arXiv:2601.11863v1 [cs.IR] 17 Jan 2026

利用元数据改进检索增强生成

Raquib BinYousuf[1 [0009−0001−1245−0450]]、Shengzhe Xu[1 [0009−0008−8589−6526]]、Mandar Sharma[1 [0000−0002−7012−9323]]、Andrew Neeser[1]、Chris Latimer[2]和 Naren Ramakrishnan[1 [0000−0002−1821−9743]]

[1] 弗吉尼亚理工大学，弗吉尼亚州，美国 {raquib, shengzx, mandarsharma, aneeser24, naren}@vt.edu [2] Vectorize.io，科罗拉多州，美国 chris.latimer@vectorize.io

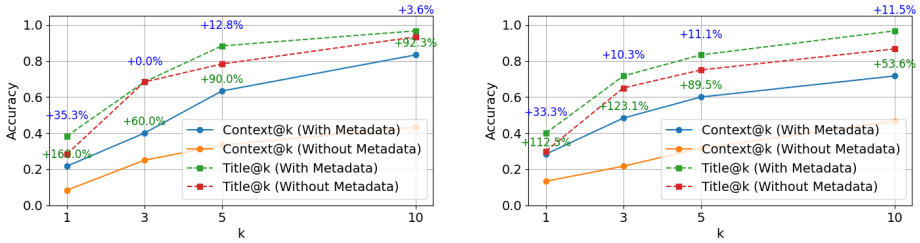
摘要。 检索增强生成系统依赖检索语义相关的文档块，以支持大语言模型生成准确且有依据的输出。在监管申报文件等结构化且重复性高的语料中，仅凭文本块相似度往往无法区分语言重叠的文档。实践者常将元数据展平并纳入输入文本，作为一种启发式方法，但这种做法的影响和权衡仍未得到充分理解。我们对元数据感知检索策略进行了系统研究，将纯文本基线与直接嵌入元数据的方法进行比较。评估涵盖元数据即文本（前缀和后缀）、将元数据和内容融合到单一索引中的双编码器统一嵌入、双编码器后期融合检索，以及元数据感知查询改写。通过多种检索指标和问题类型进行评估后，我们发现，添加前缀和统一嵌入始终优于纯文本基线；统一嵌入有时甚至优于添加前缀的方法，同时也更易于维护。除实证比较外，我们还分析了嵌入空间，发现整合元数据能够提高检索效果：增强文档内聚性、减少文档间混淆，并拉大相关与不相关文本块之间的距离。字段级消融实验表明，结构线索能够提供强有力的消歧信号。我们的代码、评估框架和 RAGMATE-10K 数据集已公开托管[3]。

关键词： 检索增强生成（RAG）· 元数据感知检索 · 稠密检索 · 查询改写 · 基准数据集

1 引言

大语言模型（LLMs）已成为信息获取的核心，使系统能够回答问题、总结文档，并对

[3] https://github.com/raquibvt/RAGMate



(a) 一般问题 (b) 深入问题

图 1：在不同查询类型下，使用和不使用元数据时检索中的 Context@K 和 Title@K 指标改进（双编码器统一嵌入）

非结构化语料库 [29, 4]进行推理。检索增强生成（RAG）架构通过先检索相关上下文，再基于这些上下文生成内容来扩展这些模型 [22]。这种混合设置已成为开放域问答、长文档理解和特定领域推理的主流范式 [15, 11]。

然而，RAG 流水线的可靠性仅与其所检索到的上下文相当。当文档语料库具有重复性或语义歧义时，例如监管申报文件、法律记录或科学论文，检索器往往会返回表面上相关但实质上无助的文本块 [15, 28]。在这些情况下，文档结构或实体上下文中的细微差异（例如公司名称、财年或章节标题）对于检索正确的信息至关重要。

一个特别典型的例子是美国证券交易委员会（SEC）的 10-K 表格申报文件。这些文档遵循固定模板，在不同公司和年份之间重复使用相同表述，并包含细微且依赖于结构的变化。这些特征对传统的基于向量的检索构成了挑战，因为这种检索依赖块级语义嵌入 [12, 15]，而块级语义嵌入往往缺乏足够的区分能力。许多消歧信号已经存在于结构化元数据中，例如申报年份、表格类型、章节标题和行业分类，但这些字段通常仅用于检索后过滤 [35]。

例如，考虑以下查询：“该公司识别出哪些与供应链中断相关的风险？”如果没有元数据，不同公司、年份和章节中的许多文本块都包含几乎相同的表述，导致仅基于语义的检索器返回看似合理但不正确的上下文。公司、财年和章节标题等元数据可以消除查询意图的歧义，将检索锚定到正确的文件和章节。

因此，美国证券交易委员会文件已成为使用大语言模型进行金融问答的常见测试平台 [19, 20, 7],，并与更广泛的金融基准并列 [31, 10]。这些研究共同凸显了 RAG 流水线在重复性领域专用语料库上的机遇与局限性。受近期上下文微调和长上下文提示研究的启发 [1, 5],，我们提出以下问题：

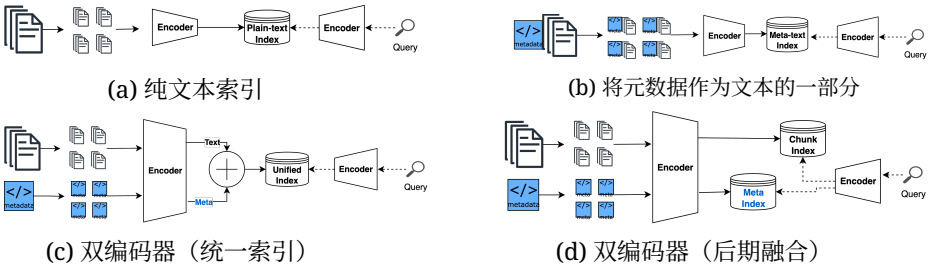


图 2：文档与元数据嵌入检索策略的概念概览。虚线箭头表示查询传播。

能否将元数据作为 *RAG* 流水线中的一等输入，而不仅仅是过滤器，还将其作为嵌入式内容，以提升检索质量和下游答案的正确性？

我们研究了元数据感知检索是否能从更模块化的设计中受益，例如将内容和元数据分开嵌入的双编码器，或明确呈现元数据线索的查询改写。作为一种简单的基线，我们还评估了直接将元数据作为文本添加到文本块中的做法。具体而言，在文本块前添加元数据可以获得较高的检索准确率，但计算成本较高，因为任何元数据更新都需要对整个索引重新嵌入。为减轻维护负担，我们提出了一种双编码器框架，其中元数据和内容分别嵌入，然后再进行组合。元数据嵌入只需针对每个字段计算一次，并可与预先计算好的文本嵌入融合，从而避免对完整语料库进行代价高昂的重新嵌入。在这些变体中，统一嵌入将元数据和内容向量相加并纳入单一索引，其准确率与在文本块前添加元数据相当，有时甚至更高，同时简化了在线服务。相比之下，后期融合评分的竞争力较弱。

除了实证比较之外，我们还分析了元数据集成为何能改善检索。通过对嵌入空间进行统计分析，我们表明，元数据能够提高文档内聚合度、减少文档间混淆，并扩大相似度分数方差，从而形成更具判别性的检索几何结构。这项分析证实，元数据并非只是增加词元，而是会重塑嵌入景观，提升相关与不相关文本块之间的可分性。

我们将贡献总结如下：

– 据我们所知，这是首项系统性研究，表明元数据信号能够显著提升检索增强生成系统中的检索效果。– 提出一种双编码器框架，分别对元数据和内容进行嵌入，在提升索引维护效率的同时，取得与前缀方法相当、有时甚至更好的准确率。

– 通过统计性的嵌入空间分析解释这些提升，证实文档内聚合度提高、文档间混淆减少，且相关与不相关文本块之间的区分更加清晰。– 字段级消融实验表明，公司和年份字段能够提供强有力的消歧信号，而章节标题带来的提升则较为有限。– 发布 RAGMATE-10K，这是一个富元数据基准，包含基于文本块的问答，可用于可复现的评估。

我们回顾相关工作（§2），介绍 metadata-as-text 和双编码器设计（§3.2），描述 RAGMATE-10K 数据集及其设置（ §4 ），呈现结果（ §5 ），分析嵌入几何（ §5.1），报告针对章节标题的消融实验（§6），最后总结局限性和未来工作。

2 相关工作

信息获取长期以来一直推动着从搜索引擎 [27, 8]到各种下游应用 [17, 24]的技术发展。注意力机制模型 [30, 12] 和现代基于聊天的大语言模型 [29, 4] 的出现，使检索增强生成（RAG） [22] 成为一种核心范式。独立运行的大语言模型往往会产生幻觉，这在医学 [33], 、法律 [28],、情报分析 [32], 和软件工程 [21], 等高风险领域尤为棘手；RAG 通过将生成结果建立在外部知识之上来缓解这一问题。相关方法既包括微调检索器 [25, 5] ，也包括目前占主导地位的无需重新训练的流水线 [15,11]。

在检索本身，不同方法对知识的表示方式各不相同：有些方法对原始文档块进行嵌入 [15], 有些使用（问题、答案）对 [35],还有些则生成合成查询或摘要，以改善匹配效果 [14, 6, 26]。这些方法涵盖稠密嵌入和稀疏的基于关键词的检索。近期研究（如 EnrichIndex [9]）进一步探索基于离线 LLM 的增强方法，通过生成摘要、用途和合成问答对来强化第一阶段检索，同时避免在线调用 LLM 的成本。

大多数既有研究仅间接利用元数据，要么将其用于过滤，要么通过后期融合信号加以利用。FinanceBench [19],、SEC-QA [20], 和 SecQue [7], 等金融问答基准，以及更广泛的金融领域数据集 [31, 10],，凸显了重复性强的领域专用语料库所带来的挑战，但尚未系统评估元数据集成。尽管近期有些方法利用 LLM 生成的额外语义上下文来丰富文档，但并未单独考察结构化元数据作为检索信号的作用。我们的工作有所不同：我们将元数据视为一等检索信号，并比较简单的“元数据即文本”策略与模块化双编码器设计。

3 元数据作为一等信号

我们从最简单的元数据集成方法开始：在嵌入之前，将结构化元数据字段直接拼接到文本块中。这

种技术将元数据视为语义输入的一部分（元数据即文本，MaT），并将其编码到与内容相同的向量空间中。

3.1 将元数据作为文本：最简基线

设语料库为一组带有相关结构化元数据的 N 文档块：

$$D \;=\; \{(m_i, c_i)\}_{i=1}^{N},$$

其中， c_i 是第 i个文本块的文本， m_i 是结构化字段的键值映射（例如，公司、表格_类型、章节、年份）。

我们定义一个序列化函数，用于根据元数据生成简洁易读的标头：

$$s(m_i) \;=\; \text{“}company : \{...\}; form : \{...\}; section : \{...\}; year : \{...\}''.$$

利用这个标头，我们通过串接运算符构造带有元数据前缀的文本块字符串：

$$\tilde{c}_i \;=\; \texttt{concat}\big(s(m_i),\, c_i\big).$$

我们还评估了一种后缀变体，即将元数据标头追加到文本块之后： $\tilde{c}_i^{\text{suf}} = \texttt{concat}\big(c_i,\;\; s(m_i)\big)$。我们在表 2 中报告前缀（$\tilde{c}_i^{\text{pre}} = \texttt{concat}\big(s(m_i)\;\; c_i\big)$）和后缀两种方式的结果，而图表使用前缀变体，除非另有明确说明。

现成的文本编码器 f_θ （冻结）将字符串映射为 d 维向量：

$$\tilde{\mathbf{e}}_i \;=\; f_\theta(\tilde{c}_i) \in \mathbb{R}^d.$$

给定用户查询 q，我们计算其嵌入 $\mathbf{e}_q = f_\theta(q)$，并在单个 MaT索引上依据余弦相似度进行检索：

$$\text{Score}_{\text{MaT}}(q, i) \;=\; \cos\big(\mathbf{e}_q,\, \tilde{\mathbf{e}}_i\big), \quad \text{rank by Score}_{\text{MaT}}.$$

3.2 双编码器：模块化集成

将元数据展平并纳入文本块文本中（第 3.1 节）可以提升检索效果，但计算成本较高，因为任何元数据更新都需要对整个文本块索引重新生成嵌入。为此，我们设计了双编码器方法，分别对内容和元数据进行嵌入，使更新更加轻量且模块化。在此框架下，我们首先介绍一种统一单索引方案，直接在嵌入空间中合并两种信号，在保持在线服务简洁性的同时避免高成本的重新索引。接着，我们将其与一种在查询时合并分数的后期融合双编码器进行比较，最后介绍能在查询中凸显元数据线索的查询侧策略。

通过加权和融合实现统一单索引 设语料库为 $D = \{(m_i,\ c_i)\}_{i=1}^{N}$，其中 c_i 是文本块文本，m_i 是键值元数据映射（例如公司、表格、年份、章节）。我们将内容和元数据编码到同一个d维空间中：

$$\mathbf{e}_i^{\text{text}} = f_\theta^{\text{text}}(c_i), \qquad \mathbf{e}_i^{\text{meta}} = f_\theta^{\text{meta}}(m_i).$$

我们对两个向量进行 L2 归一化，并取凸组合，构建一个单一的融合索引：

$$\hat{\mathbf{e}}_i^{\text{text}} = \frac{\mathbf{e}_i^{\text{text}}}{\|\mathbf{e}_i^{\text{text}}\|_2}, \qquad \hat{\mathbf{e}}_i^{\text{meta}} = \frac{\mathbf{e}_i^{\text{meta}}}{\|\mathbf{e}_i^{\text{meta}}\|_2},$$

$$\mathbf{e}_i^{\text{sum}}(\alpha) = \frac{\alpha\,\hat{\mathbf{e}}_i^{\text{text}} + (1-\alpha)\,\hat{\mathbf{e}}_i^{\text{meta}}}{\|\alpha\,\hat{\mathbf{e}}_i^{\text{text}} + (1-\alpha)\,\hat{\mathbf{e}}_i^{\text{meta}}\|_2}, \qquad \alpha \in [0,1]. \tag{1}$$

查询时，我们使用文本编码器对查询进行一次嵌入，并通过与融合索引计算余弦相似度来检索。由于文档嵌入已经过 L2 归一化，因此查询向量不进行归一化也不会影响余弦相似度下的排序：

$$\text{Score}_{\text{sum}}(q, i; \alpha) \;=\; \cos\big(\mathbf{e}_q^{\text{text}},\ \mathbf{e}_i^{\text{sum}}(\alpha)\big). \tag{2}$$

对于内积距离，$\mathbf{e}_q^{\text{text}}$和$\mathbf{e}_i^{\text{sum}}(\alpha)$都应进行 L2 归一化，以模拟余弦相似度。

公式 (1)–(2) 得到一个维度为 d的单一索引，无需像拼接那样将维度加倍。它们使元数据和文本在融合前保持分离，并避免运行时进行评分融合。

采用后期融合评分的双编码器 或者，我们也可以维护两个索引（内容和元数据），并在查询时合并评分。后期融合会在查询时公开 α，这有助于诊断，但需要进行两次查找。给定一个查询文本嵌入$\mathbf{e}_q^{\text{text}}$和文档嵌入（$\mathbf{e}_i^{\text{text}}$，$\mathbf{e}_i^{\text{meta}}$），我们计算

$$\text{Score}_{\text{late}}(q,\ i;\ \alpha) \;=\; (1-\alpha)\cos\big(\mathbf{e}_q^{\text{text}},\ \mathbf{e}_i^{\text{text}}\big) + \alpha\cos\big(\mathbf{e}_q^{\text{text}},\ \mathbf{e}_i^{\text{meta}}\big), \qquad \alpha \in [0,1]. \tag{3}$$

3.3 查询侧策略：元数据感知改写

我们使用基于大语言模型的改写算子，将模式线索（例如公司、表格、年份、章节）纳入查询，然后使用文本编码器对其进行嵌入：

$$\phi_{\text{text}}(q) = \texttt{Reformulate}(q).$$

这个经过改写的查询既可用于元数据即文本检索，也可用于双编码器变体，但会增加查询时的额外开销。查询改写通过一次大语言模型调用来实现，该调用以元数据模式（表1）和每个字段的一小组示例值为条件。给定一个查询，大语言模型会提取明确的元数据约束，并重写查询以纳入这些约束。随后，检索使用与其他所有方法相同的冻结文本编码器对改写后的查询进行嵌入，因此检索流程保持不变。

3.4 元数据集成的嵌入空间理论

我们考虑一种融合元数据的嵌入 $\tilde{\mathbf{e}}_i^\star$，它通过词元级前缀添加（MaT）或向量级融合（Unified，见第 3.2 节），将结构化元数据融入文本块嵌入 $\mathbf{e}_i = f_\theta(c_i)$。设 $d \in \mathcal{D}$ 表示一份文档（例如某家公司某一年的美国证券交易委员会文件），$i \in d$表示属于 d的一个文本块。以下命题阐述了这类嵌入如何重塑相似度格局。

命题 1（文档内聚合度提高）。

$$\mathbb{E}_{i,j\in d}[\cos(\tilde{\mathbf{e}}_i, \tilde{\mathbf{e}}_j)] \;>\; \mathbb{E}_{i,j\in d}[\cos(\mathbf{e}_i, \mathbf{e}_j)]\,.$$

元数据将文本块锚定到其文档身份，使它们在嵌入空间中彼此靠近。

命题 2（文档间混淆减少）。

$$\mathbb{E}_{i\in d_1,\, j\in d_2,\, d_1\neq d_2}[\cos(\tilde{\mathbf{e}}_i, \tilde{\mathbf{e}}_j)] \;<\; \mathbb{E}_{i\in d_1,\, j\in d_2,\, d_1\neq d_2}[\cos(\mathbf{e}_i, \mathbf{e}_j)]\,.$$

元数据提供了具有区分性的线索（公司、年份、章节），可减少不同文档之间的虚假相似性。

命题 3（分数方差增加）。

$$\mathrm{Var}[\cos(\mathbf{e}_q, \tilde{\mathbf{e}}_i)] \;>\; \mathrm{Var}[\cos(\mathbf{e}_q, \mathbf{e}_i)]\,, \qquad q \sim \textit{typical queries}.$$

统一嵌入通过凸权重 α 在仅内容信号和仅元数据信号之间进行插值，因此只要满足 $\alpha < 1$，就具备上述性质。MaT 通过令牌级注入实现类似效果，而 Unified 则通过可调权重进行向量级融合。实际上，这会创建一个结构更清晰的空间，使相关候选项与不相关候选项之间的区分更加明显。

4 方法论

4.1 数据集：RAGMATE-10K

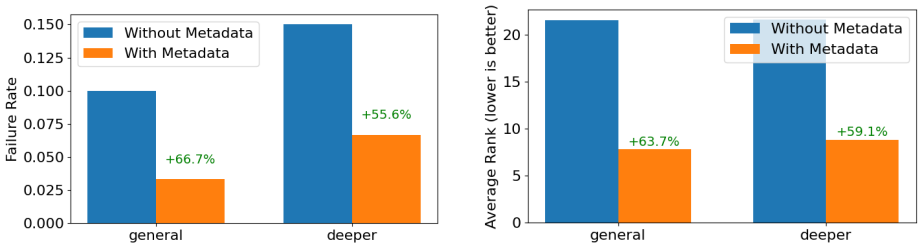
我们推出了 RAGMATE-10K，这是一个由 SEC 10-K 文件构成的数据集，旨在评估元数据感知检索。它包含来自五家美国科技公司（Apple、Alphabet、Adobe、Oracle、Nvidia）的 25 份文件，每份文件都被分割为不重叠的 350 个令牌文本块，重叠部分为 50 个令牌。由此得到 $N = 4$，490 个检索单元，每个单元表示为一个元组（m_i，c_i），其中 c_i 是文本内容，m_i 是其结构化元数据（公司、年份、章节、表格类型）。RAGMATE-10K 公开提供。[3]

我们创建了 30 个人工编写的模板，可实例化为针对特定公司和年份的问题，涵盖通用（例如，业务概述）和深入（例如，风险因素）信息需求。评估中排除 Apple 的申报文件，以避免数据污染，最终保留 120 个测试查询。

标准答案通过限制语言模型只能使用目标申报文件中的文本块来生成。模型必须引用支持性文本块，从而为检索准确率和答案依据提供监督。

表1：所有实验中使用的扁平元数据架构。

字段	描述	示例
公司_名称	申报实体名称	Alphabet公司
表格_类型	SEC表格类型	10-K
章节	文档章节标题	项目 1 - 业务
财年_年份_结束	财年结束日期	12-31
报告_的_期间	报告期结束日期 2023-12-31	
提交_日期	美国证券交易委员会文件提交日期	2024年-01-31
交易所_上市信息 在	[纽约证券交易所]上市的公开交易所	
SIC_代码	行业分类	计算机



(a) 各类别的检索失败率。 (b) 首次匹配的平均排名。

图 3：采用双编码器统一嵌入方法时，不同查询类型下相对于普通基线的检索性能对比。

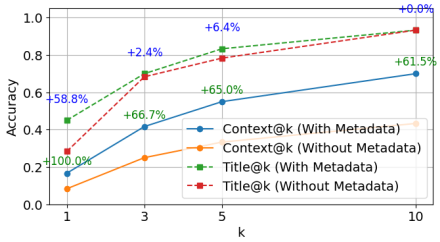
4.2 实现细节

我们使用冻结的文本编码器 f_θ 和固定的检索流程，以隔离元数据设计的影响。每个检索单元都是一对 $(m_i,\ c_i)$，其中 c_i 是文档文本的一个文本块，而 m_i 是一个扁平的键值元数据字典。

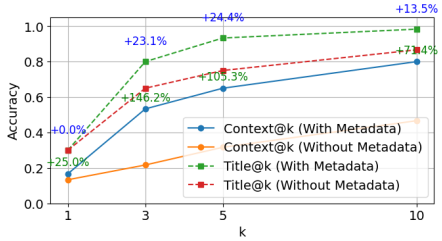
我们评估了两种具有代表性的嵌入模型：OpenAI 的 text-embedding-3-small（维度为 1536） [2] 和 BAAI 的 bge-m3（维度为 1024） [3], 。后者是一款强大的开源检索器，针对多语言和跨领域检索进行了优化。这两种编码器均以冻结形式使用，未经微调。

元数据 m_i 以扁平的键值字典表示，独立于文本块内容。对于基于文本的变体，这些字段会按固定顺序序列化为标头；对于双编码器变体，则会原样传递给元数据编码器。我们使用余弦相似度进行 top-K 检索，并改变 $K \in \{1, \ldots, 10\}$。我们使用四项指标，根据真实支持文本块评估检索质量：

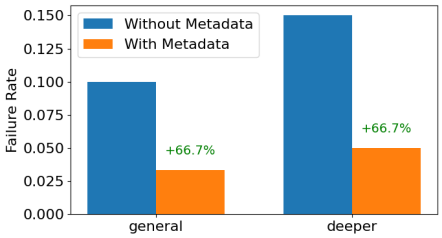
– **Context@K**：前 K 个检索结果中是否至少有一个文本块支持标准答案。– **Title@K**：前 K 个文本块中是否至少有一个来自正确的文档（公司和年份）。– **平均匹配排名**：排名最高的支持性文本块的平均排名位置（越低越好）。



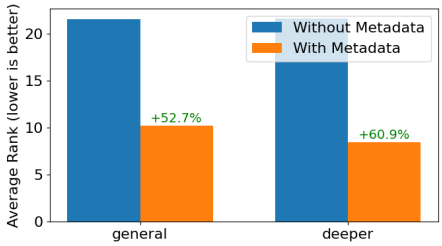
(a) Context@K 和 Title@K（一般问题）。



(b) Context@K 和 Title@K（深入问题）。



(c) 按类别划分的检索失败率。



（d）首次匹配的平均排名。

图 4：使用元数据文本化（前缀）时，不同查询类型下相对于普通基线的检索性能对比

– **检索失败率**：未检索到任何相关文本块的查询所占比例（越低越好）。

5 项发现

我们的实验表明，元数据感知检索始终优于纯内容基线。在各种策略中，有两种表现尤为突出：基于前缀的元数据文本化（MaT）和双编码器统一嵌入。

统一嵌入是最有效且最实用的方法。通过将元数据和内容向量融合到单一索引中，它们实现了与前缀方法相当或更高的准确率，同时在索引维护和在线服务方面具有明显优势。这使统一嵌入成为现实世界检索增强生成系统部署的有力候选方案，尤其适用于元数据会随时间不断变化的场景。

与此同时，元数据即文本仍然是一种简单且高性能的基线。直接拼接能够可靠地提升检索准确率，无需架构变更或额外基础设施，因此成为免训练基线的理想选择。

后期融合双编码器和元数据感知查询改写的表现不如这些方法。对融合权重 $\alpha \in \{0.0, 0.1, \ldots, 1.0\}$ 进行扫描后发现，中等取值（$\alpha \approx 0.3$–0.6）效果最佳，这证实元数据应当补充而非主导内容信号。实际应用中，后期融合优势不大：统一的嵌入在单一索引中也能实现类似的平衡，同时避免额外的

表 2：两种嵌入模型在截断值 k=5 下的检索性能摘要。双编码器结果以 $\alpha = 0.5$ 报告。

(a) OpenAI text-embedding-3-small

方法	通用				更深入			
	上下文@5 ↑	标题@5 ↑	平均排名 ↓	失败 ↓	上下文@5 ↑	标题@5 ↑	平均排名 ↓	失败 ↓
无元数据	33.33	78.33	21.61	10.00	31.67	75.00	21.63	15.00
元数据后缀	48.33	88.33	11.49	1.67	53.33	86.67	10.52	3.33
双编码器（后期融合）	36.67	78.33	10.80	15.00	38.33	78.33	15.43	15.00
双编码器（重新表述）	41.38	82.76	11.04	22.41	38.60	78.95	13.89	19.30
元数据前缀	**55.00**	**83.33**	**10.22**	**3.33**	**65.00**	**93.33**	**8.46**	**5.00**
双编码器（统一）	**63.33**	**88.33**	**7.84**	**3.33**	**60.00**	**83.33**	**8.84**	**6.67**

(b) BAAI bge-m3

方法	通用				更深入			
	上下文@5 ↑	标题@5 ↑	平均排名 ↓	失败 ↓	上下文@5 ↑	标题@5 ↑	平均排名 ↓	失败 ↓
无元数据	26.67	93.33	16.45	30.00	33.33	95.00	15.95	31.67
元数据后缀	33.33	95.00	17.93	6.67	45.00	95.00	11.00	11.67
双编码器（后期融合）	30.00	88.33	13.67	35.00	31.67	91.67	19.37	31.67
双编码器（重新表述）	25.86	86.21	15.74	34.48	26.32	94.74	16.94	40.35
元数据前缀	**61.67**	**96.67**	**6.97**	**3.33**	**58.33**	**95.00**	**10.98**	**5.00**
双编码器（统一）	**55.00**	**93.33**	**10.73**	**8.33**	**51.67**	**95.00**	**10.09**	**10.00**

查找和延迟。不过，扫描 α 仍有助于诊断元数据的边际价值（见图5）。

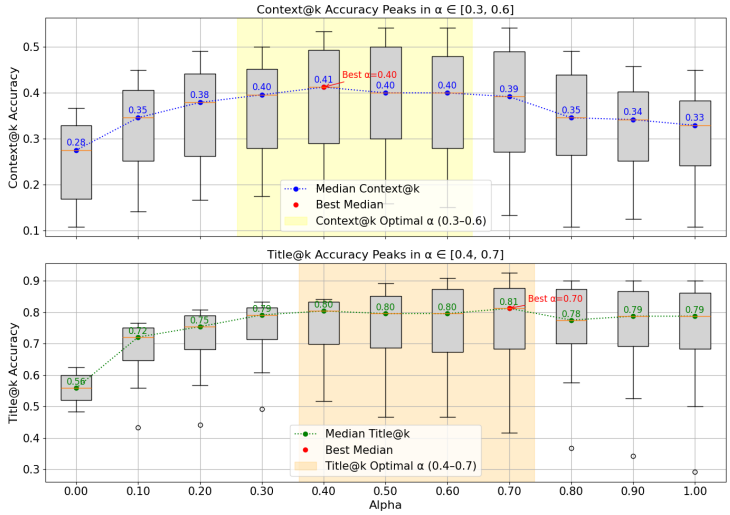


图 5：采用后期融合评分的双编码器在不同元数据权重 α 下的检索性能。适度赋予元数据权重可提升结果；完全依赖内容或元数据都会降低性能。

在两种嵌入模型中——OpenAI 的 text-embedding-3-small（1536 维）和 BAAI 的 bge-m3（1024 维）——元数据增强策略始终优于普通基线，

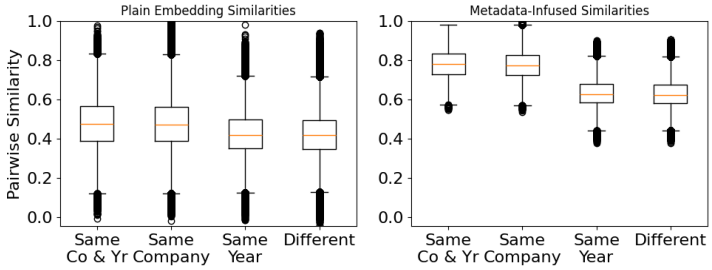


图 6：统一嵌入的嵌入空间分析（组内／组间文档相似度）

对于需要进行公司级或章节级消歧的复杂查询，性能提升最为显著。图表 1、3 和 4 展示了这些策略在不同查询类型上的性能对比趋势。

表 2 报告了 $k{=}5$ 处的代表性指标，以简洁的数值形式展示不同方法之间的性能差异。这些核心结果为后续章节中对权衡和消融实验的深入分析奠定了基础。

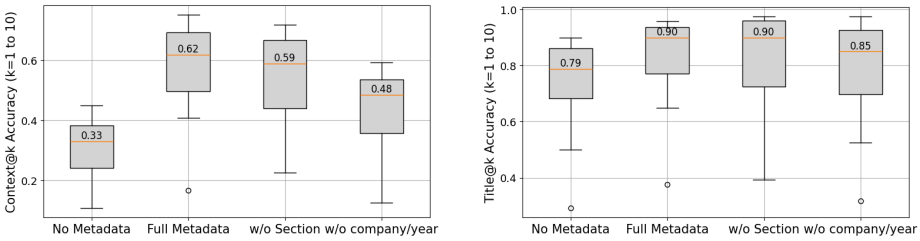
5.1 嵌入空间分析

我们通过计算元数据即文本（MaT）和统一嵌入（公式 1）下的成对余弦相似度，检验命题 1–3。对于每对文本块，我们计算两种相似度：普通 $\cos(\mathbf{e}_i, \mathbf{e}_j)$，以及元数据增强的 $\cos(\tilde{\mathbf{e}}_i^{\mathrm{MaT}}, \tilde{\mathbf{e}}_j^{\mathrm{MaT}})$ 或 $\cos(\tilde{\mathbf{e}}_i^{\mathrm{Unif}}, \tilde{\mathbf{e}}_j^{\mathrm{Unif}})$。我们将样本对划分为同一公司及年份（正样本）和不同公司或年份（负样本），并量化二者之间的区分度。

下面报告双编码器统一嵌入的结果；MaT 变体呈现出相似趋势。元数据提高了所有分层中的相似度，尤其是正样本对的相似度，从而拉大了差距。例如，正样本与负样本之间的平均间隔增至三倍（$0.054 \to 0.152$），Cohen's d 从小效应（0.45）提升至极大效应（2.25），曲线下面积则从 0.63 升至 0.94。如

表 3：根据成对余弦相似度计算的同一公司及年份（正样本）与不同公司或年份（负样本）样本对之间的区分度。我们报告普通基线值，以及统一嵌入和前缀嵌入的相对改进幅度（Δ）。箭头表示数值越高（↑）或越低（↓）越好。

指标	纯文本	Δ 双编码器（统一）	Δ 元数据前缀
平均边际（↑）	0.054	+0.098	**+0.101**
Cohen's d（↑）	0.450	**+1.800**	+1.440
费舍尔得分 F (↑)	0.102	**+2.425**	+1.692
曲线下面积 (↑)	0.625	**+0.315**	+0.283
KS 距离 (↑)	0.190	**+0.520**	+0.461
尾部概率质量 $\Pr[\cos \geq 0.8]$，正值 (↑)	0.007	**+0.407**	+0.108
尾部概率质量 $\Pr[\cos \geq 0.8]$, neg (↓)	0.000	+0.006	**+0.000**



(a) Context@K 准确率（$k = 1$ 至 10）　(b) Title@K 准确率（$k = 1$ 至 10）。

图 7：元数据消融对检索性能的影响

表 3 显示，两种元数据策略相较于普通嵌入均带来了显著提升；统一嵌入在绝大多数指标上优于添加前缀的做法，同时也更易于维护。

6 元数据字段的影响：消融研究

并非所有元数据字段对检索性能的贡献都相同。这里我们重点关注两类信号：公司和年份等全局标识符，以及来自章节标题的局部上下文。

从 SEC 文件等冗长且重复的文档中提取的文本块，往往会失去其上下文锚点。例如，“我们相信我们的战略正在奏效”这句话出现在“风险因素”章节和“管理层讨论与分析”中时，含义可能截然不同。我们区分两类上下文：i）全局上下文，由公司和年份等字段提供，用于标识整个文档；ii）局部上下文，由章节标题等字段提供，用于说明文本块在更广泛结构中的位置。为检验它们各自的作用，我们比较了四种条件：i）无元数据（基线）：不含元数据的纯文本块；ii）完整元数据：包含所有字段，包括章节标题；iii）不含章节标题：除章节标题外的完整元数据；iv）不含公司和年份：除公司和年份外的完整元数据。所有嵌入均使用 OpenAI 的 text-embedding-3-small 模型，并采用 MaT 表述；检索则使用第 4 节中的指标进行评估，重点关注 Context@K 和 Title@K。

图表 7a 和 7b 展示了结果。公司和年份提供了最强的消歧信号：移除它们会同时降低 Title@K 和 Context@K。相比之下，移除章节标题只会使 Context@K 略有下降，对 Title@K 则没有影响，这表明全局标识符决定了文档级准确率，而章节线索主要有助于分块级定位。

7 结论、局限性与未来工作

我们重新审视了元数据在检索增强生成（RAG）中作为一等检索信号的作用，并以 SEC 文件作为测试平台。在两种嵌入模型中，我们发现，将元数据与内容一同嵌入能够持续改善检索效果。简单地为

元数据添加前缀效果显著，但采用统一嵌入的双编码器能够达到或超过其准确率，同时更易于维护。字段级消融实验表明，公司和年份是强有力的消歧信息，而章节标题的作用仅较为有限；嵌入空间分析则显示，元数据通过提高文档内相似度并减少跨文档混淆，改善了嵌入空间的几何结构。

我们也承认本研究存在局限性。本研究聚焦于 SEC 表格 10-K 文件。这是一个刻意设计的高难度压力测试语料库，其特点是模板僵化、词汇重复度高，且文档级区别细微，常常会干扰纯语义检索。尽管不同领域的绝对增益可能有所差异，但这些最坏情况有助于我们隔离元数据集成的影响。我们预计，将元数据视为一等检索信号（尤其是通过统一嵌入实现）的相对优势也能推广至其他结构化语料库，例如科学文章、法律记录和技术手册。标准答案以半监督方式生成，但这一局限性仍反映了将大语言模型用作评判器或预言机的常见做法 [34, 13, 23, 16, 18]。此外，我们评估了冻结编码器，但未探索微调或端到端训练。

总体而言，我们的结果表明，有效的元数据集成并不需要复杂的架构：拼接或统一融合已经能够在准确性与实用性之间实现良好平衡。未来的研究方向包括自适应加权、更丰富的元数据模态（表格、图表），以及对下游生成质量进行人工评估。

利益披露。 作者声明不存在与本文内容相关的竞争性利益。

参考文献

1. 介绍上下文检索，https://www.anthropic.com/news/contextual- retrieval 2. 向量嵌入 - OpenAI API，https://platform.openai.com 3 3 4 3 2025年。BAAI/bge-m · Hugging Face（9月），https://huggingface.co/BAAI/bge-m。Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., 等：GPT-4技术报告。arXiv预印本 arXiv:2303.08774（2023）5. Asai, A., Wu, Z., Wang, Y., Sil, A., Hajishirzi, H.：Self-RAG：通过自我反思学习检索、生成和批判（2024）6. Baek, J., Jeong, S., Kang, M., Park, J.C., Hwang, S.：知识增强型语言模型验证。载于：2023年自然语言处理经验方法会议论文集。第1720–1736页（2023）7. BenYoash, N., Brief, M., Ovadia, O., Shenderovitz, G., Mishaeli, M., Lemberg, R., Sheetrit, E.：SecQue：评估现实世界金融分析能力的基准。载于：第四届生成、评估与指标（GEM^2）研讨会论文集。第212–230页（2025年）8. Brin, S., Page, L.：大规模超文本网络搜索引擎的剖析。《计算机网络与ISDN系统》**30**(1-7)，107–117（1998）

9. Chen, P.B., Wolfson, T., Cafarella, M., Roth, D.：EnrichIndex：使用大语言模型离线丰富检索索引。arXiv预印本 arXiv:2504.03598（2025年）10. Chen, Z., Chen, W., Smiley, C., Shah, S., Borova, I., Langdon, D., Moussa, R., Beane, M., Huang, T.H., Routledge, B.R., 等：FinQA：一个针对金融数据进行数值推理的数据集。载于：2021年自然语言处理经验方法会议论文集。第3697–3711页（2021年）11. Cuconasu, F., Trappolini, G., Siciliano, F., Filice, S., Campagnano, C., Maarek, Y., Tonellotto, N., Silvestri, F.：噪声的力量：重新定义检索增强生成系统的检索。载于：第47届国际ACM SIGIR 信息检索研究与发展会议论文集。第719–729页（2024年）12. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.：BERT：用于语言理解的深度双向Transformer 预训练。载于：2019年计算语言学协会北美分会年会：人类语言技术论文集，第1卷（长篇与短篇论文）。第4171–4186页（2019年）13. Fu, J., Ng, S.K., Jiang, Z., Liu, P.：GPTScore：按需评估。载于：2024年计算语言学协会北美分会年会：人类语言技术论文集（第1卷：长篇论文）。第6556–6576页（2024年）14. Gao, L., Ma, X., Lin, J., Callan, J.：无需相关性标签的精准零样本稠密检索。载于：第61届计算语言学协会年会论文集（第1卷：长篇论文）。第1762–1777页（2023年）15. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H., Wang, H.：面向大语言模型的检索增强生成：综述。arXiv预印本 arXiv:2312.10997 **2**(1)（2023年）16. Hackl, V., Müller, A.E., Granitzer, M., Sailer, M.：GPT-4是可靠的评分者吗？评估GPT-4文本评分的一致性。载于：Frontiers in Education。第8卷，第1272229页。Frontiers Media SA（2023年）17. Hu, Y., Koren, Y., Volinsky, C.：针对隐式反馈数据集的协同过滤。载于：2008年第八届IEEE国际数据挖掘会议。第263–272页。IEEE（2008年）18. Huang, H., Bu, X., Zhou, H., Qu, Y., Liu, J., Yang, M., Xu, B., Zhao, T.：关于用于大语言模型评估的LLM-as-a-judge的实证研究：微调评判模型并不能普遍替代GPT-4。载于：计算语言学协会研究成果：ACL 2025。第5880–5895页（2025年）19. Islam, P., Kannappan, A., Kiela, D., Qian, R., Scherrer, N., Vidgen, B.：FinanceBench：金融问答新基准。arXiv预印本 arXiv:2311.11944（2023年）20. Lai, V., Krumdick, M., Lovering, C., Reddy, V., Schmidt, C., Tanner：SecQA：用于金融问答的系统性评估语料库。载于：第十届金融科技与自然语言处理研讨会论文集。第221–236页（2025年）21. Lee, C.T., Neeser, A., Xu, S., Katyan, J., Cross, P., Pathakota, S., Norman, M., Simeone, J.C., Chandrasekaran, J., Ramakrishnan, N.：大语言模型能在电子表格中找到方向吗？载于：2025年IEEE/ACM第47届软件工程国际会议（ICSE）论文集。ICSE ’25，IEEE计算机学会，加拿大安大略省渥太华（2025年），https://www.computer.org/csdl/proceedings-article/icse/2025/056900a638/251mGdNO8uY 22. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., 等：检索增强生成

知识密集型自然语言处理任务。《神经信息处理系统进展》**33**，9459–9474（2020） 23. Li, J., Sun, S., Yuan, W., Fan, R.Z., Zhao, H., Liu, P.: 用于评估对齐的生成式评判器。arXiv 预印本 arXiv:2310.05470（2023） 24. Li, L., Chu, W., Langford, J., Schapire, R.E.: 一种用于个性化新闻文章推荐的上下文老虎机方法。载于：第19届万维网国际会议论文集。第661–670页（2010） 25. Lin, X.V., Chen, X., Chen, M., Shi, W., Lomeli, M., James, R., Rodriguez, P., Kahn, J., Szilvasy, G., Lewis, M., 等：Ra-DIT：检索增强双重指令微调。载于：第十二届国际学习表征会议（2023） 26. Luo, L., Li, Y.F., Haffari, G., Pan, S.: 图推理：忠实且可解释的大语言模型推理。arXiv 预印本 arXiv:2310.01061（2023） 27. Page, L., Brin, S., Motwani, R., Winograd, T.: PageRank引文排名：为网络带来秩序。技术报告，斯坦福信息实验室（1999） 28. Ryu, C., Lee, S., Pang, S., Choi, C., Choi, H., Min, M., Sohn, J.Y.: 大语言模型的基于检索的评估：韩国法律问答案例研究。载于：2023年自然法律语言处理研讨会论文集。第132–137页（2023） 29. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., 等：Llama：开放且高效的基础语言模型。arXiv 预印本 arXiv:2302.13971（2023） 30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: 注意力就是你所需要的一切。《神经信息处理系统进展》**30**（2017） 31. Xie, Q., Han, W., Chen, Z., Xiang, R., Zhang, X., He, Y., Xiao, M., Li, D., Dai, Y., Feng, D., 等：FinBen：面向大语言模型的综合金融基准。《神经信息处理系统进展》**37**，95716–95743（2024） 32. Yousuf, R.B., Defelice, N., Sharma, M., Xu, S., Ramakrishnan, N.: 利用大语言模型增强支持对多个文档进行分析推理。载于：2024年IEEE大数据国际会议（BigData）。第1892–1901页。IEEE（2024） 33. Zakka, C., Shad, R., Chaurasia, A., Dalal, A.R., Kim, J.L., Moor, M., Fong, R., Phillips, C., Alexander, K., Ashley, E., 等：Almanac——面向临床医学的检索增强语言模型。NEJM AI **1**(2)，AIoa2300068（2024） 34. Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., 等：使用MT-Bench和Chatbot Arena评判LLM-as-a-judge。《神经信息处理系统进展》**36**，46595–46623（2023） 35. Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., 等：由简到繁提示能够提升大语言模型的复杂推理能力。arXiv 预印本 arXiv:2205.10625（2022）