arXiv:2603.04799v1 [cs.DB] 5 Mar 2026

超越线性 LLM 调用：一种高效且有效的语义过滤器范式

Nan Hou 香港中文大学
nhou@se.cuhk.edu.hk

Kangfei Zhao 北京理工大学
zkf1105@gmail.com

Jiadong Xie 香港中文大学
jdxie@se.cuhk.edu.hk

Jeffrey Xu Yu 香港科技大学（广州）
jeffreyxuyu@hkust- gz.edu.cn

摘要

大语言模型正越来越多地用于大型语料库上的语义查询处理。已有研究提出了一组源自关系代数的语义算子，为表达此类查询提供统一接口，其中语义过滤算子是基石。给定一张表 T以及一个自然语言谓词 e，对于关系中的每个元组，语义过滤器的执行过程是构造一个将谓词 e 与其内容相结合的输入提示，查询大语言模型，并获得二元决策。然而，这种逐元组评估需要对表进行完整的线性扫描，从而带来难以承受的延迟和词元成本。尽管近期工作试图优化语义过滤，但仍未突破线性大语言模型调用的障碍。为此，我们提出聚类-采样-投票（CSV），一种新框架，它将大语言模型调用降至亚线性复杂度，同时提供误差保证。CSV将元组嵌入语义簇，采样一个小子集进行大语言模型评估，并通过两种提出的投票策略推断簇级标签：UniVote均匀聚合标签，SimVote则按语义相似度对投票加权。此外，CSV会对模糊簇触发重新聚类，以确保在多样化数据集上的稳健性。在真实世界数据集上开展的结果表明，与最先进方法相比，CSV将大语言模型调用次数减少了1.28-355×，同时在准确率和F1 分数方面保持了相当的有效性。

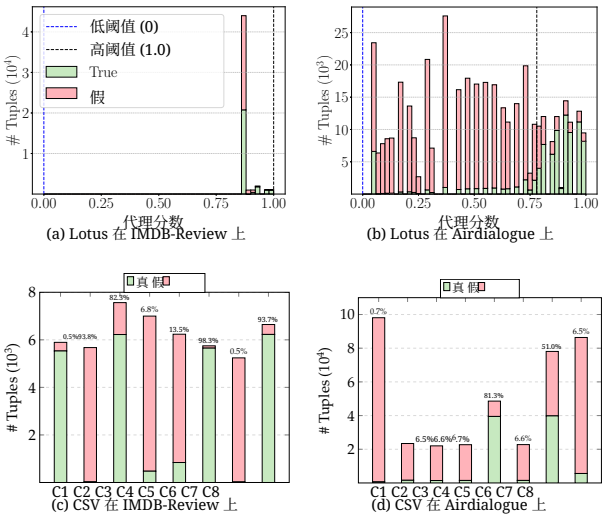
PVLDB 引用格式：
Nan Hou、Kangfei Zhao、Jiadong Xie 和 Jeffrey Xu Yu。《超越线性 LLM 调用：一种高效且有效的语义过滤范式》。PVLDB, 20(1)：XXX-XXX，2026年。doi:XX.XX/XXX.XX

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

1 引言

大语言模型（LLM）已迅速成为现代数据处理中的重要工具，展现出前所未有的能
能够理解自然语言查询并利用其内嵌的世界知识进行推理。许多复杂的

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。请访问 https://creativecommons.org/licenses/by-nc-nd/4.0/ 查看本许可协议副本。如需进行超出本许可协议范围的使用，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。出版权已授权给 VLDB Endowment。《VLDB Endowment 会议录》，第 20 卷，第 1 期，ISSN 2150-8097。doi:XX.XX/XXX.XX



(a) Lotus 在 IMDB-Review 上 (b) Lotus 在 Airdialogue 上
(c) CSV 在 IMDB-Review 上 (d) CSV 在 Airdialogue 上

图 1：Lotus 与我们的方法（CSV）的数据标签分布

分析应用已借助大语言模型得到增强，包括多文档摘要 [50]，因果关系抽取 [9]，和法律合同分析 [8]。随着半结构化和非结构化数据持续增长，分析系统越来越需要支持批量语义查询处理，即通过基于大语言模型的语义转换来评估海量数据。与遵循符号推理的经典关系算子不同，语义查询处理侧重语用学，并依赖语境、共享知识和共识进行推理。

为弥合语义查询处理与可扩展数据系统之间的差距，近年来提出了许多系统 [3, 10, 24, 25, 33, 38–40, 43]，使用户能够通过 DataFrame API 或带有用户定义函数（UDF）的扩展 SQL 来指定语义查询。尽管它们在查询表达方式和底层系统设计上各不相同，但仍可从其支持的查询中提炼出一组核心语义转换运算符。与关系运算符类似，Lotus [33] 定义了一组语义算子，包括语义过滤器、语义映射、语义连接等，从而构成一种声明式且可组合的查询接口，支持在算子层面进行算法优化。在这些算子中，语义过滤器是关系代数中选择运算的对应算子，也是几乎所有系统中查询的基石。给定一个包含一个或多个文本属性的表 T和一个自然语言（NL）谓词 e，语义过滤器通过大语言模型的判断，返回T中所有文本属性满足该谓词 e 的元组。用户可以利用此算子直接表达二元分类。例如，在 IMDB-Review 数据集 [30], 中，

带有自然语言谓词“评论是正面的”的语义过滤器可用于情感分析。与关系算子的求值不同，语义算子会将自然语言谓词的上下文与输入元组结合起来，同时利用大语言模型从大型语料库中学习到的知识。因此，其输出本质上会受到提示结构、大语言模型骨干和模型配置的影响，存在不确定性。

所有现有语义过滤系统采用的传统方法，都是线性扫描表 T，并针对每个元组调用一次大语言模型。当表规模较大时，这种方法会导致查询延迟较高，且词元消耗巨大。为提高语义过滤器的查询效率，Lotus [33] 提出了一种两阶段模型级联算法，考虑到 LLM 推理是查询处理过程中的瓶颈。该算法采用轻量级代理模型（通常是小型大语言模型）预先筛选元组，并收集部分元组以学习阈值，从而将不确定的元组转发给能力更强的大语言模型进行最终验证。然而在实践中，我们发现，该级联算法在真实世界数据集上面临查询精度与效率方面的两难困境。如图 1(a) 和 1(b) 所示，代理分数介于低阈值和高阈值之间的所有元组都必须转发给能力更强的大语言模型。在 Lotus 中，即便在最佳情况下，所有数据也必须由代理模型进行线性处理。在最坏情况下，阈值校准不佳会导致几乎所有数据都被传递给能力更强的大语言模型，不仅抵消了预期的效率提升，还会加剧整体推理成本。这个问题至关重要；然而，基础语义过滤算子的优化却不幸地一直被忽视。

为此，我们希望从算法层面重新审视语义筛选条件的处理优化，并由此提出一个问题：**能否在为语义筛选器提供误差保证的同时，降低大语言模型调用的复杂度？** 对于关系过滤，数据库系统会在查询属性上构建 B+-树或位图索引，以局部索引查找取代全表扫描。对于语义过滤，尽管现有语义分析系统 [27] 允许用户使用各种预训练嵌入模型构建语义索引，但能否利用向量化语义索引加速算法，以及如何利用它们减少语义过滤中的大语言模型调用，仍不明确。此外，理论误差分析与算法设计紧密交织，虽更具挑战性，却至关重要。由于大语言模型采用了复杂的学习机制，用户通常将其视为黑盒，因此，查询准确率保证是系统和算法设计者必须考虑的问题。尽管从深度学习的角度难以洞察模型的判断机制，但从筛选算法的角度提供严格的误差保证，有助于解读查询结果，并指导算法和系统的配置。

我们优化语义过滤的方法，首先受到近期批处理场景下大语言模型推理领域进展的启发 [28, 31, 35]。鉴于大语言模型的推理成本高昂，近期研究探索了输入缓存策略来加速在线查询，即将历史输入查询及其推理结果保存在缓存中。新查询到达时，从缓存中检索与其语义相似的查询，并直接使用这些查询的缓存结果作为代理输出。输入缓存能够加速推理，其背后的直觉是：输入越相似其大语言模型输出就越相似。对于语义过滤查询，其大语言模型调用的输入由三部分组成：系统提示（即用于语义过滤的统一指令）、谓词表达式和一个元组的值。对于给定的过滤谓词，所有元组的系统提示和谓词表达式都相同。最终决定大语言模型调用结果的是元组值，而语义相似度可以通过聚类方法来捕捉。

本文提出了一种新的范式，称为聚类-采样-投票（*CSV*），用于大语言模型驱动的语义过滤处理，旨在减少大语言模型调用次数，同时保持准确率。CSV 框架基于这样一种观察：语义相似的输入往往会从大语言模型获得一致的输出。利用这一特性，CSV 能够在相似元组之间摊销推理成本，同时保持较高的标注保真度。该框架包含三个关键阶段：1 聚类：使用预训练编码器对关系中的所有元组进行嵌入，并采用 K-means 等算法将其分组成语义相似的簇。此步骤在线下执行，可在不同查询间重复使用。2 采样：从每个簇中抽取少量元组作为代表，并使用大语言模型进行评估。3 投票：对于每个簇中剩余的元组，根据大语言模型对采样集的评估结果推断其标签，采用多数投票（均匀投票）或相似度加权投票（基于相似度的投票）。为确保正确性，CSV 提供了理论保证，可限定投票结果与预期大语言模型输出之间的差异。这通过约束每个簇内的标签频率分布来实现：如果某个簇缺乏语义纯度（即标签未能清晰区分），框架就会递归地将数据集重新聚类为多个子集；在最坏情况下，对于含糊不清的子集，框架会退回到直接使用大语言模型进行评估。此外，我们的理论分析明确建立了预期误差与采样比例之间的联系，为平衡理论保证与大语言模型调用次数提供了直观的方法。如图 1(c) 和 1(d) 所示，两个数据集中八个簇的标签分布表明，大多数簇都由单一标签主导，从而验证了聚类步骤的有效性。在少数情况下，例如图 1(d) 中的簇 7，其标签纯度较低，框架会触发在线重新聚类以保持准确率。这种自适应机制使 CSV 能够灵活、稳健地适用于多种数据集和谓词。Th

e con tributions o f thi s paper are summar ized as follows.

本文的贡献总结如下：**(1) 算法开发**。我们设计了一种用于大语言模型驱动的语义过滤的新算法，在平均情况下可将大语言模型调用次数降低至次线性规模。**(2) 理论分析**。我们对所提出的算法进行了详细的理论分析，明确揭示了误差界与采样比例之间的关系。**(3) 实验验证**。我们在多个数据集和基准测试上进行了大量实验，以展示所提方法的有效性和效率。与现有方法相比，我们的结果表明，执行时间、大语言模型调用次数和词元消耗均显著减少。

2 预备知识与现有方法

语义算子是一种对关系数据进行的声明式转换，它由自然语言表达作为

谓词。在本文中，我们研究由大语言模型（LLM）驱动的语义过滤器。

2.1 问题陈述

我们用 $T(A_1, \cdots, A_j)$，或简称为 T，表示一张表，其中 $A_1, \cdots, A_j$ 是 T 的属性。$t \in T$ 表示表 T 中的一个元组，而 $t(A_j)$ 表示元组 t 中属性 A_j 的值。$M : \mathrm{T} \mapsto \mathrm{T}$ 是一个预训练大语言模型，它接受输入并在文本空间 T 中生成输出。

语义过滤器。给定一个自然语言表达 e，语义过滤器 σ_M 通过大语言模型M对 T的一个或多个文本属性进行筛选，类似于关系选择算子，返回 T中满足由 e定义的语义谓词的所有元组。在式 (1) 的算子中，$M(t, e)$ 表示一次大语言模型调用，其中元组 t中涉及的属性，以及表达式 e和一条指令，共同构成大语言模型的输入，提示大语言模型判断 t 是否满足语义谓词 e。

$$\sigma_{M(e)}(T) = \{t \in T \mid M(t, e) = \texttt{True}\}. \tag{1}$$

2.2 现有方法与问题

在算法层面，语义过滤有三种现有方法 [25, 33, 49], ; 由于其他方法侧重于系统级优化而非算法改进，因此本文不予讨论 [25, 38]。

参考算法 [25, 33, 49]: 处理语义过滤器的一种直接算法是为每个 $t \in T$依次或通过表扫描并行调用一次大语言模型 $M(t, e)$。该算法总共需要进行 $O(|T|)$ 次大语言模型调用，成本极高。

Lotus [33]: 为降低逐元组大语言模型评估的成本，Lotus 提出了一种两阶段算法，利用成本较低的代理大语言模型，避免对 T中的每个元组都调用成本高昂的裁决大语言模型。对于语义过滤器 $\sigma_{M(e)}$，Lotus 首先从 T中抽取固定数量的元组，并学习两个阈值 τ_+和 τ_-，分别作为真和假的决策边界。对于每个元组 $t \in T$，系统都会调用代理大语言模型计算代理分数，该分数通常根据大语言模型输出的对数似然得出，可从开源大语言模型或某些应用程序接口中获取。如果代理分数低于 τ_-，Lotus 就将结果标记为假；如果高于 τ_+，则将结果标记为真；代理分数落在模糊区间$[\tau_-, \tau_+]$内的元组则会转交给能力更强的裁决大语言模型进行最终验证。

问题: 尽管 Lotus 提供了准确率保证 [18], , 但其实际性能可能会因多种原因而下降。首先，代理模型的输出并不总是与真实标签保持一致的良好对齐。在有利的情况下（例如图 1(b) 中的 Airdialogue），元组分布在整个得分范围内，真元组的得分通常高于假元组，因此可以轻松学习到合适的阈值，从而有把握地标记相当一部分元组。然而，在常见情况下（例如图 1(a) 中的 IMDB-Review），代理分数集中在狭窄的区间内（即 $[0.8, 0.85]$），而真元组的得分并不总是高于假元组。这种重叠会使基于小样本的阈值学习不稳定，并可能产生退化的阈值，实质上退回到 Reference 算法。其次，用于拟合 τ_+和 τ_-的采样过程可能与典型的语义过滤器工作负载不匹配。Lotus 的采样策略专为需要提取低选择性元组 [21] 的场景而设计。对于标签分布较为均衡的语义过滤器（如图 1(a) 和 1(b) 所示），这种策略可能会产生有偏样本，进而使阈值趋向极端值，降低代理模型有把握地排除负例的能力。第三，两阶段级联算法并未突破线性大语言模型调用这一瓶颈，甚至可能增加总成本。Lotus 必须对每个元组调用代理模型，而代理模型本身仍是一个推理成本相当高的大语言模型。当阈值校准不佳时，大量元组会被转发给裁决大语言模型，从而额外产生一次近乎线性的处理过程。在这种情况下，两阶段级联的成本可能高于直接应用 Reference 算法。

BARGAIN [49]: 类似地，BARGAIN 首先对每个元组 $t \in T$ 调用代理大语言模型，以获得代理预测及其对应的代理分数。随后，它根据预先设定的区间（例如，20 个分数子区间）将元组划分到不同的分数区域。BARGAIN 从代理分数最高的区域开始采样，并逐步扩展至分数较低的区域。在每个区域内，它会对元组进行采样，并调用裁决大语言模型，以检验该区域中的元组是否能够达到用户指定的质量目标。区域级流程会持续进行，直到 BARGAIN 确定一个最低代理分数阈值，使得纳入任何分数更低的区域都会导致无法满足质量目标。最后，将代理分数低于该阈值的元组交由裁决大语言模型进行最终验证。

问题: 由于 BARGAIN 需要像 Lotus 一样，使用大量代理大语言模型扫描数据集，因此在线性处理上会产生相当大的开销。值得注意的是，近期研究 [14, 15]指出，神经网络的置信度分数往往校准不佳，无法准确反映真正正确的概率，这是因为神经网络天生过度自信。这一问题可能会影响 BARGAIN 的性能，因为代理分数构成了其方法的基础。

3 我们的 SemanticFilter 方法

本节提出一种基于聚类-采样-投票（CSV）范式的新型语义过滤器算法，旨在解决第 §2 节中讨论的问题。总体而言，我们的方法以基于嵌入的分组取代了针对每个元组的大语言模型代理调用。我们使用嵌入模型为每个元组计算一个嵌入；这些嵌入可以离线生成（例如，在创建表或插入元组时生成），其成本显著低于调用大语言模型来获取代理分数。我们对元组嵌入进行聚类，并根据簇内真或假的占优情况，为每个簇推断一个标签。如图 1(c) 和 (d) 所示，这使我们能够在无需大语言模型调用的情况下，为大量元组添加标签。

我们的 CSV 范式基于这样一种直觉：嵌入相似的元组往往会引出大语言模型相似的响应；因此，基于嵌入的分组可以作为一种初步近似。这种直觉常用于推理缓存：当两个提示词在语义空间中距离较近时，它们的输出在含义上很可能相似，因此可以复用先前的推理结果 [28, 31, 35]。在语义过滤中，系统提示和谓词表达式是固定的，只有元组内容会在不同调用中变化。因此，大语言模型输出的变化完全由元组驱动，而

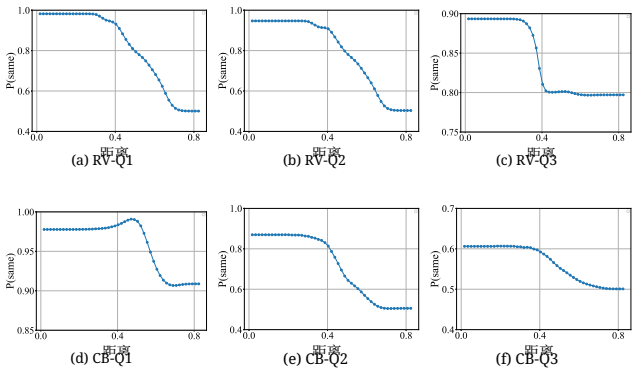


图 2：距离上的概率分布

元组之间的语义接近度可以通过嵌入空间中的距离来衡量。

我们通过量化多个谓词中嵌入距离与标签一致性之间的关系，对这一假设进行实证验证。对于每个元组，我们使用 E5-Large [44] 计算其嵌入，并从大语言模型获取真标签的概率。对于每对元组（t_i，t_j），我们计算其嵌入之间的欧几里得距离，以及这两个元组具有相同真或假标签的概率。将所有成对距离的范围划分为 50 个区间后，图 2 展示了在两个数据集的 6 个查询中，标签一致性的均值概率随距离增大而变化的趋势。不同谓词呈现出一致的规律：嵌入距离越大，标签一致性的概率越低。影响程度因谓词而异——例如，RV-Q1、RV-Q2 和 CB-Q2 的下降更为陡峭，而 RV-Q3 和 CB-Q1 的下降则较为缓慢——但总体趋势始终稳定。这些发现为使用基于嵌入的聚类作为语义过滤中摊销大语言模型评估开销的机制提供了定量支持。接下来，我们将在 §3.1 介绍 CSV 的详细信息，并在 §3.2 提供其理论分析。

3.1 聚类-采样-投票范式

顾名思义，聚类-采样-投票（CSV）范式分两个阶段处理语义过滤查询：离线聚类阶段和在线采样与投票阶段。在离线阶段，我们采用 K-means 等聚类算法，将表 T 中的元组划分为若干簇，这一过程与查询无关。在线阶段，我们从每个簇中采样一个子集，调用大语言模型评估这些采样元组是否满足语义谓词，并通过投票汇总结果，以确定该簇中的其余元组。由于投票机制会直接跳过对未采样元组的大语言模型调用，为了保持较高的准确率，我们使用两个阈值 ub 和 lb，限制对置信度较低的元组进行这种直接过滤。此外，在接下来的理论分析部分，我们证明误差由阈值 ub 和 lb保证。该算法会对这些低置信度元组启动回退机制，通过在线细粒度重新聚类、重新采样和重新投票来完善判定。如果不确定性仍然存在，且递归深度达到其

算法 1： SemanticFilter(T, e, M, k, ξ)

输入 ：输入表 T、表达式 e、大语言模型 M、簇数 k，以及采样比例 ξ

输出： 输出表 R

1 初始化 $R \leftarrow \emptyset$；
2 为每个元组生成嵌入 $\mathbf{e}_i$ $t_i \in T$；
3 将 $\{(\mathbf{e}_1, t_1), \cdots, (\mathbf{e}_n, t_n)\}$ 划分为 k 个簇
通过对 $\{\mathbf{e}_1, \cdots, \mathbf{e}_n\}$ 执行 k-均值聚类，得到 $\mathbf{C} = \{C_1, \cdots, C_k\}$；
4 初始化 $Q \leftarrow \mathbf{C}$；
5 **当** Q 非空时**执行**
6 初始化 U $\leftarrow \emptyset$；
7 **对于** $C_j \in Q$ **执行**
8 Q.remove(C_j)；
9 以比例 ξ 随机抽样子集 $C'_j \subset C_j$；
10 分配 $O \leftarrow \emptyset$；
11 **对于**（$\mathbf{e}_i$，t_i）$\in C'_j$ **do**
12 $o_i \leftarrow M(t_i, e)$；
13 $O \leftarrow O \cup \{o_i, t_i, \mathbf{e}_i)\}$；
14 **如果** o_i 是真**则**
15 $R \leftarrow R \cup \{t_i\}$；
16 $C^*, R^+ \leftarrow$ 投票(O, C_j, C'_j)；
17 $R \leftarrow R \cup R^+$；
18 **如果** C^* 不为空**则**
19 U $\leftarrow$ U $\cup C^*$；
20 **如果** C^* 不为空**则**
21 通过将 U 划分为新的簇 $\{C_{j_1}, \cdots, C_{j_m}\}$ k-均值；
22 $Q \leftarrow \{C_{j_1}, \cdots, C_{j_m}\}$；
23 返回 R；

最大阈值，框架最终会回退到直接调用大语言模型，以处理剩余的歧义元组。

算法 1 展示了语义过滤器中 CSV 的处理流程，给定一张表 T、一个语义表达式 e、一个大语言模型 M，以及簇数量参数 k。在离线阶段，该算法采用嵌入模型将每个元组 t_i 编码为向量表示 $\mathbf{e}_i$（第 2 行），然后根据嵌入的语义相似度，使用 K-means 将元组划分为 k 个簇 $\{C_1, \cdots, C_k\}$（第 3 行）。在在线采样与投票阶段，我们使用投票策略逐一处理每个簇。这里，我们使用集合 Q来维护所有待处理的簇。对于每个簇层，我们初始化一个临时集合 U（第 8 行），用于收集所有需要进一步细化的低置信度元组。对于一个簇 C_i，我们首先以采样率 ξ均匀抽取一批元组 C'_i（第 9 行）。随后，我们调用大语言模型 M，根据语义谓词 e 分别推断每个采样元组的结果（第 10–15 行），并将结果保存在临时集合 O中。通过基于大语言模型的语义过滤器的采样元组会直接插入结果表 R（第 14–15 行）。对于其余元组 $C_j \backslash C'_j$，算法引入投票机制，即第 16 行中的 Vote 函数。该函数以采样元组的结果、当前簇中的所有元组以及采样元组作为输入。该函数返回两个

算法 2： UniVote(O, C, C')

输入： 一组采样元组及大语言模型调用结果 O，一个包含其嵌入的元组簇 C，该包含其嵌入的元组采样子集 C'

输出： 带有其嵌入的未决元组 C^* 和正元组 R^+

$O^+ \leftarrow \{o_i = \text{真} \mid (o_i, t_i, \mathbf{e}_i) \in O\}$;
得分 $\leftarrow \frac{|O^+|}{|O|}$;
如果 得分 $\geq$ 上界 **那么**
  $C^* \leftarrow \emptyset, R^+ \leftarrow \{t_i \mid (t_i, \mathbf{e}_i) \in C \setminus C'\}$;
否则如果 得分 $\leq$ 下界 **那么**
  $C^* \leftarrow \emptyset, R^+ \leftarrow \emptyset$;
else
  $C^* \leftarrow C \setminus C', R^+ \leftarrow \emptyset$;
返回 C^*, R^+;

算法 3： SimVote(O, C, C')

输入： 一组采样元组及大语言模型调用结果 O，一个包含其嵌入 C的元组簇，一组包含其嵌入 C'的元组采样子集

输出： 带有嵌入 C^*的未决元组和正元组 R^+

$C^* \leftarrow \emptyset, R^+ \leftarrow \emptyset$;
对于 $(t_j, \mathbf{e}j) \in C \setminus C'$ **执行**
  得分 $\leftarrow 0$;
  范数 $\leftarrow \sum_{t_k \in C'}$ 相似度$(\mathbf{e}_j, \mathbf{e}_k)$;
  对于 $(o_i, t_i, \mathbf{e}_i) \in O$**执行**
    标志 $\leftarrow 1$ 如果 o_i 为真则为 0;
    得分 $\leftarrow$ 得分 $+ \frac{\text{相似度}(\mathbf{e}_i, \mathbf{e}^j) \text{ 范数} \cdot \text{标志}}{}$;
  如果 得分 > 上界 **则**
    $R^+ \leftarrow R^+ \cup \{t_j\}$;
  否则如果 得分 < 上界 $\wedge$得分 > 下界 **则**
    $C^* \leftarrow C^* \cup \{(t_j, \mathbf{e}j)\}$;
返回 C^*, R^+;

元组集，C^* 和 R^+，分别表示需要进行回退验证的低置信元组集，以及经投票判定通过过滤器的高置信元组集。我们将 R^+ 追加到结果表中（第 17 行）。对于低置信度元组，我们将所有这类集合收集到 U 中（第 18 行）。处理完当前簇层后，如果 U 非空，我们就使用 K-means 对 U 整体重新分区（第 19 行），并将生成的子簇放回集合 Q，以便进行下一轮迭代（第 20 行）。随后，按照相同的流程处理每个新子簇（第 10–19 行），即均匀采样与投票，直至达到最大分裂上限。我们为聚类与投票算法提供了两种投票策略，即均匀投票（算法 2 ）和基于相似度的投票（算法 3 ）。这两种策略都使用两个阈值 $ub, lb \in$（0, 1）来评估投票信念。下文将详细介绍这两种算法。

均匀投票。 给定采样元组 O的结果，均匀投票计算通过过滤器的元组比例（算法 2 第 1 行）。将其与两个阈值 ub和 lb 进行比较，得到式 (3) 所示的三种情况。

$$\text{score}(t_i) = \frac{|O^+|}{|O|}, \forall t_i \in C \setminus C', \tag{2}$$

$$\hat{M}(t_i; e) = \begin{cases} \text{True} & \text{if score}(t_i) \geq ub \\ \text{False} & \text{if score}(t_i) \leq lb \\ \text{Undetermined} & \text{if score}(t_i) \in (lb, ub) \end{cases}. \tag{3}$$

这里，我们用 $\hat{}M(t, e)$ 表示元组 t 关于 e 的投票结果。在每种情况下，均匀投票都会返回被视为通过过滤器的相应元组 R^+，以及未决元组 C^*（第 3–8 行）。在均匀投票中，C'中采样元组的结果贡献相等，且投票结果 $\hat{}M(t; e)$ 对簇中所有剩余元组都相同。为实现细粒度决策，我们进一步提出一种基于相似度的投票机制，该机制考虑采样元组与剩余元组之间的嵌入相似度。

基于相似度的投票。 算法 3 展示了基于相似度的投票流程，该流程还会返回两个集合 R^+ 和 C^*，作为额外结果以及未决元组。在算法 3 中，对于每个剩余元组 t_i，我们根据每个采样元组 t_j 与 t_i之间的归一化相似度对其投票结果进行汇总，并将得分函数重新表述为式 (4)

$$\text{score}(t_i) = \sum_{t_j \in C'} \frac{\text{sim}(e_i, e_j)}{\sum_{t_k \in C'} \text{sim}(e_i, e_k)} \mathbb{I}(M(t_j, e)), \forall t_i \in C \setminus C', \tag{4}$$

其中 $\mathbb{I}(M(t_j, e))$ 是对采样元组进行大语言模型调用的恒等函数。类似地，得分函数按照公式 (3) 与 lb和 ub进行比较。

与均匀投票相比，当初始聚类效果不佳时，基于相似度的投票能够提供稳健性。具体而言，当一个簇内正标签与负标签的分布不满足我们的标准，即 lb和 ub 时，均匀投票会重新聚类整个簇，而基于相似度的投票可能对某些项成功投票。这减少了后续步骤中未处理数据的数量。例如，§ 4.2 所示的 AD-Q1 和 AD-Q4 结果表明，在减少大语言模型调用方面，基于相似度的投票优于均匀投票。

示例 *3.1*。（聚类-采样-投票）。图 3 展示了在 IMDB-Review 数据集上进行聚类与均匀投票的运行示例。该数据集由一个元组表组成，每个元组代表一条电影评论。此处应用的语义过滤查询是“评论是正面的”。我们首先基于元组嵌入，使用 K-均值聚类将该表划分为 8 个簇。每个簇的标签分布如图 1(c) 所示。接下来，我们分别对每个簇进行投票。我们以簇 C3（绿色高亮）和 C8（蓝色高亮）为例。在此运行示例中，我们设定 $lb = 0.15$ 和 $ub = 0.85$。在簇 C8 中，我们均匀采样了 101 个元组，并通过对每个元组分别进行大语言模型调用获得其语义过滤结果，例如，对于元组“this file is outstanding...”，语义过滤器返回真。在采样元组中，有 99.01% 被标记为真，超过了上界（0.9901 > ub），因此我们将该簇中的所有元组归类为真。对于簇 C3，其中均匀采样了 101 个元组，

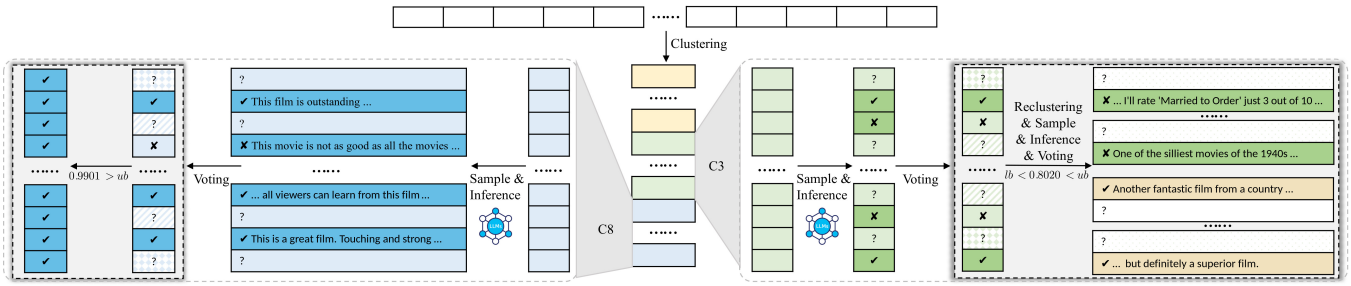


图 3：在 IMDB-Review 数据集上使用 UniVote 的聚类-采样-投票（CSV）运行示例

80.20% 被分类为真。由于 $lb < 0.8020 < ub$，我们采用重新聚类策略。该策略产生了两个簇，在图 3 最右侧以绿色和黄色区分。类似地，我们从这两个簇中的每个簇采样元组，利用大语言模型对单个元组进行分析，随后进行投票。

复杂度分析。给定表 T 以及采样率 ξ，CSV 范式在最佳情况下发起 $O(\xi|T|)$ 次大语言模型调用，在最坏情况下发起 $O(|T|)$ 次大语言模型调用。

CSV 框架可通过为多列生成融合语义嵌入来支持多列筛选条件。例如，将多列文本与列名拼接后，把长上下文输入预训练语言模型以生成逐行嵌入，或通过层次聚类为所有列的嵌入建立索引。此外，我们的方法可结合增量聚类算法来维护簇，从而扩展支持元组的插入/删除。我们的更新处理区分小批量与大批量更新。(1) 小规模、偶发更新。当更新批次较小时，假定簇质心保持不变。新数据被插入到质心最近的簇中，并复用已有的簇级投票结果。若某个簇此前未达成共识，则采用线性增长的大语言模型调用来纳入这些少量更新，同时将开销保持在较低水平。(2) 较大规模的周期性更新。对于较大的更新批次，我们使用小批量 K-means 更新聚类。后续的采样、投票和重新聚类与底层数据完全兼容。我们会缓存原始数据集中采样元组的大语言模型处理结果，以便潜在复用。对于更新后数据集中的样本，将调用大语言模型处理缓存中缺失的元组。对于删除，被删除的元组将被标记，并在簇数量达到特定数值时进行合并。

3.2 对投票策略 ξ 设置的分析

§3.1介绍了两种投票策略，即UniVote和SimVote　　.

这两种策略都由一个关键参数——采样比例 ξ——决定，该参数会直接影响时间复杂度（需要 $O(\xi|T|)$ 次大语言模型调用）和理论误差（后续讨论将对此进行详细阐述）。具体而言，投票策略从给定的簇中抽取一个元组子集，其数量占该簇全部元组的比例为 ξ；随后，将抽取的元组输入大语言模型，并根据采样结果推断簇级标签。ξ的选择会自然带来权衡。较大的 ξ会增加大语言模型调用次数，但能降低估计误差；反之，较小的ξ会减少大语言模型调用次数，但代价是误差更高。然而在实践中，用户通常缺乏选择 ξ所需的先验知识，难以兼顾成本与质量。

为了进行有原则的参数调优，我们引入一个用户友好的参数 ϵ，表示理论分析中由用户指定的误差容限。接下来，我们推导 ξ与 ϵ之间的理论关系，从而能够针对给定的 ϵ选择合适的采样比例 ξ，使我们的CSV 框架以高概率达到期望的准确率。

首先，我们介绍理论分析的基石，即引理 3.2中的Bernstein不等式 [5]，该不等式将在后文中频繁使用。

引理 3.2.（*Bernstein*不等式 *[5]*）给定 $X = \{x_1, \cdots, x_n\}$ 是规模为 n 的有限总体，其中 $x_1, \cdots, x_n$ 是均值为 μ 的二元变量。令$\hat{}\mu$为从 X进行无放回简单随机抽样得到的 k个变量 $c_1, \cdots, c_k$的均值。对于任意$\epsilon > 0$，我们有

$$Pr\left[|\hat{\mu} - \mu| \geq \epsilon\right] \leq 2exp\left(-\frac{k\epsilon^2}{2\hat{\sigma}^2 + 2R\epsilon/3} \cdot \frac{n-k}{n-1}\right),$$

其中，$\hat{}\sigma$ 是 k 个采样变量c_1、$\cdots$、c_k的样本标准差，即$\hat{}\sigma^2 = \frac{1}{k-1}\sum_{i=1}^{k}(c_i - \hat{}\mu)^2$；$R$是$x_i$的几乎必然界，这意味着 $|x_i| \leq R$几乎必然成立。

UniVote分析。首先，我们分析UniVote下的SemanticFilter。在此，给定 C_j的 $X = \{x_1$、$\cdots$、$x_n\}$，其中 x_1、$\cdots$、x_n为二元变量，并且对于每个$(\mathbf{e}_i, t_i) \in C_j$，都有 $x_i = \mathbb{I}(M(t_i, e))$。令 $\mu(X)$表示 X中变量的均值，令 $f(t_i) \in \{0, 1\}$表示$(\mathbf{e}_i, t_i)$是否被UniVote纳入 R^+。具体而言，当 $|O^+|/|O| \geq ub$时，$f(t_i) = 1$；当 $|O^+|/|O| \leq lb$时，$f(t_i) = 0$，其中 $|O| = k$且 $\xi = |O|/|C_j|$。根据Bernstein界，当采样率 ξ足够大时，我们确认对于每个$(\mathbf{e}_i, t_i)$，$f(t_i)$都是 x_i的准确估计量。

定理3.3。给定一个簇 $C = \{(\mathbf{e}_1, t_1), \cdots (\mathbf{e}_n, t_n)\}$以及参数 lb、ub、l和$\mathbb{I}$ M e $\in C$ x_i。令 $X = \{$、$\cdots$、$\}$属于 C，其中，对于每个 $(\mathbf{e}_i, t_i)$，$((t_i,))$均为二元。假设 ξ 满足

$$\xi \geq \frac{1}{2} - \sqrt{\frac{1}{4} + \ln l\left(\frac{2\hat{\sigma}^2}{\epsilon^2} + \frac{2}{3\epsilon}\right)}.$$

那么，当 $\mu(O) \leq lb$ 或 $\mu(O) \geq ub$时，以下不等式以至少 $1 - 2l^n$ 的概率成立：

$$Pr[f(t_i) \neq x_i] \leq \max(lb + \epsilon, 1 - (ub - \epsilon)).$$

定理3.3给出了投票决策 $f(t_i)$ 与大语言模型的真实输出 x_i 不一致的概率上界。关键在于，样本均值 $\mu(O)$ 以高概率集中在真实总体的均值 $\mu(X)$ 附近。对 ξ 的条件确保我们抽取了足够多的元组，使 Bernstein不等式能够保证这种集中在容差 ϵ 范围内。因此，如果投票给出了一个标签，则误差上界为 $\max(lb+\epsilon, 1-(ub-\epsilon))$，且这一保证以至少 $1-2l^n$ 的概率成立。**证明概要：** 设 $\mu(O)$ 为从中抽取的 k 个元素的均值

$$X, \text{i.e., } k = \xi n \geq \tfrac{1}{2}n - n\cdot\sqrt{\tfrac{1}{4} + \ln l\left(\tfrac{2\hat{\sigma}^2}{\epsilon^2} + \tfrac{2}{3\epsilon}\right)} \text{ and } \mu(O) = |O^+|/|O|.$$

对于 $\mu\ O\ \leq lb \quad f\ t_i\ = 0$ 。$Pr[f\ t_i\ \neq$ ( ) 的情况，$x_i] = Pr[x_i \neq 0] = Pr[x_i = 1] = \mu\ X$ $\dot{P}r[Pr[f\ t_i\ \neq$ 我们有 ( ) 因此，( ) ( )。接着，我们有 ( ) $x_i] \geq lb+\epsilon] = Pr[\mu\ X\ \geq lb+\epsilon] \leq Pr[\mu\ X\ \geq \mu\ O\ +\epsilon] \leq$ ( ) ( ) ( ) ( )。根据 $Pr[|\mu\ X - \mu\ O\ | \geq \epsilon]$ 引理 ., 我们有 ( ) $\mu(O)| \geq \epsilon] \leq 2\exp\left(-\frac{3}{2}\frac{2}{k\epsilon^2}\cdot/3\cdot\frac{n-k}{n-1}\right) < 2\exp(n\ln l) = 2l^n$。 $Pr[|\mu\ X -$

对于另一种情况，即 $\mu(O) \geq ub$，我们有 $f(t_i) = 1$。因此，$Pr[f(t_i) \neq x_i] = Pr[x_i \neq 1] = Pr[x_i = 0] = 1-\mu(X)$。接着，我们有 $Pr[Pr[f\ t_i\ \neq x_i] \geq 1 - ub-\epsilon\ ] = Pr[1-\mu\ X\ \geq 1 - ub-\epsilon\ ] =$ ( ) ( ) ( ) $Pr[\mu\ X\ \leq ub-\epsilon] \leq Pr[\mu\ X\ \leq \mu\ O\ -\epsilon] \leq Pr[|\mu\ X\ -\mu\ O\ | \geq \epsilon]$ ( ) ( ) ( ) ( ) ( ) ( )。基于前面的分析，我们已经证明 $Pr[|\mu(X) - \mu(O)| \geq \epsilon] \leq 2l^n$。□

注意，在 $lb < |O^+|/|O| < ub$ 的情形下，SemanticFilter 会继续重新划分簇，同时保持 lb 和 ub 的值不变。因此，分析仍然一致。因此，此处我们仅关注 $\mu(O) = |O^+|/|O|$ 满足 $\mu(O) \leq lb$ 或 $\mu(O) \geq ub$ 的情况。

对 SimVote 的分析。接下来，我们分析采用 SimVote 的 SemanticFilter。我们考虑引理 3.2 的加权版本。

推论 3.4。设 $X = \{x_1, \ldots\ldots, x_n\}$ 是一个大小为 n 的有限总体，其中 $x_1, \ldots\ldots, x_n$ 是均值为 μ 的二元变量。设 $c_1, \ldots\ldots, c_k$ 是从 X 中无放回抽取的样本，且每个 c_i 都关联一个权重 w_i，其中 $w_i \geq 0$ 和 $\sum_{i=1}^k w_i = 1$。设 $\hat{}\mu$ 为 $c_1, \ldots\ldots, c_k$ 的均值，即 $\hat{}\mu = \frac{1}{k}\sum_{i=1}^k c_i$；设 $\hat{}\mu_w$ 为 c_i 在权重 w_i 下的加权均值，即 $\hat{}\mu_w = \frac{1}{k}\sum_{i=1}^k w_i c_i$。对于任意 $\epsilon > 0$，有

$$Pr[|\hat{\mu}_w - \mu| \geq \epsilon] \leq 2\exp\left(\frac{-3k\epsilon^2}{(6\hat{\sigma}^2 + 2\epsilon)v}\cdot\frac{n-k}{n-1}\right),$$

其中，$\hat{\sigma^2} = \frac{1}{k-1}\sum_{i=1}^k (c_i - \hat{}\mu)^2$ 是样本标准差，且 v 是满足 $\max_i w_i \leq \frac{v}{k}$ 的常数。

推导可写为 $\hat{}\mu_w - \mu = \sum_{i=1}^k y_i$，其中 $y_i =$ $w_i\ c_i - \mu$ 0 3 2 y_i ( ) 的期望为 。应用引理 . 并对方差进行界定，即可证明推论 3.4。完整的推导和详细的证明步骤见附录 A

这里，给定簇 C_j 的 $X = \{x_1, \cdots, x_n\}$，其中 x_1、$\cdots$、x_n 为二元变量，且对于每个 (e_i, t_i)，$x_i = \mathbb{I}(M(t_i, e)) \in C_j$。令 $g(t_i) \in \{0, 1\}$ 表示 (e_i, t_i) 是否被 SimVote 纳入 R^+，即当 $\text{score}(t_i) \geq ub$ 时，$g(t_i) = 1$；当 $\text{score}(t_i) \leq lb$ 时，$g(t_i) = 0$，其中 $\text{score}(t_i)$ 定义见公式 (4)。

引理 *3.5.* 给定簇 $C = \{(e1, t_1), \cdots, (e_n, t_n)\}$ 及其子集 $O \subseteq C$，该子集从 C 中均匀采样。令 C 的 $X(C) = \{x_1, \cdots, x_n\}$，其中对每个 $(e_i, t_i) \in C$，$x_i = \mathbb{I}(M(t_i, e))$ 为二元，且 μ(得分$(C \setminus O)$) 为 $(e_i, t_i) \in C \setminus O$ 上得分 (t_i)（定义见式 *4*）的均值。则有 μ(得分$(C \setminus O)) = \mu(X(O))$。

证明概要： 由于 $\mathbb{I}(M(t_j, e)) = x_j$，我们有

$$|C \setminus O| \cdot \mu(\text{score}(C \setminus O)) = \sum_{t_i \in C\setminus O}\sum_{t_j \in O}\frac{\text{sim}(t_i, t_j)}{\sum_{t_k \in O}\text{sim}(t_i, t_k)}x_j$$

$$= \sum_{t_j \in O} x_j \cdot \left(\sum_{t_i \in C\setminus O}\frac{\text{sim}(t_i, t_j)}{\sum_{t_k \in O}\text{sim}(t_i, t_k)}\right). \quad (5)$$

由于从 C 中均匀采样得到 O，因此式 (5) 右侧为 $=$ $\sum_{t_j \in O} x_j |C\ \ O| \quad \mu \quad C\ \ O\ = \sum_{t_j \in O} x_j = \mu\ X\ O$ □ \，于是，(得分(\)) (())。

根据推论 3.4 和引理 3.5，我们确认，当采样率 ξ 足够大时，对于每个 (e_i, t_i)，$g(t_i)$ 都是 x_i 的准确估计量。

定理 *3.6*。给定一个簇 $C = \{(e1, t_1), \cdots, (e_n, t_n)\}$ 和参数 lb、ub、l、v 和 ϵ。令 $X = $ 为 C 的 $\{x_1, \cdots, x_n\}$，其中对于每个 $(e_i, t_i) \in C$，$x_i = M(t_i, e)$ 都是二元的。假设 ξ 满足

$$\xi \geq \frac{1}{2} - \sqrt{\frac{1}{4} + \frac{vlnl(6\hat{\sigma}^2 + 2\epsilon)}{3\epsilon^2}}$$

那么，当 $score(t_i) \leq lb$ 或 $score(t_i) \geq ub$ 时，以下不等式以至少 $1-2l^n$ 的概率成立：

$$Pr[g(t_i) \neq x_i] \leq \max(lb+\epsilon, 1-(ub-\epsilon)).$$

定理 3.6 将理论保证扩展到了 SimVote。虽然每个元组都获得了个性化得分，但引理 3.5 保证这些得分集中在与均匀采样相同的均值附近。由于加权机制产生的方差，定理 3.6 中关于 ξ 的条件比定理 3.3 中的条件稍复杂。尽管如此，最终的保证形式相同。

证明概要：我们有 $k = \xi n \geq n/2 - \sqrt{n^2/4 + n^2 vlnl(6\hat{}\sigma^2 + 2\epsilon)/3\epsilon^2}$。对于情形 $score(t_i) \leq lb$，我们有 $g(t_i) = 0$。因此，$Pr[g(t_i) \neq x_i] = Pr[x_i \neq 0] = Pr[x_i = 1] = \mu(X)$。接着，

$Pr[Pr[g\ t_i\ \neq x_i] \geq lb+\epsilon] = Pr[\mu\ X\ \geq lb+\epsilon] \leq Pr[\mu\ X\ \geq$ ( ) ( ) ( ) (t_i) $score$ $X\ -score$ $+\epsilon] \leq Pr[|\mu($ ( ) $(t_i)| \geq \epsilon]$。根据引理 .5，我们有 $\mu(score(t_i)) = \mu(X)$；根据推论 3.4，我们有 $Pr[|\mu\ X\ -score\ t_i\ | \geq \epsilon] \leq 2exp\ \frac{-3k\epsilon^2}{6\sigma^2+2\epsilon}\ v\ \frac{n-k}{n-1} < 2\quad n\quad l\ =$ ( ) $2l^n$ $score\ t_i\ \geq ub$ $g\ t_i\ = 1$ ( ) ((ˆ) ·) exp(ln)。对于另一种情形 ( )，我们有 ( )。因此，$Pr[g(t_i) \neq x_i] = Pr[x_i \neq 1] = Pr[x_i = 0] = 1-\mu(X)$。接着我们有 $Pr[Pr[g(t_i) \neq x_i] \geq 1-(ub-\epsilon)] = Pr[1-\mu(X) \geq 1 - ub-\epsilon\ ] = Pr[\mu\ X\ \leq ub-\epsilon] \leq Pr[\mu\ X\ \leq score\ t_i\ -\epsilon] \leq$ ( ) ( ) ( ) ( ) ( ) ( )。 $Pr[|\mu\ X\ - score\ t_i\ | \geq \epsilon]$ 基于上述分析，我们已经证明 $Pr[|\mu(X) - score(t_i)| \geq \epsilon] \leq 2l^n$。

4 实验研究

本节介绍 §4.1 中的实验设置，并从以下几个方面全面评估我们的方法：❶ 在语义过滤器上比较基线方法，并在 §4.2 中展示整体有效性和效率的结果。❷ 在 §4.3 中分析超参数对 CSV 的影响。❸ 在 §4.4 中考察 CSV 中重新聚类的效果。❹ 在 §4.5 中探讨实际结果与理论结果之间的差异。❺ 在 §4.6 中评估 CSV 对多种合成查询的泛化能力。❻ 在附录 B 中研究采用不同嵌入模型和大语言模型的影响。

4.1 实验设置

数据集。 接下来 [6, 27, 33, 49] ，我们评估了五个用于语义过滤的数据集。1**IMDB-Review：** IMDB-Review [30] 数据集是一个源自影评领域的二元情感分类基准。它包含 50,000 条带标签的评论，这些评论均为用户在互联网电影数据库（IMDB）上撰写的完整影评，每条评论表达了对某部电影的正面或负面观点。该数据集在两种情感类别中的样本数量均等。2 **Codebase：** Codebase 数据集 [23] 由一个名为 user 的结构化表格构成，包含 9,378 行，其中有一列名为 AboutMe，存放较长的自我介绍文本。3 **Airdialogue：** Airdialogue 数据集 [45] 是一个源自航空票务领域的分类任务。它包含 402,035 条唯一对话，每条对话均标注为五个可能类别之一：预订（51.40%）、取消（1.46%）、无_航班（23.08%）、无_预订（23.89%）和变更（0.17%）。4 **TC：** Twitter仇恨言论与冒犯性语言（TC）数据集 [11] 是一个源自社交媒体领域的分类基准。它包含 24,783 条带标签的推文，每条推文均被标注为以下三个类别之一：仇恨言论、冒犯性语言或两者皆非。该数据集通过在 Twitter 上使用与仇恨言论相关的关键词收集推文，随后通过众包进行标注。5 **Fever：** 事实提取与验证（Fever）数据集 [41] 是一个用于事实核查领域声明验证的大规模基准，由改写维基百科中摘录的句子构建而成。在本研究中，我们使用经过清理的版本，其中已移除不完整条目，以确保一致性和可靠性。

查询。 表1总结了我们实验中使用的14个测试查询。对于 RV-Q2/Q3 和 CB-Q1/Q2/Q3，真实标签并未公开；因此，我们使用 GPT-4o 作为代理，通过直接调用大语言模型来获取这些标签。在这12个查询中，Fever 是一个引用两列的多列语义过滤器，其余查询均为单列谓词。针对 Airdialogue 的任务是一个多类分类问题，用于确定对话的类别。我们将该任务分解为四个二元语义过滤器查询 AD-Q1/Q2/Q3/Q4，每个查询都是针对以下四个类别之一的谓词：预订/取消/no_航班/no_预订。除了表1所示的谓词外，我们还在所有 Airdialogue 查询的提示词前添加以下指令："以下是航空票务代理与客户之间的一段对话。对话结果将属于以下5个类别之一。[预订]：代理已为客户预订航班（不包括更改航班）。 [取消]：代理取消了客户现有的有效预订。 [更改]：代理更改了客户现有的航班预订，并成功找到一个新航班。 [no_预订]：客户想要更改或取消航班，但该客户名下没有有效预订。[no_航班]：客户想预订从出发地到目的地的航班，但出发地与目的地之间没有航班。"

算法。 我们将所提出的方法与以下现有方法进行比较。1 **Reference**：根据公式 (1) 中的定义，我们依次为每个元组调用大语言模型，并获取结果，以此为语义过滤器建立基线。2 **Lotus**：Lotus 为语义过滤器引入了最先进的方法 [33]。我们采用该方法作为相应任务的基线。3 **BARGAIN**：BARGAIN [49] 采用了与 Lotus 类似的模型级联框架，同时为精确率、召回率和准确率提供了强有力的理论保证。我们将 BARGAIN 的准确率目标设为 0.85，容差设为 0.05。对于 Lotus 和 BARGAIN，我们均使用其各自源代码仓库中提供的默认超参数。这两种方法都使用 LlaMA-3.2- 3B 进行初步过滤，并使用 LLaMA-3.1-8B 作为更大的模型进行二次处理。

对于我们的方法，5 **UniCSV** 和 6 **SimCSV**是语义过滤器（算法 1）的两种算法，分别通过 UniVote（算法 2）和 SimVote（算法 3）实例化。默认情况下，我们使用 LLaMA-3.1-8B [42] 作为基础大语言模型，并使用 E5-Large [44] 作为嵌入模型。当输入文本超过嵌入模型支持的上下文长度时，会将其切分为较小的文本块，并分别对每个文本块进行嵌入。最终嵌入由所有文本段嵌入的均值计算得出，以确保不同长度的输入具有一致的表示。聚类时，我们采用 [34] 的 K 均值算法，并将默认簇数量设为 4。为避免过滤过程中进行过多重新聚类，重新聚类的最大迭代次数限制为 3。一旦剩余簇中发生投票失败，我们便采用顺序大语言模型调用，以确保误差有界。为更好地适应谓词的多样性，我们进一步引入 BM25 分数 [16] 来衡量词汇相似度，并将其作为聚类的距离度量，与嵌入的欧几里得距离联合使用。引入超参数 λ 来平衡不同距离的相对贡献，即，λ用于 L2 距离，而 $(1-\lambda)$ 用于 BM25。对于 CB-Q1 和 TC，我们设为 $\lambda = 0.4$；对于其余查询，则设为 $\lambda = 1.0$。其他超参数如下：$lb = 0.15$、$\xi = 0.005$、最小_样本 = 101。

评估指标。 我们通过准确率、召回率和 F1 分数评估不同方法的效果；通过执行时间、大语言模型调用次数和令牌消耗评估效率。这种多维度评估能够均衡分析预测有效性与计算效率。对于效率，执行时间衡量完成语义算子所需的端到端实际运行时间。大语言模型调用次数指大语言模型的总调用次数。为确保一致性，我们根据发出的提示词数量统计调用次数，不论单个提示词中是否批量处理了多个元组。令牌消耗统计处理的词元总数，包括输入（提示词）和输出词元。

实现细节 实验在配备单块 80GB NVIDIA A100 GPU 的本地环境中进行，并使用 vLLM [22] 作为推理引擎。对于所有方法，批量提示处理均通过 LiteLLM [1], 的批量_补全 方法实现，温度设为 0.7，输出词元上限设为 32。

4.2 SemanticFilter 的总体评估

本节将从效率和有效性两方面，对比我们的两种方法 UniCSV 和 Sim-CSV 与基线方法。

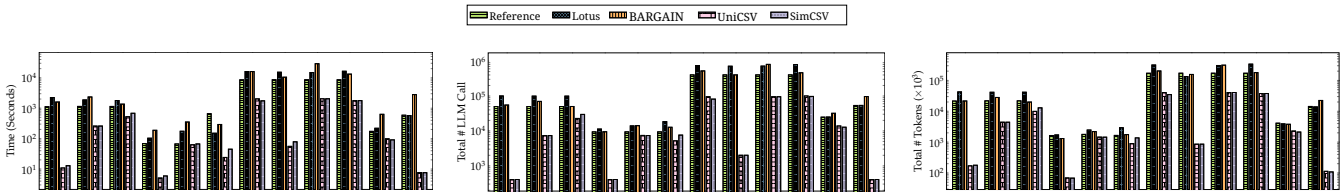
效率。 图 4 展示了不同方法的大语言模型调用次数、执行时间和令牌消耗。UniCSV 和 SimCSV 通过减少大语言模型调用次数，显著提升了效率：与 Reference 相比，减少了 1.28-200× 倍；与 Lotus 相比，减少了 1.81-355× 倍；与 BARGAIN 相比，减少了 1.68-200× 倍。由于大语言模型调用主导了运行时间

表1：测试语义查询的概况

数据集	查询	查询类型	谓词	真实值
IMDB-Review	RV-Q1	语义过滤器	这条{评论}是积极的。	可用
	RV-Q2	语义过滤器	该{评论}显式建议他人观看这部电影。	GPT-4o
	RV-Q3	语义过滤器	该{评论}侧重于技术细节等事实性描述。	GPT-4o
Codebase	CB-Q1	语义过滤器	{关于我}包含一个社交媒体链接。	GPT-4o
	CB-Q2	语义过滤器	此{关于我}文本表明，用户要么有计算机科学方面的经验，要么对计算机科学领域感兴趣。	GPT-4o
	CB-Q3	语义过滤器	{关于我}侧重于事实性身份信息。	GPT-4o
TC	TC	语义过滤器	{评论}包含仇恨言论或冒犯性语言。	可用
Airdialogue	AD-Q1/Q2/Q3/Q4	语义过滤器	该对话的结果是预订/取消/无_航班/无_预订。	可用
Fever	Fever	多列过滤器	The{声明}由{证据}支持。	可用

表2：语义过滤查询的准确率和 F1 分数

	方法	RV-Q1	RV-Q2	RV-Q3	CB-Q1	CB-Q2	CB-Q3	AD-Q1	AD-Q2	AD-Q3	AD-Q4	TC	Fever
Acc.	Reference	0.9480	0.8229	0.9097	0.9731	0.7782	0.7033	0.8972	0.9139	0.8112	0.8737	0.7203	0.8258
	Lotus	0.9480	0.8679	0.9948	0.9595	0.6853	0.6996	0.8938	0.9066	0.9192	0.8690	0.5196	0.7396
	BARGAIN	0.9186	0.7881	0.9948	0.9681	0.5308	0.6493	0.7423	0.8972	0.3885	0.9563	0.7872	0.5177
	UniCSV	0.9307	0.7723	0.9527	0.9673	0.7810	0.7090	0.8948	0.9193	0.8723	0.8975	0.7477	0.7526
	SimCSV	0.9305	0.7718	0.9398	0.9671	0.7813	0.7376	0.8968	0.9193	0.8722	0.8977	0.7799	0.7524
F1	Reference	0.9489	0.7633	0.0880	0.6344	0.7557	0.6374	0.9056	0.8165	0.7002	0.1841	0.8060	0.8947
	Lotus	0.9489	0.7113	0.0972	0.4947	0.6115	0.3107	0.9029	0.8048	0.8393	0.1788	0.6007	0.7955
	BARGAIN	0.9140	0.5616	0.0296	0.5954	0.1979	0.0947	0.6878	0.7722	0.3429	0.3697	0.8589	0.5831
	UniCSV	0.9336	0.7097	0.0880	0.0623	0.7601	0.6340	0.9039	0.8249	0.7800	0.2179	0.8348	0.8588
	SimCSV	0.9335	0.7092	0.0906	0.0679	0.7555	0.6589	0.9052	0.8249	0.7799	0.2175	0.8550	0.8586



（a）查询时间　（b）大语言模型调用总数　（c）词元总数

图 4：语义过滤器的查询时间、大语言模型调用次数和令牌用量对比

表 3：超参数对 UniCSV 和 SimCSV 性能的影响

参数	值	RV-Q1 Acc.	RV-Q1 F1	CB-Q2 Acc.	CB-Q2 F1	AD-Q2 Acc.	AD-Q2 F1	RV Acc.	RV F1	CB-Q2 Acc.	CB-Q2 F1	AD-Q2 Acc.	AD-Q2 F1
		UniCSV						SimCSV					
# 簇	2	0.9079	0.9138	0.7780	0.7492	0.9168	0.8210	0.9082	0.9140	0.7739	0.7450	0.9168	0.8209
	4	0.9307	0.9336	0.7852	0.7601	0.9192	0.8248	0.9305	0.9335	0.7803	0.7539	0.9193	0.8249
	8	0.9367	0.9367	0.7829	0.7597	0.9202	0.8264	0.9388	0.9384	0.7774	0.7534	0.9202	0.8264
	16	0.9424	0.9438	0.7815	0.7564	0.9200	0.8262	0.9379	0.9396	0.7788	0.7546	0.9199	0.8260
采样比率 ξ (‰)	5	0.9307	0.9336	0.7852	0.7601	0.9192	0.8248	0.9305	0.9335	0.7803	0.7539	0.9193	0.8249
	10	0.9304	0.9334	0.7794	0.7518	0.9192	0.8248	0.9459	0.9470	0.7756	0.7493	0.9193	0.8249
	15	0.9304	0.9334	0.7802	0.7537	0.9193	0.8250	0.9439	0.9453	0.7828	0.7568	0.9192	0.8248
	20	0.9308	0.9337	0.7803	0.7536	0.9191	0.8246	0.9306	0.9335	0.7827	0.7565	0.9191	0.8248
	25	0.9307	0.9336	0.7788	0.7513	0.9192	0.8248	0.9307	0.9337	0.7861	0.7603	0.9191	0.8246
下界	0.10	0.9302	0.9333	0.7803	0.7547	0.9193	0.8249	0.9312	0.9341	0.7814	0.7551	0.9193	0.8249
	0.15	0.9307	0.9336	0.7852	0.7601	0.9192	0.8248	0.9305	0.9335	0.7803	0.7539	0.9193	0.8249
	0.20	0.9304	0.9334	0.7852	0.7595	0.9193	0.8249	0.9304	0.9334	0.7839	0.7575	0.9193	0.8248
	0.50	0.9304	0.9334	0.7959	0.7956	0.9193	0.8249	0.9304	0.9334	0.7953	0.7950	0.9193	0.8345

和令牌用量，这些减少使所有数据集的性能都直接提升了 1 至 3 个数量级。例如，在 RV-Q1 上，UniCSV 和 SimCSV 仅需调用 404 次，便可在 13 秒内完成，令牌用量约为 17 万。相比之下，基线方法需要调用数万次，运行时间超过 1,000 秒，令牌用量超过 2,000 万。Lotus 和 BARGAIN 产生的较高成本源于得益于其两阶段模型级联过滤。在许多情况下，较小的3B模型无法学习到精确度足够高的阈值，导致超过一半的数据再次被路由到较大的8B模型。在最糟糕的情况下，即 Lotus 的 RV-Q1 和 BARGAIN 的 AD-Q3，较小的模型未能过滤掉任何元组，从而使所有三项指标的成本都增加。

有效性。表2报告了所有方法在12个语义过滤查询上的准确率和F1 分数。总体而言，UniCSV 和 SimCSV 的表现与 Reference 相当，且优于 Lotus 和 BARGAIN，CB-Q1 除外。相比之下，BARGAIN 的质量相较于 Reference 表现出很强的不稳定性：虽然它在 AD-Q4 上取得了较高的 F1 分数和准确率，但在 CB-Q2 和 AD-Q3 上却明显退化。我们推测，这种不稳定性是因为它将大语言模型的逻辑值用作谓词的置信度信号，而由于神经网络固有的过度自信，这种做法并不可靠 [14, 15]。一个显著的例外是，我们的方法在 CB-Q1 上 F1 分数较低。该查询显式提及实体“社交链接”，因此选择性极低，仅为 0.033。在默认设置 $lb = 0.15$ 下，正例稀少导致召回率因采样失败而不稳定（许多样本中几乎没有正例，甚至没有正例），而后续的投票过度自信又绕过了重新聚类。当我们将 lb 更改为 0.01 时，F1 分数达到 0.5825，但约有一半的元组通过线性大语言模型调用处理。

在聚类效果良好的数据集上，UniCSV 与 SimCSV 在准确率和 F1 分数方面差异甚微，表明两者的有效性相当。在 CB-Q2 和 AD-Q1 等聚类效果较差的情形下，SimCSV 略胜一筹，展现出对含噪或模糊簇边界更强的适应能力。这种稳健性提升了挑战性条件下的过滤效率。此外，UniCSV 和 SimCSV 表现出很强的稳定性：在不同数据集、查询和规模（从数千条数据，例如 CB，到数十万条数据，例如 AD）下，两者都能持续减少时间和令牌消耗，同时保持有效性。

4.3 超参数分析

我们分析以下三个影响 UniCSV 和 SimCSV 有效性与效率的超参数：簇数量、采样比例和下界。

(1) 簇数量。 如表 3 所示，增大簇大小可以提升实际性能，但不会影响理论误差界。当簇数量从 2 增至 4 时，准确率和 F1 分数显著提升，此后提升幅度逐渐减小。

为提高效率，如图 5 所示，簇数量不会影响大语言模型调用总数，例如 RV-Q1 和 AD-Q2，因为采样比例决定了每簇的查询数，从而保持稳定
的采样量。在 CB-Q2 这类较小的数据集中，每簇样本数主要由$min_$样本调控，因此簇数量增加时，大语言模型调用总数也会线性增加。此外，在 AD-Q2 中，将簇数量设为 2 会因语义区分不足而触发重新聚类，从而增加大语言模型调用、运行时间和令牌用量。

（2）采样比例。如表 3 所示，采样比例几乎不会影响 UniCSV 和 SimCSV 的准确率与 F1 分数，这表明较低的采样比例已足够。我们将在 §4.5 的后续实验中对此进行深入探讨。

在图 5 中，采样比例的增加通常会使大多数数据集上的大语言模型调用呈线性增长。而在 CB-Q2 上，尽管采样比例增加，大语言模型调用仍保持稳定。这种稳定性源于该数据集规模相对较小，其中大多数簇的采样数量为最小$_$采样 。值得注意的是，在使用 SimCSV 的 RV-Q1 上，当

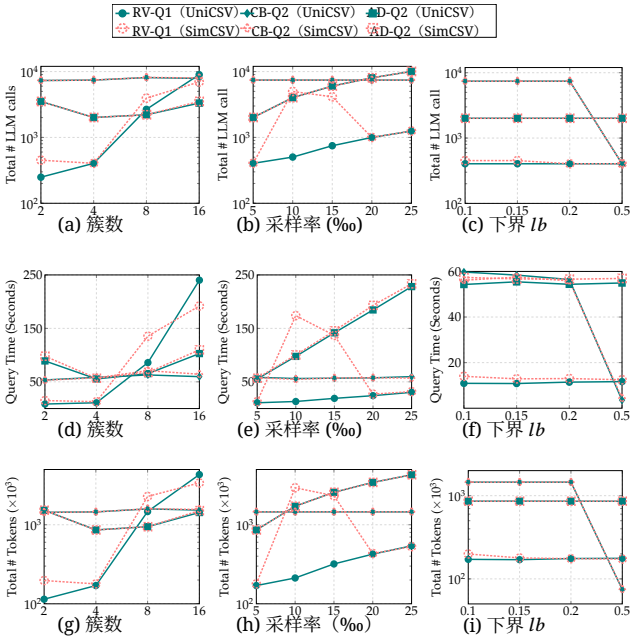


图 5：语义过滤器的超参数对查询时间、大语言模型调用次数和令牌用量的影响

采样率达到 10‰ 时，大语言模型调用会出现显著上升。这一激增归因于部分回退到线性处理，即便在单个簇内部也会发生。与 UniCSV 不同，SimCSV 采用更具适应性的投票策略，当共识较弱时，即使在内部连贯聚类中也可能触发更多大语言模型查询。这说明 SimCSV 对簇内分歧较为敏感，尤其是当采样密度超过揭示模糊案例的阈值时。

(3) 下界。 默认情况下，我们设置 $ub = 1 - lb$。当 lb 不超过 0.2 时，大语言模型调用次数、准确率和 F1 分数保持稳定。一旦 lb 达到 0.5——即无论置信度如何都强制投票——大语言模型调用次数便会大幅减少，例如在 CB-Q2 上，因为该 lb 值避免了线性处理并利用了聚类。此外，对于 CB-Q2 查询，将 lb 设为 0.5，即规定仅根据共识进行投票，而不考虑置信度，最终得到的准确率和 F1 分数均高于基线方法。这表明，在直接 LLM 推理不可靠的情况下（即对于 CB-Q2 查询），利用基于聚类的共识可以使过滤性能超越传统方法。结果表明，在这些参数的广泛取值范围内，准确率和 F1 分数都相对稳定，体现了该算法对采样率和投票阈值中等幅度波动的稳健性。

4.4 重新聚类的影响

我们通过消融实验评估递归重聚类机制的影响，比较启用与不启用重新聚类时的有效性指标（准确率和F1分数）以及效率指标（大语言模型调用次数和重新聚类时间）。在启用重新聚类时，我们使用默认 $lb = 0.15$。当重新聚类

表 4：重新聚类消融实验

	方法	RV-Q1	RV-Q3	CB-Q1	CB-Q2	CB-Q3	AD-Q2
Acc.	UniCSV（不含 RC）	0.9229	0.9939	0.9677	0.7475	0.6403	0.9243
	SimCSV（不含 RC）	0.9230	0.9939	0.9677	0.7469	0.6403	0.9240
	UniCSV（含 RC）	0.9301	0.9527	0.9673	0.7813	0.7090	0.9193
	SimCSV（使用 RC）	0.9305	0.9398	0.9671	0.7813	0.7376	0.9193
F1	UniCSV（不使用 RC）	0.9272	0	0.0619	0.7010	0.5302	0.8347
	SimCSV（不使用 RC）	0.9273	0	0.0619	0.7003	0.5302	0.8345
	UniCSV（使用 RC）	0.9336	0.0880	0.0623	0.7601	0.6340	0.8249
	SimCSV（使用 RC）	0.9335	0.0906	0.0679	0.7555	0.6589	0.8249
# 调用次数	UniCSV（不含 RC）	404	404	404	404	404	2008
	SimCSV（不含 RC）	404	404	404	404	404	2008
	UniCSV（含 RC）	404	22760	404	7423	5196	2008
	SimCSV（含 RC）	404	29683	404	7378	7586	2008
RC 时间 (%)	UniCSV（带 RC）	0	0.44	0	1.47	3.27	0
	SimCSV（带 RC）	0	0.29	0	1.37	2.05	0

表 5：理论与实践之间的差距

ϵ	UniCSV Est.	UniCSV Err.	UniCSV ξ（‰）	SimCSV Est.	SimCSV Err.	SimCSV ξ(‰)
0.10	0.9991	0.0628	13.0	0.9980	0.0622	26.4
0.15	0.9997	0.0633	6.4	0.9997	0.0635	12.7
0.20	0.9997	0.0636	3.9	0.9997	0.0637	7.8
0.25	0.9999	0.0637	2.7	0.9999	0.0635	5.4
0.30	0.9999	0.0635	2.0	0.9999	0.0637	4.0

被禁用时，我们将 $lb = ub = 0.5$固定。该配置迫使算法无论置信度如何都做出投票决策，从而绕过递归重聚类机制。结果汇总于表4。我们在所评估的查询中观察到两种截然不同的行为模式。对于查询RV-Q1、CB-Q1、AD-Q2，重新聚类未被触发，两种配置的性能几乎相同。在RV-Q1和AD-Q2中，初始聚类产生了足够纯净的纯簇，投票置信度始终超过阈值边界，因此额外的大语言模型调用和运行时间开销均为零。在CB-Q1中，低选择性导致的采样失败造成了过度自信的投票结果，重新聚类未被触发。当对CB-Q1使用$lb = 0.01$时，将触发重新聚类，准确率得到提升。相比之下，当禁用重新聚类时，查询RV-Q3、CB-Q2和CB-Q3的性能显著下降。在CB-Q3中，准确率最多下降9.7%，F1 分数下降超过12%。此外，启用重新聚类时，大语言模型调用次数显著增加，表明这些情况下相当一部分元组需要更细粒度的簇细化。这凸显了该机制在纠正低置信度分配、并在初始聚类无法清晰分离语义类别的查询中维持预测质量方面的关键作用。重要的是，尽管大语言模型调用次数增加，重新聚类的计算开销仍然不大。即使在要求最高的场景中，重新聚类也仅占总运行时间的不到3.3%（CB-Q3），表明迭代细化过程相对于占主导的LLM推理成本所增加的延迟可以忽略不计。

4.5 理论分析

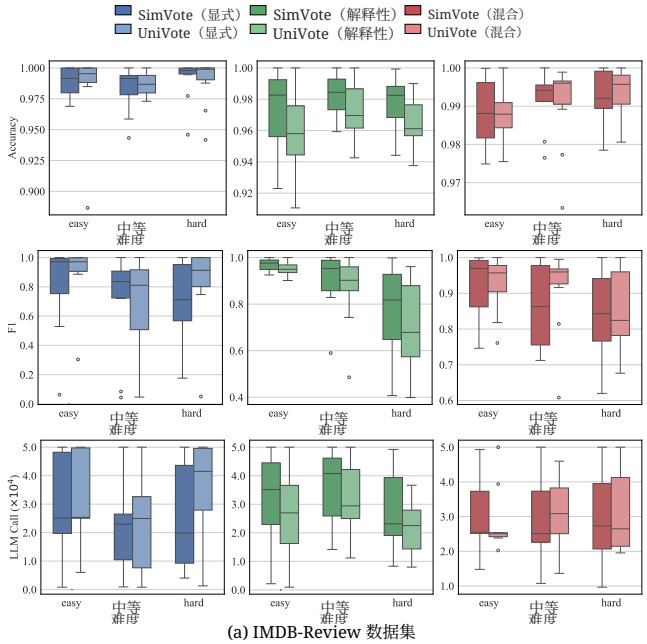
为探究理论与实践之间的差距，我们使用投票算法 UniCSV 和 SimCSV 分析了 RV 上一个具有代表性的簇。该簇包含 14,608 个嵌入，其置信区间测得为 0.9942。具体而言，我们改变理论误差界 ϵ，根据定理 3.3 和 3.6 计算所需的采样比例 ξ，随后报告经验投票输出（记为 Est.），以及大语言模型推断输出与投票输出之间的误差（记为 Err.）。具体而言，Est. 是满足 $f(t_i) = x_i$ 的采样 x_i 所占的比例，而 Err. 是采样实例与所有实例之间的差异。默认设置 $lb = 0.15$ 和 $ub = 1 - lb$，并设置 $l = 0.9996$，即失败概率小于 0.579%。

结果如表 5 所示。对于 UniVote 和 SimVote，实际误差远小于理论保证误差；增大理论误差界 ϵ 可显著降低所需的采样比例。因此，这表明较小的采样比例足以保证较强的理论误差界，同时保持较高的实证有效性。此外，尽管 SimVote 理论上产生的误差界较宽松，但其采样率 ξ 始终高于 UniVote。尽管如此，SimVote 的实证表现仍与 UniVote 相当，这表明即使在放宽理论约束的情况下，其语义投票机制依然稳健。

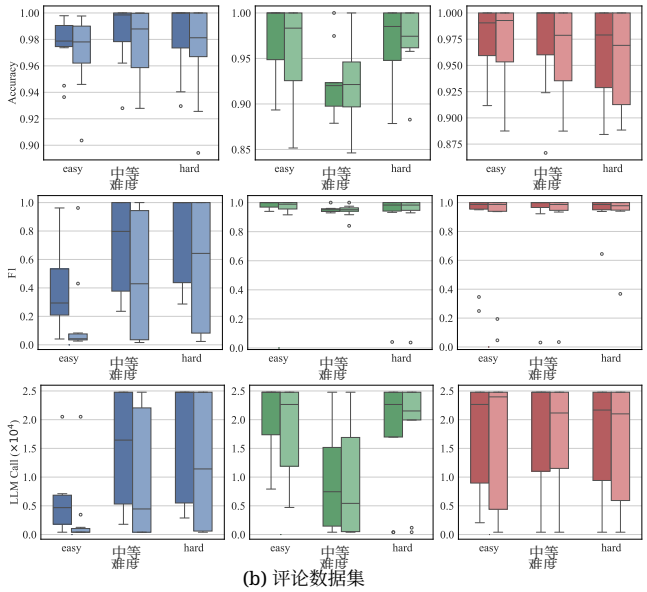
4.6 合成查询测试

为评估 CSV 在更广泛、更多样的谓词空间中的表现，我们构建了一个合成查询集，其中对语义和难度进行可控变化。具体而言，我们指示 GPT 5.2 为两个数据集（IMDB-Review 和 TC）生成三类查询：(1) 明确提及元组文本中实体/概念的查询（显式 谓词）；(2) 表达抽象、主观、解释性语义的查询（解释性 谓词）；(3) 将明确提及与解释性语义相结合的查询（混合 谓词）。每类查询均设置三个难度级别（简单/中等/困难），难度反映语义的微妙程度和语言复杂度。每个（类型、难度）组包含 10 个不同的查询，因此每个数据集有 90 个查询，两个数据集合计 180 个查询。我们通过向 GPT-5.2 提供采样元组和少样本示例来生成这些查询。我们分别在这些合成查询上评估 UniCSV、SimCSV 和线性大语言模型调用基线（Reference）。对于具有较强词汇锚定的谓词（显式和混合），我们采用结合词汇与语义信号（BM25相似度与欧几里得距离的加权和）的混合聚类距离；对于解释性谓词，则采用标准欧几里得距离。我们使用 Reference 的结果作为真实值的代理。

图 6 展示了跨数据集、查询类型和难度级别的准确率、F1 分数以及大语言模型调用次数的箱线图。总体而言，UniCSV 和 SimCSV 在合成数据集上实现了较高的准确率，同时始终比基线 Reference 减少大语言模型调用次数。我们还观察到一些细致的失效模式，这有助于刻画我们方法的普适性。对于困难的解释性/混合谓词，相较于简单/中等难度的谓词，F1 分数有所下降；初步分析表明，这是由极低的选择性造成的，因为真阳性样本极少，导致随机样本很难包含足够的正例来进行可靠投票。在此类情况下，由于标签偏斜（真阴性样本较多），准确率仍可能保持较高水平，而 F1 分数则揭示出少数正例的遗漏。类似现象也出现在显式谓词中，此时 CSV 需要更多的大语言模型调用才能达到可比的 F1 分数。BM25 与嵌入距离的混合提升了聚类质量



(a) IMDB-Review 数据集



(b) 评论数据集

图 6：UniCSV 与 SimCSV 在合成查询上的表现

这些谓词的聚类质量。总之，这些结果验证了 CSV 支持更广泛的谓词空间，同时揭示了我们方法的一个明显局限区间。

5 相关工作

基于机器学习的一般分析型查询处理。随着深度学习的成功，视频数据库管理系统（VDBMS） [4, 17, 18, 20, 36, 47] 应运而生，专为视频数据分析而设计和开发。在这些系统中，深度学习模型作为数据库管理系统中的用户定义函数，用于图像分类、目标检测等。面向视频分析查询，这些系统采用了广泛的优化技术，包括对预处理和查询流水线的端到端优化 [20], 、声明式查询提示 [36],、细粒度模型选择 [7],、查询无关语义索引 [4], ，以及针对特定算子的快速评估算法 [4, 17]。除了视频分析之外，概率谓词 [12, 29, 48] 也与机器学习模型结合，用于更广泛的应用场景；在这些场景中，会构建成本更低的代理模型，并利用它们之间的相关性。针对机器学习与关系算子的混合查询，RAVEN [32] 会对机器学习查询和关系查询进行交叉优化，利用数据统计信息在多次执行中进行选择。此外，一些系统还采用自适应查询处理，根据运行时统计信息动态调整查询计划和资源分配，或选择成本更低的代理模型。ABAE [19], 和InQuest [37]则通过引入基于代理模型的采样聚合算法，专注于优化采用机器学习模型的聚合查询。

基于大语言模型的语义查询处理。 大语言模型为结构化数据 [13, 26] [3, 39, 43] [39], [24], 和非结构化数据开启了一种新的查询范式，即语义查询处理。DocETL、ZenDB、DocWanger [40], Aryn [3] 和 Unify [43] 利用大语言模型，通过语义查询对非结构化文本进行分析，其查询语言涵盖从扩展 SQL 方言 [24], 声明式操作 [39] 到自然语言 [3, 40, 43]。在这些文档分析系统中，大语言模型也被用作查询评估过程中的优化代理，例如查询规划 [3], 查询改写、提示优化和操作分解 [40]。UQE [10] 通过允许自然语言连接和选择谓词来扩展 SQL，从而本质上支持由大语言模型实现的语义连接和语义过滤器。Palimpzest [25] 是一个由大语言模型驱动的分析系统，通过逻辑优化和物理优化来降低大语言模型调用的查询时间和财务成本。Abacus [38] 是构建在 Palimpzest 之上的优化器，它将帕累托优化算法与成本估算集成到查询规划中。Lotus [33] 在表中形式化了基于大语言模型的语义操作。此外，Lotus 针对高开销的语义操作提出了优化算法，采用批量推理和大语言模型级联。BARGAIN [49] 采用与 Lotus 相同的模型级联框架。除准确率之外，它还为语义操作的召回率、精确率和准确率提供可证明的界限。 [2] 评估了配备 pgAI 的 PostgreSQL 以及配备 FlockMTL 的 DuckDB 的语义查询性能 [13], , 突出了强制结构化输出、批处理和查询规划方面的挑战。在这些系统中，语义过滤器是基本算子。

6 结论

本文提出了一种高效且可扩展的语义过滤方法——聚类-采样-投票（CSV），旨在解决将大语言模型集成到数据系统中的核心计算挑战。通过对相似元组进行聚类、采样代表元组，并通过具有理论保证的投票推断标签，CSV 在保持相当有效性的同时，减少了冗余的大语言模型调用。大量实验验证了我们提出的 CSV 的有效性：与 Reference 相比，大语言模型调用次数减少至 1.28-200×；与 Lotus 相比，减少至 1.81-355× 。

参考文献

[1] 2023。LiteLLM：统一的大语言模型 API。https://github.com/BerriAI/litellm。 [2] Kerem Akillioglu、Anurag Chakraborty、Sairaj Voruganti 和 M. Tamer Özsu。2025年。面向大语言模型查询的关系数据库管理系统研究挑战。arXiv:2508.20912 [cs.DB] [3] Eric Anderson、Jonathan Fritz、Austin Lee、Bohou Li、Mark Lindblad、Henry Lindeman、Alex Meyer、Parth Parmar、Tanvi Ranade、Mehul A. Shah、Benjamin Sowell、Dan Tecuci、Vinayak Thapliyal 和 Matt Welsh。2024年。大语言模型驱动的非结构化分析系统设计。CoRR abs/2409.00847（2024年）。arXiv:2409.00847 [4] Jaeho Bang、Gaurav Tarlok Kakkar、Pramod Chunduri、Subrata Mitra 和 Joy Arulraj。2023年。SEIDEN：重新审视视频数据库系统中的查询处理。VLDB基金会会刊 16，9（2023年），2289–2301。 [5] Rémi Bardenet 和 Odalric-Ambrym Maillard。2015年。不放回抽样的集中不等式。（2015年）。 [6] Asim Biswal、Liana Patel、Siddarth Jha、Amog Kamsetty、Shu Liu、Joseph E Gonzalez、Carlos Guestrin 和 Matei Zaharia。2024年。仅有文本转SQL还不够：使用 TAG 统一人工智能与数据库。arXiv 预印本 arXiv:2408.14717（2024年）。 [7] Jiashen Cao、Karan Sarkar、Ramyad Hadidi、Joy Arulraj 和 Hyesoon Kim。2022年。FiGO：视频分析中的细粒度查询优化。载于 SIGMOD。ACM，559–572。 [8] Ilias Chalkidis、Abhik Jana、Dirk Hartung、Michael J. Bommarito II、Ion An- droutsopoulos、Daniel Martin Katz 和 Nikolaos Aletras。2022年。LexGLUE：英语法律语言理解基准数据集。载于 ACL会议论文集 2022。计算语言学协会，4310–4330。 [9] Haoang Chi、He Li、Wenjing Yang、Feng Liu、Long Lan、Xiaoguang Ren、Tongliang Liu 和 Bo Han。2024年。揭示大语言模型中的因果推理：现实还是幻象？神经信息处理系统进展 37（2024年），96640–96670。 [10] Hanjun Dai、Bethany Wang、Xingchen Wan、Bo Dai、Sherry Yang、Azade Nova、Pengcheng Yin、Mangpo Phothilimthana、Charles Sutton 和 Dale Schuurmans。2024年。UQE：面向非结构化数据库的查询引擎。神经信息处理系统进展 37（2024年），29807–29838。 [11] Thomas Davidson、Dana Warmsley、Michael W. Macy 和 Ingmar Weber。2017年。自动化仇恨言论检测与冒犯性语言问题。载于 ICWSM。AAAI出版社，512–515。 [12] Dujian Ding、Sihem Amer-Yahia 和 Laks V. S. Lakshmanan。2022年。关于机器学习模型上的高效近似查询。VLDB基金会会刊 16，4（2022年），918–931。 [13] Anas Dorbani 、 Sunny Yasser 、 Jimmy Lin 和 Amine Mhedhbi。2025 年。超越 Quacking：将语言模型和检索增强生成深度集成到 DuckDB 中。VLDB基金会会刊 18，12（2025年），5415–5418。 [14] Yarin Gal 和 Zoubin Ghahramani。2016年。作为贝叶斯近似的丢弃法：表示深度学习中的模型不确定性。载于 ICML 2016论文集，第48卷。JMLR.org，1050–1059。 [15] Jiahui Geng、Fengyu Cai、Yuxia Wang、Heinz Koeppl、Preslav Nakov 和 Iryna Gurevych。2024年。大语言模型置信度估计与校准综述。载于 NAACL会议论文集 2024。计算语言学协会，6577–6595。doi:10.18653/V1/2024.NAACL-LONG.366 [16] Karen Sparck Jones、Steve Walker 和 Stephen E. Robertson。2000年。信息检索的概率模型：发展与比较实验——第1部分。《信息处理与管理》36，6（2000年），779–808。 [17] Daniel Kang、Peter Bailis 和 Matei Zaharia。2019年。BlazeIt：针对基于神经网络的视频分析优化声明式聚合查询和限制查询。VLDB基金会会刊 13，4（2019年），533–546。 [18] Daniel Kang、Edward Gan、Peter Bailis、Tatsunori Hashimoto 和 Matei Zaharia。2020年。使用代理进行带保证的近似选择。VLDB基金会会刊 13，11（2020年），1990–2003。 [19] Daniel Kang、John Guibas、Peter Bailis、Tatsunori Hashimoto、Yi Sun 和 Matei Zaharia。2021年。利用高开销谓词加速近似聚合查询。VLDB基金会会刊 14，11（2021年），2341–2354。 [20] Daniel Kang、Ankit Mathur、Teja Veeramacheneni、Peter Bailis 和 Matei Zaharia。2020年。联合优化基于深度神经网络的视觉分析中的预处理与推理。VLDB基金会会刊 14，2（2020年），87–100。 [21] Daniel Kang、Francisco Romero、Peter D. Bailis、Christos Kozyrakis 和 Matei Zaharia。2022年。VIVA：交互式视频分析的端到端系统。载于 CIDR。www.cidrdb.org。 [22] Woosuk Kwon、Zhuohan Li、Siyuan Zhuang、Ying Sheng、Lianmin Zheng、Cody Hao Yu、Joseph Gonzalez、Hao Zhang 和 Ion Stoica。2023年。使用 PagedAttention 实现大语言模型服务的高效内存管理。载于第29届操作系统原理研讨会。611–626。 [23] Jinyang Li、Binyuan Hui、Ge Qu、Jiaxi Yang、Binhua Li、Bowen Li、Bailin Wang、Bowen Qin、Ruiying Geng、Nan Huo 等。2024年。大语言模型已经能够充当数据库接口了吗？面向大规模数据库知识文本转SQL的大型基准测试。神经信息处理系统进展 36（2024年）。 [24] Yiming Lin、Madelon Hulsebos、Ruiying Ma、Shreya Shankar、Sepanta Zeighami、Aditya G. Parameswaran 和 Eugene Wu。2024年。迈向准确高效的大语言模型文档分析。CoRR abs/2405.04674（2024年）。arXiv:2405.04674 [25] Chunwei Liu、Matthew Russo、Michael Cafarella、Lei Cao、Peter Baile Chen、Zui Chen、Michael Franklin、Tim Kraska、Samuel Madden、Rana Shahout 等。2025年。Palimpzest：通过声明式查询处理优化人工智能驱动的分析。载于 CIDR。2. [26] Shu Liu、Asim Biswal、Amog Kamsetty、Audrey Cheng、Luis Gaspar Schroeder、Liana Patel、Shiyi Cao、Xiangxi Mo、Ion Stoica、Joseph E. Gonzalez 和 Matei Zaharia。2025年。优化关系型数据分析工作负载中的大语言模型查询。载于第八届机器学习与系统会议。 [27] Shicheng Liu、Jialiang Xu、Wesley Tjangnaka、Sina J. Semnani、Chen Jie Yu 和 Monica Lam。2024年。SUQL：使用大语言模型在结构化和非结构化数据上进行对话式搜索。载于 NAACL。计算语言学协会，4535–4555。 [28] Xutong Liu、Baran Atalar、Xiangxiang Dai、Jinhang Zuo、Siwei Wang、John Lui、Wei Chen 和 Carlee Joe-Wong。2025年。低成本大语言模型服务的语义缓存：从离线学习到在线适应。arXiv预印本 arXiv:2508.07675（2025年）。 [29] Yao Lu、Aakanksha Chowdhery、Srikanth Kandula 和 Surajit Chaudhuri。2018年。利用概率谓词加速机器学习推理。载于 SIGMOD。ACM，1493–1508。 [30] Andrew L. Maas、Raymond E. Daly、Peter T. Pham、Dan Huang、Andrew Y. Ng 和 Christopher Potts。2011年。用于情感分析的词向量学习。载于第49届计算语言学协会年会：人类语言技术论文集。142–150。 [31] Ramaswami Mohandoss。2024年。面向大语言模型应用的基于上下文的语义缓存。载于2024年IEEE人工智能会议。IEEE，371–376。 [32] Kwanghyun Park、Karla Saur、Dalitso Banda、Rathijit Sen、Matteo Interlandi 和 Konstantinos Karanasos。2022年。机器学习预测查询的端到端优化。载于 SIGMOD。ACM，587–601。 [33] Liana Patel、Siddharth Jha、Melissa Z. Pan、Harshit Gupta、Parth Asawa、Carlos Guestrin 和 Matei Zaharia。2025年。语义算子及其优化：迈向具有准确率保证的基于人工智能的数据分析。VLDB基金会会刊 18，11（2025年），4171–4184。 [34] Fabian Pedregosa、Gaël Varoquaux、Alexandre Gramfort、Vincent Michel、Bertrand Thirion、Olivier Grisel、Mathieu Blondel、Peter Prettenhofer、Ron Weiss、Vincent Dubourg、Jake VanderPlas、Alexandre Passos、David Cournapeau、Matthieu Brucher、Matthieu Perrot 和 Edouard Duchesnay。2011年。Scikit-learn：Python机器学习工具。机器学习研究期刊 12（2011年），2825–2830。 [35] Sajal Regmi 和 Chetan Phakami Pun。2024年。GPT语义缓存：通过语义嵌入缓存降低大语言模型成本和延迟。arXiv预印本 arXiv:2411.05276（2024年）。 [36] Francisco Romero、Johann Hauswald、Aditi Partap、Daniel Kang、Matei Zaharia 和 Christos Kozyrakis。2022年。通过声明式模型关系优化视频分析。VLDB基金会会刊 16，3（2022年），447–460。 [37] Matthew Russo、Tatsunori Hashimoto、Daniel Kang、Yi Sun 和 Matei Zaharia。2023年。加速非结构化数据流上的聚合查询。VLDB基金会会刊 16，11（2023年），2897–2910。 [38] Matthew Russo、Sivaprasad Sudhir、Gerardo Vitagliano、Chunwei Liu、Tim Kraska、Samuel Madden 和 Michael Cafarella。2025年。Abacus：面向语义算子系统的基于成本的优化器。arXiv预印本 arXiv:2505.14661（2025年）。 [39] Shreya Shankar、Tristan Chambers、Tarak Shah、Aditya G. Parameswaran 和 Eugene Wu。2025年。DocETL：面向复杂文档处理的智能体式查询重写与评估。VLDB基金会会刊 18，9（2025年），3035–3048。 [40] Shreya Shankar、Bhavya Chopra、Mawil Hasan、Stephen Lee、Björn Hartmann、Joseph M. Hellerstein、Aditya G. Parameswaran 和 Eugene Wu。2025年。使用 DocWrangler 引导语义数据处理。CoRR abs/2504.14764（2025年）。arXiv:2504.14764 [41] James Thorne、Andreas Vlachos、Christos Christodoulopoulos 和 Arpit Mittal。2018年。FEVER：用于事实抽取与验证的大规模数据集。载于 NAACL-HLT。计算语言学协会，809–819。 [42] Hugo Touvron、Thibaut Lavril、Gautier Izacard、Xavier Martinet、Marie-Anne Lachaux、Timothée Lacroix、Baptiste Rozière、Naman Goyal、Eric Hambro、Faisal Azhar 等。2023年。Llama：开放且高效的基础语言模型。arXiv预印本 arXiv:2302.13971（2023年）。 [43] Jiayi Wang 和 Jianhua Feng。2025年。Unify：非结构化数据分析系统。载于 ICDE。IEEE，4662–4674。 [44] Liang Wang、Nan Yang、Xiaolong Huang、Binxing Jiao、Linjun Yang、Daxin Jiang、Rangan Majumder 和 Furu Wei。2022年。通过弱监督对比预训练生成文本嵌入。arXiv预印本 arXiv:2212.03533（2022年）。 [45] Wei Wei、Quoc Le、Andrew Dai 和 Jia Li。2018年。AirDialogue：目标导向对话研究环境。载于2018年自然语言处理实证方法会议论文集。3844–3854。 [46] Shitao Xiao、Zheng Liu、Peitian Zhang 和 Niklas Muennighoff。2023年。C-Pack：推进通用中文嵌入的资源包。arXiv: 2309 07597 [ ]。cs.CL Ziyang Xiao、Dongxiang Zhang、Zepeng Li、[47] Sai Wu、Kian-Lee Tan 和 Gang Chen。2023年。DoveDB：声明式低延迟视频数据库。VLDB基金会会刊 16，12（2023年），3906–3909。

[48] Zhihui Yang、Zuozhi Wang、Yicong Huang、Yao Lu、Chen Li 和 X.Sean Wang。2022年。利用相关代理模型优化机器学习推理查询。*VLDB*基金会会刊15，10（2022年），2032–2044。[49] Sepanta Zeighami、Shreya Shankar 和 Aditya G. Parameswaran。2025年。降低成本，而非准确率：具有保证的大语言模型驱动的数据处理。*ACM*数据管理会刊3，6（2025年），1–26。doi:10.1145/3769776

[50] Jingqing Zhang、Yao Zhao、Mohammad Saleh 和 Peter Liu。2020。Pegasus：使用抽取的间隔句进行预训练以实现抽象式摘要。载于国际机器学习会议。PMLR，11328–11339。[51] Yanzhao Zhang、Mingxin Li、Dingkun Long、Xin Zhang、Huan Lin、Baosong Yang、Pengjun Xie、An Yang、Dayiheng Liu、Junyang Lin、Fei Huang 和 Jingren Zhou。2025年。Qwen3 Embedding：通过基础模型推进文本嵌入与重排序。*arXiv* 预印本 *arXiv:2506.05176*（2025年）。

推论 3.4 的证明

推论 A.1. 设 $X=\{x_1, ..., x_n\}$ 为一个规模为 n 的有限总体，其中 $x_1, \cdots, x_n$ 为均值为 μ 的二元变量。设 $c_1, ..., c_k$ 从 X 中无放回抽样得到，且每个 c_i 关联一个权重 w_i，其中 $w_i \geq 0$ 且 $\sum_{i=1}^{k} w_i = 1$。设 $\hat{\mu}$ 为 $c_1, \cdots, c_k$ 的均值，即 $\hat{\mu} = \frac{1}{k}\sum_{i=1}^{k} w_i c_i$。对于任意 $\epsilon > 0$，我们有

$$Pr[|\hat{\mu}_w - \mu| \geq \epsilon] \leq 2\exp\left(\frac{-3k\epsilon^2}{(6\hat{\sigma}^2 + 2\epsilon)v} \cdot \frac{n-k}{n-1}\right),$$

其中 $\hat{\sigma}^2 = \frac{1}{k-1}\sum_{i=1}^{k}(c_i - \hat{\mu})^2$ 是样本标准差，v 是一个常数，使得 $\max_i w_i \leq \frac{v}{k}$。

证明概要： $\hat{\mu}_w - \mu = \sum_{i=1}^{k} w_i c_i - \sum_{i=1}^{k} w_i \mu = \sum_{i=1}^{k} w_i (c_i - \mu)$。令 $y_i = w_i(c_i - \mu)$，我们有 $\hat{\mu}_w - \mu = \sum_{i=1}^{k} y_i$。根据大数定律，$y_i$ 的均值等于 0，因为 c_i 是从均值为 μ 的 X 中采样得到的。对 y_i 应用引理 3.2，我们有

$$Pr\left[\left|\frac{1}{k}\sum_{i=1}^{k} y_i\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{k\epsilon^2}{2\hat{\sigma}_y^2 + 2R\epsilon/3} \cdot \frac{n-k}{n-1}\right). \quad (6)$$

由于 x_i 的 c_i 独立同分布，$\hat{\sigma}_y^2 = \frac{1}{k-1}\sum_{i=1}^{k}(w_i(c_i-\mu))^2 = \frac{1}{k}\hat{\sigma}^2\sum_{i=1}^{k} w_i^2$。由于 $|y_i| = w_i(c_i - \mu) \leq 1$，$R \leq \max_i w_i \leq \frac{v}{k}$，$Pr[|\hat{\mu}_w - \mu| \geq \epsilon] = Pr\left[\left|\sum_{i=1}^{k} y_i\right| \geq k\epsilon\right]$，且 $\sum_{i=1}^{k} w_i^2 \leq v/k$，根据公式（6），我们有

$$Pr[|\hat{\mu}_w - \mu| \geq \epsilon] = Pr\left[\left|\sum_{i=1}^{k} y_i\right| \geq \epsilon\right] = Pr\left[\left|\frac{1}{k}\sum_{i=1}^{k} y_i\right| \geq \frac{\epsilon}{k}\right]$$

$$\leq 2\exp\left(-\frac{k(\epsilon/k)^2}{2\hat{\sigma}_y^2 + 2\epsilon/3k} \cdot \frac{n-k}{n-1}\right) = 2\exp\left(\frac{-3k\epsilon^2}{(6\hat{\sigma}^2 + 2\epsilon)v} \cdot \frac{n-k}{n-1}\right).$$

□

B 嵌入模型与大语言模型的影响

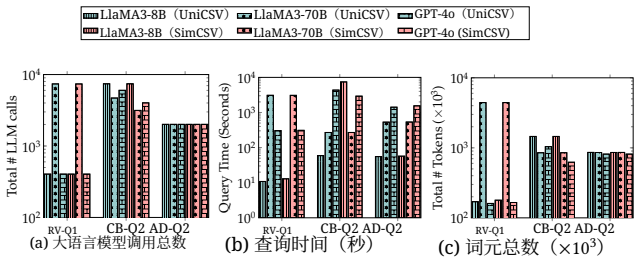
（1）不同嵌入模型的影响。我们通过在所有数据集上比较 BGE-Large-en（BGE） [46], Qwen-0.6B（通义千问） [51], 以及默认的 E5-Large（E5）[44] 模型，研究嵌入模型如何影响 UniCSV 和 SimCSV，结果见表 7 和图 8。就以准确率和 F1分数衡量的有效性而言，差异相对较小。BGE 在 RV-Q1 上略优，但在 AD-Q2 上表现略差。这些结果表明，嵌入质量对最终结果质量的影响有限。效率指标则显示出更为显著的差异。由于 BGE 的嵌入生成的连贯聚类较少，需要更多大语言模型查询才能解决，因此它在 RV-Q1 和 AD-Q2 上都会导致资源利用率上升。通义千问在 RV-Q1 上表现欠佳。尽管如此，与基线方法相比，所有嵌入模型仍保持显著更高的效率，加速比介于 1.1× 到 18.95×之间，这表明即使嵌入模型不够理想，投票策略依然稳健。

（2）不同骨干模型的影响。我们评估了三种骨干大语言模型的性能：LlaMA3-8B、量化为 int4 的 LlaMA3-70B，以及 GPT-4o。LlaMA 模型通过本地 vLLM 引擎部署，而 GPT-4o 则通过远程调用使用。

结果详见表6和图7。三种模型在效率和有效性指标方面表现相当，这表明 UniCSV 和 SimCSV 对模型选择并不过分敏感。LlaMA3-70B 在 RV-Q1 上表现出色，说明其更大的容量使其能够在复杂情形下进行更精细的语义辨别。GPT-4o 在 CB-Q2 上优势显著，准确率比其他模型高约 10%。这一提升可能源于 GPT-4o 在处理复杂推理任务方面的出色能力。然而，GPT-4o 并非总是占优；许多查询用较小的模型就能充分解决，而且使用 GPT-4o 有时会触发内容管理限制。在效率方面，CB-Q2 和 AD-Q2 的大语言模型调用量与令牌用量总体保持一致。在 RV-Q1 上，LlaMA3-70B 的调用次数更多，反映出它对细微语义边界的敏感性。尽管如此，其速度仍比基线方法快 6.76× 倍。不同模型的查询延迟各不相同，这是由模型规模、服务器吞吐量和访问限制等因素造成的。

表6：不同大语言模型骨干网络的语义过滤器

算法	模型	RV-Q1 准确率	RV-Q1 F1	CB-Q2 Acc.	CB-Q2 F1	AD-Q2 准确率	AD-Q2 F1
UniCSV	LlaMa-8B	0.9307	0.9336	0.7799	0.7517	0.9190	0.8248
	LlaMa-70B	0.9520	0.9523	0.7715	0.8079	0.9194	0.8251
	GPT-4o	0.9306	0.9335	0.8870	0.8814	0.9194	0.8250
SimCSV	LlaMa-8B	0.9305	0.9335	0.7803	0.6988	0.9193	0.8249
	LlaMa-70B	0.9521	0.9524	0.7701	0.8068	0.9194	0.8251
	GPT-4o	0.9305	0.9335	0.8666	0.8687	0.9193	0.8250

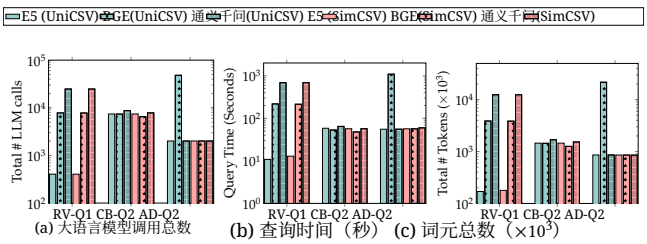


(a) 大语言模型调用总数 (b) 查询时间（秒） (c) 词元总数（$\times 10^3$）

图 7：不同大语言模型骨干网络的比较

表 7：采用不同嵌入模型的语义过滤器

算法。	模型	RV-Q1 准确率	RV-Q1 F1	CB-Q2 准确率	CB-Q2 F1	AD-Q2 准确率	AD-Q2 F1
UniCSV	E5	0.9307	0.9336	0.7852	0.7601	0.9192	0.8248
	BGE	0.9527	0.9531	0.7827	0.7594	0.9122	0.8036
	Qwen	0.9227	0.9235	0.7777	0.7540	0.9172	0.8192
SimCSV	E5	0.9305	0.9335	0.7815	0.7555	0.9193	0.8249
	BGE	0.9520	0.9524	0.7806	0.7587	0.8947	0.7755
	Qwen	0.9223	0.9231	0.7794	0.7500	0.9172	0.8192



(a) 大语言模型调用总数 (b) 查询时间（秒） (c) 词元总数（$\times 10^3$）

图 8：不同嵌入模型的比较