HyperJoin：大语言模型增强的超图链接预测，用于可连接表发现

Shiyuan Liu 悉尼科技大学
shiyuan.liu-1@student.uts.edu.au

Jianwei Wang 新南威尔士大学
jianwei.wang1@unsw.edu.au

Xuemin Lin ACEM，上海交通大学 xuemin.lin@sjtu.edu.cn

Lu Qin 悉尼科技大学
lu.qin@uts.edu.au

Wenjie Zhang 新南威尔士大学
wenjie.zhang@unsw.edu.au

Ying Zhang 悉尼科技大学 ying.zhang@uts.edu.au

arXiv:2601.01015v1 [cs.CL] 3 Jan 2026

摘要

作为数据湖管理中的一项关键任务，可连接表发现已受到广泛关注。现有的基于语言模型的方法通过结合离线列表示学习与在线排序，取得了显著成效，但其设计未能充分考虑底层的结构交互：（1）在离线阶段，它们直接将表建模为彼此孤立的列或成对列，因此难以捕捉丰富的表间和表内结构信息；（2）在在线阶段，它们仅依据查询与候选列之间的相似度对候选列排序，忽视了候选列之间的相互作用，导致结果集缺乏一致性。为解决这些局限，我们提出 HyperJoin，这是一种由大语言模型（LLM）增强的**超**图框架，用于**可连**接表发现。具体而言，我们首先构建一个超图，利用表内超边和大语言模型增强的表间超边对表进行建模。进而，我们将可连接表发现任务表述为在所构建超图上的链接预测。随后，我们设计了 HIN（分层交互网络），通过结合跨超边的全局消息传递，以及列与超边之间的局部消息传递，学习具有表达力的列表示。为增强结果列的一致性和内部一致性，我们将在线排序转化为一个具备一致性感知能力的 top-K 列选择问题。接着，我们引入一个重排序模块，利用最大生成树算法剪除噪声连接并最大限度地提升一致性。实验结果表明，HyperJoin 表现优异，相较于最佳基线，平均提升了 21.4%（Precision@15）和 17.2%（Recall @15）。

PVLDB 参考格式：
Shiyuan Liu、Jianwei Wang、Xuemin Lin、Lu Qin、Wenjie Zhang 和 Ying Zhang。HyperJoin：用于可连接表发现。PVLDB，17(9)：2227 - 2240，2024。
doi:10.14778/3665844.3665853

PVLDB 工件可用性：
源代码、数据和/或其他工件已发布于 https://github.com/T-Lab/HyperJoin。

本作品依据知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议授权。访问 https://creativecommons.org/licenses/by-nc-nd/4.0/ 查看该许可协议副本。如需进行该许可协议未涵盖的任何使用，请发送电子邮件至 info@vldb.org 获取许可。版权归所有者/作者所有。出版权授权给 VLDB 基金会。VLDB 基金会会议录，第 17 卷，第 9 期，ISSN 2150-8097。doi:10.14778/3665844.3665853

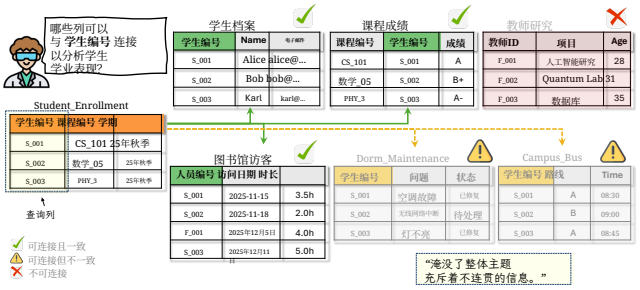


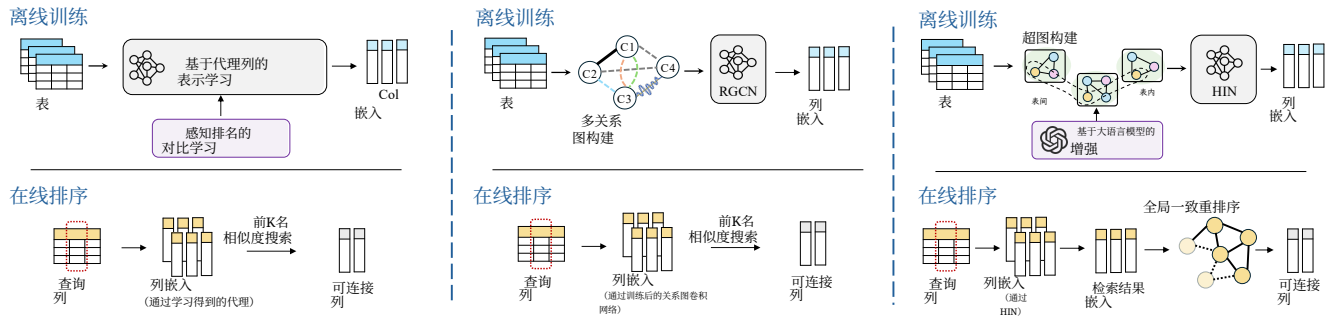
图 1：数据湖中的可连接表发现示例。

1 引言

开放数据门户和企业数据湖的蓬勃发展，带来了海量表格数据，为数据分析和机器学习创造了巨大机遇。然而，这些表往往彼此割裂，难以利用其他表中的互补信息来增强分析。先前的研究 [7, 10, 12, 26, 31] 指出了这一问题的普遍性，并报告称，约 89% 的现实世界商业智能项目涉及多个需要进行连接的表。在这项工作中，我们研究可连接表发现。给定一个表仓库 T 和一个查询列 C_q，可连接表发现旨在从 T 中找到列 C，这些列与 C_q具有大量语义匹配的单元格。因此，包含列 C的表可以与查询表连接，以丰富特征并增强下游数据分析或机器学习任务。

示例 1. 如图 *1* 所示，一位数据科学家使用学生_选课 表分析学生的学业表现，并搜索与学生_编号 （C_q）可连接的列。数据湖中包含若干语义上可连接的列（绿色勾号）。然而，尽管来自宿舍_维护 和校园_巴士等表的列可连接，但它们源自不相关的服务领域，可能会因信息不连贯而干扰分析（黄色警告）。因此，有效的连接发现方法应在识别可连接列的同时，确保返回结果保持连贯，从而更好地支持分析。

认识到可连接表搜索的重要性和实际价值，研究人员已提出一系列方法（深入讨论请参阅综述与基准论文 [13]）。早期的非学习方法通常将可连接表发现视为集合相似度搜索问题 [23]。其中具有代表性的方法包括，



(a) Snoopy (b) Omnimatch (c) HyperJoin（本文方法）

图 2：可连接表发现方法的框架比较。

JOSIE [49] 和 LSH Ensemble [50], , 它们都将每一列建模为不同值的集合，并根据值的重叠情况识别可连接列，同时借助倒排索引或基于 MinHash 的局部敏感哈希提高效率。然而，这些方法将连接类型限制为基于精确字符串匹配的等值连接，忽视了许多语义匹配单元格（例如，“NY”在语义上与“New York”匹配），因此低估了列的可连接性。

近年来，研究者提出了基于预训练语言模型（PLMs）的方法，以利用预训练语言模型（例如 BERT [14] 和 DistilBERT [33]）的语义能力，更有效地发现可连接表。代表性方法包括 DeepJoin [16], 、Snoopy [19] 和 Omnimatch [25]。如图 2(a) 和图 2(b) 所示，这些方法通常结合离线训练阶段与在线排序阶段，以兼顾效率和效果。在离线阶段，模型经过训练以学习列表示，将每列的内容编码为稠密向量。在在线阶段，给定一个查询列，系统会根据查询列与候选列之间的相似度来识别可连接列。

动机。 尽管这些基于预训练语言模型的方法取得了令人鼓舞的表现，但其设计未能充分考虑表内潜在的结构交互，而这些交互对于准确发现可连接表至关重要。具体而言：

首先，在离线训练阶段，这些方法通常将表划分为彼此孤立的列或成对的列。这种方法难以捕捉丰富的表内结构信息（例如列之间的高阶依赖关系，以及表与列之间的层次关系）和表间结构信息（例如全局连接拓扑）。因此，学习得到的表示缺乏足够的表达能力，无法捕捉这些复杂的依赖关系，导致准确率不够理想。具体而言，DeepJoin [16] 和 Snoopy [19] 分别处理各列以学习其表示。尽管 Omnimatch [25]尝试利用图来学习并建模列之间的关系，但它仍主要关注成对关系。

其次，在在线排序阶段，现有方法通常仅根据查询与候选列之间的相似度分数进行简单的 top-K 排序。这种方法忽略了候选列之间的相互关联和潜在依赖关系，因此会产生不连贯的结果集。整合这类列可能引入显著噪声，使无关信息掩盖整体洞见。例如，如图 1 所示，当查询为 Student$K$ ID 时，top-k 检索可能会包含来自 Campus_Bus 和 Dorm_Maintenance 的列。尽管这些列可连接，但由于它们源自不相关的领域，且无法支持对学生学业表现的预期分析，因此应排在较低位置。

挑战。 要设计一种准确的可连接表发现方法，面临以下两个挑战：

挑战一：如何通过统一框架捕捉复杂的结构关系，从而有效地进行离线表示学习？ 表存储库呈现出复杂的结构关系，既包括表间关系，也包括表内关系。此外，这些关系并非彼此孤立，而是高度相互依赖。例如，只有结合细粒度的表内语义，上下文中的表间列可连接性才有意义，因此孤立的建模方法并不适用。因此，一个重要挑战在于设计统一框架，联合利用这些复杂的结构关系来实现可连接表发现。

挑战 *II*：如何从大型数据湖中有效且高效地全局搜索相关且连贯的可连接表？ 数据湖通常包含数百到数千个异构表。考虑到相互之间的上下文，潜在连接路径的数量会呈指数级增长。此外，正如我们在定理4中证明的，确定能够最大化联合连贯性的最优 K列子集是 NP 难问题，因此在大型数据湖中进行穷举搜索难以实现。

我们的方法。 针对上述挑战，我们提出了 HyperJoin，这是一种基于**超**图的新型框架，将**可连接**表搜索问题重新表述为超图上的链接预测，从~~而~~更好地利用表的结构上下文。如图 2(c)所示，与以往基于 PLM 的方法类似，HyperJoin 也采用两阶段架构：离线表示学习阶段和在线排序阶段。

为解决挑战 I，在离线阶段，我们提出使用超图来建模表的结构先验并学习列表示。形式上，超图以列作为节点，并包含两种类型的超边：(1)大语言模型（*LLM*）增强的表间超边和 (2) 表内超边。具体而言，大语言模型增强的超边连接属于同一可连接列连通分量的所有列，从而显式建模表间可连接性信息。我们首先利用大语言模型增强数据

，生成语义等价的列名变体，然后将每个原始列及其大语言模型生成的变体视为统一的连接键实体，并利用这一扩展集合构建表间超边，以便在各表及其大语言模型生成的替代项之间传播可连接性信号。此外，表内超边连接同一张表中的所有列，以捕捉列之间的高阶关系以及列与表之间的关系。我们还设计了 HIN（分层交互网络），通过与两种类型超边相对应的分层消息传递来运行：首先在表内超边和大语言模型增强的表间超边中聚合列信息，然后通过全局混合步骤实现超边之间的全连接交互，使每一列都能更好地利用表内架构关系和表间可连接性信号。

为解决挑战 II，在在线阶段，我们引入一个全局一致重排序模块，以同时提升效率和准确性。具体而言，我们首先将全局一致性可连接表选择任务形式化为一个具有一致性感知能力的 top-K 列选择问题，在个体查询相关性与候选项间一致性之间进行权衡。我们证明了该问题是 NP 难。因此，所提出的全局一致重排序模块首先构建一个覆盖查询和目标列的加权完全图，以捕获第一阶段 top-B 检索（$K < B$）后各列的结构上下文。随后采用一种高效的贪心近似算法，该算法利用最大生成树（MST）算法剪除噪声连接并最大化全局匹配得分。**贡献**。 主要贡献如下：

• 我们提出 HyperJoin，这是一种用于可连接表发现的新型基于超图的框架，通过双类型超图利用结构先验。它包含离线表示学习阶段和在线排序阶段。

• 在离线阶段，我们构建一个包含大语言模型增强的表间超边和表内超边的超图，并设计 HIN，以实现有效的消息传递并学习具有表达力的列表示。

• 在在线阶段，我们提出了一种全局一致性重排序模块，利用基于最大生成树的高效贪心近似算法，以可行的方式发现连贯结果列。

• 在 4 个基准数据集上的实验表明，HyperJoin 表现优异：与此前最先进的方法相比，Precision@15 平均提升 21.4%，Recall@15 平均提升 17.2%。

2 预备知识

2.1 问题定义

我们遵循可连接表发现的典型设定 [13, 15–17, 19]。数据湖包含一个表仓库 $\mathbf{T} = \{T_1, T_2, \ldots, T_M\}$，其中有 M 个表。每个表 T_i 由 n_i 行（元组）和 m_i 列（属性）组成，其中每个单元格值表示为 c_{ij}。我们聚焦于文本列，并从 T 中提取所有此类列，构成列仓库 $\mathbf{C} = \{C_1, C_2, \ldots, C_N\}$，其中包含所有表中的 N 列。每个列 C表示为一组单元格值 $\{c_1, c_2, \ldots, c_{|C|}\}$，也可能包含相关元数据（例如列名），其中 $|C|$ 表示列 C的基数（单元格数量）。我们用 C_q 表示查询列。$\mathbf{R}^*$ 和

表 1：符号及其描述

记号	描述
$\mathcal{T}$	表仓库
T	由多个列组成的表
$\mathcal{C}$	列仓库（T 中的所有列）
C	由一组单元格值表示的列
C_q	查询列
$\|C\|$	列 C的基数（单元格数量）
$\mathbf{h}_C \in \mathbb{R}^d$	列 C的嵌入
$J(C_q, C_t)$	C_q 与 C_t之间的可连接性分数
K	要返回的结果数量
$\mathbf{R}, \mathbf{R}^*$	大小为 K的结果集 / 最优结果集
$\mathbf{H}(V, E)$	具有 V 个节点和 E条超边的超图
E_{inter}	表间超边
E_{intra}	表内超边
$f^{\theta}(\cdot)$	具有参数 θ的神经网络模型
λ	重排序目标中的连贯性权重

R 分别用于表示真实最优结果集和预测结果集。接下来，我们给出可连接表发现的正式定义。

定义 1（成对列可连接性）。给定查询列 C_q和来自存储库$\mathcal{C}$的候选列 C_t，其成对可连接性分数 $J(C_q, C_t)$ 定义为查询列中每个值的最佳匹配相似度的平均值：

$$J(C_q, C_t) = \frac{1}{|C_q|} \sum_{v_q \in C_q} \max_{v_t \in C_t} sim(v_q, v_t) \tag{1}$$

该方向性分数衡量 C_q被 C_t覆盖的程度。$sim(v_q, v_t) \in [0, 1]$ 是语义值对应关系，用于量化两个单元格值 v_i和v_j之间的语义相关性。

定义 2（可连接表发现）。给定查询列 C_q、表仓库$\mathcal{T}$和整数 K，可连接表发现的任务是选出一个大小为 K 的结果集$\mathcal{R}^* \subseteq \mathcal{C}$，使得每个$C_t \in \mathcal{R}$ 都与 C_q具有较高的可连接性。

2.2 超图

超图是图的一种推广，其中一条边（称为超边）可以连接任意数量 的顶点，而不只是两个 [3, 27]。形式上，超图 $\mathbf{H} = (V, E)$由一个顶点集 V和一个超边集 E构成，其中每条超边 $e \in E$ 表示多个顶点之间的群组级 关系（即 $\emptyset \neq e \subseteq V$）。这种能力使超图成为对现实世界数据中的高阶交互进行建模的天然工具，例如共同署名 [44] 和生物复合体 [29]。

2.3 语言模型与当前最先进技术

语言模型是现代自然语言处理（NLP）的基石，其根本目标是对文本序列的概率分布进行建模 [24]。该领域普遍采用的一种架构是自回归模型，它通过根据

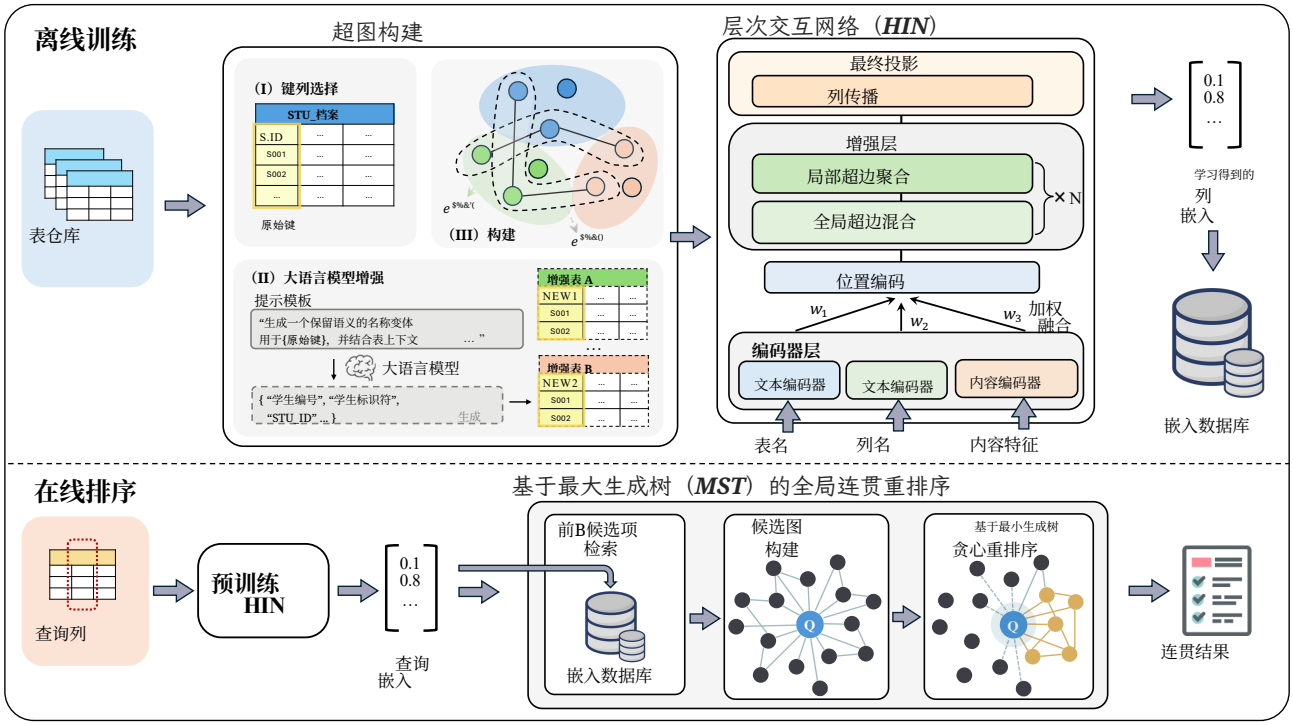


图 3：HyperJoin 框架概览。

前文上下文 [40, 41] 迭代预测下一个词元来生成序列。在这些基础之上，该领域已从预训练语言模型发展到大语言模型 [38, 39, 47]。

预训练语言模型（PLM），例如自编码模型 BERT [14] 及其变体 DistilBERT [33] 和 MPNet [35],，通过从海量文本语料库中学习上下文化表示，推动了自然语言处理的发展。这些模型将文本序列映射为能够捕捉语义相似度的稠密向量，通常需要针对下游应用进行特定任务的微调。相比之下，大语言模型（例如 GPT系列 [32]）在规模上实现了重大飞跃，其参数量通常达到数百亿甚至数千亿。尽管大语言模型与预训练语言模型采用相同的预训练范式，但其庞大的规模带来了“涌现能力”，例如零样本推理和上下文学习，使其无需显式更新参数便能解决复杂任务。

最先进技术。当前可连接表发现的最先进解决方案大多基于预训练语言模型，并遵循通用的离线/在线范式。在离线阶段，模型经过训练（或微调），并预先计算存储库中所有列的嵌入。通常还会构建近似最近邻索引（例如 FAISS 库中的 IVFPQ [22] 或 HNSW [30] ），以实现高效检索 [42]。在线阶段，使用相同的模型对查询列进行编码，并从索引中检索出与其最相似的前 k 个列，作为可连接表。

尽管这些方法共享相同的框架，但它们获取列嵌入的方式各不相同。DeepJoin [16] 对 DistilBERT 进行微调，将列嵌入固定维度的向量中。它通过一组选项将列内容和元数据转换为文本序列，以提供上下文信息。Snoopy [19] 引入代理列，以克服直接使用 PLM 编码的局限性。它通过考虑排名的对比学习，学习一组代理列嵌入。Omnimatch [25] 构建了一个相似度图，其中节点表示列，边则编码多种信号。它使用关系图卷积网络（RGCN）学习能够聚合邻域信息的列嵌入。

3 概述

HyperJoin 的整体架构如图 3 所示。我们不再将表视为孤立的对象或仅考虑成对列，而是将整个数据湖建模为一个超图，其中节点表示列，超边则显式编码表内架构约束和表间相关性。因此，我们将可连接表发现任务重新表述为超图上的链接预测问题。为有效解决这一链接预测任务，我们设计了 HIN。该网络通过执行分层消息传递，学习具有丰富表达能力的列嵌入。它由三个关键组件组成：（1）位置编码模块，用于捕捉表的身份和全局结构角色；（2）局部超边聚合模块，用于在特定的表内和表间上下文中传播信息；（3）全局超边混合模块，用于实现超边之间的全连接交互。在在线搜索阶段，我们引入全局一致性重排序模块，以选出可连接的列集合。我们将该任务表述为候选图上的一致性感知 top-K 列选择问题，并使用基于最小生成树的算法求解。

4 离线表征学习 ING

本节介绍 HyperJoin 的离线表示学习阶段。本节分为两个主要部分。首先，第 4.1 节介绍如何从数据湖构建统一超图

，包括初始列特征编码、由大语言模型增强的表间超边和表内超边构建。其次，第 4.2 节介绍在所构建超图上运行的分层交互网络，并详细说明网络架构和训练目标。

4.1 超图构建

为了更全面地捕获数据湖的结构上下文，我们将数据湖建模为一个超图，其中单条超边可以同时连接多个列，从而能够统一建模这些复杂的结构关系。

超图定义。形式上，我们有一个超图 $\mathrm{H} = (\mathrm{V}, \mathrm{E}, \mathbf{X}^v, \mathbf{X}^e, \mathbf{\Pi})$，其中 $\mathrm{V} = \{v_1, \ldots\ldots, v_N\}$ 表示 N节点的集合，每个节点对应数据湖中的一列；$\mathrm{E} = \{e_1, \ldots\ldots, e_M\}$ 表示 M超边的集合。每个节点 $v_i \in \mathrm{V}$ 都关联一个初始特征向量 $\mathbf{x}_{v_i}^v \in \mathbb{R}^d$，而 $\mathbf{X}^v \in \mathbb{R}^{N\times d}$按行堆叠所有节点特征。每条超边 $e_j \subseteq \mathrm{V}$ 表示一个列上下文（例如表内或表间上下文），并可选择性地关联一个初始超边特征向量 $\mathbf{x}_{e_j}^e \in \mathbb{R}^{d_e}$，其中$\mathbf{X}^e \in \mathbb{R}^{M\times d_e}$ 按行堆叠所有超边特征。超边可以重叠，这意味着一列可以参与多个上下文组。超图结构由节点—超边关联矩阵 $\mathbf{\Pi} \in \{0, 1\}^{N\times M}$表示，其中 $\mathbf{\Pi}_{ij} = 1$当 $v_i \in e_j$时取值为 1，否则为 0。

超图构建。H 的构建分为三个步骤：(1) 我们通过融合来自模式名称和数据内容的语义信号，编码初始节点特征 $\mathbf{X}^v$；(2) 基于物理表模式构建表内超边 $\mathrm{E}_{\text{intra}}$；(3) 通过识别由连接关系关联的列组来构建大语言模型增强的表间超边 $\mathrm{E}_{\text{inter}}$，并使用大语言模型生成的语义变体加以增强。

步骤 *1*：初始列特征编码。数据湖中的列包含来自多个来源的丰富语义信息：表名提供高层次的上下文，列名描述具体语义，而单元格值则提供直接证据。稳健的初始表示应融合所有这些来源，为后续的结构感知学习奠定坚实基础。

每个列节点 v_i 都通过融合三个组成部分——表名、列名和单元格值——来初始化其特征向量 $\mathbf{x}_{v_i}^v$。我们使用两个相互独立的文本编码器对表名和列名进行编码，分别得到 $\mathbf{h}_i^{\text{table}} \in \mathbb{R}^{d_0}$ 和 $\mathbf{h}_i^{\text{col}} \in \mathbb{R}^{d_0}$。对于单元格值，我们通过简单的统计池化聚合预训练的值嵌入，获得内容表示，并将结果输入两层 MLP，得到 $\mathbf{h}_i^{\text{content}} \in \mathbb{R}^{d_0}$。令 $\mathrm{S} = \{\text{table}、\ \text{col}、\ \text{content}\}$ 表示这些组成部分，令$\mathbf{w} \in \mathbb{R}^3$为可学习的融合权重；我们计算

$$\mathbf{h}_i^{(0)} = \sum_{s\in\mathcal{S}} \mathrm{softmax}(\mathbf{w})_s \cdot \mathbf{h}_i^s, \tag{2}$$

随后通过线性投影得到 $\mathbf{x}_{v_i}^v = \mathbf{W}_0\mathbf{h}_i^{(0)} \in \mathbb{R}^d$。将 $\mathbf{x}_{v_i}^v$堆叠后，得到 H 中使用的节点特征矩阵 $\mathbf{X}^v \in \mathbb{R}^{N\times d}$。

步骤 *2*：表内超边。对于每个表 T，我们构建一个表内超边 $e_T^{\text{intra}} = \{v_i \mid v_i\text{对应于 } T \text{ 中的一列}\}$。这样做基于两点考虑：(i) 表会自然地将语义相关的列归为一组；(ii) 同一表中共现的列通常会共享超越连接关系的上下文。因此，这些超边能够捕获模式级共现信号。

步骤 *3*：表间超边。为构建 $\mathrm{E}_{\text{inter}}$，我们构建辅助连接图 $\mathrm{G}_{\text{join}} = (\mathrm{V}, \mathrm{P}_{\text{join}})$，其中 $(v_i, v_j) \in \mathrm{P}_{\text{join}}$ 表示对应的列对可连接。我们使用并查集获取 G_{join} 的连通分量 $\{\mathrm{C}_k\}$；每个包含 $|\mathrm{C}_k| \geq 2$ 的分量都会导出一个表间超边 $e_k^{\text{inter}} = \mathrm{C}_k$，并将 $\mathrm{E}_{\text{inter}}$ = 设置为 $\{e_k^{\text{inter}}\}$。为在缺少真实连接信息的情况下增强可连接性信号，我们通过以下步骤，利用大语言模型生成的语义变体来构建连接图：(i) 键列选择。我们使用标准的键性启发式方法（例如完整性/唯一性信号）识别候选连接键列；(ii) 大语言模型增强。对于每个选定的连接键列 C_{key}，我们使用大语言模型生成一个保持语义不变的名称变体 C'_{key}，并以表上下文（例如表名/列名和样例行）为条件。提示模板见框 4.1；(iii) 表间超边生成。我们向 G_{join} 添加边 $(C_{\text{key}}, C'_{\text{key}})$，然后重新计算连通分量，以形成最终的 $\mathrm{E}_{\text{inter}}$。

提示设计

输入上下文。列名、表名、表中所有列名，以及 3–5 行采样数据。
转换指导。生成一个语义等价的变体，并遵循常见的数据库命名约定：

• 分隔符变更（例如，CustomerID → Customer_ID、customer-id）• 缩写（例如，CustomerID → CustID）• 大小写约定（例如，CustomerID → customerId、CUSTOMERID）• 同义词（例如，CustomerID →ClientID）**输出**。以结构化格式（例如，JSON：{"perturbed":"Cust_ID"}）返回一个最合理的变体。

理论分析。我们对为何结合表内上下文和表间上下文能够改进可连接表发现提供了形式化论证，并证明先前的最先进方法是我们这一形式化方法的特例。所有证明均见我们的在线完整版 [28]。

设定。 令 $Y_{ij} \in \{0, 1\}$ 表示两列（v_i, v_j）是否可连接。令 $\mathbf{x}_i$为列 v_i 的初始特征。对于每列 v_i，定义由构建的超图诱导的两个上下文变量：(i) 其表内上下文 $\mathrm{N}^{\text{intra}}(v_i)$（与 v_i 位于同一张表中的列集合），以及 (ii) 其表间上下文 $\mathrm{N}^{\text{inter}}(v_i)$（通过可连接性与 v_i 相连的列集合，例如位于同一连接组件中的列）。我们将用于判断可连接性的可用信息记为一个特征集：

$$\begin{aligned}\mathcal{F}_{\text{single}}(i, j) &= \{\mathbf{x}_i, \mathbf{x}_j\},\\ \mathcal{F}_{\text{multi}}(i, j) &= \Big\{\mathbf{x}_i, \mathbf{x}_j, \mathcal{N}^{\text{intra}}(v_i), \mathcal{N}^{\text{intra}}(v_j),\\ &\qquad \mathcal{N}^{\text{inter}}(v_i), \mathcal{N}^{\text{inter}}(v_j)\Big\}.\end{aligned} \tag{3}$$

令 $\Psi^*(\mathrm{F})$ 表示特征集 F 下可实现的最小期望发现风险：

$$\Psi^*(\mathcal{F}) = \inf_g \mathbb{E}\big[\ell(g(\mathcal{F}), Y_{ij})\big],$$

其中，g 是任意可测评分函数，$\ell(\cdot)$是有界损失（例如逻辑损失或 0-1 损失）。

定理1。 对于上述定义的两个特征集 $\mathcal{F}_{multi}$ 和 $\mathcal{F}^{single}$，贝叶斯最优风险满足

$$\Psi^*(\mathcal{F}_{multi}) \le \Psi^*(\mathcal{F}_{single}).$$

此外，如果新增的上下文是有信息的，即 $\mathbb{P}(Y_{ij}=1 \mid \mathcal{F}_{multi}) \neq \mathbb{P}(Y_{ij}=1 \mid \mathcal{F}^{single})$ 以非零概率成立，则不等式是严格的。

定理 2。 考虑我们的框架，该框架通过在构建的超图上运行的模型 f_θ 来学习列嵌入。*DeepJoin* 和 *Snoopy* 可视为通过将所使用的上下文限制为 $\mathcal{F}^{single}$（即仅进行按列编码，而不使用 $\mathcal{N}^{intra}$ 或 $\mathcal{N}^{inter}$）得到的特例。

4.2 HINModel 设计

我们介绍我们的 HIN，它在构建的超图上运行，以学习具有表现力的列表示。我们的架构由三个组件组成：(1) 位置编码、(2) 局部超边聚合和 (3) 全局超边混合。

位置编码。 标准的消息传递图神经网络（GNNs）存在位置无感知问题 [20], ：即使节点在可连接性图中扮演不同的全局角色，具有相同局部邻域的节点也可能获得相同的表示。这一局限性由一维 *Weisfeiler–Lehman*（*1-WL*）测试 [43]进行了形式化刻画。为缓解这一问题，我们引入两种位置编码：(i) 表级位置编码和 (ii) 列级拉普拉斯位置编码。

表级位置编码。 我们维护一个可学习的查找矩阵 $\mathbf{E}_{\text{tbl}} \in \mathbb{R}^{|\mathcal{T}|\times d}$，其中每一行存储一个可训练的 d 维向量，对应一个表 ID。对于属于表 $t(v)$ 的列 v，我们通过以下方式注入表的身份信息：

$$\mathbf{h}_v^{\text{tbl-pe}} = \mathbf{x}_v^v + \alpha \cdot \mathbf{E}_{\text{tbl}}[t(v)], \tag{4}$$

其中 α 是可学习标量权重（初始化为 0.1），并与模型联合优化。

列级位置编码。 我们采用拉普拉斯位置编码来捕捉可连接性拓扑中列的全局结构角色。我们构建一个成对列图 $\mathrm{G}_{\text{col}} = (\mathrm{V}, \mathrm{P}^{\text{join}})$，其中每条边 $(v_i, v_j) \in \mathrm{P}^{\text{join}}$ 表示对应的列对可连接。我们通过设置 $A_{ij} = A_{ji} = 1$（当且仅当 $(v_i, v_j) \in \mathrm{P}^{\text{join}}$ 时），否则设为 0，来构建对称邻接矩阵 $\mathbf{A} \in \{0,1\}^{N\times N}$。我们计算归一化拉普拉斯矩阵

$$\mathbf{L}_{\text{norm}} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{5}$$

其中 $\mathbf{I}_N$ 是 $N \times N$ 单位矩阵，$\mathbf{D}$ 是对角度矩阵。我们提取最小的 k 个特征向量 $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$，并通过一个两层 MLP 对它们进行投影：

$$\mathbf{PE}_{\text{col}}(v_i) = \mathbf{W}_2 \cdot \sigma\Big(\mathbf{W}_1 \cdot [\mathbf{v}_1[i], \ldots, \mathbf{v}_k[i]]^\top\Big). \tag{6}$$

其中，$\mathbf{W}_1 \in \mathbb{R}^{d/2\times k}$ 和 $\mathbf{W}_2 \in \mathbb{R}^{d\times d/2}$ 是可学习的投影矩阵，$\sigma(\cdot)$ 是非线性激活函数（例如 ReLU）。这种编码捕获了可连接性图的全局谱结构。为提高效率，我们在线下预先计算特征向量，并缓存 $\mathbf{PE}_{\text{col}}$，以便在训练期间查找。

统一位置编码。 最终的位置编码为：

$$\mathbf{h}_v = \mathbf{x}_v^v + \alpha \cdot \mathbf{E}_{\text{tbl}}[t(v)] + \beta \cdot \mathbf{PE}_{\text{col}}(v), \tag{7}$$

其中，$\mathbf{x}_v^v$ 是原始初始列嵌入，$\mathbf{E}_{\text{tbl}}[t(v)]$ 是包含列/节点 v 的表的表级嵌入，$\mathbf{PE}_{\text{col}}(v)$ 是基于拉普拉斯的列位置编码，β 是一个可学习标量权重，其初始值与 α 相同。它们共同使模型能够自动调整每个位置编码信号的强度。

局部超边聚合。 传统图神经网络会受到过度挤压问题的影响 [4, 9], ，即远距离节点的信息在经过多层传播后会被指数级压缩。我们采用基于层次化超边的方法来聚合局部和全局信息：首先将列聚合为 M 超边（$M \ll N$），然后在超边级进行消息传递，从而缩短路径长度并缓解信息瓶颈。

阶段 *1*：节点级特征变换。 给定经过 PE 增强的嵌入 $\{\mathbf{h}_v^{(0)}\}_{v\in\mathcal{V}}$，我们应用 $L = 2$ 层节点变换：

$$\mathbf{h}_v^{(\ell)} = \text{LayerNorm}(\sigma(\mathbf{W}^{(\ell)}\mathbf{h}_v^{(\ell-1)} + \mathbf{b}^{(\ell)})), \tag{8}$$

其中 $\sigma(\cdot)$ 是 ReLU 激活，$\mathbf{h}_v^{(0)} = \mathbf{h}_v$。与在此阶段聚合邻居信息的传统图神经网络层不同，我们将聚合推迟到第 2 阶段，在超边级进行。

第 *2* 阶段：聚合至超边级。我们通过均值池化将节点嵌-入聚合到超边级：

$$\mathbf{x}_{e_j}^e = \frac{1}{|e_j|}\sum_{v\in e_j} \mathbf{h}_v^{(L)}, \tag{9}$$

其中 $e_j \subseteq \mathrm{V}$ 表示第 j 个超边的节点集。令 $\mathbf{H} \in \mathbb{R}^{N\times d}$ 按行堆叠节点嵌入，即 $\mathbf{H}[v,:] = \mathbf{h}_v^{(L)}$；令 $\mathbf{X}^e \in \mathbb{R}^{M\times d}$ 按行堆叠超边嵌入，即 $\mathbf{X}^e[j,:] = \mathbf{x}_{e_j}^e$。这可以通过稀疏矩阵乘法高效计算：

$$\mathbf{X}^e = \mathbf{D}_e^{-1}\mathbf{\Pi}^\top\mathbf{H}, \tag{10}$$

其中 $\mathbf{\Pi} \in \{0,1\}^{N\times M}$ 是节点–超边关联矩阵，$\mathbf{D}_e \in \mathbb{R}^{M\times M}$ 是对角矩阵，且 $(\mathbf{D}_e)_{jj} = \sum_{i=1}^N \mathbf{\Pi}_{ij} = |e_j|$。每个节点最多属于两个超边（一个表内超边，至多一个表间超边），因此 $\mathbf{\Pi}$ 非常稀疏，时间复杂度为 $O(\|\mathbf{\Pi}\|_0 \cdot d)$，其中 $\|\mathbf{\Pi}\|_0 \le 2N$。

我们将聚合后的超边嵌入表示为 $\mathbf{Z}^{(0)} = \mathbf{X}^e$，并按超边类型对其进行划分。表间超边和表内超边捕获的是本质上不同的关系：跨表连接语义与表内模式语义。我们采用不同的线性变换来学习领域特定模式（小写 =表示单个嵌入，大写 =表示堆叠矩阵）：

$$\tilde{\mathbf{Z}}_{\text{inter}} = \mathbf{Z}_{\text{inter}}\mathbf{W}_{\text{inter}}, \tag{11}$$

$$\tilde{\mathbf{Z}}_{\text{intra}} = \mathbf{Z}_{\text{intra}}\mathbf{W}_{\text{intra}}, \tag{12}$$

并将它们拼接起来：$\tilde{\mathbf{Z}} = [\tilde{\mathbf{Z}}_{\text{inter}};\ \tilde{\mathbf{Z}}_{\text{intra}}] \in \mathbb{R}^{M\times d}$。

全局超边混合。虽然在超边图上进行局部消息传递能够捕捉短程依赖关系，但要建模长程可连接关系，我们仍需在超边之间进行全局信息交换。然而，标准的超图消息传递本质上是局部的（通常每层传递 1 跳），因此需要多层才能覆盖全局，这会增加计算量，并可能加剧过度挤压。受 MLP-Mixer [36], 的启发，我们设计了一种全局混合机制，可在单层中实现全连接通信，同时通过结构感知注意力保留超图结构先验。

混合器架构。我们的混合器每层包含两个依次执行的操作：超边之间的超边交互（HI），以及在每个超边嵌入内部进行的超边精炼（HR）。

对于 HI，我们采用结构感知多头自注意力，将内容相似性与结构邻近性相结合。为捕捉结构关系，我们首先构建超边邻接矩阵 $\mathbf{A}_{\text{hyperedge}} \in \mathbb{R}^{M\times M}$，其中条目 $[i, j]$ 反映超边之间共享列的数量 i，并且 j：

$$\mathbf{A}_{\text{hyperedge}} = \text{RowNormalize}\Big(\mathbf{\Pi}^T\mathbf{\Pi} - \text{diag}(\mathbf{\Pi}^T\mathbf{\Pi})\Big). \tag{13}$$

我们通过加性偏置项将这一结构先验注入多头自注意力：

$$\text{head}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_k}} + \lambda\cdot\mathbf{A}_{\text{hyperedge}}\right)\mathbf{V}_i, \tag{14}$$

其中 $\mathbf{Q}_i$，$\mathbf{K}_i$，$\mathbf{V}_i$ 分别是第 i 个头的查询、键和值矩阵，λ 是一个初始化为 0.5 的可学习标量。这样一来，即使超边之间的内容相似性较低，只要结构连接较强，它们仍然可以交换信息。我们使用 $h = 8$ 个头，并通过残差连接合并输出：$\mathbf{Z}_{\text{HI}} = \tilde{\mathbf{Z}} + \text{Concat}(\text{head}_1,\ldots,\text{head}_h)\mathbf{W}_O$。

对于 HR，我们采用两层 MLP：

$$\mathbf{Z}_{\text{out}} = \mathbf{Z}_{\text{HI}} + \mathbf{W}_2\cdot\text{GELU}(\mathbf{W}_1\cdot\text{LayerNorm}(\mathbf{Z}_{\text{HI}})^T)^T, \tag{15}$$

其中 $\mathbf{W}_1 \in \mathbb{R}^{4d\times d}$ 和 $\mathbf{W}_2 \in \mathbb{R}^{d\times 4d}$ 分别将维度扩展并以 4 倍因子投影回原维度。我们堆叠 $L_{\text{mixer}} = 2$ 层，所有层共享 $\mathbf{A}^{\text{hyperedge}}$。

将信息传播回列。我们将增强后的超边嵌入传播回列：

$$\mathbf{h}_v^{\text{hyperedge}} = \frac{1}{|\mathcal{N}(v)|}\sum_{j\in\mathcal{N}(v)}\mathbf{W}_{\text{h2c}}\mathbf{z}_{e_j}^{(\tilde{L_{\text{mixer}}})}, \tag{16}$$

其中，$\mathcal{N}(v) = \{j \mid \mathbf{\Pi}_{vj} = 1\}$ 表示包含列 v 的超边集合，$\mathbf{W}_{\text{h2c}} \in \mathbb{R}^{d\times d}$ 用于将超边嵌入映射到列级空间。最终的列嵌入结合了原始信息与超边增强信息：

$$\mathbf{h}_v^{\text{final}} = \text{L2Norm}(\text{LayerNorm}(\mathbf{h}_v + \mathbf{h}_v^{\text{hyperedge}})). \tag{17}$$

残差连接保留了原始语义，而 L2 归一化则支持高效的余弦相似度搜索。

HIN 的整体前向过程在在线完整版附录的算法 2 中进行了总结 [28]。以下定理表明，我们的模型比 1-WL测试更具表达能力。证明见在线完整版 [28]。

定理3（HIN 严格强于 1-WL消息传递）。设$\mathcal{H}_{base}$为在可连接性图上运行、使用原始节点特征$\mathbf{h}^{(0)}$（即 *1-WL*初始化输入）的置换等变消息传递编码器假设类。设$\mathcal{H}^{pe}$为通过向输入添加以下内容而得到的类

$$PE(v) = \alpha\mathbf{E}_{tbl}[t(v)] + \beta\mathbf{PE}_{col}(v), \tag{18}$$

并采用单射节点级更新（如 *HIN* 中所用）。假设存在两个节点 u和 v，在原始初始化下它们在 *1-WL* 意义上不可区分，但 $PE(u) \neq PE(v)$。那么 $\mathcal{H}_{base} \subsetneq \mathcal{H}^{pe}$。

复杂度分析。在阶段 1 中，计算前 k 个拉普拉斯特征向量是离线完成的（对于小图使用稠密$O(N^3)$，对于大型/稀疏图则使用稀疏迭代求解器）；在线位置注入的开销为 $O(Nd)$。在阶段 2 中，L 层节点级变换的开销为 $O(LNd^2)$，通过 $\mathbf{X}^e = \mathbf{D}_e^{-1}\mathbf{\Pi}^\top\mathbf{H}$将节点池化到超边的开销为 $O(\|\mathbf{\Pi}\|_0 d)$。在阶段 3 中，每个混合器层的开销为 $O(M^2d + Md^2)$，因此总开销为 $O(L_{\text{mixer}}(M^2d + Md^2))$。因此，整体在线复杂度为 $O(LNd^2 + L_{\text{mixer}}(M^2d + Md^2))$，其中 $\|\mathbf{\Pi}\|_0 \le 2N$，且通常为 $M \ll N$（例如，$M \approx 0.1N$–$0.3N$）。

训练目标。我们采用无标签的自监督训练策略，将每个表拆分为两个相互重叠的子集，并将连接键列（包括由大语言模型生成的名称变体）视为正样本对，同时从同一拆分中的非键列以及不同表中采样负样本。我们的训练目标是鼓励模型为可连接列对分配比不可连接列对更高的相似度，且两者之间至少相差一个间隔。给定由算法 2 生成的最终列嵌入$\mathbf{h}_C^{\text{final}}$，我们将列 C_a 和 C_b 之间的余弦相似度定义为

$$s(C_a, C_b) = \frac{\mathbf{h}_{C_a}^{\text{final}\top}\mathbf{h}_{C_b}^{\text{final}}}{\left\|\mathbf{h}_{C_a}^{\text{final}}\right\|_2\left\|\mathbf{h}_{C_b}^{\text{final}}\right\|_2}. \tag{19}$$

令 $\mathbf{D}_{\text{train}}^+$ 表示可连接（正）样本对集合，$\mathbf{D}_{\text{train}}^-$ 表示不可连接（负）样本对集合。对于每个锚点列 C_i，我们采样一个正样本 C_i^+，满足（C_i，C_i^+）$\in \mathbf{D}_{\text{train}}^+$；并采样一个负样本 C_i^-，满足（C_i，C_i^-）$\in \mathbf{D}_{\text{train}}^-$。我们优化基于间隔的三元组排序损失

$$\mathcal{L}_{\text{triplet}} = \frac{1}{|\mathcal{D}_{\text{train}}^+|}\sum_{(C_i,C_i^+)\in\mathcal{D}_{\text{train}}^+}\Big[m + s(C_i, C_i^-) - s(C_i, C_i^+)\Big]_+, \tag{20}$$

其中，$[\cdot]_+ = \max(\cdot, 0)$，且 m 为间隔超参数。

5 在线排序

我们介绍 HyperJoin 的在线排序阶段，该阶段能够从大规模数据湖中高效检索连贯且可连接的列。第 5.1 节讨论目标函数及其计算复杂度。随后，第 5.2 节介绍一种高效的全局一致性重排序模块：首先通过相似度搜索检索一个规模适中的候选池并构建候选图，然后使用结合最小生成树的贪心算法进行重排序。

5.1 难度分析

给定查询列 C_q、快速相似度搜索阶段返回的初始候选池 $\mathbf{C}_B$ 以及目标规模 K，我们的目标是从 $\mathbf{C}_B$中选取一个子集 $\mathbf{R}^* \subseteq$，使其 $|\mathbf{R}^*| = K$，并且既与 C_q高度相关，又能彼此连接，构成一致性结果集。为量化相关性和一致性，我们使用列之间的成对亲和度分数 $w(\cdot,\cdot)$，其中 $w(C_i, C_j) \in \mathbb{R}^+$反映了它们的可连接性和语义相似度。一致性结果集既应与查询相关，其内部也应具有良好的连通性。自然地，可以通过以下目标函数为大小为 K 的集合评分，以刻画这种权衡：

$$\mathcal{R}^* = \underset{\mathcal{R}\subseteq C_B,|\mathcal{R}|=K}{\arg\max}\ G(\mathcal{R}), \tag{21}$$

其中目标函数 $G(\mathbf{R})$ 为：

$$G(\mathcal{R}) = \sum_{C\in\mathcal{R}} w(C_q, C) + \lambda\cdot\text{Coherence}(\mathcal{R}) \tag{22}$$

第一项汇总单个查询的相关性，第二项则衡量选定列之间的互联紧密程度。超参数 $\lambda \geq 0$ 控制相关性与一致性之间的权衡。

在式 21 下选择最优的 K 列子集是一个组合优化问题。以下定理对求解此类问题的难度进行了分析。证明见在线完整版 [28]。

定理4（连贯性感知选择的 NP难性）。当一致性（R）定义为诱导子图$\{C_q\} \cup \mathcal{R}$上以 C_q 为根的最大生成树的权重时，在基数约束$|\mathcal{R}| = K$ 下最大化式 *21* 中的目标函数 G（R）是 *NP* 难的。

5.2 全局连贯搜索

给定查询列 C_q，离线阶段会为数据湖中的每一列 C 提供结构感知嵌入 h_C。查询时，我们旨在求解式 21 中的一致性感知 top-K 选择目标。定理4表明，选择大小为 K 的最优子集是 NP 难的。因此，我们采用兼顾可扩展性和有效性的两阶段框架：（1）检索一个规模适中的候选池，并构建加权候选图；（2）基于最小生成树执行贪心重排序，返回全局一致的 top-K 集合。

候选图构建。给定查询列 C_q，数据湖包含 N 个列。为了避免难以处理的搜索空间 $\binom{N}{K}$，我们首先通过高效的相似度搜索检索一个规模适中的候选池，大小为 B（其中 $K < B \ll N$），然后仅在此候选池中执行考虑一致性的优化，从而得到候选集 C_B。

定义3（候选图）。给定查询列 C_q和通过相似度搜索得到的候选集 C_B（通常 $|C_B| = B > K$），候选图定义为加权无向图$\mathcal{G}_C = (\mathcal{V}_C, \mathcal{E}_C, w)$，其中 $\mathcal{V}_C = \{C_q\} \cup C_B$ 包含查询列和所有候选列，而边集 $\mathcal{E}_C = \mathcal{E}_{query} \cup \mathcal{E}_{cand}$ 由查询到候选边和候选间边组成。权重函数 w： $\mathcal{E}_C \to \mathbb{R}^+$ 为每条边赋予一个正权重：

$$w(C_i, C_j) = sim(h_{C_i}, h_{C_j}), \tag{23}$$

其中 h_{C_i}、h_{C_j}是由 *HIN* 学习得到的结构感知的列嵌入（第 *4.2* 节）。

候选图通过以下步骤构建。首先，对于给定的查询列 C_q，我们根据其学习到的嵌入之间的余弦相似度，在数据湖中的所有列上执行高效的相似度搜索。由此得到按相关性排序的前B 个候选项列表 $\mathrm{C}_B = \{C_1, C_2, \ldots, C_B\}$，从而确保高召回率。其次，我们建立查询到候选边 $\mathrm{E}^{query} = \{(C_q, C) \mid C \in \mathrm{C}_B\}$，将查询列与所有候选项连接起来，并为其分配边权重 $w(C_q, C_i) = \mathrm{sim}(h_{C_q}, h_{C_i})$，以反映每个候选项与查询的相关性。第三，为捕捉候选项间一致性，我们在所有候选项对之间建立候选间边 $\mathrm{E}_{cand} = \{(C_i, C_j) \mid C_i, C_j \in \mathrm{C}_B, i \neq j\}$，其边权重 $w(C_i, C_j) = \mathrm{sim}(h_{C_i}, h_{C_j})$ 反映两个候选项之间的可连接性和语义相似度。该阶段通过将选择范围限制在前B 个候选项内，将搜索空间从 $\binom{N}{K}$ 缩减至 $\binom{B}{K}$，同时保持高召回率。

基于最小生成树的贪心重排序。在构建候选图 G_C后，核心挑战是高效地选择一个集合 $\mathrm{R} \subseteq \mathrm{C}_B$，使其最大化公式 21 中的目标函数 G (R)。这需要实例化连贯性项 Coherence(R)，用于量化选定列在联合分析中整合得有多好。一种看似直接的确保一致性的方法，是要求返回的列构成一个稠密团，即每一对列都具有很强的可连接性。对于真实数据湖而言，这一假设过于严格，因为其中的连接图通常较为稀疏，并且通过桥接表进行多跳连接的情况很常见。

相反，一致性应确保 R 中的每一列都能通过一系列强连接关系从查询 C_q 到达。对此概念进行量化的自然方式，是计算在 $C_q \cup \mathrm{R}$ 上的诱导子图中构建的最大生成树 (MST) 的权重。MST 提取能够连接所有列且不形成环的最强边集，确保从查询锚点到每个结果都可达，同时兼容真实数据湖中常见的多跳连接，并避免稠密子图度量过于严格的问题。形式化地，对于顶点集 $\mathrm{V}' = C_q \cup \mathrm{R}$，令 $\mathrm{G}_C[\mathrm{V}']$ 表示 G_C中由 V' 诱导的子图。R 的一致性定义为 $\mathrm{G}_C[\mathrm{V}']$的 MST 权重：

$$\mathrm{Coherence}(\mathcal{R}) = \max_{T \in \mathcal{S}(\mathcal{V}')} \sum_{(u,v) \in E(T)} w(u, v), \tag{24}$$

其中，S(V′) 是 $\mathrm{G}_C[\mathrm{V}']$的所有生成树构成的集合，$E(T)$ 表示树 T的边集。

对于在线搜索而言，通过为每个候选子集 R 显式构建 MST 来计算 G(R)，计算开销过高。因此，我们提出一种贪心算法，将 Prim 最小生成树构造的逻辑与子集选择巧妙结合，从而无需在每次迭代时从头重新计算 MST。该算法通过模拟以下过程，逐步构建结果集 R：从查询 C_q 开始生长的树根

- **初始化：** 从 $\mathrm{R} = \emptyset$ 开始，并将当前连通分量定义为 $S = \{C_q\}$。由于 S 最初仅包含查询节点，因此任意候选节点 C 的一致性增益项都只是 $w(C, C_q)$；因此，第一个选中的节点是查询相关性最高的节点。
- **选择规则：** 每一步，对每个候选节点 $C \in \mathrm{C}_B \setminus \mathrm{R}$，计算边际增益

$$\Delta(C \mid \mathcal{R}) = w(C_q, C) + \lambda \cdot \max_{C' \in S} w(C, C'), \tag{25}$$

其中 $S = \{C_q\} \cup \mathrm{R}$。第一项表示与查询的相关性，第二项衡量 C 与当前分量 S之间可能的最强连接——这与 Prim 算法中的割步骤相呼应：该算法通过连接跨越生长中树的权重最大的边来添加新节点。随后，算法选择候选节点

$$C^* = \arg\max_{C \in C_B \setminus \mathcal{R}} \Delta(C \mid \mathcal{R}).$$

- **维护：** 更新结果集 $\mathrm{R} \leftarrow \mathrm{R} \cup \{C^*\}$ 和连通分量 $S \leftarrow S \cup \{C^*\}$。这种增量扩展确保每个新加入的节点都通过当前可用的最强边与现有结构相连，从而逐步构建高权重的连通性骨架。重复选择和维护步骤，直到 $|\mathrm{R}| = K$。

算法 1： 全局连贯搜索

输入： 查询列 C_q、嵌入数据库 $\{h_C \mid C \in \mathbf{C}\}$，目标规模 K、候选规模 B、连贯性权重 λ。

输出： 最终结果集 R。

// **阶段 1：候选图构建**

$\mathbf{C}_B \leftarrow$ 基于 h_{C_q} 进行 Top-B 相似度搜索

构建候选图 $\mathbf{G}_C = (\{C_q\} \cup \mathbf{C}_B, E_C, w)$

计算边的边权重 $w(C_i, C_j) \leftarrow \mathrm{sim}(h_{C_i}, h_{C_j})$

// **阶段 2：基于最小生成树的贪心重排序**

$\mathbf{R} \leftarrow \emptyset$

$U \leftarrow \mathbf{C}_B$ // 未选候选项

遍历 $C \in U$ **执行**

  $\mathrm{rel}[C] \leftarrow w(C_q, C)$

  $\mathrm{key}[C] \leftarrow w(C_q, C)$ // 最佳连接到 $S = \{C_q\}$

对于 $i = 1, \ldots, K$ **do**

  // 选择边际增益最大的候选项

  $C^* \leftarrow \arg\max_{C \in U} (\mathrm{rel}[C] + \lambda \cdot \mathrm{key}[C])$

  $\mathbf{R} \leftarrow \mathbf{R} \cup \{C^*\}$

  $U \leftarrow U \setminus \{C^*\}$

  // 更新剩余候选项的键

  对每个 $C \in U$ **执行**

    如果 $w(C, C^*) >$ 键$[C]$ **则**

      键$[C] \leftarrow w(C, C^*)$

返回 $\mathcal{R}$

这一迭代过程确保最终集合 R 通过一棵由每一步选出的最强边构成的树与 C_q 相连。最大值运算序列隐式构建了一个以 C_q 为根的连通性骨架，用以近似最小生成树的增益；如有需要，可在诱导子图 $\mathbf{G}_C[\{C_q\} \cup \mathbf{R}]$ 上运行最小生成树算法，以获得最终集合的精确基于最小生成树的一致性，而无需在在线搜索过程中对每个中间子集都重新计算。

算法描述。 完整的全局连贯搜索过程总结于算法 1 中，该算法同时整合了候选图构建和基于最小生成树的重排序。给定一个查询列 C_q后，算法首先通过相似度搜索检索 Top-B 个最相关的候选项，并构建加权候选图 $\mathbf{G}_C$（第 1–3 行）。在重排序阶段（第 4–16 行），算法通过迭代选择边际增益最高的候选项来逐步构建结果集 R，其中边际增益的定义见公式 25。该增益结合了候选项与 C_q 的相关性，以及其与当前树（包括 C_q 自身）的最强连接，从而模拟以查询为根的最大生成树的 Prim 式扩展。该过程持续进行，直到选出 K 列，最终得到一个同时针对相关性和内部可连接性进行优化的一致性结果集。

近似保证。 我们为贪心重排序算法提供理论依据。我们用 $\Delta(C \mid \mathbf{R})$ 表示式 25 中定义的代理边际增益。我们证明，式 25 中的贪心规则优化了一个易于处理的代理函数，其边际增益可证明是基于最小生成树的一致性项所贡献的真实边际增益的下界。证明见在线完整版 [28]。

表 2：基准数据集统计信息

数据集	表数	列数	查询数	连接对数
USA	2,270	21,702	20	15,584
CAN	9,590	70,982	30	57,124
UK_SG	1,166	11,293	20	8,164
Webtable	2,922	21,787	30	21,029

引理 1 (最小生成树增益下界)。 对于任意 $S \ni C_q$ 和 $v \notin S$，令 $MST(S)$ 为 $\mathcal{G}_C[S]$ 上的最大生成树权重。则

$$MST(S \cup \{v\}) - MST(S) \;\geq\; \max_{u \in S} w(v, u).$$

定理 5 (代理边际增益性质)。 令 $S = \{C_q\} \cup \mathcal{R}$，并令 C^* 为通过最大化式 *25* 中的代理边际增益 $\Delta(C \mid \mathcal{R})$ 所选出的候选项。则

$$G(\mathcal{R} \cup \{C^*\}) - G(\mathcal{R}) \;\geq\; \Delta(C^* \mid \mathcal{R}).$$

此外，如果 $u_t \in \arg\max_{u \in S_{t-1}} w(C_{(t)}u)$，那么 $\{(C_{(t)}u_t)\}_{t=1}^{K}$ 会在 $\{C_q\} \cup \mathcal{R}$ 上形成一棵生成树，并且

$$Coherence(\mathcal{R}) = MST(\{C_q\} \cup \mathcal{R}) \;\geq\; \sum_{t=1}^{K} w(C^{(t)}, u_t).$$

此分析表明，式 25 是 $G(\cdot)$ 中真实边际增益的保守估计：在每次迭代中，目标函数至少增加计算所得的替代增益。此外，所维护的连接在最终顶点集上形成一棵生成树，其总权重下界为基于最小生成树的一致性。

复杂度分析。 每次查询的时间复杂度主要由两个部分决定：候选项检索和图构建成本 $O(Nd + N\log B + B^2 d)$，以及贪心重排序成本 $O(BK)$。具体而言，阶段 1 通过对N列进行相似度搜索来检索 B 个候选项（$O(Nd + N\log B)$），并构建一个包含 $O(B^2)$ 条边的候选图，其权重计算耗时 $O(B^2 d)$。阶段 2 为每个候选项维护其与当前树的最佳连接（键）；在 K 次迭代中，每次都以 $O(B)$ 的时间选取边际增益最大的候选项，并以 $O(B)$ 的时间更新其余候选项的键，总成本为 $O(BK)$。整体复杂度为 $O(Nd + N\log B + B^2 d + BK)$，空间复杂度为 $O(B^2)$，用于存储候选图的成对权重；对于典型参数而言，这种开销是高效的。

6 实验评估

6.1 实验设置

数据集。我们在四个真实世界的数据湖基准数据集上评估可连接表发现。其中三个基准数据集来自美国、加拿大以及英国/新加坡的政府开放数据门户 [1]；为方便起见，我们分别称它们为 USA、CAN 和 UK_SG。我们还使用 Webtable [2]，这是一个源自 WDC Web 表语料库的基准数据集，并且只保留英文关系型网页表格。对于每个表，我们按照基准元数据中的规定提取关键列。所有基准数据集都提供表标题、列名和单元格上下文等元数据，并已用于以往的研究 [5, 15, 16, 37, 46, 49, 50]。表 2 报告了它们的统计信息。

基线方法。我们比较以下方法。(1) JOSIE [49]：利用倒排索引和发布列表进行基于令牌的集合相似性搜索，并采用基于代价模型的贪心合并算法。(2) LSH Ensemble [50]：使用局部敏感哈希进行基于 MinHash 的包含搜索，并采用动态分区。(3) DeepJoin [16]：用于列嵌入的预训练 Transformer 模型。(4) Snoopy [19]：一种先进的列嵌入模型，利用代理列改进语义连接发现。(5) Omnimatch [25]：一种多关系图神经网络，结合多种相似性信号进行表匹配和可连接性发现。

指标。 遵循先前的工作 [13], ，我们使用 Precision@K 和 Recall@K 评估检索质量。令 Q 表示查询列集合，$\mathrm{R}_K(q)$表示针对查询 q检索到的 Top-K 列集合，$\mathrm{R}^*(q)$表示 q的真实可连接列集合。我们计算：

$$\text{Precision@K} = \frac{1}{|Q|}\sum_{q\in Q}\frac{|\mathcal{R}_K(q)\cap\mathcal{R}^*(q)|}{K}, \tag{26}$$

$$\text{Recall@K} = \frac{1}{|Q|}\sum_{q\in Q}\frac{|\mathcal{R}_K(q)\cap\mathcal{R}^*(q)|}{|\mathcal{R}^*(q)|}. \tag{27}$$

我们报告 $K\in${5、15、25}的结果，以评估不同截断位置下的性能。

实现细节。 我们使用 Python 3.9 和 PyTorch 2.0.1 实现 HyperJoin。对于初始列编码，我们使用带有可学习令牌嵌入的文本编码器（词表大小为 ~1500）对表名和列名进行编码，随后进行均值池化。对于列内容特征，我们使用预训练的 fastText 嵌入对单元格值进行编码，然后通过两层 MLP、ReLU 激活和 Dropout 对其进行聚合与投影。模型架构由 2 个超图卷积层和 2 个 MLP-Mixer 层组成，嵌入维度为 512。我们采用三级位置编码，维度为 16，用于表级和列级结构信息；该编码通过在可连接图上进行拉普拉斯特征向量分解计算得到。训练时，我们采用 Adam 优化器，学习率为 4×10^{-4}，批次大小为 64，训练 30 个训练轮次。我们使用边界值为 1.0 的三元组损失进行对比学习，并启用困难负样本挖掘。Dropout 比率设置为 0.05，以防止过拟合。对于自监督学习，我们通过 API 使用温度 0.7 和 top-p 采样 0.9 的 DeepSeek-V3 生成增强列。对于全局一致性重排序，我们将候选规模设置为 $B=50$，将连贯性权重设置为 $\lambda=1.0$。所有实验均在一台工作站上进行，该工作站配备 Intel Xeon E E-2488 CPU（8 个核心、16 个线程）、一块具有 15GB 内存的 NVIDIA A2 GPU，以及 128 GB 内存。

6.2 总体评估

实验 1：总体性能比较。表 3 和图 4 展示了所有方法在四个基准数据集上的全面比较。总体而言，HyperJoin 在 *Webtable*、美国、加拿大 和英国_新加坡上均取得了最佳 Precision@K 和 Recall@K。如表中所示，LSH 在传统基线方法中表现最佳，而基于预训练语言模型的基线方法整体上优于传统方法。在基于预训练语言模型的模型中，Snoopy 的结果最佳。值得注意的是，HyperJoin 进一步超越了 Snoopy，取得了最高的总体准确率。具体而言，在 $K=15$，与 Snoopy 相比，HyperJoin 的 P@15 平均提高了 21.4 个百分点，R@15 平均提高了 17.2 个百分点。这些提升在政府来源基准数据集上最为显著，HyperJoin 的 P@15 比 Snoopy 高出最多 28.7 个百分点（在英国_新加坡上）。这源于政府数据的固有特性，其特点是具有强烈的结构和语义规律，例如一致的列命名、丰富的元数据以及清晰的领域层级。HyperJoin 的大语言模型增强的超图和分层消息传递专为捕捉这些精确的上下文信号而设计。相比之下，Webtable 主要由数值内容构成，此类语义信号本就较少，但 HyperJoin 仍通过有效利用可用的结构和元数据上下文取得了显著提升。这些稳定的差距表明，HyperJoin 不仅准确率更高，而且在异构数据集上的泛化能力也更强。与传统的集合重叠方法（JOSIE、LSH Ensemble）相比，HyperJoin 的提升幅度更大，在各数据集上的平均 R@25 增幅超过 40%。

我们进一步分析了当发现的表数量有限（较小的 K）或相对较多（较大的 K）时，HyperJoin 的优势是否更加明显。在 $K=5$ 时，HyperJoin 的平均 P@5 达到 87.7%，P@5 和 R@5 分别比 Snoopy 高 10.5 和 3.0 个百分点，这表明在发现范围有限时，其早期阶段的排序质量更高。随着 K 增加，这一优势变得更加显著，尤其是在召回率方面。在 $K=25$ 时，HyperJoin 的平均 P@25 和 R@25 分别比 Snoopy 高 15.1 和 19.5 个百分点，表明它能够发现显著更多的可连接表，同时保持较高的精确率。这一趋势在 UK_SG 上尤为明显：召回率差距从 +6.4 个百分点（$K=5$ 时）扩大到 +34.8 个百分点（$K=25$ 时）。总的来说，结果表明 HyperJoin 即使在较小的 K 下也具有竞争力，而在较大的 K 下优势更为显著。这说明，随着更广泛的发现使得仅依赖相关性已不足够，联合建模相关性与一致性会越来越有益。

实验 2：消融研究。我们通过消融研究分析 HyperJoin 中各组件的作用。表4报告了四个数据集上的 P@15 和 R@15。首先，我们通过比较完整模型与去除一致性重排序（CR）的模型，考察 CR 的效果。在所有数据集上，CR 都能稳定提升性能，P@15 平均提高 +4.1。其在加拿大上的收益最大（P@15：75.56→81.78，+6 22 15 51 73→55 89 +4 16 .；R@：.., .），而在美国上的提升也很明显（P@15 提高 +4.66，R@15 提高 +3.22），这表明集合级一致性能够抑制那些局部看来合理、但整体上不一致的候选项。随后，我们在与去除 CR 相同的设置下移除关键模块，以了解性能提升的来源。移除 HIN 会导致显著且稳定的性能下降。在 Webtable 上降幅最大（P@15：71.33→51.11，下降 20.22；R@15：67.09→46.88，下降 20.21），这凸显了细粒度交互建模对于过滤异构网页表格中大量表面相似干扰项的重要性。移除超图（HG）模块的影响则因数据集而异：在英国和新加坡上的降幅尤为显著（P@15：86.67→59.67；R@15：75.23→53.19），在美国上的影响也不容忽视（P@15：下降 18.34；R@15：下降 12.86），这表明当全局规律性较强时，结构信息至关重要。总体而言，消融实验表明，HIN 是各数据集上的核心贡献因素；HG 对结构更规整的语料库尤为重要；而 CR 进一步与二者互补，提升了集合级一致性。

表 3：四个基准数据集上的性能比较。

方法	Webtable						USA						CAN						UK_SG					
	精确率			召回率			精确率			召回率			精确率			召回率			精确率			召回率		
	@5	@15	@25	@5	@15	@25	@5	@15	@25	@5	@15	@25	@5	@15	@25	@5	@15	@25	@5	@15	@25	@5	@15	@25
JOSIE	35.3	25.1	15.9	10.9	23.2	24.2	76.0	74.7	68.6	17.7	52.2	79.9	56.7	52.4	44.1	13.1	35.8	49.5	27.0	28.7	23.2	8.8	27.0	34.9
LSH	58.7	51.4	46.7	17.4	40.3	52.5	81.0	72.1	72.0	18.5	41.8	50.5	68.7	65.7	60.7	13.6	35.7	46.4	26.0	26.3	26.4	6.0	14.4	22.2
DeepJoin	69.3	55.8	45.9	22.1	51.0	68.1	87.0	68.7	48.8	20.2	48.1	57.1	64.0	53.3	42.4	14.7	36.8	47.6	66.0	55.0	42.2	19.6	48.5	60.2
Snoopy	71.3	55.3	46.3	22.8	50.0	68.3	86.0	74.7	63.8	19.9	51.9	74.0	79.3	65.1	58.7	18.0	45.0	66.7	72.0	60.3	44.0	20.8	53.5	64.0
Omnimatch	70.0	59.1	45.3	21.8	55.6	68.7	63.0	74.7	70.4	14.6	52.2	82.0	60.7	62.4	56.5	13.3	41.0	61.4	56.0	48.3	39.2	16.5	41.8	55.3
HyperJoin	81.3	74.7	53.6	25.9	69.8	80.7	92.0	95.3	83.6	21.3	66.4	97.1	83.3	81.8	65.9	18.9	55.9	74.2	94.0	89.0	70.2	27.2	77.1	98.8

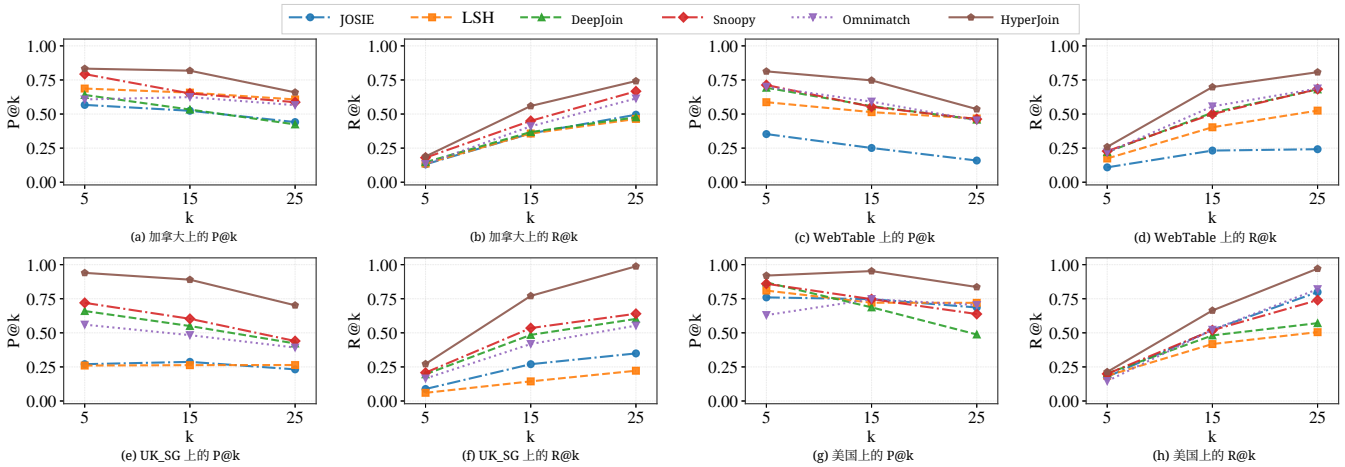


图 4：不同数据集和 k值下的性能比较。

表4：使用 P@15 和 R@15（百分比）进行的消融研究。我们通过比较有无 CR（一致性重排序）的完整模型，量化 CR 的效果；并通过移除 HIN 或 HG（超图）评估核心模块的消融效果。

变体	Webtable		USA		CAN		UK_SG	
	P@15	R@15	P@15	R@15	P@15	R@15	P@15	R@15
Full	**74.67**	**69.75**	**95.33**	**66.42**	**81.78**	**55.89**	**89.00**	**77.08**
不含 CR	71.33	67.09	90.67	63.20	75.56	51.73	86.67	75.23
不含 HIN	51.11	46.88	75.67	52.83	60.44	41.54	69.67	58.73
不含 HG	55.11	49.75	72.33	50.34	64.89	45.01	59.67	53.19

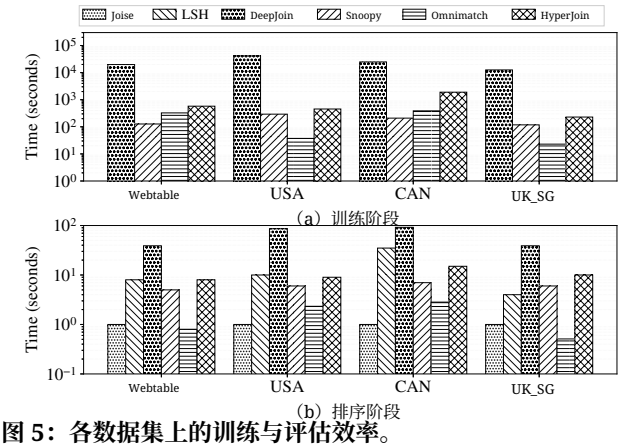
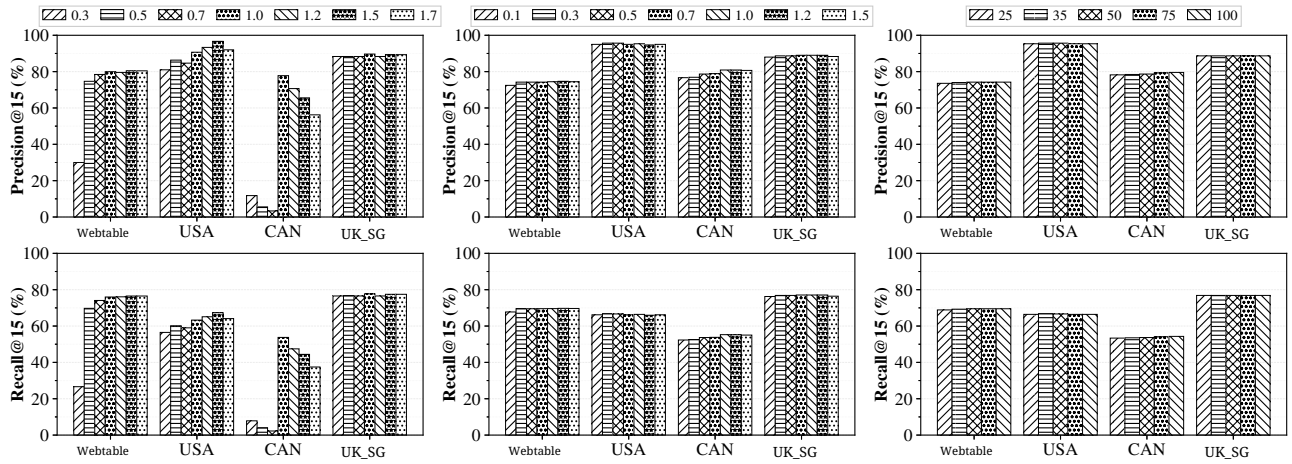


图 5：各数据集上的训练与评估效率。

实验 3：效率评估。图 5 比较了 HyperJoin 与基线方法在训练和评估阶段的效率。在训练阶段，HyperJoin 在四个数据集上的训练时间为 3.8–31.6 分钟（例如，在加拿大上为 31.6 分钟），显著快于需要数小时的 DeepJoin（例如，在加拿大上为 6.9 小时，在美国上为 11.5 小时）。这是意料之中的，因为 DeepJoin 会进行模型微调（基于 DistilBERT），其训练成本远高于轻量级训练目标。此外，由于需要执行大量成对列操作来构建列相似度图，Omnimatch 产生了相当大的离线开销；在我们的设置中，仅此预处理就需要需要数小时（例如，美国数据集上为 30.94 小时，加拿大数据集上为 43.37 小时）。在排序阶段，HyperJoin 在 8–15 秒内即可回答查询；在较大的数据集上，它始终比 DeepJoin 快，同时与局部敏感哈希和 Snoopy 的耗时处于同一数量级。请注意，报告的 DeepJoin 时间是端到端测量的，包含每次运行时重建目标端索引的耗时，而不仅仅是在线 top-k 检索步骤。



(a) 三元组损失边界（m） **(b) 连贯性权重（λ）** **(c) 候选规模（B）**

图6：数据集上的超参数敏感性分析。(a) 三元组损失边界 m，(b) Global Cohenrent Seaech 连贯性权重 λ，以及 (c) Global Cohenrent Seaech 候选规模 B。

6.3 鲁棒性评估

实验4：三元组损失边界 m。 我们在四个数据集上评估了三元组损失边界 $m \in \{0.3, 0.5, 0.7, 1.0, 1.2, 1.5, 1.7\}$。如图6(a)所示，三元组损失边界至关重要。边界过小会导致性能严重下降，甚至模型崩溃，在加拿大和 Webtable 上尤为明显（例如，在 $m = 0.7$ 时，加拿大的 P@15 降至 3.33%、R@15 降至 2.25%；在 m =0.3 时，Webtable 的 P@15 降至 30.00%、R@15 降至 26.63%）。当边界为 $m = 1.0$时，性能趋于稳定，在各数据集上都能持续取得可靠结果，并避免灾难性失效。更大的边界可能进一步提升部分语料库上的表现（例如， $m = 1.5$ 在美国数据集上表现最佳，达到 96.67% / 67.36%，并将 Webtable 的表现提升至 80.44% / 76.52%），但可能会稍微损害加拿大数据集上的表现，对英国和新加坡数据集的影响则很小（在 $m = 1.0$ 时达到峰值，为 89.67% / 77.71%）。总体而言，我们在所有实验中均采用 $m = 1.0$ 作为稳健的默认值。

实验5：连贯性权重 λ。 我们在四个数据集上评估 $\lambda \in \{0.1, 0.3, 0.5, 0.7, 1.0, 1.2\ 1.5\ 6,\}$。图(b)表明，各数据集的最优值有所不同，但在 $\lambda \in [0.7, 1.2]$ 附近存在一个稳定的平台区间。在此范围内，UK_SG 和加拿大通常表现最佳；Webtable 的表现则持续提升至 $\lambda = 1.2$（74.67% P@15 / 69.74% R@15），随后变化不大。相比之下，美国偏好较小的 λ（在 $\lambda = 0.3$–0.5 附近表现最佳，达到 95.67% / 66.66%），并且在数值较大时性能略有下降，这表明连贯性会导致过度正则化。加拿大从连贯性中获益显著，并在 $\lambda = 1.0$–1.2 时达到峰值（当 $\lambda = 1.0$ 时，P@15 为 80.89%，R@15 为 55.26%），之后略有下降。总的来说，我们将 $\lambda = 1.0$ 设为稳健的默认值，它在三个数据集上都能取得接近最优的结果，同时在美国数据集上也保持竞争力。

实验6：候选规模 B。 我们为 Global Cohenrent Seaech 设置了不同的候选池大小 $B \in \{25,35,50,75,100\}$。如图6(c)所示，性能相对于 B 基本保持稳定，在不同数据集上仅有轻微波动（P@15 的最大变化为 ≤1.34，R@15 的最大变化为 ≤0.94）。具体而言，结果在中等候选池大小时便迅速趋于饱和：UK_SG 在所有 B 下均保持不变，而 Webtable/USA 在约 $B \approx 35$–50 时趋于平稳。CAN 随着候选池增大呈现轻微的单调提升，但超过 $B = 50$ 后的增益微乎其微。总的来说，我们将 $B = 50$ 设为默认值，以兼顾近乎最优的质量和计算成本。

7 相关工作

现有的可连接表发现方法 [6, 8, 13, 15, 17, 21] 大致可分为传统集合重叠方法和基于深度学习的方法。传统方法（例如 Jaccard相似度 [18], 、包含系数 [11, 45, 48], 和局部敏感哈希 [50],）使用基于集合的度量以及精确或模糊字符串匹配来衡量列重叠。尽管这些方法具有良好的可扩展性，但它们依赖句法匹配，无法捕捉语义关系。基于深度学习的方法通过两种主要范式来解决这一局限。预训练语言模型方法（例如 DeepJoin [16] 和 Snoopy [19], ）采用 Transformer模型（如 BERT ）将列值编码为语义嵌入。图神经网络方法（例如 OmniMatch [25], ）将表存储库建模为多关系图，并使用关系图卷积网络 [34]学习结构感知的列嵌入。据我们所知，HyperJoin 是首个用于可连接表发现的基于超图的框架。它在统一的表示学习阶段同时建模表内和表间结构交互，并进一步提出一种基于最小生成树的全局一致重排序策略，以返回全局一致的连接候选。

8 结论

本文针对数据湖中的可连接表发现问题，提出了基于超图的框架 HyperJoin。HyperJoin 采用两阶段设计，包括离线表示学习和在线排序。在离线阶段，我们提出一种大语言模型增强的超图，通过双类型超边对表内结构和表间可连接性进行建模，并利用 HIN 学习表达能力强的列表示。在在线阶段，我们引入高效的全局一致性重排序模块，以生成全局一致的结果集，兼顾可扩展性与有效性。在四个公开基准数据集上的大量实验表明，HyperJoin 始终优于最先进的方法，整体上取得更高的精确率和召回率，同时返回更加连贯且语义相关的连接候选。

参考文献

[1] [不详.]. 政府开放数据门户（美国、加拿大、英国、新加坡）。 https://www.data. gov/; https://open.canada.ca/; https://data.gov.uk/; https://data.gov.sg/. [2] [不详.]. WebTable。 https://webdatacommons.org/webtables/. [3] Naheed Anjum Arafat、Arijit Khan、Arpit Kumar Rai 和 Bishwamittra Ghosh。2023年。基于邻域的超图核分解。VLDB 基金会会议录 16, 9 (2023), 2061–2074。[4] Adrián Arnaiz-Rodríguez、Ahmed Begga、Francisco Escolano 和 Nuria Oliver。2022年。Diffwire：通过 lov\'asz 界进行归纳式图重连。arXiv 预印本 arXiv:2206.07369 (2022)。[5] Chandra Sekhar Bhagavatula、Thanapon Noraset 和 Doug Downey。2015年。Tabel：网页表格中的实体链接。载于国际语义网会议。施普林格，425–441。[6] Alex Bogatu、Alvaro AA Fernandes、Norman W Paton 和 Nikolaos Konstanti-nou。2020年。数据湖中的数据集发现。载于2020年 IEEE 第36届国际数据工程会议（ICDE）。IEEE，709–720。[7] Chengliang Chai、Lei Cao、Guoliang Li、Jian Li、Yuyu Luo 和 Samuel Madden。2020年。人在环路的离群点检测。载于2020年 ACM SIGMOD国际数据管理会议论文集。19–33。[8] Tianji Cong、Fatemeh Nargesian 和 HV Jaga dish。2023年。Pylon：数据湖中的语义表联合搜索。arXiv 预印本 arXiv:2301.04901 (2023)。[9] Andreea Deac、Marc Lackenby 和 Petar Veličković。2022年。扩展图传播。载于图学习会议。PMLR，38–1。[10] Dong Deng、Raul Castro Fernandez、Ziawasch Abedjan、Sibo Wang、Michael Stonebraker、Ahmed K Elmagarmid、Ihab F Ilyas、Samuel Madden、Mourad Ouz-zani 和 Nan Tang。2017年。Data Civilizer系统。载于 CIDR。[11] Dong Deng、Albert Kim、Samuel Madden 和 Michael Stonebraker。2017年。SILK-MOTH：一种利用最大匹配约束查找相关集合的高效方法。VLDB 基金会会议录 10, 10 (2017)。[12] Yuhao Deng、Chengliang Chai、Lei Cao、Nan Tang、Jiayi Wang、Ju Fan、Ye Yuan 和 Guoren Wang。2024年。Misdetect：利用早期损失进行迭代式错误标签检测。计算机协会（ACM）。[13] Yuhao Deng、Chengliang Chai、Lei Cao、Qin Yuan、Siyuan Chen、Yanrui Yu、Zhaoze Sun、Junyi Wang、Jiajun Li、Ziqi Cao 等。2024年。Lakebench：用于发现数据湖中可连接表和可合并表的基准数据集。VLDB 基金会会议录 17, 8 (2024), 1925–1938。[14] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。BERT：面向语言理解的深度双向变换器预训练。载于2019年北美计算语言学协会会议：人类语言技术论文集，第1卷（长篇和短篇论文）。4171–4186。[15] Yuyang Dong、Kunihiro Takeoka、Chuan Xiao 和 Masafumi Oyamada。2021年。数据湖中的高效可连接表发现：一种基于高维相似度的方法。载于2021年 IEEE 第37届国际数据工程会议（ICDE）。IEEE，456–467。[16] Yuyang Dong、Chuan Xiao、Takuma Nozawa、Masafumi Enomoto 和 Masafumi Oyamada。2023年。DeepJoin：利用预训练语言模型发现可连接表。VLDB 基金会会议录 16, 10 (2023), 2458–2470。[17] Grace Fan、Jin Wang、Yuliang Li、Dan Zhang 和 Renée J Miller。2023年。通过上下文化列级表示学习，从数据湖中发现感知语义的数据集。VLDB 基金会会议录 16, 7 (2023), 1726–1739。[18] Raul Castro Fernandez、Jisoo Min、Demitri Nava 和 Samuel Madden。2019年。Lazo：一种基于基数的方法，用于联合估计 Jaccar d 相似度和包含度。载于2019年 IEEE 第35届国际数据工程会议（ICDE）。IEEE，1190–1201。[19] Yuxiang Guo、Yuren Mao、Zhonghao Hu、Lu Chen 和 Yunjun Gao。2025年。Snoopy：通过代理列实现高效且有效的语义连接发现。IEEE模式分析与机器智能汇刊（2025）。[20] Xiaoxin He、Bryan Hooi、Thomas Laurent、Adam Perold、Yann LeCun 和 Xavier Bresson。2023年。将 ViT/MLP-Mixer 推广到图。载于国际机器学习会议。PMLR，12724–12745。[21] Xuming Hu、Shen Wang、Xiao Qin、Chuan Lei、Zhengyuan Shen、Christos Faloutsos、Asterios Katsifodimos、George Karypis、Lijie Wen 和 Philip S Yu。2023年。利用表格表示学习实现自动表联合搜索。载于计算语言学协会论文集：ACL 2023。3786–3800。[22] Herve Jegou、Matthijs Douze 和 Cordelia Schmid。2010年。用于最近邻搜索的乘积量化。IEEE模式分析与机器智能汇刊 33, 1 (2010), 117–128。[23] Aamod Khatiwada、Roee Shraga、Wolfgang Gatterbauer 和 Renée J Miller。2022年。数据湖表集成。VLDB 基金会会议录 16, 4 (2022), 932–945。[24] Omar Khattab 和 Matei Zaharia。2020年。ColBERT：通过 BERT 上下文化后期交互实现高效且有效的段落搜索。载于第43届国际ACM SIGIR信息检索研究与发展会议论文集。39–48。[25] Christos Koutras、Jiani Zhang、Xiao Qin、Chuan Lei、Vasileios Ioannidis、Christos Faloutsos、George Karypis 和 Asterios Katsifodimos。2025年。OmniMatch：数据产品中的可连接性发现。VLDB 基金会会议录 18, 11（2025年），4588–4601。[26] Eugenie Y Lai、Yeye He 和 Surajit Chaudhuri。2025年。Auto-Prep：面向自助式商业智能的数据准备步骤整体预测。arXiv 预印本 arXiv:2504.11627（2025年）。[27] Geon Lee、Jihoon Ko 和 Kijung Shin。2020年。超图模体：概念、算法与发现。VLDB 13，12（2020年），2256–2269。[28] Shiyuan Liu、Jianwei Wang、Xuemin Lin、Lu Qin、Wenjie Zhang 和 Ying Zhang。2026年。HyperJoin：完整版。 https://github.com/T-Lab/HyperJoin/tree/main/ full version。[29] ¯Jose Lugo-Martinez、Daniel Zeiberg、Thomas Gaudelet、Noël Malod-Dognin、Natasa Przulj 和 Predrag Radivojac。2021年。利用超图小图核对生物网络进行分类。Bioinformatics 37，7（2021年），1000–1007。[30] Yu A Malkov 和 Dmitry A Yashunin。2018年。使用分层可导航小世界图实现高效、稳健的近似最近邻搜索。IEEE模式分析与机器智能汇刊 42，4（2018年），824–836。[31] Fatemeh Nargesian、Erkang Zhu、Renée J Miller、Ken Q Pu 和 Patricia C Arocena。2019年。数据湖管理：挑战与机遇。VLDB 基金会会议录 12，12（2019年），1986–1989。[32] Alec Radford。2018年。通过生成式预训练提升语言理解能力。预印本（2018年）。[33] Victor Sanh、Lysandre Debut、Julien Chaumond 和 Thomas Wolf。2019年。DistilBERT：BERT 的精简版本，更小、更快、更便宜、更轻量。arXiv 预印本 arXiv:1910.01108（2019年）。[34] Michael Schlichtkrull、Thomas N Kipf、Peter Bloem、Rianne Van Den Berg、Ivan Titov 和 Max Welling。2018年。使用图卷积网络对关系数据建模。载于：欧洲语义网会议。施普林格，593–607。[35] Kaitao Song、Xu Tan、Tao Qin、Jianfeng Lu 和 Tie-Yan Liu。2020年。MPNet：面向语言理解的掩码与置换预训练。神经信息处理系统进展 33（2020年），16857–16867。[36] Ilya O Tolstikhin、Neil Houlsby、Alexander Kolesnikov、Lucas Beyer、Xiaohua Zhai、Thomas Unterthiner、Jessica Yung、Andreas Steiner、Daniel Keysers、Jakob Uszkoreit 等。2021年。MLP-Mixer：一种用于视觉的全 MLP 架构。神经信息处理系统进展 34（2021年），24261–24272。[37] Fei Wang、Kexuan Sun、Muhao Chen、Jay Pujara 和 Pedro Szekely。2021年。利用多粒度图表示学习检索复杂表格。载于：第44届国际ACM SIGIR信息检索研究与发展会议论文集。1472–1482。[38] Jianwei Wang、Kai Wang、Xuemin Lin、Wenjie Zhang 和 Ying Zhang。2024年。利用预训练图变换器进行高效无监督社区搜索。arXiv 预印本 arXiv:2403.18869（2024年）。[39] Jianwei Wang、Kai Wang、Xuemin Lin、Wenjie Zhang 和 Ying Zhang。2024年。十亿规模的神经属性社区搜索。ACM数据管理会议录 1，4（2024年），1–25。[40] Jianwei Wang、Kai Wang、Ying Zhang、Wenjie Zhang、Xiwei Xu 和 Xuemin Lin。2025年。基于高阶消息传递的大语言模型增强混合类型数据插补。arXiv 预印本 arXiv:2501.02191（2025年）。[41] Jianwei Wang、Ying Zhang、Kai Wang、Xuemin Lin 和 Wenjie Zhang。2024年。利用不确定性驱动网络进行缺失数据插补。ACM数据管理会议录 2，3（2024年），1–25。[42] Yiqi Wang、Long Yuan、Wenjie Zhang、Zi Chen、Xuemin Lin 和 Qing Liu。2024年。化繁为简：大型道路网络上的高效 Top-K 近邻搜索。VLDB 基金会会议录 17，13（2024年），4683–4695。[43] Boris Weisfeiler 和 Andrei Leman。1968年。将图约化为规范形式及其中出现的代数。nti，Series 2，9（1968年），12–16。[44] Hanrui Wu、Yuguang Yan 和 Michael Kwok-Po Ng。2022年。面向顶点和超边的超图协作网络。IEEE模式分析与机器智能汇刊 45，3（2022年），3245–3258。[45] Mohamed Yakout、Kris Ganjam、Kaushik Chakrabarti 和 Surajit Chaudhuri。2012年。InfoGather：通过与网页表格进行整体匹配实现实体增强和属性发现。载于：2012年ACM SIGMOD国际数据管理会议论文集。97–108。[46] Pengcheng Yin、Graham Neubig、Wen-tau Yih 和 Sebastian Riedel。2020年。TaBERT：文本与表格数据联合理解的预训练。载于：ACL。计算语言学协会，8413–8426。[47] Haochen Zhang、Yuyang Dong、Chuan Xiao 和 Masafumi Oyamada。2023年。大语言模型作为数据预处理器。arXiv:2308.16361（2023年）。[48] Meihui Zhang、Marios Hadjieleftheriou、Beng Chin Ooi、Cecilia M Procopiuc 和 Divesh Srivastava。2010年。论多列外键发现。VLDB 基金会会议录 3，1-2（2010年），805–814。[49] Erkang Zhu、Dong Deng、Fatemeh Nargesian 和 Renée J Miller。2019年。JOSIE：用于在数据湖中查找可连接表的重叠集合相似性搜索。载于：2019年国际数据管理会议论文集。847–864。[50] Erkang Zhu、Fatemeh Nargesian、Ken Q Pu 和 Renée J Miller。2016年。LSH Ensemble：互联网规模领域搜索。VLDB 基金会会议录 9，12 2016( )。

9 附录

9.1 HIN 的整体前向过程。

算法 2 概述了 HIN 的完整前向传播。第 2–7 行通过离线预计算拉普拉斯特征向量，并添加表级和列级编码来注入位置编码。第 9–14 行执行节点级图神经网络变换，并通过领域特定变换聚合到超边级。第 16–24 行通过结构感知注意力和通道混合执行全局超边混合。第 26–29 行通过残差连接将超边嵌入传播回各列。最后，第 31 行返回最终嵌入。

9.2 理论结果的证明

本节汇集了正文中略去的证明。为便于阅读，我们按照相应定理所在的章节对证明进行归类。

9.2.1 定理*1*的证明。

设定。令 $Y_{ij} \in \{0,1\}$ 表示两列（v_i, v_j）是否可连接。令 $\mathbf{x}_i$ 表示列 v_i 的初始特征。对于每一列 v_i，定义由构建的超图所诱导的两个上下文变量：(i) 其表内上下文 $\mathrm{N}^{\mathrm{intra}}(v_i)$（与 v_i 位于同一张表中的列集合），以及 (ii) 其表间上下文 $\mathrm{N}^{\mathrm{inter}}(v_i)$（通过可连接性与 v_i 相连的列集合，例如位于同一个连接组件中）。我们将用于判定可连接性的可用信息记为一个特征集：

$$\begin{aligned} \mathcal{F}_{\mathrm{single}}(i,j) &= \{\mathbf{x}_i, \mathbf{x}_j\}, \\ \mathcal{F}_{\mathrm{multi}}(i,j) &= \Big\{\mathbf{x}_i, \mathbf{x}_j,\ \mathcal{N}^{\mathrm{intra}}(v_i), \mathcal{N}^{\mathrm{intra}}(v_j), \\ &\qquad \mathcal{N}^{\mathrm{inter}}(v_i), \mathcal{N}^{\mathrm{inter}}(v_j)\Big\}. \end{aligned} \tag{28}$$

令 $\Psi^*(\mathrm{F})$ 表示在特征集 F 下可达到的最小期望发现风险：

$$\Psi^*(\mathcal{F}) \;=\; \inf_{g}\ \mathbb{E}\big[\ell(g(\mathcal{F}), Y_{ij})\big],$$

其中，g 是任意可测评分函数，$\ell(\cdot)$ 是有界损失（例如 logistic 损失或 0-1 损失）。

定理1。对于上述两个特征集 $\mathcal{F}_{multi}$ 和 $\mathcal{F}^{single}$，贝叶斯最优风险满足

$$\Psi^*(\mathcal{F}_{multi}) \;\le\; \Psi^*(\mathcal{F}_{single}).$$

此外，如果新增上下文具有信息量，即 $\mathbb{P}(Y_{ij}=1 \mid \mathcal{F}_{multi}) \neq \mathbb{P}(Y_{ij}=1 \mid \mathcal{F}^{single})$ 以非零概率成立，那么该不等式是严格的。

证明。任何预测器 $g_{\mathrm{single}}(\mathrm{F}^{\mathrm{single}})$ 在 $\mathrm{F}_{\mathrm{multi}}$ 下也可行，只需定义 $g_{\mathrm{multi}}(\mathrm{F}_{\mathrm{multi}}) = g_{\mathrm{single}}(\mathrm{F}^{\mathrm{single}})$，即忽略额外的上下文变量。因此，

$$\begin{aligned} \Psi^*(\mathcal{F}_{\mathrm{multi}}) &= \inf_{g_{\mathrm{multi}}} \mathbb{E}\big[\ell\big(g_{\mathrm{multi}}(\mathcal{F}_{\mathrm{multi}}), Y_{ij}\big)\big] \\ &\le \inf_{g_{\mathrm{single}}} \mathbb{E}\big[\ell\big(g_{\mathrm{single}}(\mathcal{F}_{\mathrm{single}}), Y_{ij}\big)\big] = \Psi^*(\mathcal{F}_{\mathrm{single}}). \end{aligned} \tag{29}$$

如果在给定新增上下文的条件下，Y_{ij} 的条件分布发生变化，那么在 $\mathrm{F}_{\mathrm{multi}}$ 下的贝叶斯决策规则与在 $\mathrm{F}^{\mathrm{single}}$ 下的贝叶斯决策规则会在一个测度非零的集合上不同，从而产生严格更小的贝叶斯风险。□

算法 2：分层交互网络（HIN） 输入：超图 $\mathrm{H} = (\mathrm{V}, \mathrm{E}, \mathrm{X}^v, \mathrm{X}^e, \Pi)$、表 ID、可连接对 $\mathrm{P}^{\mathrm{join}}$ 输出：最终列嵌入 $\{\mathrm{h}_v^{\mathrm{final}}\}_{v\in\mathcal{V}}$ // 阶段 1：位置编码 1 预先计算拉普拉斯特征向量（离线）： 2 A $\leftarrow$ BuildAdjacencyMatrix($\mathrm{P}^{\mathrm{join}}$) 3 $\mathrm{L}_{\mathrm{norm}} = \mathrm{I}_N - \mathrm{D}^{-1/2}\mathrm{A}\mathrm{D}^{-1/2}$ 4 $\mathrm{V}^{\mathrm{pe}} \leftarrow$ EigenVectors($\mathrm{L}_{\mathrm{norm}}, k = 16$) 5 注入表级和列级位置编码： 6 对每个具有表 ID t 和索引 i 的列/节点 v 执行 7 $\mathrm{h}_v^{(0)} \leftarrow \mathrm{x}_v^v + \alpha \cdot \mathrm{E}_{\mathrm{tbl}}[t] + \beta \cdot \mathrm{MLP}^{\mathrm{pe}}(\mathrm{V}_{\mathrm{pe}}[i,:])$ // 阶段 2：局部超边聚合 8 $\mathrm{H} \leftarrow [\mathrm{h}_{v_1}^{(0)}, \ldots, \mathrm{h}_{v_N}^{(0)}]^T$ 9 对 $\ell = 1$ 到 $L = 2$ 执行 10 $\mathrm{H} \leftarrow \mathrm{LayerNorm}(\mathrm{ReLU}(\mathrm{H}\mathrm{W}^{(\ell)} + \mathrm{b}^{(\ell)}))$ 11 使用领域特定变换聚合到超边级： 12 $\mathrm{d}_e \leftarrow \Pi^T \mathbf{1}_N$ // 超边度 13 $\mathrm{D}_e \leftarrow \mathrm{Diag}(\mathrm{d}_e)$ 14 $\mathrm{X}^e \leftarrow \mathrm{D}_e^{-1}\Pi^T\mathrm{H}$ 15 $\mathrm{Z}^{(0)} \leftarrow \mathrm{X}^e$ // 混合器输入令牌 16 将 $\mathrm{Z}^{(0)}$ 划分为 $\mathrm{Z}^{(0)}$ inter 和 $\mathrm{Z}^{(0)}$ intra 17 $\tilde{\mathrm{Z}}_{\mathrm{inter}} \leftarrow \mathrm{Z}^{(0)}$ inter $\mathrm{W}_{\mathrm{inter}}$ 18 $\tilde{\mathrm{Z}}_{\mathrm{intra}} \leftarrow \mathrm{Z}^{(0)}$ intra $\mathrm{W}_{\mathrm{intra}}$ 19 $\tilde{\mathrm{Z}} \leftarrow [\tilde{\mathrm{Z}}_{\mathrm{inter}}; \tilde{\mathrm{Z}}_{\mathrm{intra}}]$ // 阶段 3：全局超边混合 20 $\mathbf{A}^{\mathrm{hyperedge}} \leftarrow$ 行归一化($\Pi \Pi^T - \mathrm{diag}(\Pi^T\Pi)$) 21 for $\ell = 1$ to $L_{mixer} = 2$ do // 带结构偏置的超边交互 22 Q, K, V $\leftarrow$ 层归一化(Z)$[\mathrm{W}_Q, \mathrm{W}_K, \mathrm{W}_V]$ 23 $\mathrm{Z}_{\mathrm{HI}} \leftarrow \tilde{\mathrm{Z}}$ + 多头$\left(\mathrm{softmax}\left(\frac{\mathrm{QK}^T}{\sqrt{d_k}} + \lambda \mathrm{A}_{\mathrm{hyperedge}}\right)\mathrm{V}\right)\mathrm{W}_O$ // 超边精炼 24 $\tilde{\mathrm{Z}} \leftarrow \mathrm{Z}_{\mathrm{HI}} + \mathrm{W}_2\mathrm{GELU}(\mathrm{W}_1$层归一化$(\mathrm{Z}_{\mathrm{HI}})^T)^T$ // 最终超边嵌入 26 令 z_{e_j} 表示 $\mathrm{Z}_{\mathrm{out}}$ 的第 j 行。 // 将超边传播回各列 25 $\mathrm{Z}_{\mathrm{out}} \leftarrow \tilde{\mathrm{Z}}$ 27 **对于** $i = 1$ 至 N **执行生硬的** 28 $\mathcal{N}(v_i) \leftarrow \{j \mid \Pi_{ij} = 1\}$ 29 $\mathbf{h}_{v_i}^{\mathrm{hyperedge}} \leftarrow \frac{1}{|\mathcal{N}(v_i)|}\sum_{j\in\mathcal{N}(v_i)} \mathbf{W}_{\mathrm{h2c}}\mathbf{z}_{e_j}$ 30 $\mathbf{h}_{v_i}^{\mathrm{final}} \leftarrow \mathrm{L2Norm}(\mathrm{LayerNorm}(\mathbf{h}_{v_i} + \mathbf{h}_{v_i}^{\mathrm{hyperedge}}))$ 31 **return** $\{\mathbf{h}_v^{\mathrm{final}}\}_{v\in\mathcal{V}}$

9.2.2 定理 *2* 的证明。

定理 2（通过结构偏置的超边交互）。考虑标准的超图上注意力模型，其实现。*DeepJoin* 和 *Snoopy* 可视为通过限制所

使用的上下文为 $\mathcal{F}^{single}$（即按列编码，不使用 $\mathcal{N}^{intra}$ 或 $\mathcal{N}^{inter}$）。

证明。通过将 E 设为 N 单节点超边 $\mathrm{E} = \{\{v_1\} \ldots, \{v_N\}\}$（或等价地禁用超边消息传递），即可实现这一点。如果每条超边都是单节点超边，那么每列恰好属于一条大小为 1 的超边。对单节点超边进行任何超边内聚合（均值/求和/注意力）都会返回列本身，因此局部超边聚合不会改变表示。此外，由于超边之间没有共享节点，超边邻接矩阵会变为对角矩阵，全局超边混合则退化为逐超边（因而逐列）的变换，不会发生跨列信息交换。因此，得到的嵌入形式为 $\mathbf{h}_i = \phi_\theta(\mathbf{x}_i)$。DeepJoin 将 ϕ_θ 实例化为一个通过自监督对比学习训练的基于预训练语言模型的列编码器，而 Snoopy 则在此类逐列嵌入的基础上进一步应用确定性的基于代理的变换。因此，这两种方法都可视为不利用 $\mathrm{N}^{\text{intra}}/\mathrm{N}^{\text{inter}}$ 的受限实例。□

9.2.3 定理*3*的证明。

定理3（HIN的表达能力严格强于1-WL消息传递）。令 $\mathcal{H}_{base}$ 为在可连接性图上运行、具有原始节点特征 $\mathbf{h}^{(0)}$（即*1-WL*初始化输入）的置换等变消息传递编码器的假设类。令 $\mathcal{H}^{pe}$ 为通过将输入扩充为

$$PE(v) = \alpha \mathbf{E}_{tbl}[t(v)] + \beta \mathbf{PE}_{col}(v), \tag{18}$$

并使用单射节点级更新（如*HIN*中所用）所得到的类。假设存在两个节点 u 和 v，使得 $\mathbf{h}_u^{(0)} = \mathbf{h}_v^{(0)}$ 和 u、v 在原始初始化下无法通过*1-WL*加以区分，但 $PE(u) \neq PE(v)$。则 $\mathcal{H}_{base} \subsetneq \mathcal{H}^{pe}$。

证明。我们首先注意到，$\mathrm{H}_{\text{base}} \subseteq \mathrm{H}^{\text{pe}}$，因为 H_{base} 中的任意编码器都可以通过忽略位置项在 H^{pe} 中实现（例如，设置 $\alpha = \beta = 0$）。

由于 H_{base} 中的编码器具有置换等变性，并且 u，v 在具有相同原始特征的情况下无法通过1-WL加以区分，因此它必须在每一层为 u 和 v 输出相同的表示；因此，H_{base} 中的任何假设都无法区分 u 和 v。

相比之下，在 H^{pe} 下，增强后的输入满足

$$\mathbf{h}_u = \mathbf{h}_u^{(0)} + \mathrm{PE}(u) \neq \mathbf{h}_v^{(0)} + \mathrm{PE}(v) = \mathbf{h}_v.$$

通过单射节点级更新，这一区别得以保留，因此存在 H^{pe} 中的一个假设，其输出能够区分 u 和 v（例如，通过线性读出实现）。因此，$\mathrm{H}_{\text{base}} \subsetneq \mathrm{H}^{\text{pe}}$。□

9.2.4 定理*4*的证明。

定理4（连贯性感知选择的 NP难性）。在基数约束 $|\mathcal{R}| = K$ 下，最大化公式 *21* 中的目标函数 $G(\mathrm{R})$ 是 *NP* 难的，其中 *Coherence*（R）被实例化为在由 $\{C_q\} \cup \mathcal{R}$ 上的诱导子图中、以 C_q 为根的最大生成树的权重。

证明。我们通过从 0–1 背包问题归约来证明 NP难性。给定一个包含物品 {1、……，n} 的背包实例，其中每件物品 i 的整数权重为 w_i，价值为 v_i，容量为 W，目标是选择一个物品子集，使其总重量至多为 W，并使总价值最大化。

我们按如下方式构造选择问题的一个实例。我们创建一个对应于查询列 C_q 的根节点。对于每个项目 i，我们创建一条包含 w_i 个候选节点的路径 $\{u_{i,1}、\ldots\ldots、u_{i,w_i}\}$。我们在

C_q $u_{i\,1}$ 0 j = 和 之间添加 1 . . . $w_i - 2$ u_i u_j 0

一条边，其权重为 。对于每个 、 、，我们添加一条边（、j、、j+1），其权重为 。我们通过将边（u_{i,w_i-1}、u_{i,w_i}）的权重设为 v_i，将该项目的价值赋予路径上的最后一条边。我们进一步添加 W 个虚拟候选节点 $\{d_1、\ldots\ldots、d_W\}$，并将每个 d_t 与 C_q 连接起来，边权重为 0。我们将结果大小设为 $K = W$，并将所有查询相关性权重设为零，因此目标函数完全由连贯性项决定。

我们声称，在 $\{C_q \cup \mathrm{R}\}$ 上的诱导子图中，一致性价值等于由 R 编码的背包物品的总价值。由于一致性是在诱导子图上计算的，任何被选中的节点都必须仅通过 $\mathrm{R} \cup \{C_q\}$ 中的节点与 C_q 连通。如果 R 包含 u_{i,w_i}，那么要使 u_{i,w_i} 在诱导子图中与 C_q 连通，R 必须包含整个前缀 $\{u_{i,1}, \ldots\ldots, u_{i,w_i-1}\}$。因此，要获得最后一条边上的正权重 v_i，就必然恰好需要消耗 w_i 个被选中的节点。选择路径的任何真前缀都不会产生正贡献，因为前缀上的所有边权重均为 0。

给定任意可行背包解 S with $\sum_{i \in S} w_i \leq W$ 我们通过纳入 S 中各物品路径上的所有节点，并用虚拟节点填满剩余预算，构造选择集 R，使得 $|\mathrm{R}| = W$。在诱导子图中，以 C_q 为根的最大生成树包含每个所选物品路径的最后一条边，贡献恰好为 $\sum_{i \in S} v_i$；其他所有必需的边权重均为 0，因此一致性等于背包价值。反过来，对于任意大小为 W 的选择集 R，我们提取相应的物品子集 S，其中包含满足 $u_{i,w_i} \in \mathrm{R}$ 的所有索引 i。根据诱导子图上的连通性要求，这意味着路径上的所有 w_i 节点都被选中，因此 $\sum_{i \in S} w_i \leq W$。此时，R 的一致性恰好为 $\sum_{i \in S} v_i$，因为正权重只出现在所选物品路径的最后一条边上。

因此，若能最优地求解选择问题，就能最优地求解 0–1 背包问题，而后者是 NP 难问题。因此，考虑一致性的 top-K 选择问题是 NP 难的。□

9.2.5 定理 *5* 的证明。

引理 1 (MST 增益下界)。对于任意 $S \ni C_q$ 和 $v \notin S$，令 $MST(S)$ 为 $\mathcal{G}_C[S]$ 上的最大生成树权重。则

$$MST(S \cup \{v\}) - MST(S) \geq \max_{u \in S} w(v, u).$$

证明。令 T 为 $\mathrm{G}_C[S]$ 的最大生成树，令 $u^* \in$ 为 $\arg\max_{u \in S} w(v, u)$。则 $T \cup \{(v, u^*)\}$ 是 $\mathrm{G}_C[S \cup \{v\}]$, 的一棵生成树，因此 $\mathrm{MST}(S \cup \{v\}) \geq \mathrm{MST}(S) + \max_{u \in S} w(v, u)$。□

定理 5（代理边际增益性质）。设 $S = \{C_q\} \cup \mathcal{R}$，并设 C^* 是通过最大化式 *25* 中的代理边际增益 $\Delta(C \mid \mathcal{R})$ 选出的候选项。则

$$G(\mathcal{R} \cup \{C^*\}) - G(\mathcal{R}) \geq \Delta(C^* \mid \mathcal{R}).$$

此外，如果 $u_t \in$ 参数最大值$_{u \in S_{t-1}} w(C_{(t)}u)$，则 $\{(C_{(t)}u_t)\}_{t=1}^{K}$ 在 $\{C_q\} \cup \mathcal{R}$ 上构成一棵生成树，并且

$$Coherence(\mathcal{R}) = MST(\{C_q\} \cup \mathcal{R}) \geq \sum_{t=1}^{K} w(C^{(t)}, u_t).$$

证明。

$G(\mathrm{R} \cup \{C^*\}) - G(\mathrm{R}) = w(C_q, C^*) + \lambda(\text{最小生成树}(S \cup \{C^*\}) - \text{最小生成树}(S))$

且引理 1 给出最小生成树$(S \cup \{C^*\})$ – 最小生成树$(S) \geq$ $\max_{u \in S} w(C^*, u)$，因此第一个结论成立。第二个结论成立，因为每次迭代都会向以 C_q为根的连通分量添加一个新顶点和一条连接边，而最小生成树在所有生成树中权重最大。 □