

基于前缀剪枝的大规模向量空间相似检索框架

刘健博¹, 邓凌风², 李文海², 田野³

(1. 武汉数博科技有限责任公司, 湖北 武汉 430205; 2. 武汉大学 计算机学院, 湖北 武汉 430072; 3. 湖北开放大学 软件工程学院, 湖北 武汉 430074)

摘要: 针对大规模文本集合下基于权重的相似性查询问题, 提出一种支持前缀剪枝的高效检索框架。首先给出向量空间模型下相似性及其带权前缀定义, 理论证明了带权前缀剪枝的正确性; 其次, 面向大规模文本查询, 提出一种新的倒排索引结构, 利用索引叶节点维护记录的前缀权重, 并基于该索引构建高效的相似检索算法; 最后, 在TF/IDF权重策略下证明该方法能够有效支持大规模带权相似检索。结果表明, 其查询效率较Lucene的归并验证策略提升了5倍以上。

关键词: 前缀剪枝; TF/IDF; 向量空间; 倒排索引; 信息检索; 数据库

DOI: 10.11907/rjdk.241015

开放科学(资源服务)标识码(OSID):

中图分类号: TP311.13

文献标识码: A

文章编号: 1672-7800(2024)006-0092-06



A Large-Scale Vector Space Similarity Retrieval Framework Based on Prefix Pruning

LIU Jianbo¹, DENG Lingfeng², LI Wenhai², TIAN Ye³

(1. Wuhan DNect Technology Ltd., Wuhan 430205, China; 2. School of Computer Science, Wuhan University, Wuhan 430072, China; 3. School of Software Engineering, Hubei Open University, Wuhan 430074, China)

Abstract: Aiming at the problem of weight-based similarity query under large-scale text collection, an efficient retrieval framework supporting prefix pruning is proposed. Firstly, we give the definition of similarity and its weighted prefix under the vector space model, and theoretically prove the correctness of weighted prefix pruning; then, for large-scale text query, we propose a new inverted index structure, use the index leaf nodes to maintain the prefix weights of the records, and construct efficient similarity retrieval algorithms based on the index; finally, we prove that the method can effectively support large-scale similar retrieval with weights, and the results show that its query efficiency is more than 5 times higher than that of Lucene's subsumption verification strategy.

Key Words: prefix-based pruning; TF/IDF; vector space model; inverted index; information retrieval; database

0 引言

伴随着数据挖掘的快速发展和互联网中文本数据的与日俱增^[1], 数据库中将产生大量带有描述性质的文本数据^[2]。面对这类数据, 一种有效的方法是对数据进行分词, 使得每个记录包含若干具有(半)结构的串字段。其中, 每个字段包含一定数量的标记(token或term), 或者通过轮排的方式得到每个字符串字段的gram^[3]。在此基础上定义集合函数, 如Jaccard、Cosine、Dice或编辑距离等, 对

串之间的相似性进行度量。例如, 给定两个集合{“武汉”, “大学”}和{“华中”, “科技”, “大学”}, 基于Jaccard度量两者交集数为2、并集为3, 二者相似性为 $2/3 \approx 0.67$ 。基于这类度量可以得到与给定查询(串)相似的记录, 尽管这些相似记录与原查询并不完全相同。这类基于计数的集相似查询(Set Similarity Search)丰富了传统数据库的查询手段, 已得到国内外学者的高度重视, 是数据库近年来的研究热点之一^[4-5]。

在图像检索和语义深度检索中, 同样存在大量的相似性查询需求。与串相似查询的不同之处在于, 这类相似性

收稿日期: 2024-01-26

基金项目: 武汉市重点研发计划项目(2023010402040006)

作者简介: 刘健博(1986-), 男, 博士, 武汉数博科技有限责任公司技术总监, 研究方向为人工智能、大数据; 田野(1980-), 男, 博士, 湖北开放大学软件工程学院高级工程师, 研究方向为人工智能、行业数字化应用。本文通讯作者: 田野。

检索认为不同集合中的元素具有不同的重要性^[6-7],从而应当在度量集合之间相似性时被赋予不同的权重(Weight)。文献[8]提出面向应用上下文的权重模型或概率模型,提升检索召回率 Recall 和精度 Precision 等。文献[9]为了适应向量空间的检索问题,将数据库内部的索引结构替换成机器学习模型。文献[10]通过优化索引结构的方式减小索引大小,而文献[11-12]通过改进索引结构提升检索效率。此外,文献[13-14]采用改进的哈希方法解决相似检索问题。这类研究普遍具有较强领域知识背景,而目前跨领域和数据集的带权搜索尚处于研究起步阶段。其中,一个核心问题在于如何针对权重设计高效的相似性查询方法,以快速响应大规模数据环境下的相似检索需求^[15]。

本文创新之处在于提出一种相似性检索框架,该框架引入带权前缀结构,在倒排索引基础上优化检索效率,构建一种高效相似检索算法。该框架支持向量模型权重的前缀定义,适用于大规模向量数据的检索问题。

1 基于权重的相似性检索

相似性检索问题具有一定的共性,本文首先对相似性检索的一般特征进行了总结归纳,给出相似性的定义,该定义不仅适用于基于计数的集相似性检索问题,也适用于基于向量空间模型的向量相似性检索问题。

对于一组富含文本的字符集,可以通过常用的符号化(Tokenization)方法将各实体记录中的串转换为一定数量的符号集合。因此,一个实体中的串可被转换为一个符号集合,该符号集合称为串集,其中每个串集由若干符号构成^[16]。

1.1 基于权重的相似性检索问题定义

考虑相似性查询通常需要一个全局序(Token order),定义带权符号上下文中的记录和查询(相似性查询针对给定查询串,查询串集与记录同构)串集如下所示^[17-20]。

定义1(串集) 任意串集 $R = \{r_1, r_2, \dots, r_{|R|}\}$ 由一个全局带权符号集 Σ 的非空子集构成,由此可得: $\forall r \in R \Rightarrow r \in \Sigma$ 。

串集可作为类型被定义为记录的任意字段。假定数据集 $R = \{R_1, \dots, R_n\}$ 由 n 个记录构成,其中每个记录由一个标示字段(id)和串集字段构成。

定义2(偏重度) Σ 中每个符号具有全局序 $O(t) \in \mathbb{N}$ 和权重 $W(t) \in \mathbb{R}$,符号 t 在 R 中的偏重度为 $F(t, R) \in \mathbb{R}$ 。

传统符号计数的相似检索可作为该定义的一个特例,即对任意 R 有 $\forall t \in R: F(t, R) = 1, W(t) = 1$ 。在信息检索应用,可以为每个符号定义任意的权重,也可在不同的串集中对某符号指定不同的偏重度。在此基础上,任意串集的长度可作如下定义。

定义3(串集的长度) 任意串集 $R = \{r_1, r_2, \dots, r_{|R|}\}$ 长度定义为:

$$\text{len}(R) = \sum_{t \in R} F(t, R)W(t) \quad (1)$$

即串集 R 中所有符号的权重与其在 R 中偏重度的乘积和,在此基础上引入 Cosine 定义任意串集带权相似性。

定义4(串集带权相似性) 任意两个串集 $R, S \in \mathbb{R}$ 带权相似度为:

$$\text{sim}(R, S) = \frac{(\sum_{t \in R \cap S} \sqrt{F(t, S)W(t)F(t, R)W(t)})^2}{\text{len}(R)\text{len}(S)} \quad (2)$$

该定义的分子源于向量内积(令任意串集中符号的权重及其所在串集的偏重度映射到全局序 O),其分母为两个串集的长度积。易证,当且仅当 R, S 完全相同时其相似度为1,而其正交时相似度为0。从传统众包(Crowd Sourcing)^[15]的集合定义可解释为:两个串集无共享符号时相似度为0;二者符号偏重度相同时为1;其他情况则二者相似度介于0和1之间。

在此基础上,一个查询 Q 具有与 R 中任意记录的串集字段相同的定义。检索目标为针对给定记录集 R 检索与 Q 的相似度不小于给定阈值 τ 的记录,即得到所有满足下述条件的结果集 $\mathfrak{S} = \{R\}$ 。

$$\text{sim}(R, S) \geq \tau | R \in R \quad (3)$$

1.2 基于权重的相似性计算示例

为了便于理解,本示例采用计数方式表示。如图1中的记录所示,基于定义2设定“武汉”“华中”和“科技”的权重为4、“大学”的权重为2,定义 R_3 中“武汉”的偏重度为2、其“大学”的偏重度为1;同样定义 R_2 中3个符号的偏重度均为1。则 R_2 和 R_3 的长度分别为 $\text{len}(R_2) = 4 + 4 + 2 = 10$ 、 $\text{len}(R_3) = 4 \times 2 + 2 \times 1 = 10$ 。

上述相似性检索方法兼顾了符号自身权重和符号在串集中的偏重度,能够更为有效地回答富含语义的文本检

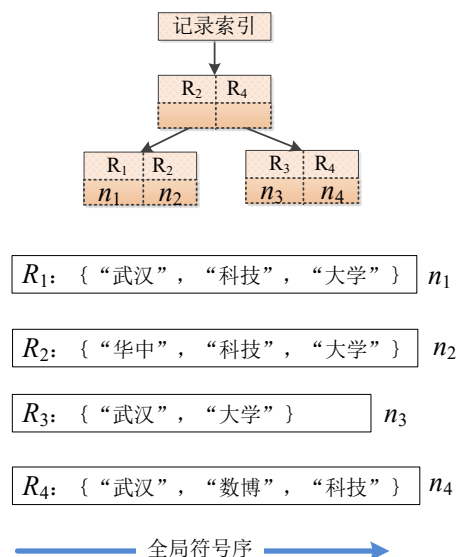


Fig. 1 String set record index

图1 串集记录索引

索,是目前较为主流的相似度检索方法。如给定查询 $\{\text{“华中”}, \text{“师范”}, \text{“大学”}\}$,其中 “师范” 的偏重度为1,其他符号的偏重度为4,则易得 $\tau = 0.4$ 下结果为 $R_2 | \text{sim}(Q, R_2) \approx 0.5$ 。

2 带权倒排索引

为实现基于符号的快速记录检索,本文首先为每个串集引入一个短小的签名(Signature)前缀,然后基于前缀进行高效剪枝。相似性查询通常基于倒排索引,给定符号集 Σ ,倒排索引由所有符号的倒排链表构成,其中每个链表为包含某个符号所有记录的数组构成。为支持大规模数据,倒排索引结构中的每个倒排链表以块的方式组织容纳于顺序的磁盘块中^[16]。

通过在倒排表中引入权重,本文给出带权索引结构示意如图2所示。其中中间节点由传统B树索引构成,用以快速访问任意给定的符号下属链表。该框架将倒排表设计为两层结构,其中的叶子节点由两个元素 $\langle ptr, lens \rangle$ 的单元序列构成。 ptr 指出倒排表在磁盘中的首地址, $lens$ 为所指倒排表的长度分段指针数组。当需要搜索一个指定长度的起始位置时,可以基于长度数组快速遍历到该长度的起始块地址和块内偏移。

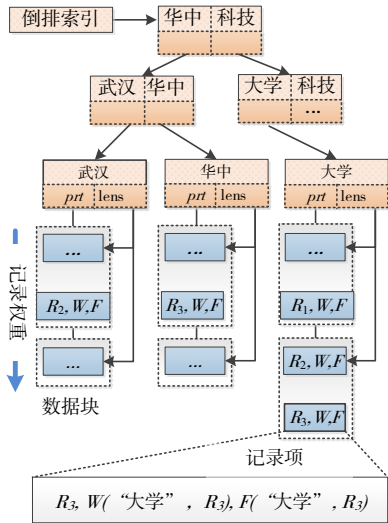


Fig. 2 Weighted index structure retrieval framework

图2 带权索引结构检索框架

2.1 带权前缀

前缀可以为一个全局有序的符号集下某个串集的一个最小前缀符号子集。前缀查询广泛存在于计数相似查询上下文中,以长度剪枝为基础,检索系统可高效基于前缀和长度对数据集进行快速过滤。

定理1(长度过滤) 给定一个查询,则相似串集长度应满足:

$$\text{len}(R) \in (0, \text{len}^{\circ}(Q)] \quad (4)$$

对于式(2)而言,大于等于 $\tau > 0$ 的约束可得 $(\sum_{t \in R \cap Q} \sqrt{F(t, Q)W(t)F(t, R)W(t)})^2 \geq \tau \text{len}(R)\text{len}(Q)$ 。假

定该式中任意符号 $t \in R \cap Q$ 在左侧所有属于 R 的串集中偏重度最大值为 $\max_{R \in R} F(t, R)$,则通过替换其中的 $F(t, R)$ 可得,当 $\tau > 0$ 时任意相似串集具有上界 $\text{len}^{\circ}(Q)$ 。

$$\text{len}^{\circ}(Q) = \frac{(\sum_{t \in Q} \sqrt{\max_{R \in R} F(t, R)F(t, Q)W(t)})^2}{\tau \text{len}(Q)} \quad (5)$$

由式(4)可知,任意满足长度约束的记录应当被独立读取和顺序扫描。而上述长度数组可辅助实现长度剪枝。在此基础上,一个前缀可以定义为满足长度上界的串集的最小有序前缀符号集,其定义如下。

定义5(前缀构造性质) 假定给定查询 Q 和 $\forall R \in R$ 中所有串集依从一个全局符号序 O ,则 Q 的带权前缀长度为:

$$\min_j (j! \sum_{t \in Q[j:|Q|]} \sqrt{W^2(t)F(t, Q) \max_{R \in R} F(t, R)}) < U_{\tau, Q} \quad (6)$$

其中, $U_{\tau, Q} = \sqrt{\tau \text{len}^{\circ}(Q)\text{len}(Q)}$ 为由式(5)决定的前缀内积上界。对于串集 $\forall R \in R$ 中符号集,取 Q 中的至少 j 个前缀符号扫描 R ,则 R 中所有与 Q 相似的串集必将被包含在内。因为,式(6)的极值能保证 Q 的后缀长度小于式(5)给出的上界约束。换言之,当考虑 Q 的前缀符号子集,若某记录 $\forall R \in R$ 无符号包含于 $Q[1:j]$,则无论 R 如何构成,其均与 Q 不相似。基于这一前缀构造性质^[15, 17],可得定理2。

定理2(构造性质) 若给定查询 Q 和 $\forall R \in R$ 中所有串集依从符号序 O ,基于式(6)前缀剪枝不会丢失结果。

给定任意查询,该定律给出了一种前缀策略,即算法可顺序扫描给定查询的带权前缀保证与查询前缀不相交的串集与查询必不相似。与此同时,一个更强的约束可参考位置过滤定义如下^[20-21]。

定理3(位置过滤) 任意给定查询 Q 与某记录 $\forall R \in R$ 的前缀和后缀长度应满足:

$$\left(\sum_{R[i]=Q[j]} \sqrt{W^2(t)F(t, Q)F(t, R)} \right)^2 + \text{len}(R[i:|R|])\text{len}(Q[j:|Q|]) \geq \tau \text{len}(R)\text{len}(Q) \quad (7)$$

上述3个定理能够支撑已有基于前缀的剪枝方法。下文将在常用的倒排索引中引入权重。

2.2 带权倒排索引结构

参考传统倒排索引结构,本文的索引数据节点以块方式组织,其中每个元素由一个三元组 $\langle id, W, F \rangle$ 构成,记录标示 id 给出包含一个符号(如图中“大学”)所有记录的主键。此外,通过权重 W 和符号在串集中的偏重度 F 根据式(5)和式(7)对数据进行剪枝。同时,将 W 分为两部分,分别保存主键 id 的串集总长度 $T = \text{len}(id)$ 和当前符号在串集的后缀长度 $W_{t, id} = \text{len}(id(t, :))$ 。以图2为例,记录 R_3 在符号“大学”中的倒排元素为 $\langle R_3, 10; 0, 1 \rangle$ 。其含义为:符号“大学”在记录 R_3 中的偏重度为1,该记录的符号集总长度为10,其中在符号“大学”之后的后缀长度为0。

倒排表对符号的访问时间复杂度为 $O(\log(m))$,其中

m 为所有符号的总数量。倒排表的总体空间复杂度为 $O(3mn)$, 即为所有符号集的所有符号维护 3 个单元的剪枝辅助信息。一般任意符号集仅包含 Σ 的一个小子集, 因此倒排表空间远小于线性复杂度。

3 相似检索框架

现阶段主流的相似检索引擎 Lucene (<https://apache.lucene.org>) 采用的策略是: 通过高效按位归并, 得到所有与已知查询存在至少一个符号交集的串集(文档)候选空间; 然后逐次对串集计算相似度评分(Score)并最终判定记录是否与给定查询相似。Lucene 在扫描过程中遍历所有至少与查询有一个符号交集的记录。图 3 中假定查询包含 8 个符号, 则需要逐次累计内积的计算复杂度为全部 16 个(在 8 个符号倒排链表中的)记录。该方法的优点是无需在倒排链表中维护辅助信息, 但缺点是需要访问和累计计算的复杂度较高, 复杂度为 $l|Q|$, 其中 l 为全集 Σ 中每个符号的平均串集数量。

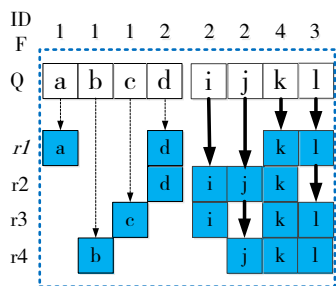


Fig. 3 Schematic diagram of inverted index retrieval

图3 倒排索引检索执行示意图

为提高检索效率, 本文提出的相似检索框架结合上文的 3 个定理, 在保证效率和准确的前提下, 使用前缀和长度过滤器(filters)对记录进行剪枝, 以达到减少比对规模的目的, 相似检索框架包括过滤和求精两个阶段。

基于前缀的剪枝算法可遵循前缀—长度—位置过滤进行剪枝(Pruning), 根据定理 2 可得, 剪枝操作不会出现“误滤”(False Negative)的可能, 但是存在一定的“误漏”(False Positive)。“误漏”在相似性查询中是被允许的, 为保证最终结果均满足式(3), 需要在最终结果集上进行求精(Verification)。该过程需要访问候选串集的后缀符号, 并逐一累计验证^[18]。检索过程示意如图 4 所示, 根据定理 1 和定理 3, 可以保证大多数情况下满足前缀和长度过滤器(filters)的记录在最终求精前被剪枝掉, 从而降低求精过程中访问串集索引的复杂度。根据定理 2 可知, 这一操作不会带来“误滤”, 即其能够基于最终的求精保证最终结果正确。

3.1 过滤阶段

过滤阶段算法的核心在于给出了一个能够处理向量空间下相似度查询的框架, 具体过程如算法 1 所示。

算法 1 基于前缀剪枝的向量相似检索算法

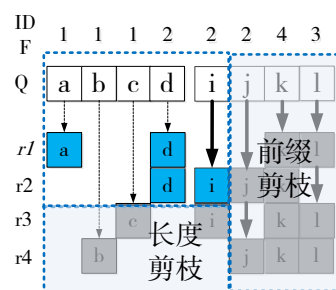


Fig. 4 Schematic diagram of prefix-based weighed inverted index retrieval

图4 前缀带权倒排索引检索执行示意图

输入: 查询符号集 Q , 带权倒排索引 $WIDX$;

输出: 候选结果集 C : id 与相似度 sim 的哈希表

1. 对 Q 中所有符号按照全局序排列并计算其前缀长度; 依次对每个符号执行下述操作:

2. 若符号 t 在 Q 中的位置小于等于式(6)确定的前缀长度, 访问倒排表得到 $WIDX[t]$ 的首块地址, 并逐一扫描块中记录的三元组描述 id, W, F ;

3. 若一个记录 id 在 C 中不存在, 则基于式(7)左侧计算符号集 id 与查询内积项; 否则累计该内积项并更新哈希表。同时, 通过 W 得到 id 的后缀长度, 并基于式(7)判定 id 的累计内积和后缀长度是否满足该式, 若不满足则从 C 中移除 id 。

4. 当 t 中某个记录的长度 W 大于式(5)给出的上界, 且 t 在 Q 中的位置小于式(6)确定的前缀长度, 则跳转至 2; 反之算法返回 C 进入后续求精过程。

该算法有效利用了上文中给出的 3 个定理的约束: 第 2 步基于式(4)得到长度的上界, 从而为扫描过程确定了一个长度终止条件; 第 3 步基于定理 3 的位置过滤思想, 通过逐次判断每个新加入或更新的候选串集的后缀长度是否能够在阈值条件得到满足, 以剪枝那些与给定查询在阈值条件下不可能相似的串集; 第 3 步中通过确定前缀的方式保证算法扫描的倒排链表仅局限于给定查询的一个前缀子集上。基于定理 3 可知, 这一基于前缀剪枝的过程能够保证没有“误滤”, 从而保证没有任何与已知查询相似的串集被剪枝, 进而保证算法的正确性。

算法 1 中用以支撑上述定理的核心要素, 即记录 id 、记录的后缀长度 W 和符号在记录中的偏重度 F , 均以三元组方式保存于带权倒排索引中, 因而算法可以在扫描倒排表的符号链表过程中被增量执行, 有效保证了算法的可用性。长度和位置过滤能够降低过滤计算的复杂度, 按照符号的频率增序构建 Σ 的全局序。实践过程中的算法复杂度明显较这一度量小, 算法 1 至多具有 $l|P_c(Q)|$ 的计算复杂度。

3.2 求精阶段

由于算法 1 仅考虑 Q 的前缀, 在没有串集后缀符号的前提下无法准确通过式(2)计算候选集 C 中任意记录与 Q 之间的相似度, 因而过滤阶段返回的候选集 C 中的串集为可能与给定查询 Q 相似的 R 中的记录。

对候选集 C 的求精过程如下: 首先, 通过访问串集索

引,依次得到候选集 C 中每个 id 对应的串集序列;然后,基于各串在串集中的偏重度和符号的权重,基于式(2)得到每个串集与 Q 之间的相似度;最后,产生与 Q 之间相似度满足式(3)的相似检索结果。

4 实验验证

为了便于与传统检索方法进行比较,本文采用TF/IDF算法,将文本内容转换为带权的空间向量,然后对方法的执行效率、影响因素和负载情况进行分析。实验环境为GenuineIntel CPU,主频3.3 GHz,内存容量为128 GB,磁盘为4块SCSI磁盘单盘IO吞吐率180 MB/s。所有程序采用Java1.8进行编译,考虑查询执行时间和IO统计次数,采用4 KB数据块维护记录和倒排索引。

4.1 实验数据

实验选取的Memetracker数据集(<https://snap.stanford.edu/data/memetracker9.html>)为一个大规模网上问答数据集,具有较为丰富的语义信息。选取其中2009年2-4月的Quotes和Phases域形成4 000万测试记录,对算法性能和扩展性进行验证实验。数据符号分布如表1所示,可以看到这些大规模数据子集与总体4 000万数据大致具有相同的平均符号数量,但符号数伴随记录数量呈次线性增长。

Table 1 The result of 4 data sets of different sizes in Memetracker
表1 Memetracker中的4个不同规模数据集结果

记录数	107	2×107	3×107	4×107
平均符号	9.82	9.78	9.81	9.8
符号数	1.71×106	2.25×106	2.52×106	2.81×106

4.2 算法效率分析

实验1(算法效率分析):采用变化策略验证算法1的前缀过滤效果,其中长度剪枝为去掉算法1第4步前缀判定后的执行结果。TF/IDF权重模型下的算法执行时间上,后续实验的执行均采用100个随机采样执行的平均时间进行度量。该实验中,Lucene方法采用的版本为Lucene6。在倒排索引中维护记录段和在记录中维护词牌信息,之后采用与式(2)同形的度量方法进行比较,实验结果如图5所示。可以看到,本文所提检索方法在效率上较Lucene有5倍提升,这主要源于符号的频率在大规模数据下分布极不均衡。在此背景下,长度索引(定理1)和位置过滤(定理3)能够过滤大量结果,此外索引内置权重也加速了算法执行效率。由此可得,本文框架对效率提升较为显著。

4.3 影响因素分析

通过分析发现,检索效率的影响因素还包括TF和检索数据规模,设计两组实验分别验证两种影响因素对执行效率的影响。

实验2(TF影响分析):为了比较TF对算法效率的影响,当去掉TF的因素后,对Lucene(TF=1, Cosine度量)和上述两种所提方法进行比较。如图6所示。长度剪枝和全剪枝策略分别较Lucene的归并方法至少提升了9和15

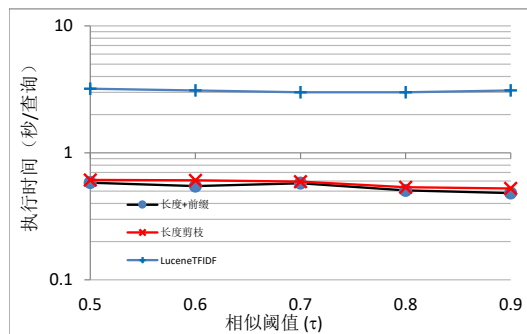


Fig. 5 Comparison with Lucene under TF/IDF weight model

图5 TF/IDF权重模型下与Lucene比较实验

倍,最大在0.9阈值时全剪枝较Lucene提升了两个数量级。由于TF在Lucene中需要访问记录中某个符号的出现频率,在去除该因素后,Lucene执行时间大致减少一半。通过实验结果以及其他计数相似性研究可知,长度和前缀在仅有符号权重时能够在较大阈值下以超线性方式减少^[17-22]。

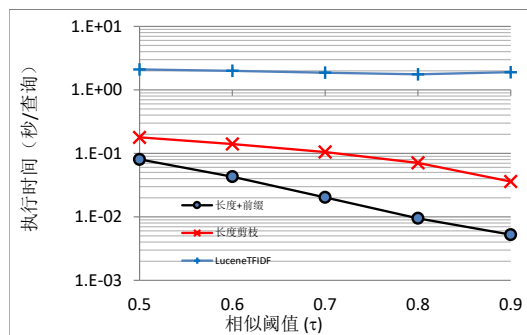


Fig. 6 Comparison with Lucene under IDF weight model

图6 IDF权重模型下与Lucene比较

实验3(数据规模影响分析):通过变换查询数据的规模,对查询时间及其主要代价构成进行验证分析。通过对表1中4个规模的数据集进行TF/IDF查询,得到图7的执行结果对比,其中每个折线为某个阈值上的检索执行时间。可以看出,伴随数据量的增长,算法执行时间近似线性增长。

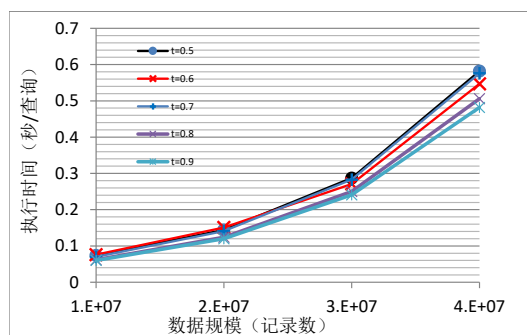


Fig. 7 Algorithm scalability under TF/IDF weight model

图7 TF/IDF下算法可扩展性比较

4.4 读写负载分析

实验4(读写负载分析):在实验3的基础上,通过记录每个查询的平均读写次数得到这些数据,通过跟踪算法访

问的倒排表的数据块评估算法数据扩展时的顺序读写代价,如图8所示。实验3和实验4的读写复杂度大致相当,计算复杂度与读写次数呈线性关系。因此,在长度索引和前缀索引作用下,计算负载与数据规模大致呈线性增长关系。

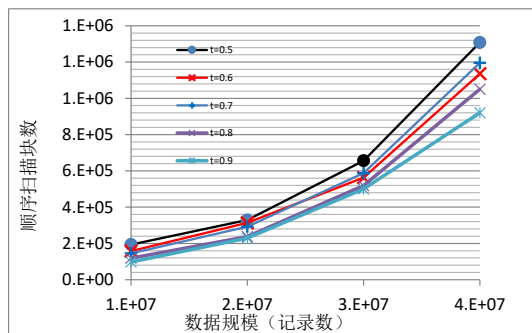


Fig. 8 Sequential I/O cost during data expansion

图8 数据扩展时的顺序读写代价

5 结语

本文提出一种支持空间模型的前缀剪枝检索框架,首先给出向量下相似性及其带权前缀定义,理论证明了带权前缀剪枝的正确性。同时,面向大规模文本查询,提出一种新的倒排索引结构,利用索引叶节点维护记录的前缀权重。基于倒排索引结构设计高效的相似检索算法,并通过实验证明该方法能有效支持大规模带权相似检索。结果表明,其查询效率较Lucene的归并验证策略提升了5倍以上。

参考文献:

- [1] AMSTERDAMER Y, DAVIDSON S B, MILO T, et al. OASSIS: query driven crowd mining[C]//Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014: 589–600.
- [2] KIM K T, RIOUX L E, TURGEON S L. Molecular weight and sulfate content modulate the inhibition of α -amylase by fucoidan relevant for type 2 diabetes management[J]. Pharma Nutrition, 2015, 3(3): 108–114.
- [3] WANG J B, GAO H, LI J Z, et al. Processing string similarity search in external memory efficiently[J]. Journal of Computer Research and Development, 2015, 52(3): 738–748.
王金宝,高宏,李建中,等. 外存中高效的字符串相似性查询处理[J]. 计算机研究与发展, 2015, 52(3): 738–748.
- [4] KOOPMAN B, ZUCCON G. Understanding negation and family history to improve clinical information retrieval[C]//Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, 2014: 971–974.
- [5] SHEN W, HAN J, WANG J. A probabilistic model for linking named entities in web text with heterogeneous information networks[C]//Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014: 1199–1210.
- [6] PAIK J H. A probabilistic model for information retrieval based on maximum value distribution[C]//Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015: 585–594.
- [7] PAIK J H. A novel TF-IDF weighting scheme for effective ranking[C]//Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2013: 343–352.
- [8] BARALIS E, CAGLIERO L, FIORI A, et al. Mwi-sum: a multilingual summarizer based on frequent weighted itemsets[J]. ACM Transactions on Information Systems (TOIS), 2015, 34(1): 1–35.
- [9] KRASKA T, BEUTEL A, CHI E H, et al. The case for learned index structures[C]//Proceedings of the 2018 International Conference on Management of Data, 2018: 489–504.
- [10] GALAKATOS A, MARKOVITCH M, BINNIG C, et al. Fiting-tree: a data-aware index structure[C]//Proceedings of the 2019 International Conference on Management of Data, 2019: 1189–1206.
- [11] ZHU M D, XU L X, SHEN D R, et al. Methods for similarity query on uncertain data with cosine similarity constraints[J]. Journal of Frontiers of Computer Science and Technology, 2018, 12(1): 49–64.
朱命冬,徐立新,申德荣,等. 面向不确定文本数据的余弦相似性查询方法[J]. 计算机科学与探索, 2018, 12(1): 49–64.
- [12] GAO Y N, YE J B, YANG N Z, et al. Middle layer based scalable learned index scheme[J]. Journal of Software, 2020, 31(3): 620–633.
高远宁,叶金标,杨念祖,等. 基于中间层的可扩展学习索引技术[J]. 软件学报, 2020, 31(3): 620–633.
- [13] WEN Q F, WANG J M, ZHU H, et al. Distributed learning to hash for approximate nearest neighbor search[J]. Chinese Journal of Computers, 2017, 40(1): 192–206.
文庆福,王建民,朱晗,等. 面向近似近邻查询的分布式哈希学习方法[J]. 计算机学报, 2017, 40(1): 192–206.
- [14] WANG Y X, TIAN J R, CHEN Z D, et al. Label enhancement based discrete cross-modal hashing method[J]. Journal of Software, 2023, 34(7): 3438–3450.
王永欣,田洁茹,陈振铎,等. 基于标记增强的离散跨模态哈希方法[J]. 软件学报, 2023, 34(7): 3438–3450.
- [15] WANDEL T S, DENG D, GERDJIKOV S, et al. State-of-the-art in string similarity search and join[J]. ACM Sigmod Record, 2014, 43(1): 64–76.
- [16] HADJIELEFTHARIOU M, CHANDEL A, KOUDAS N, et al. Fast indexes and algorithms for set similarity selection queries[C]//2008 IEEE 24th International Conference on Data Engineering, 2008: 267–276.
- [17] LI G, HE J, DENG D, et al. Efficient similarity join and search on multi-attribute data[C]//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015: 1137–1151.
- [18] WANG J, LI G, FENG J. Can we beat the prefix filtering? an adaptive framework for similarity join and search[C]//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 2012: 85–96.
- [19] WANG J, LI G, FENG J. Extending string similarity join to tolerant fuzzy token matching[J]. ACM Transactions on Database Systems (TODS), 2014, 39(1): 1–45.
- [20] XIAO C, WANG W, LIN X, et al. Efficient similarity joins for near-duplicate detection[J]. ACM Transactions on Database Systems (TODS), 2011, 36(3): 1–41.
- [21] CHAUDHURI S, GANTI V, KAUSHIK R. A primitive operator for similarity joins in data cleaning[C]//Atlanta: 22nd International Conference on Data Engineering (ICDE'06), 2006.
- [22] LESKOVEC J, BACKSTROM L, KLEINBERG J. Meme-tracking and the dynamics of the news cycle[C]//Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009: 497–506.