种批量插入相似数据模式下, HNSW 索引的召回率会发生显著下降, 经过本文实验测试, 在 GIST1M 数据集上, 最大降幅达到 9%. 经研究发现, 其召回率退化的核心原因可归结为两点: 其一是 HNSW 选择邻居关系时的所采用的启发式规则引发的过度剪枝, 即在处理过度密集的局部区域时, 该规则会误判冗余连接, 从而错误地移除能提供关键搜索方向的邻居; 其二则是相似向量连接稀疏化的问题, 即批量插入的相似数据节点难以建立足够数量的有效邻居连接, 从而在局部形成了导航能力严重受损的稀疏子图, 导致索引无法支撑起有效的全局导航. 正是由于缺乏对索引结构性问题的有效感知, 现有索引方法[30−32]在处理相似向量的批量插入时普遍暴露其性能局限, 最终导致召回率的显著下降. 鉴于此, 在不影响性能的前提下, 如何设计一种能够有效应对上述结构性问题的索引优化方法, 保证向量检索在动态场景下的召回率, 已成为一个关键问题.

为解决过度剪枝和相似向量的连接稀疏化引发的召回率退化问题, 本文提出一种基于图结构局部调整的自适应细粒度剪枝策略. 该策略通过多阶段的识别与修复框架, 在索引构建过程中对图的局部拓扑进行精准干预. 其核心机制包含以下 3 个环节.

(1) 数据驱动的局部致密区域识别. 为精准应对过度剪枝问题, 该策略通过数据驱动的预分析过程, 自适应地设定识别局部密度的阈值 β. 在此基础上, 根据节点的区域邻居平均距离与全局基线的差异来识别过度致密区域.

(2) 双重剪枝的邻居选择策略. 对致密区域的节点, 协同应用原生剪枝规则与修正的 α 剪枝规则, 通过融合两种规则所选择的候选邻居集合, 实现精度保障与探索强化的双重目标. 该方法有以下两方面的收益: 一方面维持关键局部近邻, 以提高搜索准确性; 另一方面引入结构多样性, 增强对复杂连接模式的探索能力.

(3) 基于枢纽识别的邻居补充. 为保留质量较高的邻居, 识别出应用原生剪枝规则获取的邻居结果中的枢纽节点, 将具有丰富连接的枢纽作为候选邻居. 此操作有助于提升局部拓扑的质量.

本文的主要贡献概括如下.

(1) 深入分析了 HNSW 索引在批量插入相似向量时的召回率退化问题. 通过深入的实验验证, 揭示了索引性能下降的原因: 由局部区域过度致密所引发的过度剪枝效应以及相似向量的连接稀疏化.

(2) 针对上述问题, 提出了一种基于图结构局部调整的自适应细粒度剪枝策略. 该策略构建完整优化框架, 通过数据驱动识别、双重剪枝的邻居选择以及基于枢纽识别的邻居补充方法, 有效地修复相似向量批量插入导致的结构损伤. 通过动态维护 HNSW 索引的邻居关系, 提升召回率.

(3) 在多个公开数据集上, 设计并进行详尽的实验以验证本文的优化方案. 实验结果表明, 在批量更新场景下, 相较于原 HNSW 方法, 本文的优化方法在未引入显著查询开销的前提下, 将召回率稳定提升 1%–4%. 实验验证了本文方法的有效性与高效性.

本文第 1 节描述背景和相关工作. 第 2 节通过实验发现问题, 并对现象和实验数据进行深入分析. 第 3 节将详细阐述本文所提出的索引结构优化方法及其算法实现. 第 4 节进行实验评估. 第 5 节总结本文.

1 研究背景与相关工作

本节介绍本文的研究背景与相关工作. 第 1.1 节对近似最近邻搜索和向量索引进行概述, 第 1.2 节重点聚焦于基于图的向量索引, 系统性地梳理其发展脉络与核心技术, 第 1.3 节综述与 HNSW 索引召回率优化相关的关键工作, 为本文提出的优化方法奠定基础.

1.1 近似最近邻搜索和向量索引

随着当代人工智能技术的发展, 向量嵌入已成为表征文本、图像、音视频等复杂非结构化数据的标准范式. 在这种范式下, 现实世界中的用户推荐、语义相似性检索以及多模态交互[33]等复杂场景, 能够被高效地转化为高维向量空间中的几何距离计算问题. 然而, 在高维空间中进行精确的暴力搜索会遭遇维度灾难, 其计算复杂度随着数据规模和维度的增长而急剧上升, 无法满足现代人工智能系统对低延迟、高吞吐的应用需求.

为应对这一挑战, 近似最近邻搜索技术应运而生. 其核心思想是, 在可控的精度损失范围内, 在大规模高维数据中高效地检索与查询点最接近的向量数据. 其定义如下: 在度量空间 $\mathbb{R}^d$ 中, 给定数据集 $S=\{x_1,x_2,\ldots,x_N\}\subseteq\mathbb{R}^d$, 查询 $q\in\mathbb{R}^d$ 以及近似因子 c. ANNS 的目标是找到一个或多个点 $x^*\in S$ 满足:

$$dist(q,x^*)\leqslant c\cdot dist(q,x_{\text{true}}) \tag{1}$$

其中, $dist(.,.)$ 是空间 $\mathbb{R}^d$ 的距离度量 (如欧氏距离、余弦相似度等); x_{true} 是 q 在 S 的精确最近邻.

向量索引是支撑高效近似最近邻搜索的核心数据结构. 其根本目标是通过对数据集进行预处理, 构建一种能够极大加速查询过程的辅助性结构. 在索引构建阶段, 向量索引将原始高维向量组织起来, 例如, 通过建立向量数据之间的拓扑邻接关系 (如图索引) 或空间划分关系 (如树索引、倒排索引). 这种预先建立的结构, 使得在查询阶段可以应用高效的剪枝或路由策略, 从而避免对整个数据集进行暴力扫描, 将搜索范围剪枝至一个很小的高相关性候选子集中, 最终实现数量级的查询加速.

1.2 基于图的向量索引

在前述多种向量索引中, 基于图的索引已成为当前最受关注和应用最为广泛的一类方法. 其核心思想是将数据集中的向量建模为一个邻近图 $G=(V,E)$, 其中顶点集 V 代表数据向量, 边集 E 则表示向量间的邻近关系. 通过这种方式, 近似最近邻搜索问题被转化为在图 G 上的遍历搜索问题. 相较于其他索引方法, 图索引因其在查询效率、召回精度与可扩展性之间取得了卓越的性能均衡, 而成为学术界与工业界的主选方案. 图索引的应用包含构

建与查询两个核心阶段.

在图的构建阶段, 算法为数据集中的所有节点构建高效导航的近邻图. 该构建过程是增量式的: 新节点 v 被逐一插入图中, 并为每个 v 执行邻居查找与连接操作. 具体而言, 邻居查找过程复用了查询阶段的贪心搜索机制, 即从图的随机入口点出发, 迭代搜索以定位一组距离 v 最近的候选邻居集. 随后, 算法从候选邻居中筛选预设数量的节点作为最终邻居, 并建立从 v 指向它们的有向边. 为了保证索引的存储开销与查询复杂度维持在可控范围内, 每个节点的最大出度受一个固定上限值约束. 重复此流程直至所有数据点入图, 即完成了完整近邻图的构建.

基于图的查询阶段则充分利用已构建的图结构实现高效检索, 其贪心寻路过程如图 1 所示. 查询通常从整个图中所有节点中随机选取的一个入口点开始, 迭代访问当前节点的邻居集. 每步迭代中, 算法选择离查询点更近的未访问邻居作为下一个当前节点, 并维护容量有限的动态候选结果集. 当邻居中不存在比候选集中最远点更接近查询点的新节点时, 搜索收敛并返回候选集作为近似最近邻结果. 这种基于图的贪心寻路策略通过局部搜索快速收敛至目标区域, 无须遍历全图, 实现高效率检索.

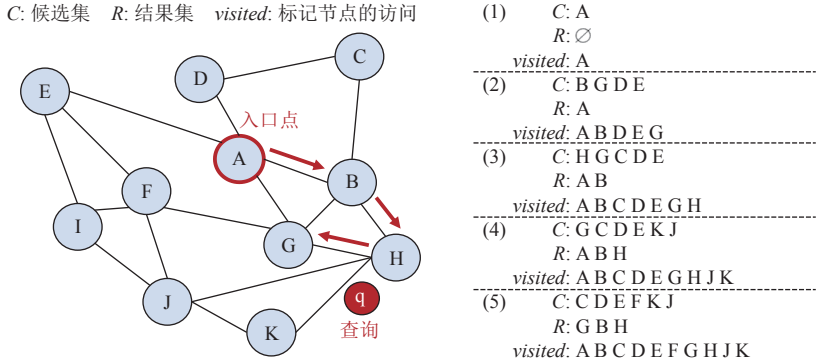


图 1 图向量索引的查询过程

基于上述基本原理, 图向量索引经历了一系列演进. 早期的探索以 Delaunay 图为代表, 该方法虽能保证搜索的精确性, 但其过高的构建复杂度使其难以应用于大规模数据集; 为缓解时间复杂度与搜索精度的固有矛盾, Malkov 等人提出了以构建小世界网络特性的邻近图为特征的 NSW 索引, 其长链接负责实现高效的全局路由, 而短链接则保障了在目标区域内的局部精确查找, 从而首次在查询效率与召回精度间取得了良好平衡; 此后的研究大多聚焦于如何构建拓扑更优的邻近图, 其中最具影响力的是 HNSW 索引.

除了在算法层面优化索引拓扑以提升效率与精度外, 索引的可扩展性, 特别是如何将图索引部署于超过单机内存的海量数据集上, 是该领域的另一大挑战. 为应对此挑战, 研究人员提出了多种基于外存的图索引方案. 其中, 以 DiskANN[34−36]和 SPTAG[37]为代表的工作影响最为深远, 其核心思想在于构建一种内存-磁盘混合架构: 在内存中仅保留一个小型的稀疏导航图, 而将包含完整连接信息的全量数据存储于磁盘. 查询时, 首先利用内存中的导航图快速定位至磁盘上的一个或多个粗粒度区域, 随后通过精心设计的磁盘 I/O 策略, 高效地读取局部图结构以完成最终的精确查找. 这种分层处理机制, 以可控的 I/O 开销与精度损失为代价, 成功地将图索引的应用扩展至 10 亿规模的向量数据集.

在众多基于图的内存索引中, 由 Malkov 等人[29]提出的 HNSW 索引结合了层级化图结构与高效的邻居选择策略, 在性能上取得了重大突破, 已成为当前应用最为广泛的向量索引之一. HNSW 的卓越性能, 主要源于以下两大核心机制.

● HNSW 的层级化图结构. 为加速搜索过程, HNSW 引入了一种多层图的组织方式. 如图 2 所示, 该结构以包含了所有数据点的底层稠密图 (第 0 层) 为基础结构, 保证了查询的高召回率. 同时, 通过对下层节点进行概率性采样, 构建出上层稀疏图作为实现远距离跳转的快速路径. 在插入新节点时, 算法会依据一个指数衰减的概率分布函数为其随机指派一个最大层数 $l_{\max}$, 并将该节点添加至从 0 到 $l_{\max}$ 的每一层图中. 该层数通过对一个服从指数

分布的随机变量进行采样而生成: 令 u 为一个服从标准均匀分布的随机变量, 即 u~U (0,1), 最大层数 $l_{\max}$ 的计算方式为:

$$l_{\max} = \lfloor -\ln u \cdot mL \rfloor \tag{2}$$

在公式 (2) 中, $\lfloor \cdot \rfloor$表示向下取整函数, 参数 mL 是控制层数分布的归一化因子, 通常取 $mL = 1/\ln M$, 其中, M 为各节点可连接的最大邻居数. 这种概率分配机制确保了绝大多数节点仅存在于底层, 而只有极少数节点能被选入高层网络. 该多层设计使得搜索可以从顶层稀疏图的高效导航开始, 逐步深入至底层稠密图进行精确查找, 从而将搜索复杂度由线性级别 O(N) 成功降低至对数级别 O(log$N$).

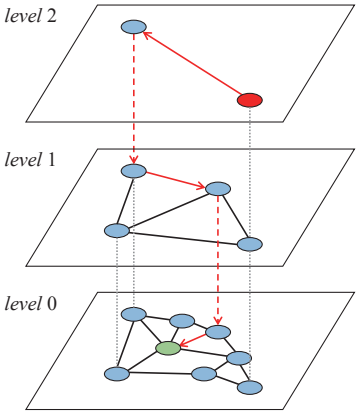
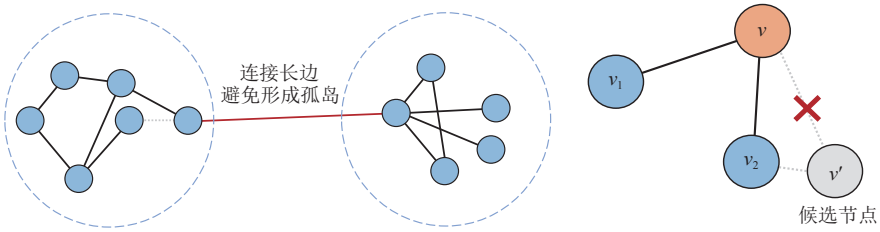


图 2 HNSW 索引查询过程

● HNSW 的启发式邻居选择规则. 为了保证图的导航质量, 尤其是在节点分布不均的区域, HNSW 设计了一种启发式邻居选择规则. 该规则的核心思想在于最大化所选邻居集的空间覆盖多样性, 而非仅追求距离上的绝对最近. 如图 3 所示, 其算法流程如下: 在为待插入节点 v 从一个候选邻居集 C 中筛选最终邻居时, 算法首先会判断 C 的基数$|C|$. 只有当$|C|$大于预设的节点最大邻居数时, 才实施启发式剪枝机制, 否则就无须剪枝. 此时, 算法将迭代地构建一个结果集 R. 在每一轮迭代中, 算法从 C 中选取当前距离 v 最近的节点并将其加入 R. 随后, 算法将对 C 中剩余的候选节点应用剪枝规则: 对于 C 中任何一个候选节点 v', 若存在结果集 R 中的已选邻居 v_i, 使得其与 v_i 的距离 $dist(v', v_i)$ 以及其与待插入节点 v 的距离 $dist(v', v)$ 满足:

$$dist(v', v_i) > dist(v', v) \tag{3}$$

在公式 (3) 中, 公式不成立时, v'被视为一个冗余的候选邻居, 算法将其从 C 中移除. 此过程反复进行, 直至结果集 R 的大小达到最大邻居数上限且满足提前终止条件.



(a) 索引的孤岛效应 (b) 启发式剪枝规则

图 3 保证图索引质量的关键

此规则在选择邻居时考虑区域覆盖 (不同邻居有一定的方向和范围辐射), 避免在局部区域出现大量冗余连接, 从而保证图高效与高质量的探索能力. 然而, 也正是因为这一规则, 在面临批量插入相似数据场景时, 可能会因

候选邻居集呈现致密的特性而发生系统性误判, 其后果是出现过度剪枝现象, 从而制约了索引在该类场景下的精准度.

● HNSW 的构建、查询流程与索引参数. 基于上述两大机制, HNSW 的索引构建与查询处理得以高效执行. 在构建索引时, 一个新节点从顶层图的入口点出发, 逐层进行贪心搜索以定位其在每一层中的插入位置, 并利用前述的启发式规则建立邻居连接; 查询过程与构建过程类似, 查询向量从顶层入口点开始, 以贪心策略迭代, 于多层图中逐步下降, 直至在第 0 层图中完成最终的近似近邻搜索并返回结果. HNSW 索引的性能主要与以下 3 个关键参数相关: *M*, 即单节点的最大邻居连接数, 其控制索引密度以平衡检索精度与计算效率; *efConstruction*, 简称 *efC*, 即建图时动态候选集大小, 其决定了图的质量与构建成本; *efSearch*, 即查询时动态候选集大小, 其直接权衡召回率与查询延迟. 此外, *mL* 参数, 即层级生成参数, 用于控制节点层数分布.

1.3 影响 HNSW 索引召回率的相关工作

HNSW 索引的召回率受到其内在的静态拓扑结构、构建参数, 以及外在的数据、查询动态特性等多重因素的复杂影响. 本节将围绕这两大方面, 对相关前沿工作进行综述.

大量研究聚焦于索引的静态拓扑结构与构建参数, 旨在通过优化图的内在属性来提升其召回性能上限: (1) 在图的拓扑连通性层面, 网络的全局连通性被认为是保证节点可达性的基础. 研究[38]指出, 图中节点的入度不足会破坏网络的全局连通性, 导致部分节点因路径缺失而难以在搜索中被发现. 类似地, HNSW 中的参数 *M* 控制着节点的邻居数量, 其值过低同样会削弱图的连通性, 从而制约召回率的上限[29]. (2) 在邻居选择与局部最优问题上, 研究[22]表明, HNSW 算法的贪心搜索特性会引发局部最优现象, 这一缺陷对其召回率形成显著约束. 为此, DiskANN[34]在其核心算法中引入参数 α 来调整长距离边的选择, 以增强跳出局部最优的能力. 这些工作表明, HNSW 的启发式邻居选择规则是决定图导航质量的关键, 优化该规则是提升召回率的核心途径之一. (3) 在索引构建的全局参数方面, Malkov 等人[29]验证了 *efConstruction* 和 *mL* 等参数的重要性, 前者决定了构建图时邻居选择的广度, 后者则影响分层结构的合理性, 二者设置不当均会导致图的初始质量下降, 进而影响最终的召回率表现.

与此同时, 另一类研究则关注动态的外部因素对召回率的潜在影响: (1) 从数据自身特性的角度来看, Elliott 等人[39]首次发现, 向量数据固有的内在维度以及其写入索引的顺序会显著影响 HNSW 的图结构, 并对召回率产生负面作用. 基于上述发现, 他们提出了一种基于内在维度计算来动态调整数据写入顺序的优化方法; (2) 关于查询的动态特性, Li 等人[40]注意到不同查询在 HNSW 中的搜索难度存在差异. 其工作通过训练机器学习模型来预测查询难度, 并对简单查询进行自适应的提前终止, 从而在保证召回率的同时优化了系统的查询延迟.

综上所述, 现有工作已从索引的静态属性与动态外部因素等多个维度, 对提升 HNSW 召回率的策略进行了相应的探索. 然而, 这些研究基于一个共同的假设: 即索引的拓扑结构在构建或优化后便相对稳定. 这种假设导致其优化策略大多属于静态、离线的场景, 或仅限于查询侧的被动调整, 因而揭示出一个关键问题: 当面向批量插入相似数据场景时, 现有研究对索引自身的结构性退化问题关注不足, 缺乏一种动态地感知并修复索引结构的内在机制.

受上述研究的启发, 本文将研究视角聚焦于批量数据更新场景下的图向量索引优化问题. 基于此视角, 本文的核心贡献在于提出并验证了一种基于图结构局部自适应调整的 HNSW 优化方法: 该方法通过在索引更新过程中赋予其局部拓扑感知与重构能力, 使其能够主动缓解因数据聚集性写入而引发的性能衰退, 从而提升 HNSW 在真实应用中的鲁棒性与召回率.

2 问题与分析

本节通过一系列对比实验, 系统性地分析 HNSW 索引在批量插入相似数据场景下的召回率退化问题. 首先, 在第 2.1 节详细阐述实验设置与数据准备, 包括数据集的选取与处理方案、构建批量负载的方法. 然后, 在第 2.2 和 2.3 节分别从宏观性能与微观拓扑两个维度, 描述与分析召回率下降与簇内连接异常等现象. 最后, 在第 2.4 节

深入地剖析这些现象, 揭示其背后的原因, 从而为第 3 节的优化方法提供理论基础与数据支撑.

2.1 实验设置

在批量插入相似数据场景下, 为分析 HNSW 索引性能下降的原因, 本节设计了对比实验, 模拟该场景来评估索引的性能和量化索引内部的结构特征. 实验从宏观性能指标与微观拓扑结构两个维度, 定量地揭示影响 HNSW 索引召回率的原因. 我们选择了多个广泛使用的代表性向量数据集, 包括 GIST1M、Msong 和 Enron (详见第 4.1 节). 这些数据集在维度、分布特性和内在结构上存在显著差异, 覆盖了较多实际应用场景. 接下来, 我们将详细阐述数据处理、负载构建与评价指标这 3 个方面的内容.

● 生成相似数据与划分数据集. 为模拟真实应用中的批量插入场景, 本节提出了构造相似数据的方法. 相似数据在较小的距离空间内存在微小变化, 因此采用一种基于维度扰动的数据生成方法. 该方法以原始数据集中的向量作为母向量, 为其生成距离上较近的相似向量. 该生成过程如下: (1) 扰动窗口: 对于选定的母向量 $v \in R^d$, 确定一个待扰动的维度子空间, 其宽度 w 由一个预设比例 r_w (默认为 0.3) 控制, w 的计算方式为 $w = \lfloor r_w \cdot d \rfloor$. 为模拟真实数据中特征重要性分布的随机性, 窗口起始维度 d_{start} 从均匀分布 U_1 中采样确定; (2) 引入有界均匀扰动: 在维度窗口 $[d_{\text{start}}, d_{\text{start}}+w)$ 内, 对每个维度添加服从均匀分布 U_2 的随机扰动, 窗口外的维度值保持不变. 该方法生成的向量为实验中的更新向量. (3) 构造查询集: 将以上扰动机制应用到原始查询集, 生成用于评估检索性能的相似查询集.

● 构建批量插入相似数据场景. 在持续写入相似数据 (上述方式产生相似数据) 时, 为精确捕捉 HNSW 索引在性能以及索引结构上的变化情况, 本节设计了一种增量式的写入负载与评估流程. 该流程实现了多次批量插入相似数据场景, 记录各级负载召回率的变化情况. 在批量插入相似向量时, 通过精确计算 HNSW 索引中数据特征的变化, 进一步分析召回率退化的原因.

实验场景和负载写入流程如下. 我们控制相似数据负载中相似数据占比, 以模拟批量写入相似数据的实际应用场景. 构建基础索引: 该过程使用原始向量构建 HNSW 索引, 未引入任何相似数据. 更新相似向量: 在基础索引的基础上, 批量插入相似数据负载. 实验定义了多级相似数据负载, 各级的相似数据占比从 1% 递增至 5%. 其中, Batch 表示在原始向量的基础上, 连续写入相似向量的场景. 各个数据集的基础向量分别如下: GIST1M 为 10 万, Msong 为 10 万, Enron 为 9 万.

在写入每个负载 (相似数据占比为 1%–5%) 后, 实验将采用相似查询集 (默认 1 000 条), 计算当前索引的召回率, 并获取邻居关系信息. 实验中设计增量式的写入负载与评估流程, 在随着相似数据在索引中占比的持续提升的过程中, 捕捉索引中呈现的召回率退化现象, 重点关注索引的以下指标: 检索召回率和相似数据邻居数 (详见第 2.3 节).

● 检索召回率. 用于衡量索引的宏观检索精度. 向量检索主要关注查全率, 即 *recall@K*, 其定义为算法返回的 *top-K* 结果集与真实 *top-K* 最近邻集之间交集的占比. 该指标直观地反映了索引成功找回真实近邻的能力.

2.2 批量插入场景下召回率的变化

本节展示了在持续插入相似数据时 HNSW 索引检索召回率的变化情况. 实验场景如第 2.1 节所述, 我们在相似数据占比为 1%–5% 负载的写入过程中测量召回率 (*recall@K*) 的变化.

图 4 展示了在插入相似数据过程中, GIST1M 数据集在索引参数为 *M*=24, *efC*=64 时召回率的变化情况. 图中每条曲线 (Batch-*efSearch*) 对应特定的查询深度参数 *efSearch* 以及 *top-K* (默认为 10), 纵坐标表示查询的召回率.

● 实验现象与数据分析. 实验结果表明, 随着相似数据占比的增加, HNSW 索引的检索召回率呈现出逐渐下降的趋势, 该现象在 3 个数据集以及不同索引参数下都普遍存在. 如图 4 所示, 当 *efSearch* 为 100 时, 相似数据占比为 1% 时的查询召回率约为 0.9, 在相似数据占比从 1% 提升至 5% 的过程中, 召回率持续下降, 最低为 0.89, 其降幅达到 9%. 即便在 *efSearch* 为 150 和 200 时, 5% 负载的召回率 (0.899 和 0.907) 相较于 1% (约 0.985) 依存在显著差距. 上述现象反映了批量插入相似数据对 HNSW 检索精度存在持续性损害.

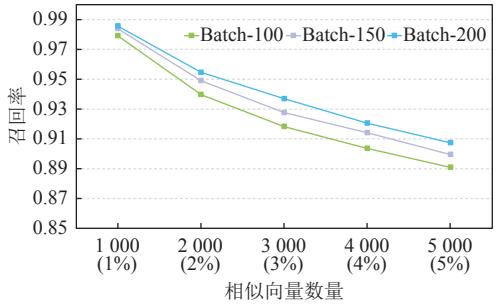


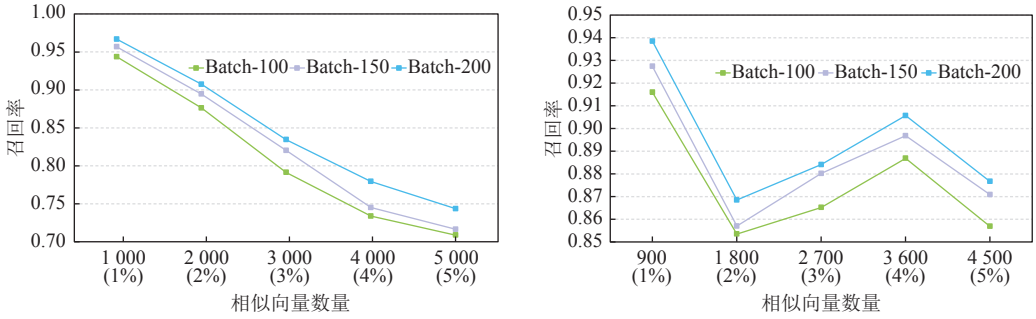
图 4　GIST1M 数据集上的平均召回率 (*M*=24, *efC*=64)

● 索引参数的敏感度分析. 实验揭示了索引构建参数对召回率退化的敏感性.

(1) 查询深度参数 *efSearch*. 增大 *efSearch* 可通过扩展搜索广度提升召回率. 例如, 在图 4 的相似数据占比为 2% 时, 将 *efSearch* 从 100 提升至 200, 召回率提升约 2%. 然而, 即便采用较高的 *efSearch* 值, 负载为相似数据占比为 1% 与 5% 时, 召回率的差距仍然十分显著. 这一结果表明, HNSW 召回率的性能损失与索引结构紧密相关, 无法仅通过增加查询时的搜索深度来弥补.

(2) 索引构建参数 *M* 与 *efConstruction*. 增大 *M* 与 *efConstruction* 可构建更高连通密度的图结构, 这会显著提升索引的鲁棒性, 并减弱性能衰减现象. 同时, 更大的 *M* 与 *efConstruction* 会增加邻居搜索的广度, 提高邻居质量, 但也会增加构建的开销.

我们在 Msong 与 Enron 数据集上进行了相同的实验. 实验结果如图 5 所示, Msong 的召回率出现平滑下降的趋势. 在 *efSearch* 为 100 时, 查询召回率从 0.95 (相似数据占比为 1%) 骤降至 0.7 (相似数据占比为 5%), 下降幅度接近 15%. 其召回率变化趋势与 GIST1M 相同, 并且下降效果更为明显; 而在维度更高且结构复杂的 Enron 数据集上, 在相似数据占比从 1% 提升到 5% 的过程中, 召回率虽然出现了波动, 我们分析了单条查询的明细数据, 发现存在部分查询向量其召回率值与平均召回率的偏差较大, 但整体趋势仍为下降. 在 *efSearch* 为 200 时, 召回率下降的幅度最大, 超过 6%.



(a) Msong召回率变化 (*M*=48, *efC*=100)　(b) Enron召回率变化 (*M*=24, *efC*=64)

图 5　不同相似数据占比下, 平均召回率下降现象

以上 3 组实验的结果都展现出召回率下降的问题, 也证实了该召回率退化是 HNSW 应对批量相似数据写入的内在结构性缺陷, 而非特定数据分布的偶然现象.

在批量插入相似数据的场景中, 实验中出现了 HNSW 索引召回率显著下降的现象, 但该现象背后的微观结构演化机制尚未明确. 基于此, 第 2.3 节将深入拓扑层面精细分析索引结构的演变, 进一步探究索引召回率退化的原因.

2.3　相似数据邻居数量的动态变化

● 相似数据邻居数. 该指标通过观测索引的微观拓扑, 量化图的局部连通性. 其定义如下: 对于批量插入相似

向量的索引, 计算该索引内相似向量节点的邻居连接数的平均值. 该指标直接反映了相似数据节点与图中其他部分建立有效连接的程度. 从理论上看, 在过度剪枝效应的影响下, 新插入的相似节点在选择邻居时会过早地丢弃大量候选邻居, 从而难以建立足够数量的多样化连接. 因此, 较低水平的平均邻居连接数, 可视为图的局部探索能力下降和结构性孤岛形成的直接表现, 亦是导致宏观召回率退化的关键微观因素.

实验结果表明, 批量插入相似数据会导致召回率的显著下降. 为探究其内在的结构性根源, 本节将聚焦于图的微观拓扑层面, 通过观测相似向量的平均邻居数量 (向量的平均邻居连接数) 这一指标, 来量化分析索引内部连接模式的动态演变. 表 1 展示了多个数据集中相似向量更新时节点的平均邻居连接和较小邻居数量向量的占比情况. 其中, 相似向量邻居数量较小向量的占比表示: 相似向量中, 向量的邻居数量小于等于 3 的向量的占比. 同理, 非相似向量邻居数量较小向量的占比代表非相似向量中的该特性.

表 1　向量邻居数量分析

数据集	索引参数	相似向量平均邻居数量	非相似向量平均邻居数量	相似向量邻居数量较小向量的占比 (%)	非相似向量邻居数量较小向量的占比 (%)
GIST1M	M=24, efC=64	6.4	9.0	60	29
Enron	M=24, efC=64	9.0	9.8	53	22
Msong	M=48, efC=100	4.6	14.5	95.7	37

随着相似数据在索引中持续累积, 新插入的相似数据节点的连接数呈现较低水平, 低于基础索引的平均邻居数. 例如, Msong 相似数据的平均邻居数为 4.6, 而基础向量的平均邻居数为 14.5, 相似向量的邻居数量较小向量的占比达到了 95.7%, 说明相似向量多数向量邻居数量都较小, 少数向量连接数量较多, 这种连接势必会影响图结构的连通性. 在 GIST1M 和 Enron 数据集上, 相似向量的平均邻居数和相似向量的邻居数量较小向量的占比都低于基础向量.

这一现象表明, 在新插入相似数据时, 索引中已有的相似数据会影响其所选邻居的质量, 并产生了显著的抑制作用. 这种微观拓扑上的连接受损现象, 与第 2.2 节的召回率退化趋势相吻合. 在更大的索引参数下, 同样表现出平均邻居数随相似数据占比增加而下降的趋势, 只是其稀疏程度减弱. 此现象进一步印证局部连通性受损的问题普遍存在, 难以仅通过调优构建参数来规避. 综上所述, 实验结果为召回率退化问题提供了更为直接的证据. 它揭示了在召回率下降的同时, 索引局部的连通性有所减弱.

2.4　召回率下降的原因分析

第 2.2 和 2.3 节的实验结果表明, 在批量插入相似向量场景下, HNSW 索引的召回率呈现显著下降的趋势; 描述邻居关系的微观数据显示, 相似数据聚集区域的节点平均邻居数量较其他区域显著偏低. 本节将深入剖析上述现象背后的成因.

● HNSW 启发式规则的系统性误判与过度剪枝. HNSW 的启发式规则的设计初衷是最大化邻居集的多样性. 然而, 当大量相似向量被集中写入索引时, 节点的候选邻居集呈现出局部过度致密的特性, 即邻居之间的距离普遍较小. 在这种高度同质化的候选环境中, 该启发式规则会发生系统性误判: 算法倾向于保留少数空间区域无重叠的邻居, 将大量连接错误地识别为冗余连接并予以剪除, 而这些连接可能提供新的探索方向.

如图 6 所示, 过度剪枝的行为切断了图中潜在的有效搜索路径. 图 6(a) 表示为向量 G 选取邻居的过程, D 已经成为 C 的邻居, 向量 G 在候选集合中, 其中, 虚线所标注的浅绿色区域为向量 D 的辐射区域, 当出现 GC>GD 的情况时, GC 不会连接, 即 G 未成为 C 的邻居. 同理, 图 6(b) 是为向量 D 选取邻居的过程, E 已经成为 D 的邻居, 向量 G 在候选集合中, 当出现 GD>GE 的情况时, GD 不会连接, 即 G 无法成为 D 的邻居. 图 6(c) 展示了从向量 A 出发查询目标点 G 时的查询路径, 其搜索路径到达节点 E, 但 E 是更远的点, 因为缺乏 CG 与 DG 这些连接边, 可能会陷入局部最优, 因此 E 未被探索, 最终导致召回失败. 在批量插入相似数据时, G 周围的更新都面临这种剪枝的影响.

● 相似向量的连接稀疏化. 在批量插入相似数据的过程中, 过度剪枝会被不断放大并累积, 逐步演变为索引结

构缺陷, 即相似向量的连接稀疏化. 该问题的形成的原因如下: 在写入较多相似向量时, 因过度剪枝导致初始邻居连接稀疏; 在后续插入向量的过程中, 当向量搜索过程收敛至该低连通区域时, 进一步加剧了过度剪枝效应, 新节点邻居的质量又进一步受到了抑制.

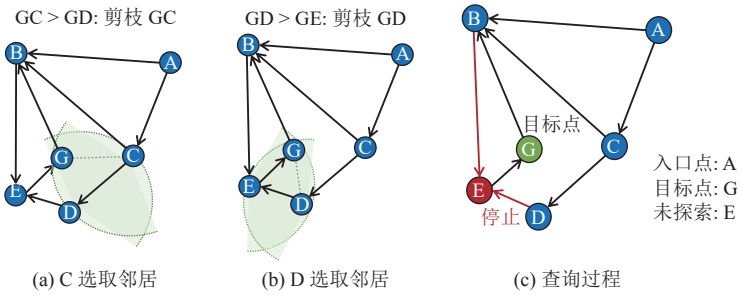


图 6 启发式规则造成的过度剪枝

当过度剪枝和连接稀疏化问题持续地累积, 最终会导致局部子图呈现出两个拓扑缺陷: 其一, 内部连接不足, 正如第 2.3 节的实验数据所示, 区域内节点的平均邻居连接数远低于邻居数量上限 M; 其二, 外部连接缺失, 即节点因邻居选择高度局限于簇内, 而难以建立起指向外部区域的关键性长连接, 阻碍了全局导航. 这一连接稀疏的局部造成了性能瓶颈: 当查询路径进入该区域后, 因缺乏多样的路径选择, 贪心搜索易陷入局部最优或过早终止, 从而解释第 2.2 节观测的召回率持续下降现象.

3 基于图结构局部调整的自适应的细粒度剪枝策略

第 2 节, 我们揭示了批量插入相似数据时 HNSW 索引结构出现的问题: 过度剪枝和相似向量连接稀疏化, 这也是该场景下索引的性能瓶颈. 针对此问题, 本文提出一种基于图结构局部调整的自适应细粒度剪枝策略. 该方案从局部拓扑入手以应对过度剪枝问题, 其核心设想在于设计一种能依据局部数据密度进行动态调整的自适应细粒度剪枝策略. 方法的核心遵循识别与修复模式: 通过数据驱动的预分析过程, 精准定位待干预的致密区域, 该区域用于识别出相似数据聚集的区域, 是划分相似向量与基础向量的依据. 针对该致密区域内的节点, 实施双重剪枝策略. 双重剪枝策略包括以下要点: 应用修正规则, 选择出候选邻居, 修正规则是放宽向量插入过程中邻居选择的剪枝的限制条件, 有效抑制过度剪枝效应; 筛选出应用原规则时的枢纽节点, 即邻居数高于平均水平的向量, 并将其与修正后的候选邻居合并, 保留潜在的有效连接. 该策略显著优化了稀疏连接下邻居的质量, 从而缓解连接稀疏化问题, 实现邻居选择中精度与多样性的有效权衡. 双重剪枝的邻居选择策略, 核心在于协同应用原生与修正的两种剪枝规则, 并合其结果, 识别并保留原生规则中的枢纽向量, 以提高邻居连接的质量与数量.

3.1 局部致密区域识别方法

首先, 识别方法需要准确的量化局部拓扑的密度, 精准识别出因相似数据聚集而形成的局部致密区域, 才能自适应的应用剪枝策略. 一种直接的识别方式是: 定义待插入节点的一跳邻居平均距离, 即该节点与其所有候选邻居间的平均距离 (图 7(a)). 然而, 该方式受候选集中异常点 (距离过小或过大) 的扰动较大, 严重影响识别结果的准确性和稳定性.

为此, 本文提出一种更具鲁棒性的识别指标, 局部平均距离, 即区域邻居平均距离 (*area_mean_dist*). 其核心思想为, 通过聚合节点邻域的整体拓扑信息, 而非只使用该节点自身的连接信息, 以评估局部区域的密度 (图 7(b)). 具体计算方式如下: 计算待插入节点候选邻居的一跳邻居平均距离, 然后计算它们的平均值. 该度量机制能够有效平滑因异常候选点引入的噪声. 该方式综合考虑了整个候选邻域的拓扑稠密特征, 具有一定的鲁棒性, 从而能够更准确地识别出向量的真实聚集区域, 减少误判的概率.

全局平均距离定义为全局邻居平均距离 (*global_mean_dist*[l]), 即对于索引的特定层级 l, 所有已存连接的长度的算术平均值. 为了保证效率, 系统仅为每一层级 l 记录当前所有连接的总数和总距离长度, 就可以增量方式计算

该全局距离. 在新增一条连接时, 仅需以 O(1) 的复杂度更新这两个统计量. 该方式使得全局平均距离通过较少计算获得, 避免了重新计算全局距离的开销.

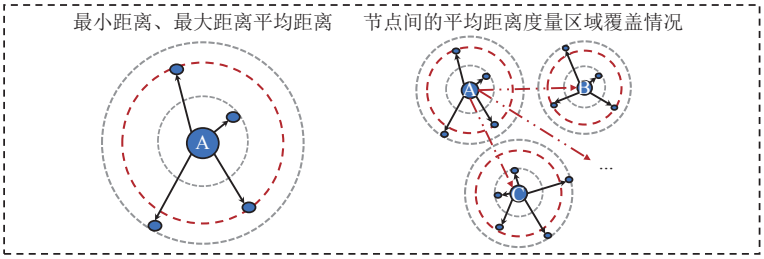


图 7 局部过度致密区域的识别方法

基于以上指标, 我们设定了区域邻居平均距离与全局平均距离的比较规则, 用于识别低密度区域. 具体识别方式如下: 新插入节点 v 的区域邻居平均距离 $area\_mean\_dist$ 与该层的全局平均距离 $global\_mean\_dist[l]$ 满足不等式:

$$area\_mean\_dist < \beta \cdot global\_mean\_dist[l] \tag{4}$$

符合公式 (4) 则判定节点 v 当前处于局部致密区域. 对于新写入的向量, 在满足公式 (4) 时, 为其应用修正的启发式剪枝规则.

识别的精准度与阈值参数 β 的设定紧密相关, 这也直接决定了算法对局部致密区域的敏感度. β 为控制识别敏感度的超参数, 数据无关的 β 值难以感知不同数据集的内在分布特性. 为此, 本文应用数据驱动的 β 值自适应选择方法: 在构建索引时, 从向量中抽取一定量样本数据, 并计算每个样本向量的区域邻居平均距离 $area\_mean\_dist$ 与全局邻居平均距离 $global\_mean\_dist[level]$ 的比值. 然后, 分析数据的分布情况, 选择合适的分位数作为 β 的取值. 该方法使得 β 不再是一个经验性的固定值, 而是能够反映当前数据集实际分布与内在统计特性的动态参数. 此设计确保优化策略的激活具备了明确的量化依据.

3.2 双重剪枝的邻居选择策略

针对识别到的致密区域节点, 本文采用一种双重剪枝的邻居选择策略. 该策略的核心在于融合两种规则的剪枝结果, 在提升连接多样性的同时, 保留关键的枢纽向量, 从而实现局部拓扑的修复和选择高质量邻居.

● 双重剪枝的邻居选择策略. 对于新写入节点, 算法协同应用原生与修正的剪枝规则, 保留潜在的有效连接, 以提升稀疏连接情况下邻居的质量:

(1) 原生剪枝规则 (α=1): 即标准的 HNSW 启发式剪枝规则. 此规则所选的候选邻居集合记为 C_{original}.

(2) 修正剪枝规则 (α>1): 如图 8 所示, 该规则通过引入一个大于 1 的剪枝因子 α, 选择性地放宽了剪枝条件. 其具体规则如下: 对于待插入节点 v, 仅当一个候选邻居 v'与结果集 R 中某个已选邻居 v_i 满足 $\alpha \cdot dist(v', v_i) > dist(v', v)$ 时, v'才被视为冗余邻居并予以剪枝. 此规则用于筛选出一组更具方向多样性的候选邻居集合, 记为 C_{alpha}.

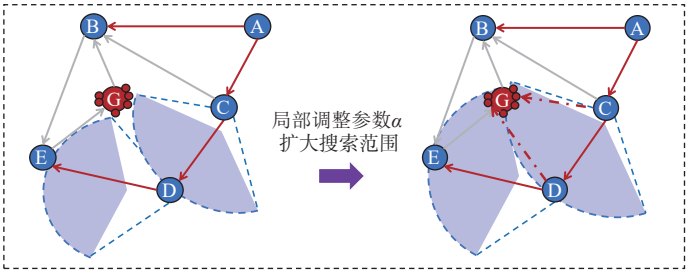


图 8 修正的启发式剪枝规则

● 基于枢纽识别的邻居补充. 应用修正规则后, 选取 C_{original} 中邻居数量不小于 $M/2$ 的节点构成枢纽节点子集 $H=\{v|v\in C_{\text{original}},|v.neighbors|\geqslant M/2\}$, 并将其合并入 C_{alpha}, 形成最终的候选邻居集 $C_{\text{final}}=C_{\text{alpha}}\cup H$.

修正剪枝规则的剪枝因子 α 决定了剪枝的宽松度, 其取值直接影响剪枝所得的候选邻居质量. 为此, 本方法基于识别出的致密区域进行局部节点采样, 通过分析样本节点间距离分布特征, 自适应确定适配当前数据集分布的 α 值.

上述双重剪枝策略, 源于对 HNSW 算法中邻居剪枝策略的深刻理解. 向量搜索过程本质上是一种贪心搜索过程, 其具有明显的路径依赖性, 即搜索着重于当前局部最优选择, 且搜索路线会影响最终结果, 因为每一步的选择依赖于之前的路径. 这意味着, 若仅采用 α>1 的修正剪枝规则, 尽管增加了邻居连接的多样性, 但也可能因在迭代早期引入了某个较远的邻居; 而改变后续的剪枝基准, 忽略了原生规则所能发现的高质量候选邻居. 因此, 本方法提出的邻居选择策略, 可以实现更为精准的权衡. 算法以具备多样邻居的 C_{alpha} 为基础, 通过精准地补入 C_{original} 中的枢纽节点来提升邻居的质量. 这种剪枝修正策略使得算法能够保留那些在原规则下可能因距离相近而被判定为冗余的连接, 但这些连接对图的连通性和导航却至关重要. 通过这样的处理, 算法有效地缓解了过度剪枝效应. 同时, 该策略还能保留原剪枝规则中较优质的邻居, 确保局部检索的精度.

3.3 算法实现与分析

下面介绍基于自适应细粒度剪枝策略的 HNSW 索引构建过程, 完整流程如算法 1 所示. 该算法接收的参数包括: 待插入向量 v、索引参数 (邻居数量上限 M 和候选邻居扩展因子 *efConstruction*)、索引最高层级 *maxLayer*、入口点 *enterPoint*、邻居度量参数 β 以及剪枝因子 α. 在向索引插入向量 v 后, 需同步更新局部和全局的距离指标.

算法 1 包含两个阶段. 第 1 阶段是入口点的定位 (第 5–8 行), 通过自顶向下的贪心搜索过程, 在各层级为待插入节点 v 确定最优的搜索起始点 *ep*. 第 2 阶段是在各个层级 l 上的邻居构建 (第 9–22 行), 该阶段是算法的核心优化部分: 第 1 步, 通过 SearchAtLayer 为 v 获取初始候选邻居集 *tempRes* (第 10 行). 第 2 步, 识别该节点是否处于致密区域 (第 11、12 行): 首先, 通过 getAreaAvgDistance 计算该节点的区域邻居平均距离 *area_mean_dist*, 将其与全局邻居平均距离 *global_mean_dist*[l] 进行比较, 若 *area_mean_dist* 低于 $\beta\cdot$*global_mean_dist*[l], 则判定该节点处于局部致密区域, 并对其实施双重剪枝的邻居选择策略 (第 13–18 行). 该策略先采用 α>1 的修正剪枝规则, 获取更多样连接的候选邻居集 C_{alpha}; 在此基础上, 再采用 α=1 的原生剪枝规则以得到 C_{original}, 并将其中邻居数大于 $M/2$ 的枢纽节点组成枢纽子集 H (第 15–17 行), 最终的候选邻居集 C_{final} 为 C_{alpha} 与 H 的并集 (第 18 行). 对于未处于致密区域的节点, 则仅对其实施 α=1 的原生剪枝规则 (第 19–21 行). 在构建完 v 的邻居关系后, 其为新邻居构建反向连接关系, 以增强图的局部连通性 (第 22–25 行). 最后, 更新索引的全局状态: 判断入口点是否需要更新, 若新节点 v 的层级 *level* 超过了最大层级 *maxLayer*, 则将其设为新的全局入口点 (第 28–31 行).

算法 1. 基于自适应细粒度剪枝优化的 HNSW 索引构建算法 (Insertion).

输入: 待插入向量 v, 邻居数量上限 M, 候选邻居扩展因子 *efConstruction*,
索引最高层级 *maxLayer*, 入口点 *enterPoint*, 邻居度量参数 β, 剪枝系数 α.

1. **begin**
2. Queue<VectorID> *tempRes* // 存储搜索过程中的候选节点
3. *level* = *getRandomLevel*() // 为节点 v 随机分配的目标层级
4. VectorID *ep* = *enterPoint* // 初始化入口点
5. **for** l = *maxLayer* **downto** *level*−1 **do:** // 阶段 1: 自顶向下搜索, 为待插入层定位入口点
6. *tempRes* = *SearchAtLayer*(v, *ep*, M, 1, l)
7. *ep* = *getClosest*(*tempRes*, 1)
8. **end for**
9. **for** l = min(*maxLayer*, *level*) **downto** 0 **do:** // 阶段 2: 逐层构建邻居连接, 并实施自适应的细粒度剪枝优化

```
        tempRes = SearchAtLayer(v, ep, M, efConstruction, l)
        area_mean_dist = getAreaAvgDistance(v, l) // 计算区域邻居平均距离, 以感知局部区域密度
        if (area_mean_dist < β·global_mean_dist[l]) then: // 对致密区域的节点, 实施双重剪枝的邻居选择策略
            C_original = getNeighborsByHeuristic(efConstruction, 1)
            C_alpha = getNeighborsByHeuristic(efConstruction, α)
            for v_i in C_original do: // 将原生规则所选的枢纽节点合并入最终候选邻居集
                if |v_i.neighbors| ⩾ M/2 then:
                    H = H ∪ {v_i}
            C_final = C_alpha ∪ H
    else: // 对未处于致密区域的节点, 仅实施原生剪枝规则
            C_final = getNeighborsByHeuristic(efConstruction, 1)
    end if
    for v_i in v.neighbors do: // 为新建立的邻居构建反向连接
        reConnection(v_i, v)
        getNeighborsByHeuristic(efConstruction, α)
    end for
    ep = getClosest(tempRes, 1) // 更新下一层的入口点
end for
if (level > maxLayer) then: // 设置新的全局入口点
    maxLayer = level
    enterPoint = v
end if
end
```

● 开销分析. 本算法在设计上严格控制了开销. 在存储开销方面, 额外存储空间仅包含各层级图结构的全局平均距离 *global_mean_dist*[*level*], 其空间复杂度为 O(1), 对系统整体存储需求的影响可忽略不计. 在计算复杂度方面, 主要开销来自 getAreaAvgDistance 函数, 而通过合理地设置 *efConstruction* 和 *M* 参数, 可以有效控制图结构的规模, 从而将该计算的时间复杂度控制在可接受范围内. 根据多个数据集上的实验观察, 被识别为致密区域并经过剪枝优化的节点占比较低 (仅为数据集的 0.5%–2%). 因此, 该优化方法引入的额外计算开销非常小, 得以维持 HNSW 索引对数级别的插入复杂度.

本节详细阐述了一种基于图结构局部调整的自适应细粒度剪枝策略, 通过识别与修复的闭环机制, 有效缓解了因过度剪枝导致的局部拓扑结构稀疏和召回率退化问题. 该策略能够动态地适应局部数据分布的变化: 在数据致密区域通过放宽剪枝条件主动保留了关键的导航连接; 而在其他区域则仍实施原生 HNSW 剪枝规则.

4 实验分析

4.1 实验设置

● 实验数据集. 为全面评估本文方法的效果, 我们在多个公开数据集 (详见表 2) 上开展实验. GIST1M 包含 100 万个 960 维 GIST 描述符[41], 这些描述符是基于一组感知维度从原始图像中提取的低维向量数据. Msong 涵盖 100 万首西方流行音乐的音频特征 (节奏、响度、淡入淡出时间等), 适用于音乐推荐系统研究. Enron 包含安然公司内部 150 名用户数据, 其中大多数是高层管理人员, 涵盖约 50 万封真实电子邮件及其元数据, 适用于文本分析与社交网络研究, 数据维度较高 (1 369). 此外, 局部本征维度 (local intrinsic dimensionality, LID) 是衡量数据局部

结构复杂性的重要指标, 能够揭示高维向量数据在局部区域的实际维度特征. LID 值较大通常表示更难区分这些向量数据, 较高的 LID 给模拟相似向量写入场景带来了挑战.

表 2 实验数据集

数据集	样本数量	向量维数	内容	LID
GIST1M[41,42]	1 000 000	960	Image	18.9
Msong[43]	1 000 000	420	Audio	9.5
Enron[44]	94 987	1 369	Text	11.7

● 实验环境. 本实验的硬件平台为: Intel(R) Xeon(R) Gold 6240M CPU @ 2.60 GHz, 内存 192 GB, 操作系统为 CentOS Linux 7.

● 评价指标. 实验中主要关注索引的检索精度、微观拓扑结构、索引构建开销及检索效率, 采用以下 4 个关键指标.

(1) 召回率 (*recall@K*). 用于衡量检索结果的准确性. 其形式化定义为算法返回的 *top-K* 结果集与真实 *top-K* 最近邻集合之间交集的基数, 与 *K* 的比值. 实验中 *top-K* 的默认值为 10.

(2) 平均邻居连接数. 其定义为计算相似更新集中所有节点建立的出边数量的算术平均值, 用于验证本文所提优化方法在修复相似向量连接稀疏的问题上的有效性.

(3) 索引构建时间. 记录了构建完整索引所需的时间, 直接反映了不同算法在索引构建阶段的计算开销.

(4) 查询执行时长/查询吞吐量 (QPS). 用于衡量检索的效率, 反映了索引处理查询请求的能力.

● 实验方法. 为全面评估本文提出的基于图结构局部调整的自适应细粒度剪枝策略的效果, 本节的实验遵循第 2.1 节的实验设置, 模拟批量插入相似数据的场景. 图 9 定义了实验场景和负载写入流程. 构建基础索引 (图 9 中的原始向量): 该过程使用原始向量构建 HNSW 索引, 未引入任何相似数据. 更新相似向量 (图 9 中的更新向量): 在基础索引的基础上, 批量插入相似数据负载. 基于以上实验设置, 分别构建两组索引: Batch 表示基于原 HNSW 算法构建的索引, Opt 表示应用本文优化方法改进的索引. 实验定义了多级相似数据负载, 各级的相似数据占比从 1% 递增至 5%. 为模拟批量更新相似向量场景, 我们分别设置数据集的基础向量如下: GIST1M 为 10 万, Msong 为 10 万, Enron 为 9 万.

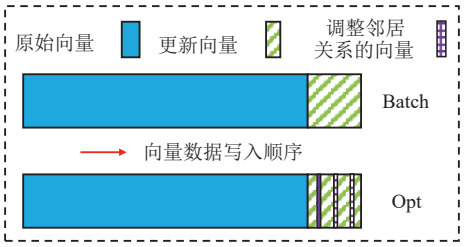


图 9 数据写入场景的实验负载

优化方法的实验流程通过对这两个索引施加相同的、批量相似数据负载操作展开. 具体而言, 实验将依次批量地向基础索引写入新数据, 来构建相似数据占比为 1%, 2%, 3%, 4% 以及 5% 的 5 个增量更新的批量负载. 在写入每批数据后, 对两个索引的当前状态进行召回率、邻居关系分析等测试. 该测试将通过相应的相似查询集, 对比分析两种算法在关键指标上的性能表现. 实验过程按如下方式进行: 首先, 实验测量并比较它们的召回率 (*recall@K*) 与查询执行时长, 定量评估本文方法在提升检索精度与速度方面的具体优势. 其次, 实验将测量并对比相似数据节点的平均邻居连接数, 从微观层面验证本文方法在修复连接稀疏化问题上的优化效果.

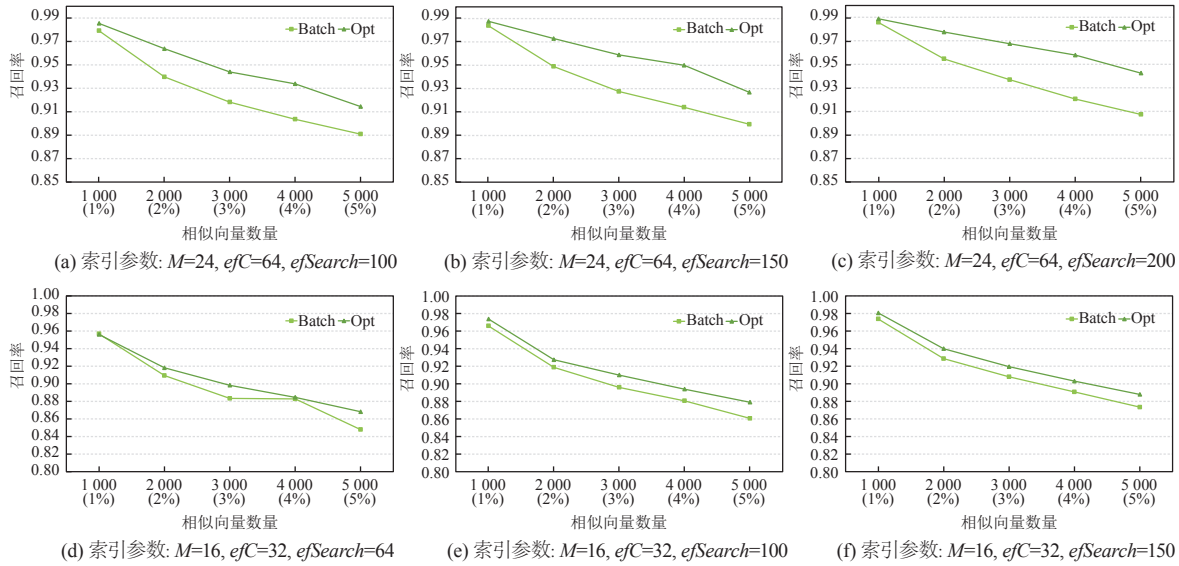
4.2 算法性能实验结果与分析

4.2.1 平均召回率的结果与分析

在批量插入相似数据场景下, 我们在多个数据集上开展横向对比实验, 以评估本文方法的性能表现. 实验比较

原生 HNSW 与本文优化方法的召回率 (*recall@K*), 实验负载为第 4.1 节中定义的 5 类负载.

图 10 展示了 GIST1M 数据集的实验结果. 在各组索引参数 (*M* 和 *efC*) 与搜索深度 (*efSearch*) 下, 经本文方法优化的索引召回率在所有负载阶段 (相似数据占比 1%–5%) 均优于原 HNSW 索引, 并且性能衰减速率也有所降低. 例如, 初始索引参数为 *M*=24, *efC*=64 时, 当 *efSearch* 为 100 时, 随着相似数据占比的增加 (2%–5%), 召回率提升分别为 2.41%, 2.58%, 302% 和 2.35%; 在 *efSearch* 为 150 时, 随着相似数据占比的增加 (2%–5%), 召回率提升分别为 2.28%, 3.06%, 3.73% 和 3.51%. 综上所述, 在不同相似数据占比下, 索引的检索召回率都得到不同程度的提升. 在写入相似数据占比较小时 (1%), 召回率的变化幅度不大. 这表明该优化方法不仅提升了基础检索性能, 还减缓了持续插入相似数据对索引性能的损害.



(a) 索引参数: *M*=24, *efC*=64, *efSearch*=100　(b) 索引参数: *M*=24, *efC*=64, *efSearch*=150　(c) 索引参数: *M*=24, *efC*=64, *efSearch*=200

(d) 索引参数: *M*=16, *efC*=32, *efSearch*=64　(e) 索引参数: *M*=16, *efC*=32, *efSearch*=100　(f) 索引参数: *M*=16, *efC*=32, *efSearch*=150

图 10　GIST1M 数据集的平均召回率

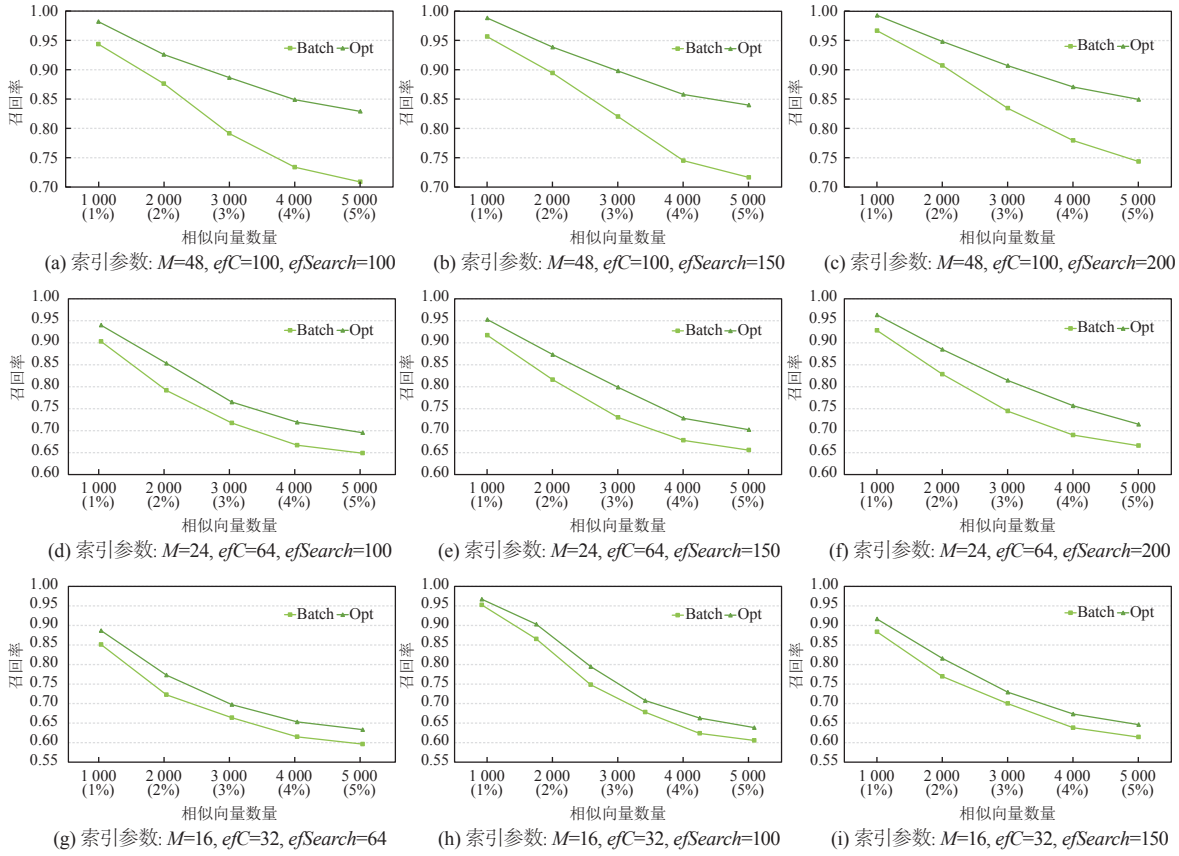
在索引配置为 *M*=24, *efC*=64 时, 增大搜索参数 *efSearch* (100–200), 召回率的提升幅度呈现出增加的趋势 (>3%). 因为增大 *efSearch*, 会增加搜索的深度, 这样向量选择邻居时候选集的质量得到了改善, 本文方法保留了原规则的中的优质邻居, 即枢纽节点, 因此, 有助于进一步提升改善向量的邻居关系质量. 在索引配置为 *M*=16, *efC*=32 时, 索引的提升幅度相对减小 (约为 1%–2%). 索引配置参数较小时, 召回率水平和邻居质量都受到了影响.

下面我们分析索引参数配置对索引的检索召回率水平的影响. 在 *M*=16, *efC*=32, *efSearch*=100, 相似数据占比为 3% 时, 索引的平均召回率水平约为 88%; 在 *M*=24, *efC*=64, *efSearch*=150, 相似数据占比为 3% 时, 索引的平均召回率水平约为 91.4%. 两种条件下索引的平均召回率相差 3.4%, 而应用本文的优化方法后, 召回率的差距为 5.5%, 在较小的索引参数时, 图的质量受到了一定的影响, 这与 HNSW 索引的原理特性相符合. 同时, 这也进一步表明本文方法能有效缓解批量插入场景下的召回率下降问题.

我们在 Msong 和 Enron 数据集上进行了相同的性能比较, 实验结果如图 11 和图 12 所示. 在 Msong 数据集上, 随着相似数据占比的增加 (2%–5%), 在 *M*=16, *efC*=32, *efSearch*=64 时, 召回率平均提升水平为 3.9%; 在 *M*=24, *efC*=64, *efSearch*=100 时, 召回率平均提升水平为 4.9%; 在 *M*=48, *efC*=100, *efSearch*=100 时, 召回率平均提升水平为 8%. 在 Msong 数据集上, 随着相似数据占比的增加, 本文方法整体上表现出较好的召回率提升的效果. 比较在不同索引参数对本文方法的影响, 在较小索引参数下 (*M*=16, *efC*=32), 索引的召回率水平偏低, 图质量不高, 其召回率提升不高 (3.9%); 当索引参数较大时 (*M*=48, *efC*=100), 召回率平均提升水平可达 8%. 这与 GIST1M 数据集上表现相同, 索引配置参数较小时, 召回率水平和邻居质量都受到了影响.

当 *efSearch* 分别为 100、150 和 200 时, 由 *M*=48, *efC*=100 构建索引的平均召回率的上升幅度分别为 8.38%、

7.81% 和 6.75%. 在其他各组索引参数下, 查询召回率也均有不同程度的提升, 最大的上升幅度为 12% (*M*=48, *efC*=100, *efSearch*=100, 相似数据占比为 5%).

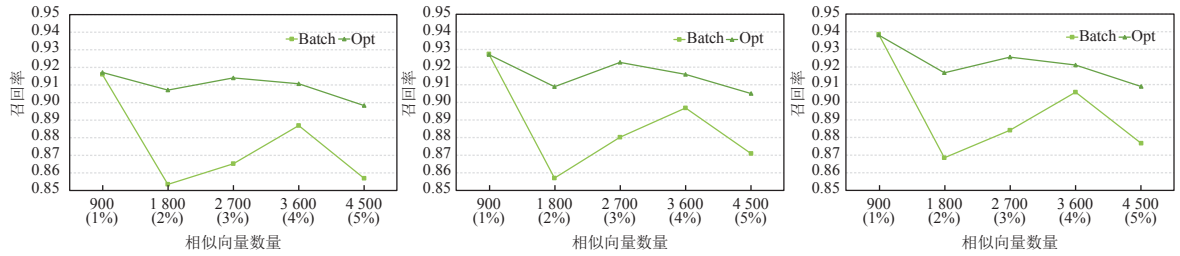


(a) 索引参数: *M*=48, *efC*=100, *efSearch*=100 (b) 索引参数: *M*=48, *efC*=100, *efSearch*=150 (c) 索引参数: *M*=48, *efC*=100, *efSearch*=200

(d) 索引参数: *M*=24, *efC*=64, *efSearch*=100 (e) 索引参数: *M*=24, *efC*=64, *efSearch*=150 (f) 索引参数: *M*=24, *efC*=64, *efSearch*=200

(g) 索引参数: *M*=16, *efC*=32, *efSearch*=64 (h) 索引参数: *M*=16, *efC*=32, *efSearch*=100 (i) 索引参数: *M*=16, *efC*=32, *efSearch*=150

图 11 Msong 数据集的平均召回率



(a) 索引参数: *M*=24, *efC*=64, *efSearch*=100 (b) 索引参数: *M*=24, *efC*=64, *efSearch*=150 (c) 索引参数: *M*=24, *efC*=64, *efSearch*=200

图 12 Enron 数据集的平均召回率

考虑到 Enron 数据的维度较高 (1 396 维), 其对索引参数较为敏感, 即在不同索引参数下的召回率水平差异较大, 实验中选择固定索引参数 (*M*=24, *efC*=64), 召回率水平约为 90%. 在 Enron 数据集上, 本文方法同样表现出了召回率提升的效果. 当 *efSearch* 分别为 100、150 和 200 时, 由 *M*=24, *efC*=64 构建索引的平均召回率的上升幅度分别为 3.37%、2.95% 和 2.74%, 其中最大的上升幅度为 5.36% (*M*=48, *efC*=100, *efSearch*=100, 相似数据占比为 2%). 在 Enron 数据集上, 随着相似数据占比的增加, 相较于 Batch 场景, 本文优化方法减小了召回率的波动, 使得召回率的变化更加平稳. 综上所述, 本文方法在多个数据集上都展现出了较好的效果, 体现了在批量更新相似向量场景下, 本文方法优化索引召回率的能力.

4.2.2 相似数据邻居数量的结果与分析

本节针对批量插入相似向量引发的连接稀疏化问题, 通过对比优化前后平均邻居连接数指标的变化, 验证了本文方法对向量邻居关系具有较好的修复能力. 实验基于 GIST1M、Enron 和 Msong 数据集, 表 3 为索引邻居关系的分析结果 (相似数据占比为 1%–5%), 实验结果表明, 所有数据集均显著改善, 相似节点的邻居的连接密度和邻居质量均有所提升, Msong 数据集提升最显著.

表 3 相似向量邻居数量的变化

数据集	索引参数	Batch平均邻居数量	Opt平均邻居数量	Batch邻居数量较小向量的占比 (%)	Opt邻居数量较小向量的占比 (%)
GIST1M	*M*=24, *efC*=64	6.4	7.7	60	59
Enron	*M*=24, *efC*=64	9.0	9.7	53	48
Msong	*M*=48, *efC*=100	4.6	12.1	95.7	78.5

在 GIST1M 和 Enron 数据集上, 相似向量的平均邻居数量分别为 6.4 和 9.0, 优化方法将其提升至 7.7 和 9.7. 在 Msong 数据集上, Batch 场景下 HNSW 的平均连接数仅为 4.6, 而本文方法将其提升至 12.1, 同时, Batch 场景下邻居数量较小向量的占比高达 95.7%, 本文方法将其降低至 78.5%, 低连接的向量数量大大降低. 因此, 本文提出的双重剪枝策略, 通过剪枝优化方法和保留枢纽向量, 不仅可以提高邻居的基础数量, 而且有助于保留关键连接边 (枢纽向量) 以提高邻居质量, 即使在大量相似数据积累的负载 (相似数据占比为 5%) 下, 仍能维持较高的图连通性. 这将有效地缓解图结构的拓扑稀疏化问题, 进而提升检索的召回率.

4.3 开销与效率评估

本节进一步评估了索引的构建开销和计算效率. 实验对比了 Batch 和 Opt 两个场景下 HNSW 索引的构建时间和查询时间/查询吞吐量 (QPS) 两个关键指标.

● 索引构建时间. 表 4 展示了原方法与本文优化方法在索引构建时的平均耗时情况, 为精确度量构建的开销, 索引使用单线程方式构建. 本文方法会引入的额外计算成本, 在构建阶段会增加少量的计算时间. 实验结果表明, 在所有数据集中, 本文方法与原 HNSW 索引的构建时间均保持在同一水平. 例如, 在 GIST1M 数据集上, 平均构建时间均为 98 s; 在高维向量 Enron (1 369 维) 上, 平均构建时间仅增加 4.2 s.

表 4 索引构建开销

数据集	索引参数	Batch构建时间 (s)	Opt构建时间 (s)
GIST1M	*M*=24, *efC*=64	98.4	98.5
Enron	*M*=24, *efC*=64	97.8	102
Msong	*M*=48, *efC*=100	78.6	80.4

参数 *M* 与 *efConstruction* 会显著影响索引的构建时间. 当采用更大的参数 (如 *M*=48, *efC*=200) 时, 其邻居数量和搜索范围均会扩大, 其构建时间也相应地增加. 但本文方法的目标是优化邻居关系, 在高参数配置下索引的邻居关系已经得到了改善, 因此额外计算成本也会相应地减少, 所以, 本文方法的计算开销不会因较大索引参数带来的基础构建开销的增加而产生明显的增长.

上述观测结果符合理论预期: 本文方法的额外开销主要是对少量枢纽节点执行邻居选择策略, 在全流程中均摊的计算开销较低, 最终实现以计算时间换取查询精度的显著提升.

● 查询执行时长/查询吞吐量 (QPS). 在 Batch 和 Opt 场景下, 表 5 和表 6 分别展示了在不同查询参数 (*efSearch* 和 *top-K*) 时索引的检索效率. 其中, *efSearch* 为查询时动态候选集的大小, *top-K* 为结果集的数量. 表 5 和表 6 中的检索时间为: 在对应场景下, 执行查询集的总检索时长, 查询集的大小为 1k.

实验数据显示, 在检索 GIST1M 和 Enron 向量索引时, 本文方法的检索时长会降低, 索引的 QPS 得到了提升. 本文方法中的双重剪枝策略和保留枢纽向量, 可以改善邻居关系, 从而减少搜索步长, 可以达到提升召回率和搜索最优邻居的目的. 本文方法在应用到 Msong 数据集时, 增加了较多的邻居数量 (如表 3 所示), 这使得查询搜索的候选向量增多, 因此增加了搜索的开销.

表 5 索引检索效率 (*top-K*=10)

数据集	索引参数	*efSearch*	Batch检索时间 (ms)	Opt检索时间 (ms)
GIST1M	*M*=24, *efC*=64	64	678	586
		100	1 672	1 415
		150	2 878	2 483
		200	4 252	3 723
Enron	*M*=24, *efC*=64	64	344	303
		100	800	696
		150	1 412	1 199
		200	2 151	1 798
Msong	*M*=48, *efC*=100	100	966	1 024
		150	1 460	1 532
		200	2 066	2 143
		300	2 886	3 011

表 6 索引检索效率 (*top-K*=20)

数据集	索引参数	*efSearch*	Batch检索时间 (ms)	Opt检索时间 (ms)
GIST1M	*M*=24, *efC*=64	64	694	571
		100	1 594	1 393
		150	2 695	2 426
		200	3 981	3 629
Enron	*M*=24, *efC*=64	64	329	303
		100	767	696
		150	1 374	1 205
		200	2 028	1 802
Msong	*M*=48, *efC*=100	100	926	966
		150	1 384	1 441
		200	1 928	1 991
		300	2 665	2 704

结合第 4.2 节中索引召回率结果, 召回率得到了提升, 查询时长减少, 因此, 相似向量的邻居关系的质量得到了改善. 本文方法会引入的额外计算成本, 为相似向量选择枢纽向量和减少空间覆盖引起的过度剪枝, 双重剪枝策略为向量保留了较优的邻居关系.

5 结 论

本文聚焦于批量插入相似数据场景下 HNSW 索引的召回率退化问题, 提出了一种基于图结构局部调整的自适应细粒度剪枝策略, 以提升其在批量更新场景下的鲁棒性与检索精度. 首先, 本文通过深入的实验与理论分析发现, 过度剪枝效应以及其引发的相似向量连接稀疏化问题是导致查询召回率下降的主要原因. 基于此发现, 本文所提出的优化策略构建了一个多阶段的识别与修复框架: 在识别阶段, 以数据驱动的方法确定诊断阈值, 并通过区域邻居平均距离实现了对致密区域的精准定位; 在修复阶段, 针对处于致密区域的节点, 采用双重剪枝的邻居选择机制. 该机制通过并行应用原生与修正的剪枝规则, 并对两者的结果进行合并, 在保留高精度近邻的同时, 融入了具有方向多样性的邻居, 从而协同地缓解了过度剪枝与连接稀疏化这两大缺陷. 最后, 在多个公开数据集上的实验评估表明, 本文提出的优化方法在不引入显著查询开销的前提下, 能够有效提升索引的召回率, 展现了其在面对批量数据更新场景的有效性与实用性.

References

[1] Tian Y, Yue ZY, Zhang RY, Zhao X, Zheng BL, Zhou XF. Approximate nearest neighbor search in high dimensional vector databases: Current research and future directions. IEEE Data Engineering Bulletin, 2023, 47(3): 39–54.

[2] Qin JB, Wang W, Xiao C, Zhang Y. Similarity query processing for high-dimensional data. Proc. of the VLDB Endowment, 2020, 13(12): 3437–3440. [doi: 10.14778/3415478.3415564]

[3] Qin JB, Wang W, Xiao C, Zhang Y, Wang YS. High-dimensional similarity query processing for data science. In: Proc. of the 27th ACM SIGKDD Conf. on Knowledge Discovery & Data Mining. Singapore: ACM, 2021. 4062–4063. [doi: 10.1145/3447548.3470811]

[4] Li W, Zhang Y, Sun YF, Wang W, Li MJ, Zhang WJ, Lin XM. Approximate nearest neighbor search on high dimensional data—Experiments, analyses, and improvement. IEEE Trans. on Knowledge and Data Engineering, 2020, 32(8): 1475–1488. [doi: 10.1109/TKDE.2019.2909204]

[5] Yasser M, Hussain KF, Ali SA. Comparative analysis of similarity methods in high-dimensional vectors: A review. In: Proc. of the 2023 Int'l Conf. on Artificial Intelligence Science and Applications in Industry and Society (CAISAIS). Galala: IEEE, 2023. 1–6. [doi: 10.1109/CAISAIS59399.2023.10270776]

[6] Zhao X, Tian Y, Huang K, Zheng BL, Zhou XF. Towards efficient index construction and approximate nearest neighbor search in high-dimensional spaces. Proc. of the VLDB Endowment, 2023, 16(8): 1979–1991. [doi: 10.14778/3594512.3594527]

[7] Wang YS, Li PF, Wang ZQ, Zhu QM. Survey on multimodal information extraction research. Ruan Jian Xue Bao/Journal of Software,

2025, 36(4): 1665–1691 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7245.htm [doi: 10.13328/j.cnki.jos.007245]

[8] Almeida F, Xexéo G. Word embeddings: A survey. arXiv:1901.09069, 2019.

[9] Grohe M. word2vec, node2vec, graph2vec, X2vec: Towards a theory of vector embeddings of structured data. In: Proc. of the 39th ACM SIGMOD-SIGACT-SIGAI Symp. on Principles of Database Systems. Portland: ACM, 2020. 1–16. [doi: 10.1145/3375395.3387641]

[10] Pan JJ, Wang JG, Li GL. Survey of vector database management systems. The VLDB Journal, 2024, 33(5): 1591–1615. [doi: 10.1007/s00778-024-00864-x]

[11] Pan JJ, Wang JG, Li GL. Vector database management techniques and systems. In: Proc. of the Companion of the 2024 Int'l Conf. on Management of Data. Santiago: ACM, 2024. 597–604. [doi: 10.1145/3626246.3654691]

[12] Zhao WX, Zhou K, Li JY, Tang TY, Wang XL, Hou YP, Min YQ, Zhang BC, Zhang JJ, Dong ZC, Du YF, Yang C, Chen YS, Chen ZP, Jiang JH, Ren RY, Li YF, Tang XY, Liu ZK, Liu PY, Nie JY, Wen JR. A survey of large language models. arXiv:2303.18223, 2023.

[13] Zhu XP, Yao HD, Liu J, Xiong XK. Review of evolution of large language model algorithms. ZTE Technology Journal, 2024, 30(2): 9–20 (in Chinese with English abstract). [doi: 10.12142/ZTETJ.202402003]

[14] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih WT, Rocktäschel T, Riedel S, Kiela D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 793.

[15] Gao YF, Xiong Y, Gao XY, Jia KX, Pan JL, Bi YX, Dai Y, Sun JW, Wang M, Wang HF. Retrieval-augmented generation for large language models: A survey. arXiv:2312.10997, 2023.

[16] Fan WQ, Ding YJ, Ning LB, Wang SJ, Li HY, Yin DW, Chua TS, Li Q. A survey on RAG meeting LLMS: Towards retrieval-augmented large language models. In: Proc. of the 30th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Barcelona: ACM, 2024. 6491–6501. [doi: 10.1145/3637528.3671470]

[17] Ren RY, Wang YH, Qu YQ, Zhao WX, Liu J, Wu H, Wen JR, Wang HF. Investigating the factual knowledge boundary of large language models with retrieval augmentation. In: Proc. of the 31st Int'l Conf. on Computational Linguistics. Abu Dhabi: ACL, 2025. 3697–3715.

[18] Liu ZY, Wang PJ, Song XB, Zhang X, Jiang BB. Survey on hallucinations in large language models. Ruan Jian Xue Bao/Journal of Software, 2025, 36(3): 1152–1185 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7242.htm [doi: 10.13328/j.cnki.jos.007242]

[19] Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975, 18(9): 509–517. [doi: 10.1145/361002.361007]

[20] Dolatshah M, Hadian A, Minaei-Bidgoli B. Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces. arXiv:1511.00628, 2015.

[21] Malkov Y, Ponomarenko A, Logvinov A, Krylov V. Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems, 2014, 45: 61–68. [doi: 10.1016/j.is.2013.10.006]

[22] Wang MZ, Xu XL, Yue Q, Wang YX. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. Proc. of the VLDB Endowment, 2021, 14(11): 1964–1978. [doi: 10.14778/3476249.3476255]

[23] Wang JD, Li SP. Query-driven iterated neighborhood graph search for large scale indexing. In: Proc. of the 20th ACM Int'l Conf. on Multimedia. Nara: ACM, 2012. 179–188. [doi: 10.1145/2393347.2393378]

[24] Gionis A, Indyk P, Motwani R. Similarity search in high dimensions via hashing. In: Proc. of the 25th Int'l Conf. on Very Large Data Bases. Edinburgh: Morgan Kaufmann Publishers Inc., 1999. 518–529.

[25] Datar M, Immorlica N, Indyk P, Mirrokni VS. Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. of the 12th Annual Symp. on Computational Geometry. New York: ACM, 2004. 253–262. [doi: 10.1145/997817.997857]

[26] Blumer A, Blumer J, Haussler D, McConnell R, Ehrenfeucht A. Complete inverted files for efficient text retrieval and analysis. Journal of the ACM (JACM), 1987, 34(3): 578–595. [doi: 10.1145/28869.28873]

[27] Gray R. Vector quantization. IEEE ASSP Magazine, 1984, 1(2): 4–29. [doi: 10.1109/MASSP.1984.1162229]

[28] Pan ZB, Wang LZ, Wang Y, Liu YC. Product quantization with dual codebooks for approximate nearest neighbor search. Neurocomputing, 2020, 401: 59–68. [doi: 10.1016/j.neucom.2020.03.016]

[29] Malkov YA, Yashunin DA. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2020, 42(4): 824–836. [doi: 10.1109/TPAMI.2018.2889473]

[30] Chen Q, Zhao B, Wang HD, Li MQ, Liu CJ, Li ZZ, Yang M, Wang JD. SPANN: Highly-efficient billion-scale approximate nearest neighbor search. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 398.

[31] Echihabi K, Zoumpatianos K, Palpanas T. New trends in high-D vector similarity search: Al-driven, progressive, and distributed. Proc. of the VLDB Endowment, 2021, 14(12): 3198–3201. [doi: 10.14778/3476311.3476407]

[32] Xu YM, Liang HY, Li J, Xu ST, Chen Q, Zhang QX, Li C, Yang ZY, Yang F, Yang YQ, Cheng P, Yang M. SPFresh: Incremental in-place update for billion-scale vector search. In: Proc. of the 29th Symp. on Operating Systems Principles. Koblenz: ACM, 2023. 545–561. [doi: 10.1145/3600006.3613166]

[33] Durante Z, Huang QY, Wake N, Gong R, Park JS, Sarkar B, Taori R, Noda Y, Terzopoulos D, Choi Y, Ikeuchi K, Vo H, Fei-Fei L, Gao JF. Agent AI: Surveying the horizons of multimodal interaction. arXiv:2401.03568, 2024.

[34] Subramanya SJ, Devvrit, Kadekodi R, Krishaswamy R, Simhadri HV. DiskANN: Fast accurate billion-point nearest neighbor search on a single node. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 1233.

[35] Singh A, Subramanya SJ, Krishnaswamy R, Simhadri HV. FreshDiskANN: A fast and accurate graph-based ANN index for streaming similarity search. arXiv:2105.09613, 2021.

[36] Gollapudi S, Karia N, Sivashankar V, Krishnaswamy R, Begwani N, Raz S, Lin YY, Zhang Y, Mahapatro N, Srinivasan P, Singh A, Simhadri HV. Filtered-DiskANN: Graph algorithms for approximate nearest neighbor search with filters. In: Proc. of the 2023 ACM Web Conf. Austin: ACM, 2023. 3406–3416. [doi: 10.1145/3543507.3583552]

[37] GitHub, Inc. SPTAG. 2025. https://github.com/microsoft/SPTAG

[38] Iwasaki M, Miyazaki D. Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. arXiv:1810.07355, 2018.

[39] Elliott OP, Clark J. The impacts of data, ordering, and intrinsic dimensionality on recall in hierarchical navigable small worlds. In: Proc. of the 2024 ACM SIGIR Int'l Conf. on Theory of Information Retrieval. Washington: ACM, 2024. 25–33. [doi: 10.1145/3664190.3672512]

[40] Li CL, Zhang MJ, Andersen DG, He YX. Improving approximate nearest neighbor search through learned adaptive early termination. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 2539–2554. [doi: 10.1145/3318464.3380600]

[41] Jégou H, Douze M, Schmid C. Searching with quantization: Approximate nearest neighbor search using short codes and distance estimators. Technical Report, RR-7020, INRIA, 2009. [doi: 10.1023/A:1011139631724]

[42] INRIA. Datasets for approximate nearest neighbor search. 2011. http://corpus-texmex.irisa.fr/

[43] MSong. Million song dataset benchmarks. 2011. http://millionsongdataset.com/

[44] Russell S, Christopher M, Jeffrey P. Enron email dataset. 2015. https://www.cs.cmu.edu/~./enron/

附中文参考文献

[7] 王永胜, 李培峰, 王中卿, 朱巧明. 多模态信息抽取研究综述. 软件学报, 2025, 36(4): 1665–1691. http://www.jos.org.cn/1000-9825/7245.htm [doi: 10.13328/j.cnki.jos.007245]

[13] 朱炫鹏, 姚海东, 刘隽, 熊先奎. 大语言模型算法演进综述. 中兴通讯技术, 2024, 30(2): 9–20. [doi: 10.12142/ZTETJ.202402003]

[18] 刘泽垣, 王鹏江, 宋晓斌, 张欣, 江奔奔. 大语言模型的幻觉问题研究综述. 软件学报, 2025, 36(3): 1152–1185. http://www.jos.org.cn/1000-9825/7242.htm [doi: 10.13328/j.cnki.jos.007242]

作者简介

王可, 博士生, CCF 学生会员, 主要研究领域为数据库系统, 向量数据库.

胡思劼, 硕士生, 主要研究领域为数据库系统, 向量索引.

胡卉芪, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为数据库, 分布式系统.

赵明昊, 博士, 助理教授, 博士生导师, CCF 专业会员, 主要研究领域为云计算/存储, 操作系统, 软件工程.

魏星, 博士, CCF 专业会员, 主要研究领域为向量数据库, 异构数据库, 机器学习.

屠要峰, 博士, 研究员, CCF 杰出会员, 主要研究领域为数据库, 大数据, 机器学习.

周烜, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为数据管理系统.