·先进计算与数据处理· 文章编号:1000-3428(2020)11-0117-07 文献标志码:A 中图分类号:TP391

基于差分编码的 RDF 分组压缩

伍伟鑫,韩京宇,朱 曼

(南京邮电大学 计算机学院,南京 210023)

摘 要: 语义网技术的发展使资源描述框架(RDF)的数据量迅速增长,导致其对存储空间与传输带宽的要求不断提高。现有的通用压缩方法和 RDF 专用压缩方法可以解决该问题,但仍存在数据冗余。为此,提出一种基于差分编码的 RDF 分组压缩算法。将 RDF 数据根据连接宾语的谓语组合进行分组,在消除宾语冗余的同时进一步减少谓语冗余。在此基础上,针对分组后得到的主语序列,通过引入差分编码技术进一步优化其存储空间。实验结果显示,与 Plain、HDT 和 HDT ++ 算法相比,该算法在结构化程度低的 Archives Hub、Linkedmdb、rdfabout 和 DBpedia 数据集中可获得平均 17% 的性能提升,在结构化程度高的 dbtune 数据集中可获得 23% 的性能提升,表明其对于不同结构化程度的数据集均具有较好的 RDF 压缩性能。

关键词: 语义网;资源描述框架;结构化程度;数据压缩;差分编码

开放科学(资源服务)标志码(OSID):

中文引用格式: 伍伟鑫,韩京宇,朱曼. 基于差分编码的 RDF 分组压缩[J]. 计算机工程,2020,46(11):117-123.

英文引用格式: WU Weixin,HAN Jingyu,ZHU Man. RDF grouping compression based on delta encoding[J]. Computer Engineering,2020,46(11):117-123.

RDF Grouping Compression Based on Delta Encoding

WU Weixin,HAN Jingyu,ZHU Man

(School of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210023,China)

【Abstract】 With the development of semantic Web technology,the volume of Resource Description Framework(RDF) data is increasing rapidly along with its demand for storage space and transmission bandwidth. Existing general compression methods and RDF-specific compression methods can solve this problem,but still suffer from a lack of data redundancy. To this end,this paper proposes an RDF grouping compression algorithm based on delta encoding. The algorithm groups RDF data according to the combination of predicates connected to the object,so as to further reduce predicate redundancy while eliminating object redundancy. On this basis,it further optimizes the storage space of the grouped subject sequence data by introducing delta coding technology. Experimental results show that,compared with the Plain,HDT and HDT ++ algorithm,this algorithm improves the performance by 17% on average in less structured datasets including Archives Hub,Linkedmdb,rdfabout and DBpedia,meanwhile improves performance by 23% on average in highly structured dataset dbtune,which demonstrates that the proposed algorithm has better RDF compression performance in datasets with different degrees of structure.

【Key words】 sematic Web; Resource Description Framework(RDF); degree of structure; data compression; delta encoding

DOI: 10.19678/j.issn.1000-3428.0056311

0 概述

为使计算机更好地理解人类创造的网络资源,TIM 等人于 1998 年提出了语义网的概念[1]。在语义网中,计算机可以理解文档中的词语和概念,从而使整个互联网成为一个通用的信息交换介质。W3C 联盟于 1999 年发布了基于 XML 语法的概念模型资源描述框架(Resource Description Framework,RDF)[2]用于实现语义网,其在逻辑上以图的形式表示,图中节点代表资源或资源的属性值,节点间连线代表节

基金项目: 国家自然科学基金(61602260);江苏省社科基金重点项目(18GLA004)。

作者简介: 伍伟鑫(1994—),男,硕士研究生,主研方向为 RDF 数据压缩、大数据管理;韩京宇,教授、博士;朱 曼,讲师、博士。

收稿日期: 2019-10-16 **修回日期:** 2019-12-16 **E-mail:** changing.xin@gmail.com

点间关系。RDF 图存储为文件时以三元组的形式描述每条数据,每个三元组由被描述的资源(主语)、资源的属性(谓语)和属性对应的值(宾语)组成[3],形式化为$(s,p,o)\in(U\cup B)\times U\times(U\cup B\cup L)$,其中,$U$为统一资源标识符,$B$为空白节点,$L$为 RDF 文本内容[4]。RDF 模型凭借简单、灵活、易扩展等特点迅速成为知识领域主流的半结构数据模式,被广泛应用于生命科学、地理学和维基百科等综合知识领域。

虽然 RDF 在逻辑层面表现优异,但其在物理层面可扩展性较差,原因在于 RDF 为了增强数据在逻辑层面的表现力而引入了大量重复的内容,即数据实际存储所占用空间远大于数据绝对空间[5],使得 RDF 数据集较其他存储格式需要更多的存储空间。例如数据集 DBpedia,其利用灵活的 RDF 结构有效地整合了维基百科中不同领域的异质资源。最新版本的 DBpedia 中包含 130 亿个三元组,描述了人类、地点、组织、电影、种类、疾病等多种信息,该数据集被应用于语义搜索、实体消歧和翻译等多种服务。尽管 DBpeida 可以通过调用不同的 API 进行在线查询,但一些需要完整数据集支持的应用和服务必须在本地存储和处理这个庞大的数据集,在资源受限的场景如轻量级的客户端或者低性能的网络场景中,如此大量的运算是一个巨大的挑战,而由此产生的 RDF 分布式管理技术[6-8]和压缩处理技术,则可作为可扩展技术来解决庞大 RDF 数据集的管理难题。

虽然目前已存在一些有效的 RDF 压缩方法,但其中仍存在不足,如包含尚未处理的冗余、压缩过程中引入了新的冗余等。为此,本文提出一种基于差分编码的 RDF 分组压缩算法。通过构建连接宾语的谓语组合,利用宾语和谓语之间相对唯一的映射关系减少不同谓语组合中的谓语冗余,并根据谓语组合将 RDF 数据分组存储,以消除宾语冗余。在此基础上,对分组后的主语序列进行差分编码,以序列间的偏移量代替数值本身,在不引入额外辅助索引的前提下优化主语序列的存储空间。

1 研究背景

RDF 压缩技术可分为物理压缩和逻辑压缩两种方法,前者通过减少符号与句法冗余,将原数据转化为更简洁的存储形式,后者致力于寻找一部分可以推导剩余数据的规则数据,最终只需要存储规则数据。

最简单的物理压缩方法是通用压缩算法,如 gzip 和 bzip2,其优点在于简单且运行效率高,可以轻易地集成到其他工作流中协同完成任务,但此类算法没有利用 RDF 数据的结构特性,并且在压缩过程中破坏了原数据的结构,无法进一步在压缩数据上进行查询检索等复杂操作。

HDT[9-10]是一种基于 BitmapTriple 的 RDF 专用物理压缩算法,其将原始 RDF 数据转化为一个包含多个深度为 3 的树的森林,每棵树的根节点代表主语,第 2 层是与根节点每个主语对应的谓语列表,第 3 层是对应每个(主语,谓语)组合的宾语列表。整个森林只需要存储谓语列表、宾语列表和对应代表分支信息的比特序列。这个简单的编码方法具有比通用压缩算法更高的压缩比率,并且可以通过遍历森林中的根节点解决压缩数据中基于主语的查询问题。HDT ++ 算法[11]在 HDT 的表达方式上进行了改进,其根据谓语组合对主语分组,大幅减少了谓语冗余,并且根据谓语对宾语进行二次分组,在分组中使用局部编码代替全局编码,以更小的比特数存储宾语,获得了比 HDT 更好的压缩效果。k^2-triple 算法[12]根据不同谓语将原始数据划分成多个以主语为横坐标、以宾语为纵坐标的二维 0-1 矩阵,对这些稀疏的二维矩阵利用 k^2-tree 算法[13]进行矩阵压缩,同时该算法可实现针对压缩数据常见的查询操作。

逻辑压缩方法的重点在于寻找规则子图的方式。文献[14-15]提出了基于 lean subgraph 的逻辑压缩算法。lean subgraph 是原始数据图的一个子图,是原始图实例的最小子图,由 lean subgraph 所移除的三元组数量强依赖于原始图的特征,最少可移除数量约为空白节点的两倍。然而,lean graph 中的部分三元组仍然可以由其他三元组推导得出,说明 lean subgraph 并不能完全移除冗余三元组[15]。基于规则的逻辑压缩 RB 算法[16]通过频繁项集挖掘技术检测 intra-property 和 inner-property 两种模式,intra-property 在指定谓语的前提下挖掘重复出现的(主语->宾语)映射,inner-property 挖掘重复出现的(主语->(谓语,宾语))映射,挖掘所得映射将转化为规则用于移除冗余数据。然而这两种模式并不都有效,只有 inner-property 可以移除大量冗余三元组。文献[17]指出频繁项集不能很好地捕获数据中的语义冗余,并提出一种更具表现力的喇叭规则,只要三元组能够匹配喇叭规则的头部即可被移除出数据集。存储的喇叭规则可以再次利用 RB 算法进行压缩,其压缩性能较单独使用 RB 算法有所提升,但同时也引入了较高的延迟。PIC 算法[18]将原始数据转化为以主语为横轴、以(谓语,宾语)二元组为纵轴的二维 0-1 矩阵,并将矩阵中的每一行 0-1 数组序列转化为一个新的三元组,原数据集的三元组可以由新产生的三元组计算得到,通过存储数量远小于原三元组的新三元组实现数据压缩。

2 相关定义

定义 1(连接宾语的谓语组合) 给定一个 RDF 数据集 $T=\{<s_i,p_i,o_i>\mid i\in\mathbb{N},s_i\in S,p_i\in P,o_i\in O\}$,

其中,S 为主语集合,P 为谓语集合,O 为宾语集合。$\forall o \in O, \exists \{p_1, p_2, \cdots, p_j\}, j \in \mathbb{N}, p_j \in P$, s. t. $< s_k, p_j, o > \in T, s_k \in S$,$p$ 组合被定义为连接宾语的谓语组合。

2.1　RDF 数据的结构化程度

结构化数据也称为行数据,是由二维表结构进行逻辑表达和实现的数据,行数据严格遵循相同的数据格式与长度规范,主要通过关系型数据库进行存储和管理。RDF 数据是一种灵活的半结构化数据,并不强求所有数据都具有相同的数据格式,因此,不同 RDF 数据集中数据的结构完整性也各不相同。

定义 2(RDF 数据的结构化程度) RDF 数据的结构化程度定义为谓语数量与连接主语的谓语组合数量的比值:$|P|/|G|$,其中,P 为谓语集合,G 为 RDF 数据集中谓语组合的集合。

2.2　基于模式的冗余

HDT ++算法[11]将与主语相关联的谓语归纳为谓语组合,再利用谓语组合对主语进行分组,使得每个主语只被存储一次,谓语通过谓语组合的形式只被存储较少的次数。这样分组的意义在于:对于不同的主语,可能存在多个谓语对其进行描述,如“人”这个主语可能存在的谓语包括身高、体重、籍贯、肤色等,在结构化程度高的数据集中,同一类型的主语大多具有相同的描述(谓语)。例如,对于 A 和 B 都存在身高、体重、肤色、籍贯的谓语描述,因此,将这样的主语由(身高、体重、肤色、籍贯)的谓语组合进行分组保存可以减少谓语的重复出现。但是在结构化程度低的数据集中,对于同一类型的不同主语,对其描述的谓语各不相同。例如,对于 A 的描述有身高、体重、肤色、籍贯,对于 B 的描述只有身高、体重、肤色,而没有籍贯的描述,对于 C 的描述只有身高、体重,这样就会产生(身高、体重、肤色、籍贯)、(身高、体重、肤色)和(身高、体重)3 种不同的谓语组合。上述 3 种谓语组合只存在少数谓语的差异,但因为组合不完全相等,所以相同的谓语就需要被存储多次,由此产生冗余。

定义 3(基于模式的冗余) 在连接主语的谓语组合中,G_m、G_n分别为第 m 个和第 n 个谓语组合,$m \neq n$,$m, n \in \mathbb{N}$。$\forall p_i \in G_m$ 且 $p_i \in G_n$,p_i表示基于模式的冗余。

2.3　差分编码

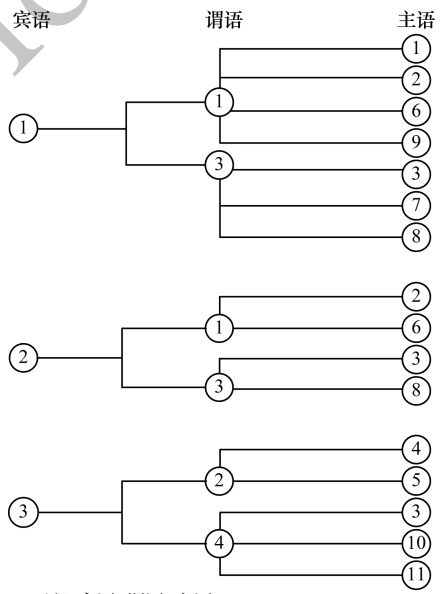
差分编码[19]又称增量编码,其是以数字序列间差异进行存储,而不存储数字本身的一种编方式,表示为:$\{a_1, a_2, \cdots, a_n\} \rightarrow \{a_1, a_2 - a_1, \cdots, a_n - a_{n-1}\}$。显然,差分编码的优点在于:当序列中数字之间的偏移量较小时,差分编码可以节省大量空间。此外,差分编码在减少序列存储空间的同时无需引入额外的中间变量,通过编码方法自身即可完成序列的编码与解码。

3　基于差分编码的 RDF 分组压缩算法

基于上节所讨论的冗余与编码方式,本文构建一种新的 RDF 数据分组表示方法用于减少低结构化程度 RDF 数据中的谓语冗余。首先归纳与宾语相对应的谓语组合,然后利用谓语组合对宾语进行分组表示,减少谓语冗余并去除宾语冗余,最后对分组后的主语序列应用差分编码,以更小的空间存储 RDF 中的全部主语。

3.1　基于谓语-宾语的分组表示

将 RDF 原始数据图转化为字典与 ID 图,字典表示为 RDF 中的 URI 或字面值与其唯一 ID 的映射,ID 图为通过 ID 表示的原始三元组关系。URI 或字面值通常是一串较长的字符串,如果重复存储将造成存储空间浪费,而通过字典将字符串映射为 ID 后只需要在字典中存储一次。本文在此基础上,将 ID 图转化为根据宾语归纳的谓语组合对宾语分组的形式,如图 1 所示。其中,树形结构的根节点为宾语,第 2 层节点为与根节点宾语对应的谓语组合,第 3 层节点为与(宾语,谓语)对应的主语列表。不同的宾语如果对应相同的谓语组合,则将其合并为一组,如图 2 所示。例如,图 1 中第 1 棵树与第 2 棵树的谓语组合相同,则将图 2 中第 1 棵树和第 2 棵树的谓语组合合二为一共同存储。



三元组(宾语, 谓语, 主语):
(1, 1, 1), (1, 1, 2), (1, 1, 6), (1, 1, 9), (1, 3, 3),(1, 3, 7), (1, 3, 8), (2, 1, 2), (2, 1, 6), (2, 3, 3), (2, 3, 8), (3, 2, 4), (3, 2, 5), (3, 4, 3), (3, 4, 10),(3, 4, 11)

图 1　基于谓语组合的 RDF 数据分组表示

Fig. 1　RDF data grouped expression based on predicate combinations

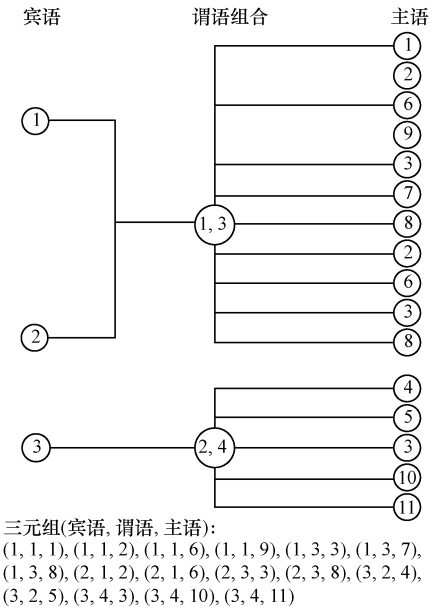


图 2 分组表示后相同谓语组合的归纳合并

Fig. 2 Induction and combination of repeat predicate combinations after grouped expression

根据谓语对宾语分组的意义在于：

1)存在一部分宾语同时与多个谓语相关联，将这些宾语的共同谓语抽取出来作为分组的规则对宾语进行分组，可以减少不同宾语的相同谓语重复存储。

2)对于结构化程度较低的数据，根据宾语归纳的谓语组合数相对于根据主语归纳的谓语组合数大幅减少。在一定方差范围内，每个宾语只与一个谓语相关联，只有部分宾语会同时与超过一个谓语相关联。主语则不然，由于主语的语义关系，每个主语会存在多个谓语来描述它的属性[20]。可根据组合数公式 C_p^n 计算可能的谓语组合数量，其中，p 代表谓语数量，n 代表与主语或宾语相关联的平均谓语数量，主语对应的谓语数远大于宾语对应的谓语数，所以，根据宾语归纳的谓语组合数量会小于根据主语归纳的谓语组合数量，也就减少了谓语冗余的数量。

3)由于宾语只与很少数量的谓语相关联，因此由宾语归纳的谓语组合中包含谓语的数量非常少，即使在不同谓语组合间存在重复谓词，重复谓词的数量也会被限制，从而限制了谓语总数的增长。

3.2 主语差分编码

原始 RDF 数据被转化为分组表示后，主语也被对应的谓语组合分到对应的分组中，为对主语进行差分编码，需要进行以下处理：

1)对主语序列按照新的顺序重新由小到大编码，使主语序列可以发挥差分编码的最大性能。值得注意的是，重新编码指的是将字典中的唯一 ID 替换为新的根据分组位置的新 ID，无须额外存储新编码的中间映射，而只需要在字典中进行 ID 更新。

2)将分组后的主语列表根据组内的(宾语，谓语)三元组划分成为不同的数字序列，对每个数字序列进行差分编码。

在图 3 中，主语序列(第 1 列)到第 2 列的转化展示了主语序列重新编码的过程，第 2 列到第 3 列的转化展示了根据(宾语，谓语)二元组划分的主语序列进行差分编码的过程。在存储到文件中时，对于每个主语序列，序列中数值以变长长度存储，由于差分编码产生的偏移量之间可能大小差距较大，因此变长长度可以最大化节省空间。图 3 中最后一列比特序列用于区分不同主语序列，一串连续的 0-bit 与一个单独的 1-bit 表示其所对应的主语序列归属于同一个(谓语，宾语)二元组。例如图 3 中比特序列的前 4 个比特数为[0,0,0,1]，说明其所对应的主语序列[1,1,1,1]皆对应二元组(1,1)，而接下来的比特序列[0,0,1]所对应的主语序列[5,1,1]则对应二元组(1,3)，可以发现二元组的第 2 个元素由 1 变成了 3。由于 3 是该组谓语组合的最后一位，因此下一个比特序列所对应的二元组变为(2,1)。在解压缩时，依此类推即可逆向还原所有三元组。

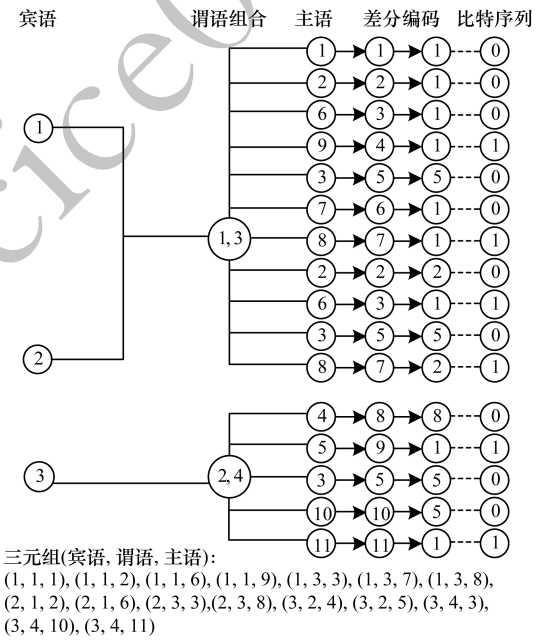


图 3 主语序列重编码及差分编码

Fig. 3 Re-encoding and delta encoding for subject sequence

算法 1 描述了本文算法的压缩过程，具体步骤如下：

1)声明 object2predicaet、objPred2subject 为映射，bitSeq 为字符串(第 1 行)。

2)遍历原始数据三元组，统计宾语与谓语的映射关系和(宾语、谓语)二元组与主语的映射关系

(第 2 行 ~ 第 5 行)。

3)将宾语、谓语映射关系中的谓语组合去重归纳得到宾语组合与对应的谓语组合(第 6 行)。

4)从宾语组合、谓语组合中生成的(宾语、谓语)二元组对主语进行归纳,在归纳过程中对每个二元组对应的主语序列进行差分编码(第 7 行和第 8 行),同时通过比特序列界定不同主语序列的界限(第 9 行)。

5)将宾语组合、谓语组合、主语序列和比特序列返回存储到文件中(第 10 行)。

算法 1　压缩算法

输入　原始三元组 triples

输出　谓语组合 predicateGroups,宾语根据谓语组合所分组合 objectGroups,对应(宾语,谓语)二元组的主语列表 subjectList,主语切换标记比特序列 bitSeq

```
1. object2predict ← map, objPred2subject ← map, bitsequence ←string
2. for subject,predicate,object← triples:
3. object2predicate[object].add(predicate)
4. objPred2subject[(object,predicate)].add(subject)
5. end for
6. predicateGroups,objectGroups ←reduce(object2predicate)
7. for (o,p) ←predicateGroup,objectGroups:
8. subjectList.add(deltaEncoding(objPred2subject[(o,p)]))
9. bitSeq.add(len(objPred2subject[(o,p)]) * '0' + '1')
10. returnobjectGroups,predicateGroups,subjectList,bitSeq
```

算法 1 的时间复杂度主要源于 2 个 for 循环和 1 个 reduce 操作。第 1 个 for 循环的执行次数为 RDF 数据中三元组数量 n,第 2 个 for 循环执行次数为 predicateGroups 中第 i 个谓语组合中谓语个数 k_i 与 objectGroups 中第 i 个宾语组合中宾语个数 m_i 的乘积之和 $\sum_{i=1}^{|G|} k_i m_i$(|G|为谓语组合的数量)。由于谓语组合和宾语组合是谓语和宾语的子集,因此 $k_i \leqslant |P|$,$m_i \leqslant |\text{Object}|$(由于宾语数量的符号|$O$|与时间复杂度的符号类似,为避免混淆,因此宾语数量的符号用|Object|代替)。当 $k_i = |P|$,$m_i = |\text{Object}|$ 时,表示只存在一个分组,$|G| = 1$,此时循环执行次数为 $|P| \times |\text{Object}|$;当 $k_i = 1$,$m_i = 1$ 时,宾语和谓语为一一映射,表示每个宾语与谓语的组合即为一个分组,$|G| = |\text{Object}| = |P|$,因此,循环执行次数为|Object|或|$P$|。reduce 操作的执行次数为 RDF 数据中宾语数量|Object|,$|\text{Object}| \ll n$,因此,算法 1 的时间复杂度为 $O(n + |\text{Object}| \times |P| + |\text{Object}|) = O(\max(n, |\text{Object}| \times |P|))$。

算法 1 的空间复杂度主要源于 object2predicate 和 objPred2subject 的临时存储。object2predicate 存储内容为宾语到谓语组合的映射,其中,key 的数量为宾语数量|Object|,value 的总数为三元组的数量 n。objPred2subject 存储内容为宾语与谓语的二元组到主语的映射,其中,所有元素的总量小于等于 3 倍的三元组数量。因此,算法 1 的空间复杂度为 $O(|\text{Object}| + n + 3n) = O(n)$。

算法 2 描述了本文算法的解压缩过程,具体步骤如下:

1)将分组表示的宾语、谓语和主语从文件中读出后,迭代每个谓语组合和对应的宾语组合生成(宾语,谓语)二元组(第 1 行和第 2 行)。

2)逐个访问主语列表,在访问过程中逐步还原差分编码(第 3 行 ~ 第 10 行)。

3)根据 bitSeq 中连续的 0-bit 和一个单独的1-bit 界定对应当前(谓语,宾语)二元组的主语序列的终止位置(第 6 行和第 13 行)。

4)将符合条件的主语与二元组组成三元组添加到 originalTriples 中并最终返回。

算法 2　解压缩算法

输入　谓语组合 predicateGroups,宾语根据谓语组合所分组合 objectGroups,对应(宾语,谓语)二元组的主语列表 subjectList,主语切换标记比特序列 bitSeq

输出　ID 表示的完整三元组 originalTriples

```
1. for (object,predicate) ← predicateGroups,objectGroups:
2. //获取每一对(宾语,谓语)二元组
3. for subject,bit← subjectList,bitSeq:
4. //通过 bit 确定 subject 的终止位置
5. if subject.forwardBit = 0:
6. subject = subject + subject.forward
7. //对差分编码进行解码
8. originalTriples←(subject,predicate,object)
9. end if
10. end for
11. //构成一条完整三元组
12. until bit = 1
13. returnoriginalTriples
```

算法 2 的时间复杂度主要源于迭代宾语、谓语二元组的 for 循环,该 for 循环执行次数与算法 1 中第 2 个 for 循环执行次数相等。因此,算法的时间复杂度为 $O(|\text{Object}| \times |P|)$。由于算法 2 的计算过程不需要除输入输出外的额外空间,因此其空间复杂度为 $O(1)$。

4　实验与结果分析

本文实验基于处理器为 Intel Core i5 3.1 GHz,内存为 16 GB 2 133 MHz LPDDR3 的计算机。分别选择不同领域、不同大小和不同结构化程度的数据集,数据集描述如表 1 所示,其中,Archives Hub 是档案文件描述数据集,Linkedmdb 是电影领域数据集,rdfabout 是综合领域数据集,dbtune 是音乐领域数据集。以上述4 个数据集分析 DGC 的实验结果,对 DGC 压缩与解压缩时间进行对比,与 HDT ++ 算法对分组数量和分组谓语总数进行对比,并与 Plain

(直接存储)、HDT、HDT ++ 算法对压缩后空间容量进行对比。

表 1 数据集描述

Table 1 Datasets description

数据集	三元组数量	主语数量	谓语数量	宾语数量
Archives Hub	431 088	51 411	141	104 408
Linkedmdb	6 147 996	694 400	222	2 052 964
rdfabout	17 188 323	3 132 667	108	4 408 000
dbtune	58 920 361	12 401 228	394	14 260 204
DBpedia	431 440 396	41 645 264	57 986	141 541 287

DGC 算法在不同数据集上压缩与解压缩的时间对比如图 4 所示。一般而言,压缩时间随着三元组数量的增长而增长,但同时也受到谓语数量的影响,从 rdfabout到 DBpedia 的时间陡增论证了这一点,DBpedia 不仅在三元组数量上是 dbtune 的7.3 倍,在谓语数量上更是有 146 倍的增长,这使得 DBpedia 存在更多的分组,每个分组中也存在着更多的谓语。因此,需要更多的计算,导致时间的陡增。解压缩时间虽然也随着数据量的增大而增多,但其增长幅度相对较小,一方面由于解压缩所需的计算量较少,另一方面可以对不同分组进行并行处理以加快速度。

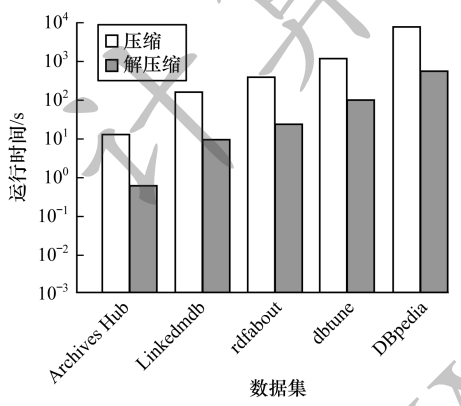


图 4 压缩与解压缩的运行时间对比

Fig. 4 Running times of compression and decompression

在 HDT ++ 连接主语的谓语组合(PS 分组)和 DGC 连接宾语的谓语组合(PO 分组)两种分组方式下,不同数据集谓语组合数量和谓语总数的对比如表 2 所示。可以看出,除数据集 dbtune 的 PO 分组数大于 PS 分组数外,其余数据集的 PO 分组数均小于 PS 分组组数,这是由于 dbtune 的结构化程度高于 Archives Hub、Linkedmdb、rdfabout 和 DBpedia,说明 dbtune 数据集中与主语相连的谓语组合较为完整,可以很好地对主语进行分组,而对于结构化程度较低的另外 4 个数据集,谓语组合不能对主语进行有效分组,但对宾语分组得到了很好的分组效果。

表 2 谓语-宾语分组与谓语-主语分组后谓语组合数与谓语总数的比较

Table 2 Comparison of the total number of predicate groups and predicates after predicate-object grouping and predicate-subject grouping

数据集	结构化程度	PS 谓语组数	PO 谓语组数	PS 谓语总数	PO 谓语总数
Archives Hub	0.23	611	234	12 035	802
Linkedmdb	0.03	8 459	1 527	135 345	13 456
rdfabout	0.35	305	263	2 316	602
dbtune	0.41	963	2 418	8 816	12 756
DBpedia	0.04	1 309 392	687 331	47 568 923	5 957 005

由表 2 可见,在 Archives Hub、Linkedmdb、rdfabout、DBpedia 数据集中,PO 谓语总数都明显低于 PS 谓语总数,其中,在 DBpedia 数据集中将语总数从 PS 分组下的千万量级减少到 PO 分组下的百万量级,说明 PO 分组确实有效减少了大量谓语冗余,只有在 dbtune 数据集中 PS 分组获得了更好的分组效果。但从另一个角度来看,dbtune 的 PO 谓语组数是其 PS 谓语组数的 2.5 倍,而 PO 的谓语总数却只是 PS 谓语总数的 1.4 倍,谓语总数的比例相对于组数的比例有所降低。反之,在 Archives Hub 中,PS 的谓语组数是其 PO 谓语组数的 2.6 倍,与 dbtune 中 PO 组数对 PS 组数的比例类似,但 PS 的谓语总数是 PO 谓语总数 15 倍,在 Linkedmdb 和 rdfabout 数据集中也有相同的结论,这是由于 PO 分组冗余增长受限的特性限制了谓语冗余的产生,无论在结构化程度高或低的数据集中,PO 分组限制谓语冗余生成的特性都在发挥作用。

不同算法的压缩结果比较如表 3 所示,其中,Plain 列表示将原始数据的 ID 图直接写入文件所需空间大小。显然,DGC 算法在所有数据集上均取得了最优结果。与 Plain 和 HDT 相比,其在不同数据集上都取得了超过 40% 的性能优化,这是因为 DGC 处理了 Plain 和 HDT 没有处理的谓语组合冗余,并利用差分编码减少了主语序列存储所需的大量空间。

表 3 不同算法的压缩结果比较

Table 3 Comparison of compression results by different algorithms MB

数据集	Plain 算法	HDT 算法	HDT ++ 算法	DGC 算法
Archives Hub	2.56	1.39	0.83	0.68
Linkedmdb	35.91	22.54	14.24	12.30
rdfabout	116.32	63.84	35.72	29.12
dbtune	400.36	242.05	132.10	101.64
DBpedia	3 497.36	1 839.08	1 523.72	1 210.40

由表 3 可见,DGC 在与 HDT ++ 的对比中也取得了平均 18% 的优化:在结构化程度低的数据集 Archives Hub、Linkedmdb、rdfabout 和 DBpedia 中,

PO 分组获得了冗余更少的谓语组合,差分编码进一步优化了压缩结果,最终两者结合后获得了平均 17% 的性能提升;在结构化程度高的 dbtune 数据集中,PO 分组相对于 PS 分组所带来更多冗余的负面影响被差分编码的优异效果所弥补,获得了 23% 的性能提升。同时,其在结构化程度更低的 Archives Hub、Linkedmdb、rdfabout 和 DBpedia 数据集的性能提升程度反而不及结构化程度更高的 dbtune,主要因为这 4 个数据集本身所包含的冗余较少,分组后谓语组合所需空间只占存储所需空间的较小部分,导致性能提升程度有限。

5　结束语

本文提出一种基于差分编码的 RDF 压缩算法。根据谓语对宾语进行分组,以减少谓语冗余,同时对分组后的主语序列进行差分编码,从而优化主语存储所需空间。实验结果表明,对于结构化程度不同的数据集,该算法较 Plain、HDT 和 HDT ++ 算法均能获得不同程度的性能提升。下一步将设计更具通用性的压缩算法以适用于更多具有不同结构化程度的数据集,并实现对压缩数据进行复杂查询的功能。

参考文献

[1] TIM B L, HENDLER J, LASSILA O. The semantic Web[J]. Scientific American, 2001, 284(5): 34-43.

[2] MANOLA F, MILLER R. RDF primer [EB/OL]. (2004-02-10)[2019-10-15]. http://www.w3.org/TR/rdf-primer/.

[3] DU Xiaoyong, WANG Yan, LÜ Bin. Research and development on semantic Web data management[J]. Journal of Software, 2009, 20(11): 2950-2964. (in Chinese)
杜小勇,王琰,吕彬. 语义 Web 数据管理研究进展[J]. 软件学报,2009,20(11):2950-2964.

[4] GUTIERREZ C, HURTADO C, MENDELZON A O, et al. Foundations of semantic Web databases[C]//Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. New York, USA: ACM Press, 2011: 520-541.

[5] SALOMON D. Data compression: the complete reference[M]. Berlin, Germany: Springer, 2007.

[6] WANG Xin, XU Qiang, CHAI Lele, et al. Efficient distributed query processing on large scale RDF graph data[J]. Journal of Software, 2019, 30(3): 498-514. (in Chinese)
王鑫,徐强,柴乐乐,等. 大规模 RDF 数据上高效率分布式查询处理[J]. 软件学报,2019,30(3):498-514.

[7] YUAN Pingpeng, LIU Pu, ZHANG Wenya, et al. A highly scalable RDF data storage system[J]. Journal of Computer Research and Development, 2012, 49(10): 2131-2141. (in Chinese)
袁平鹏,刘谱,张文娅,等. 高可扩展的 RDF 数据存储系统[J]. 计算机研究与发展,2012,49(10):2131-2141.

[8] WU Gang. Research on key technologies of RDF graph data management[D]. Beijing: Tsinghua University, 2008. (in Chinese)
吴刚. RDF 图数据管理的关键技术研究[D]. 北京:清华大学,2008.

[9] FERNÁNDEZ J D, MARTÍNEZ-PRIETO M A, GUTIERREZ C. Compact representation of large RDF data sets for publishing and exchange[C]//Proceedings of ISWC'10. Berlin, Germany: Springer, 2010: 193-208.

[10] FERNÁNDEZ J D, MARTÍNEZ-PRIETO M A, GUTIÉRREZ C, et al. Binary RDF representation for publication and exchange (HDT)[J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2013, 19: 22-41.

[11] HERNÁNDEZ-ILLERA A, MARTÍNEZ-PRIETO M A, FERNÁNDEZ J D. Serializing RDF in compressed space[C]//Proceedings of the 21st Data Compression Conference. Washington D. C., USA: IEEE Press, 2015: 1-5.

[12] ÁLVAREZ-GARCÍA S, BRISABOA N, FERNÁNDEZ J D, et al. Compressed vertical partitioning for efficient RDF management[J]. Knowledge and Information Systems, 2014, 44(2): 439-474.

[13] BRISABOA N R, CÁNOVAS R, MARTÍNEZ-PRIETO M A, et al. Compressed string dictionaries[C]//Proceedings of the 10th International Conference on Experimental Algorithms. New York, USA: ACM Press, 2011: 136-147.

[14] IANNONE L, PALMISANO I, REDAVID D. Optimizing RDF storage removing redundancies: an algorithm[C]//Proceedings of the 18th International Conference on Innovations in Applied Artificial Intelligence. New York, USA: ACM Press, 2005: 732-742.

[15] MEIER M. Towards rule-based minimization of RDF graphs under constraints[C]//Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems. Berlin, Germany: Springer, 2008: 89-103.

[16] JOSHI A, HITZLER P, DONG G. Logical linked data compression[C]//Proceedings of the 10th Extended Semantic Web Conference. Berlin, Germany: Springer, 2013: 170-184.

[17] VENKATARAMAN G, SREENIVASA KUMAR P. Horn-rule based compression technique for RDF data[C]//Proceedings of the 30th Annual ACM Symposium on Applied Computing. New York, USA: ACM Press, 2015: 396-401.

[18] ZHU M, WU W X, PAN J Z, et al. Predicate invention based RDF data compression[C]//Proceedings of the 8th Joint International Conference on Semantic Technology. Berlin, Germany: Springer 2018: 153-161.

[19] SMITH S W. The scientist and engineer's guide to digital signal processing[M]. [S. l.]: California Technical Publishing, 1997.

[20] FERNÁNDEZ J D, MARTÍNEZ-PRIETO M A, REDONDO P F, et al. Characterising RDF data sets[J]. Journal of Information Science, 2018, 44(2): 203-229.

编辑　金胡考