云环境中基于多属性排序的密文检索方案

冯贵兰[1] 谭 良[1,2]
(四川师范大学计算机学院 成都 610101)[1] (中国科学院计算技术研究所 北京 100190)[2]

摘 要 密文检索是云提供数据加密存储服务的重要辅助功能。目前云环境中密文检索的排序搜索算法只根据关键词的单一局部属性进行文档相关性分数计算,因此存在查准率不高的问题。针对这个问题设计了一种基于多属性排序的密文检索方案,其思想是首先通过文档关键词的局部属性和全局属性构建多属性特征向量安全索引,其次根据用户选择的排序方式确定各局部属性和全局属性的权值,然后调用多属性评分公式进行文档相关性分数计算,最后根据分数排序返回用户最感兴趣的检索结果。实验表明,该方法能够有效提高检索速度和检索结果的准确性。

关键词 数据加密,密文检索,云计算,多属性排序
中图法分类号 TP311 **文献标识码** A

Multi-attribute Ranked Keyword Search over Encrypted Cloud Data

FENG Gui-lan[1] TAN Liang[1,2]
(College of Computer, Sichuan Normal University, Chengdu 610101, China)[1]
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract Searching on encrypted data is an important auxiliary function for cloud. Confidentiality-preserving rank-ordered search algorithm computes document relevance scores with one keyword local attribute, so its precision is low. To solve this problem, a multi-attribute ranked keyword search over encrypted cloud data was introduced. Firstly, cloud service provider(CSP) builds secure index of multi-attribute feature vector based on keyword local and global attributes. Secondly, the local attribute and global attribute weights are determined by users' sort-by. And then, relevance score is calculated by multi-attribute ranking formula. Finally, CSP will return the interested results for the user. Experiments show that the method can improve the retrieval speed and the accuracy of search results effectively.

Keywords Data encryption, Search on encrypted data, Cloud computing, Multi-attribute ranked

1 引言

云计算是并行计算、分布式计算、网格计算的综合发展,或者说是这些计算科学概念的商业实现[1]。它主要通过将传统计算机系统的计算和存储职责从本地转移到云中,为用户节省大量的硬件成本,在业界受到了广泛的关注与认可。目前典型的云平台包括 Amazon Elastic Compute Cloud (EC2)[2],Google App Engine[3]和 Microsoft Live Mesh[4]。

随着云计算的发展,越来越多的敏感信息(例如个人医疗记录、财务信息、公司的重要文件等)被集中存储到云中[5],但当用户将数据交给云服务提供商后,人们就失去了对数据的直接控制力,可能导致个人隐私数据的泄漏和滥用。事实上,Google、Salesforce 和 MediaMax 等云服务提供商均发生过云存储泄密事件[6]。加密是保护用户数据秘密、隐私的一种较为有效的手段,然而数据加密后丧失了许多特性,导致对加密数据的检索是一项非常困难的工作。特别是在非可信环境中,如何对密文进行高效查询引起了人们普遍的关注。目前的大多数密文检索方案都不支持排序搜索,即检索出用户最需要的加密数据。尤其是在数据规模较大的云计算环境中,包含某一查询关键词的文档可能有很多个,如何在多个可能相关的文档中找出最相关的一个或若干个文档是急需解决的问题。针对上述问题,本文提出了一个基于多属性排序的密文检索方案 SESMA(Searchable Encryption Schemes based on Multi-attribute)。SESMA 可以根据用户选择的不同排序方式来进行区别排序,返回最符合用户需求的检索结果。多属性排序从关键词局部属性和全局属性等多个属性来综合体现文档特征,不仅充分考虑了同一关键词在不同文档下的差异,而且保证了文档质量和权威性。与现有方案相比,它在保证数据安全的前提下提高了查准率。

本文第2节回顾前人在该领域所作的相关工作;第3节具体介绍 SESMA 的总体模型;第4节介绍 SESMA 应用在云计算环境的具体实例;第5节介绍 SESMA 的性能评估和算法评价;最后总结全文。

2 相关工作

密文信息的检索始于2000年 Song 等人提出的线性搜索算法[7]。该算法利用伪随机序列和校验序列生成流密码,用流密码加密明文信息得到密文信息。在检索阶段,检索词和

到稿日期:2013-01-20 返修日期:2013-04-04 本文受国家自然科学基金(60970113),国家自然科学基金青年基金(60903073),四川省教育厅青年基金项目(08zb02)和四川师范大学校级项目(11KYL03)资助。
冯贵兰(1988—),女,硕士,主要研究方向为云计算、信息安全;谭 良(1972—),男,博士,教授,主要研究方向为可信计算、网络安全。

密文信息序列线性地进行模 2 加。若得到的结果满足检验关系，那么返回相应的密文信息。该方法的优点是加解密速度快，具有极强的抵抗统计分析能力。但是其在检索过程需逐次匹配密文信息，不适用于大规模数据的云计算环境。

Boneh 等人在 2004 年提出基于关键词的公钥加密算法[8]。其主要过程是首先生成公钥、私钥，然后使用公钥加密明文信息生成可检索的密文信息 E。检索过程中，运行 Trapdoor 算法，由关键词和私钥生成陷门 T。在服务器端通过对比陷门 T 和密文 E，返回相应的检索结果。该方案支持非属主用户的检索，其缺点是(1)服务器检索时需要进行大量的对数运算，检索效率很低，不适合数据规模较大的云计算环境；(2)加密后的关键词具有统计特性。

IBM 研究员 Gentry[9,10] 设计了一种基于理想格(ideal lattice)的全同态加密方案，使用户可以充分操作加密状态的数据。通过该方案，明文数据可以在不恢复明文信息的情况下被有效检索出来。但该方案太复杂且计算量太大，还不适合应用到云计算环境中。

2007 年，Swaminathan 等人提出保护隐私的排序搜索算法[11]。该算法使用保序加密[12]算法加密每一文档中关键词的词频信息，当检索请求提交到服务器端后，首先检索出含有关键词密文的加密文档，然后再将根据保序算法加密的词频进行排序处理，将评价值高的加密文档返回给用户。这种方法对多个可能相关的加密文档进行排序，把最相关的结果返回给用户，提高了检索正确率和返回率，减小了检索过程的通信量和计算量。随后，Cong 等人在其基础上提出了加密云数据安全排序搜索算法[13]，该算法根据 TFIDF 值对文档进行排序。此类算法计算量小、速度快，比较适合于云环境。但已有的文献[11,13]都只利用了文档中的关键词词频信息，仅从单个属性来衡量文档特征，它们具有如下两个缺点：(1)采用单个关键词局部属性对文档特征进行提取过于依赖关键词的重要性。(2)不考虑关键词全局属性，无法体现文档权威性。这两个缺点会影响文档的查准率。

本文针对排序搜索算法的上述问题，设计了一种基于多属性排序的密文检索方案，其思想是首先通过文档关键词的局部属性和全局属性构建多属性特征向量安全索引，其次根据用户选择的排序方式确定各局部属性和全局属性的权值，然后调用多属性评分公式进行文档相关性分数计算，最后根据分数排序返回用户最感兴趣的检索结果。

3 基于多属性排序的密文检索模型——SESMA

云计算环境中的数据存在规模大、使用频繁等特点，如何在此环境中快速准确地找到用户自己需要的数据是一个非常重要的问题。显然，在检索过程中使用需要逐次匹配密文信息的线性搜索算法和需要大量对数运算的公钥加密算法是不合适的，而已有的密文排序搜索算法尽管计算量小、速度快，但查准率不高，本文提出的基于多属性排序的密文检索模型弥补了这一缺点，其模型如图 1 所示。

该模型主要包括文档预处理和密文检索及相关性排序两大关键过程，文档预处理的主要功能是提取文档多属性特征向量，构建倒排索引。密文检索及相关性排序过程的主要功能是根据用户检索请求，使用多属性评分函数对文档相关性分数进行计算，返回排序结果。

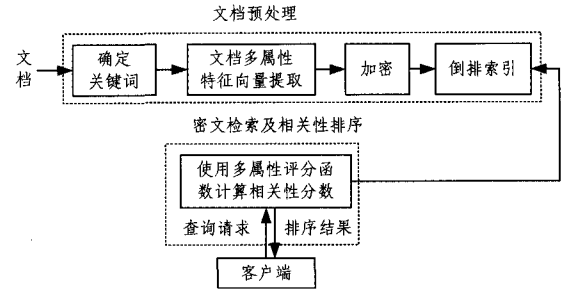

图 1 多属性排序的密文检索模型

3.1 确定关键词

首先介绍 SESMA 模型的第一步：确定关键词。该步骤以初始文档集为输入，通过分词的方法将词语从文档中切分出来，形成分词结果集合，再对分词结果集合过滤，从中提取出能准确反映文档语义的关键字。文档的分词主要是通过分词算法实现。文档分词完成后还需对分词结果进行处理。处理过程包括词条过滤和词频统计。处理完成后就提取到文档关键字集合 $K=(k_1,k_2,\cdots,k_n)$。

3.2 关键词多属性特征向量提取

其次介绍 SESMA 模型的第二步：关键词多属性特征向量提取，它是文档预处理的关键性步骤。其功能是根据文档关键词提取多属性特征向量。在介绍提取方法之前，我们先定义“关键词局部属性”、“关键词全局属性”、“局部属性特征向量”、“全局属性特征向量”、“单属性特征向量”、“多属性特征向”等概念。

定义 1(关键词局部属性) 关键词对于文档主题的贡献度受局部性因素的影响，将这些影响因素定义为关键词局部属性。如关键词的 TFIDF 值、词性、词长和位置等就是关键词局部属性。

定义 2(关键词全局属性) 关键词对于文档主题的贡献度受全局性因素的影响，将这些全局性的影响因素定义为关键词全局属性。如文档被引频次、下载次数等就是关键词全局属性。通常关键词全局属性即为文档属性。

定义 3(局部属性特征向量) 使用关键词局部属性的特征向量为局部属性特征向量。

定义 4(全局属性特征向量) 使用关键词全局属性的特征向量为全局属性特征向量。

定义 5(单属性特征向量) 若文档特征向量只使用了一个关键词属性，我们称之为单属性特征向量，如图 2 所示。

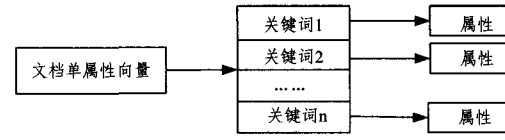

图 2 文档单属性向量

经典向量空间模型(Vector space Model，简称 VSM)[14]就是应用单属性特征向量的例子。向量空间中的 N 个文档可以用一个矩阵表示，如图 3 所示。

$$\begin{pmatrix} & T_1 & \cdots & T_t \\ D_1 & d_{11} & \cdots & d_{1t} \\ D_2 & d_{21} & \cdots & d_{2t} \\ \cdots & \cdots & \cdots & \cdots \\ D_n & d_{n1} & \cdots & d_{nt} \end{pmatrix}$$

图 3 单属性特征向量

图 3 中 $D_1, D_2, \cdots, D_n$ 表示 N 个文档，$T_1, \cdots, T_i$ 是能代表文档内容的特征项，它们可以是字、词、短语或者某种语义单元。矩阵中的一个元素对应于文档中一个词项的权重。“0”意味着该词项在文档中没有意义，或该词项不在文档中出现。常用的特征项权重计算方法为 TF-IDF 函数。

定义 6（多属性特征向量） 关键词局部属性和全局属性等多个属性组成的特征向量被定义为多属性特征向量，如图 4 所示。

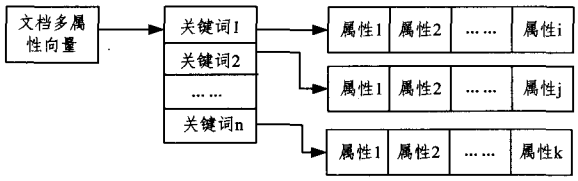

图 4 文档单属性向量

图 4 中关键 1，关键 2，…，关键词 n 是从文档中提取出的关键词，属性 1，属性 2，…，属性 n 代表关键词的局部属性和全局属性等多个属性。

关键词多属性特征向量提取的具体方法是根据关键词集合确定其局部属性和全局属性，然后形成多属性特征向量。

3.3 加密

然后介绍 SESMA 模型的第 3 步：加密。其功能是对待上传至服务器的文档集和多属性特征向量进行加密处理。

由于存储文档的服务器是不可信的，如果将文档和关键词多属性特征向量直接提交上传，那么将导致用户数据的安全和隐私受到威胁，因此需要进行加密处理从而保证信息在服务器端存储时的安全性。对称加密算法和非对称加密算法都可以实现对明文数据的加密。非对称加密算法具有较高的安全性，但是其加解密速度比对称加密算法要慢很多。对称加密算法加解密速度快，一般情况下能满足用户对安全性的要求，我们将采用对称加密算法来进行加密处理。

3.4 倒排索引

再次介绍 SESMA 模型的第 4 步：倒排索引。其功能是根据多属性特征向量建立倒排索引。

索引文件是全文检索的核心技术。在对全文建立索引方面，倒排索引（Inverted Index）[15] 应用最为广泛。倒排索引是描述一个关键字集合和一个文档集合之间的对应关系的数据结构。它是一种以关键词作为索引关键字和链表访问入口的索引结构，用来存储在全文检索下某个关键词在一个文档或者一组文档中的存储位置的映射。

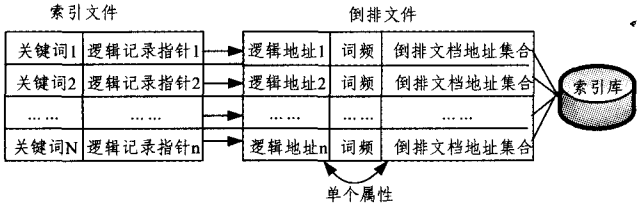

图 5 倒排索引模型示意图

如图 5 所示，传统的倒排索引结构由索引文件和倒排文件组成[16]。索引列表是文档中所有关键字的集合。它由一条条记录组成，每条记录包含关键词和该词对应的指针，指针指向倒排文件中相应的逻辑地址。倒排文件指出哪些文档含有这个关键字以及关键字的词频信息和这些文档的地址集合。从图中可以看出，通过查询词查找索引文件，再由索引文件指向倒排文件，倒排文件指向主文件，这样就可以很快地检索出关键词在哪些文档中出现了。

SESM 模型倒排索引的具体方法如图 6 所示。根据多属性特征向量建立倒排索引需要将原来倒排文件中的单属性词频改为词频、位置、被引频次和下次次数等多个属性。

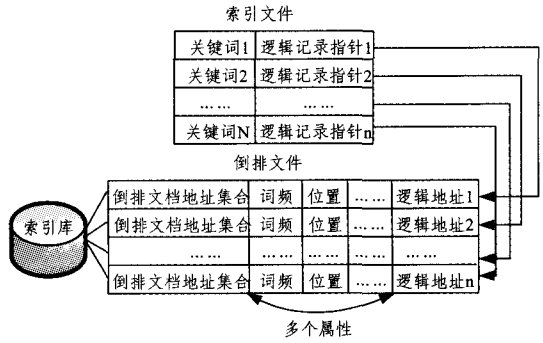

图 6 多属性倒排索引示意图

3.5 文档的评分函数

最后，SESMA 模型的第 5 步是密文检索及相关性排序。其功能是根据用户查询请求，调用多属性评分函数进行相关性排序，在这个步骤中最重要的是多属性评分函数。

在信息检索中，评分函数用于计算检索词和文档之间的相关性分数。文献[11]中的排序函数是基于关键词的频率（Term Frequency），文献[13]中的排序函数是基于关键词频率——逆文档率（TF-IDF，term frequency-inverse document frequency）。这类基于词频的单属性排序函数过于依赖关键词的重要性，使得检索查全率和查准率并不高。

本模型中的多属性评分函数将使用关键词局部属性和全局属性组成的多属性来计算文档相关性。其中每个属性的重要程度可能不同，从而引入属性权重，例如关键词 K 的属性 $\rho_1, \rho_2, \cdots, \rho_n$ 对应的权值分别为 $\gamma_1, \gamma_2, \cdots, \gamma_n$，并且 $\sum_1^n \gamma_i = 1$。文档排序算法的多属性评分函数表示为：

$$score = \sum_{i=1}^{n} \rho_i \times \gamma_i \tag{1}$$

式中，γ_i 是属性 ρ_i 的权重。在具体应用中，各个属性值的权值可以根据用户选择的不同排序方式进行动态调整。

4 SESMA 云环境应用实例

本节将 SESMA 模型应用于具体的云环境，以验证本模型的有效性和正确性。

4.1 总体流程与属性选择

在云计算环境中，SESMA 的基本架构如图 7 所示。其基本流程是首先 Owner 在客户端对要上传的文档进行预处理，为其提取局部属性特征向量，其次将文档、局部属性特征向量一起加密后存储到服务器中，再次由云服务器端提取全局属性特征向量，最后将局部属性特征向量和全局属性特征向量连结起来建立多属性特征向量安全索引。检索过程中，User 向服务器提交加密的查询请求，服务器根据用户选择的不同排序方式确定各个属性的权值后再调用多属性评分函数进行相关性计算，最后根据文档分数进行排序，将最符合用户

查询需求的结果返回给 User。

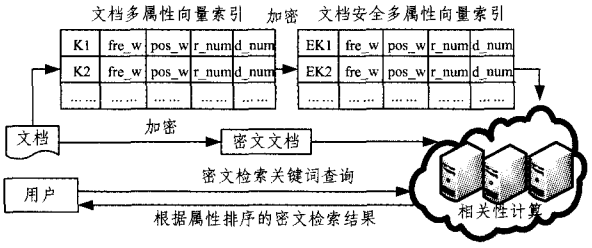

图 7　SESMA 架构及基本流程图

4.2　文档预处理

根据 SESMA 模型，第一过程就是对要加密上传的文档集合进行预处理。文档集的预处理主要包括客户端对文档生成的局部属性特征向量加密，云服务器端提取文档全局属性特征向量后形成多属性特征向量，再根据多属性特征向量为密文文档集建立密文索引表。

(1)关键词提取

以初始文档集(设为 d_i)为输入，提取出该文档的核心词汇集。采用正向最大分词法对文档页面进行自动分词。正向最大分词法的基本思想是假设词库中最长词条是 m 个字，取文档 d_i 的前 m 个字作为匹配词。查找词库，若词库中存在这样的长为 m 的一个词，则匹配成功，将前 m 个字作为一个词提取出来；若匹配不成功，则将匹配词的最后一个字去掉，用前 $m-1$ 个字进行重新匹配，如此循环，直到匹配成功。实际应用中，我们采用 MMSEG4J 实现分词。它是基于 Java 的开源中文分词组件，其 MMSEG 算法基于正向最大匹配实现中文分词，词语的正确识别率达到 98.41%。分词完成后，过滤掉与文档主题无关的一些语气词和低频词。最后将剩余的词作为文档的关键词提取出来，形成文档 d 的关键词集合 $K=(k_1,k_2,\cdots,k_n)$。

(2)文档多属性特征向量提取

首先在客户端提取文档局部属性特征向量。本实例采用关键词权值、位置权值作为局部属性。对于用户输入的搜索关键词，如果它在某文档中出现的频率越高，位置越重要，就认为该文档和关键词的相关性越好，也越能满足用户的需求。

1)关键词权值：本文计算关键词在文档中的权重采用 TF-IDF 算法[17]。TF 是指关键字在文档中出现的频率，IDF 是指关键字的逆文档频率。上传总数为 N 的文档集，该算法以关键词 K_i 在文档中出现的次数和含有关键词 K_i 的文档数的比值来衡量关键词 K_i 在文档 d 中的权重值，计算公式是

$$W_i=TF_i\times\log\left(\frac{N}{DF_i}\right) \tag{2}$$

式中，W_i 为关键词 K_i 在文档 d 中的权重值，TF_i 为关键词 K_i 在文档中出现的次数，N 是文档总数，DF_i 为含有关键词 K_i 的文档数。

2)位置权值：通常查询关键词在正文中出现的机会最大(即不是在标题、句首、居中等处)，而对在其它位置出现的情况，根据所处位置决定其权值大小。若关键词在标题处出现，则它出现在正文中的频率也很高，因为通常情况下正文都是围绕标题展开的，标题可认为是对正文的概括；若出现在文档居中的位置，那通常是文档某段正文内容的概括，相当于标题，则也说明是很重要的。根据表 1 可以对关键词出现的位置相应地赋予一定的权值。

表 1　关键词位置的权值参照表

关键词位置	权值	关键词位置	权值
标题	10	每段句首	5
每句开头	1.5	正文	1

客户端进行文档局部属性特征向量抽取的具体方法是以 d_i 的关键词为输入，生成文档局部属性特征向量。根据 d_i 的关键词分别计算词频权重 fre_w 和位置权重 pos_w。本文采用 TF-IDF 算法来计算词频权重(如式(2)所示)。通过式(2)和表 1 依次计算关键词集合 $K=(k_1,k_2,\cdots,k_n)$ 中各个关键词的词频权重和位置权重，提取出文档 d_i 的局部属性特征向量 $\vec{D}=\{(k_1,fre\_w,pos\_w),(k_2,fre\_w,pos\_w),\cdots,(k_n,fre\_w,pos\_w)\}$。

其次在云服务器端提取文档全局属性特征向量。本实例采用被引频次、下载次数作为全局属性。因为文档的被引频次、下载次数可以从整体上反映文档特征，体现文档权威程度，避免过于依赖关键词的重要性而无法保证文档的质量。

1)被引频次：本篇文档被其他文档引用的次数，与该文档质量的高度正相关。

2)下载次数：文档被其他用户下载的次数，它在直观上能够与该文档的被阅读次数相对应，从而下载次数可以反映文档的受欢迎程度。下载次数由服务器端进行记录。

云服务器端进行文档全局属性特征向量抽取的具体方法是根据文档 d_i 得到被引频次和下载次数，生成文档全局属性特征向量。

(3)客户端加密

为考虑系统的效率，采用加解密效率高的对称算法对待上传的文档集进行加密。局部属性特征向量中只有关键词部分可能泄露文档信息，词频权重值、位置权重和文档标识不会泄露文档信息，只需对关键词部分进行加密处理。采用 Trapdoor 算法对关键词部分进行加密，利用密钥 K 和关键词 W 作为输入，用 Trapdoor 算法计算可用于搜索关键词 W 的密文关键词 Ek。可得到加密后的局部属性特征向量 $\overrightarrow{Ep}=\{(Ek_1,fre\_w,pos\_w),(Ek_2,fre\_w,pos\_w)\cdots(Ek_n,fre\_w,pos\_w)\}$。最后将加密后的文档和加密局部属性特征向量一起上传到服务器中存储。

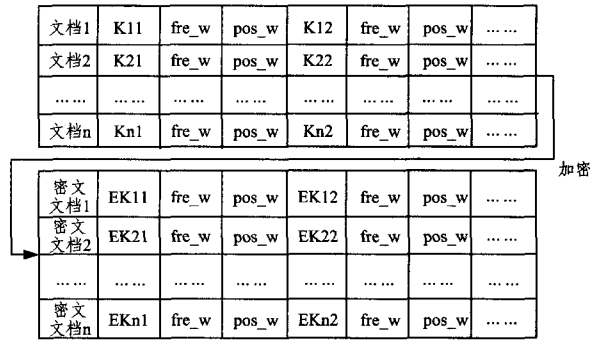

图 8　客户端数据加密示意图

图 8 中 K_{ij} 是用户分词后提取的明文关键字，EK_{ij} 是加密后生成的密文关键字。密文局部属性特征向量直接附加到每个文档后面，然后将密文文档和加密后的关键字、词频权值和位置权值提交给服务器。

(4)安全倒排索引构建

服务器接收到用户提交的密文数据后，首先得到该文档

的被引次数和下载次数，提取全局属性特征向量，再和用户提交的密文局部属性特征向量连结，形成密文多属性特征向量，其中包含的属性有词频权值、位置权值、被引次数和下载次数等4个，形如$\overrightarrow{Em}=\{(Ek_1, fre_w, pos_w, r_num, d_num), (Ek_2, fre_w, pos_w, r_num, d_num), \cdots, (Ek_n, fre_w, pos_w, r_num, d_num)\}$。云服务器端再根据密文多属性特征向量和文档标识号为密文文档集建立密文倒排索引，如图9所示。

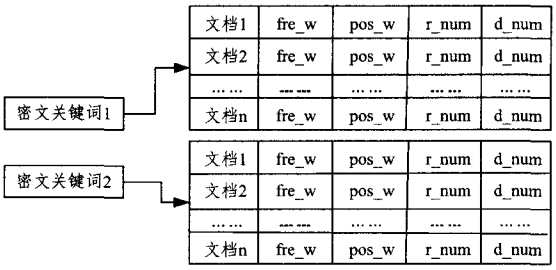

图9 安全多属性向量索引

至此，文档预处理中的功能就完成了。

4.3 密文检索及相关性排序

根据SESMA模型，第二过程是密文检索及相关性排序。其步骤是首先对用户检索请求进行处理，然后使用多属性评分函数对文档相关性分数进行计算，返回排序结果。

(1)检索词处理

本文目前只考虑单个关键词的情况。当用户想要查找某个关键词W时，首先依据密钥K和查询关键词W，采用Trapdoor算法生成关于关键字W的陷门值T_w。

然后，用户还可以选择不同的排序方式，如：默认、被引频次、下载频次等。不同排序方式使得各个属性值在评分函数中的权值不同。因为关键词出现的频率最能代表该词在文档中的重要程度，所以关键词词频权值无论在哪种排序方式上都占有最大比重。本文根据实验数据的百分比，认为不同排序模式的各个属性权值如表2所列。

表2 排序模式的属性权值参照表

排序模式\权值名称	关键词词频权值	位置权值	被引频次权值	下次次数权值
默认排序模式	0.5	0.2	0.15	0.15
被引频次排序模式	0.35	0.2	0.3	0.15
下载次数排序模式	0.35	0.2	0.15	0.3

最后向服务器提交关键字W的陷门值T_w和排序方式。

(2)密文检索与相关性排序阶段

服务器通过用户提交的陷门值T_w，查找密文索引表，获得初步的密文检索结果。然后根据用户选择的排序方式调用多属性评分函数进行相关性分数计算，进而优化初步密文检索结果，返回最符合用户需求的检索结果，具体流程如图10所示。

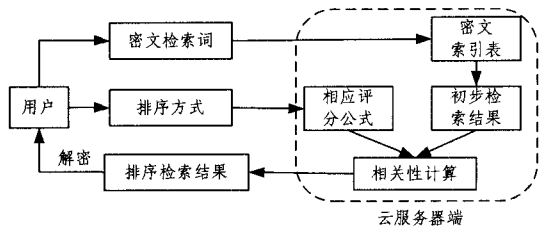

图10 密文检索与相关性排序

首先根据不同的排序模式，按照表2确定各个属性的权重值，然后调用3.5节中介绍的多属性评分函数计算文档相关性分数，最后根据相关性分数大小进行排序处理，把分数值最高的加密文档返回给用户。

5 性能分析及算法评价

5.1 性能测试

5.1.1 实验环境

实验的硬件平台环境为因特尔双核E7500处理器、2G内存，操作系统为Windows XP 32bit，语言开发环境为全文检索引擎工具包Lucene-3.6.1、开源Java PDF工具包PDFBox-0.7.3、Apache-tomcat-7.0.23、MyEclipse9.0、JDK1.7。

5.1.2 评价指标

实验采用的评测指标为信息检索系统中常用的召回率(Recall Rate)、准确率(Precision Rate)、平均查准率MAP(Mean of Average Precision)和$P@n$(返回前n个结果的准确率)。

(1)召回率的计算公式如下：

$$召回率=\frac{系统检索到的相关文件}{系统所有相关的文件总数} \tag{3}$$

(2)准确率的计算公式如下：

$$准确率=\frac{系统检索到的相关文件}{系统所有检索到的文件总数} \tag{4}$$

(3)MAP

单个主题的平均准确率是检索出每篇相关文档后的准确率的平均值。主题集合的平均准确率(MAP)是每个主题的平均准确率的平均值。MAP是反映系统在全部相关文档上性能的单值指标。系统检索出来的相关文档越靠前，MAP就可能越高。如果系统没有返回相关文档，则准确率默认为0。MAP的计算公式如下：

$$MAP=\sum_{i}^{r}\frac{1}{r}=1\frac{i}{第i个文档的位置} \tag{5}$$

(4)$P@n$

单个主题的$P@n$是系统对于该主题返回的前n个结果的准确率。主题集合的$P@n$是每个主题的$P@n$的平均值。本实验n分别取15、20、30。$P@n$的计算公式如下：

$$P@n=\frac{前n个检索结果中相关文档的个数}{n} \tag{6}$$

5.1.3 实验结果

在中国知网上随机选取了50篇文档，设置所有文档的全局属性，例如：被引次数和下载次数。对每个文档做统一预处理：中文分词、过滤停用词。在构建查询时本实验使用title、content、keywords、summary域作为查询信息。

(1)实验1：检索查询效果对比

使用相同的Query分别在单属性密文检索系统和多属性密文检索系统下进行查询，分别返回排在前15、20、30的结果，计算P/R值，进一步获得$P@10$、MAP等评价指标值，对系统进行评估。实验评测结果如表3和图11所示。

表3 实验评测结果

	MAP	P@15	P@20	P@30
单属性密文检索	0.8861	0.8000	0.7000	0.6333
多属性密文检索	0.9508	0.8667	0.8000	0.6667
提高百分比	6.47%	6.67%	10%	3.33%

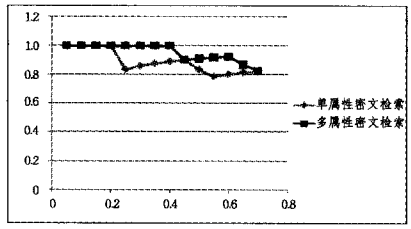

图 11　P/R 图比较

以上的实验评测对比结果表明，相比单属性密文检索系统，多属性密文检索系统在 MAP 上有了 6%左右的提高，通过 P/R 图可以看到基于多属性密文检索系统的查准率略优于前者。虽然调用多属性评分函数进行分数计算时涉及到更多的属性，但是却因为分数计算的精确性使大大缩短了排序时间，所以多属性密文检索的检索时间会优于单属性密文检索。

(2)实验 2：检索时间对比

使用相同的 Query 分别在单属性密文检索系统和多属性密文检索系统下进行查询，统计返回排在前 10、20、30、40 和 50 的结果时间(单位 ms)，比较其检索时间，如图 12 所示。

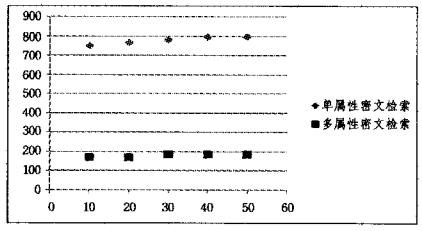

图 12　检索时间对比图

从图 12 中可以看到，单属性密文检索和多属性密文检索的检索时间与文档规模呈近似正比例关系，检索时间随着文档的增加而增加。在相同文档规模下，多属性密文检索比单属性密文检索的检索时间大大缩短了，由此可见多属性密文检索这种方法是可取的。

(3)实验 3：属性数量对检索质量的影响对比

采用相同的 Query 分别对使用 1 个属性(关键词词频)、2 个属性(关键词词频、位置)、3 个属性(关键词词频、位置和引用次数)、4 个属性(关键词词频、位置、引用次数和下载次数)的密文检索系统进行查询，计算其 $P@15$、$P@20$、$P@30$。实验评测结果如图 13 所示。

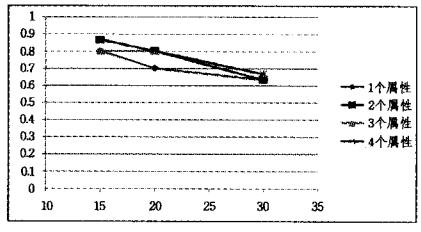

图 13　查准率对比图

从图 13 中可以看到，在同样的返回结果下，属性个数越多，查准率相对较高，但是其查准率影响的程度和选择的属性对文档贡献程度正相关。

以上的实验可以充分表明，本文提出的基于多属性排序的密文检索方案通过结合文档多属性特征来提高检索准确率的方法是可行有效的。

5.2　算法评价

一方面，在检索过程中，相对于使用需要逐次匹配密文信息的线性搜索算法和需要大量对数运算的公钥加密算法，本文提出的基于多属性排序的密文检索方案比较适合于云计算环境。具体分析如下：

(1)检索速度快，该方案在检索过程中只需要查找安全索引表，而不用像线性搜索算法那样逐次匹配密文信息，因此检索速度较快，非常适用于数据规模较大的云计算环境。

(2)计算量较小，在该方案中当检索请求提交到云服务器端后，首先检索出含有关键词密文的加密文档，然后根据多属性排序公式计算文档相关性分数，最后根据分数排序返回用户最感兴趣的加密文档。方案中多属性排序公式中的计算是线性的，与公钥加密算法检索时大量对数运算和全同态加密方案中的计算相比，计算量大大减小，适用于数据使用频繁的云计算环境。

(3)降低了网络通信量。该方案只返回最可能相关的文档，大大降低了网络负载。

(4)提高了用户满意度。我们的方案可以在给定多个可能相关文档的情况下对加密文档进行排序，进而把最可能相关的文档返回给用户，一定程度上提高了用户满意度。

另一方面，本文提出的基于多属性排序的密文检索方案相对于已有的同类单属性密文排序搜索算法性能更佳，分析如下：

(1)提高了查准率。本方案引入了包括关键词全局属性和局部属性的多属性特征向量，这使得在文档相关性分数计算时能从多个方面、多个角度综合评价文档，从而返回最符合用户需求的文档。

(2)缩短了检索时间。虽然调用多属性评分函数进行分数计算时涉及到更多的属性，但是却因为分数计算的精确性大大缩短了排序时间，从而减少了检索时间。

结束语　本文介绍了基于多属性排序的密文检索系统，采用文档多属性特征向量构建安全索引，结合文档关键词局部属性和全局属性等多属性进行相关性分数计算，返回用户最感兴趣的检索结果，一定程度上提高了用户对检索结果的满意度。实验结果证明了本文提出方法的可行性和有效性。尽管如此，本文提出的改进方法还存在不足之处：该方法中含有几个参数，需要不断实验来找到更为合理的值，以进一步提高检索的精度；在文档相关性分数计算算法中，还可以考虑查询串和检索文档中的关键词的相似程度，这将是本文下阶段研究所要做的工作。

参 考 文 献

[1]　张建勋，古志民，郑超. 云计算研究进展综述[J]. 计算机应用研究，2010，27(2)：429-433

[2]　Amazon Elastic Compute Cloud(EC2)[OL]. http://www.amazon.com/ec2/，2008-07-18

[3]　Google App Engine [OL]. http://appengine.google.com [18 Jul 2008]

[4]　Microsoft Live Mesh[OL]. http://www.mesh.com，2008-07-18

[5]　Kamara S，Lauter K. Cryptographic cloud storage [C]// Proceedings of Financial Cryptography：Workshop on Real-Life CryptographicProtocols and Standardization 2010. January 2010

(下转第 157 页)

实在分析 TBFL 方法性能方面有重要作用。

结束语 运用测试集对程序错误语句定位问题，除了上述所提及的几个主要方法，许多学者提出了许多其它行之有效的方法。但大多冗余测试用例都会伤害这些方法的功效。于是就出现了类似于 SAFL 这样为了消除冗余影响的方法。这些方法都用一些实例，或者证明其方法的功效，或者说明该方法怎样解决冗余测试的伤害问题。当然作为一门面向实际的学科，无论怎样强调实验数据都是正确的。不过科学哲学告诉我们，方法只能用理论来证明，实验只是方法的正确性的一种"验证"手段。为此，我们想在一般意义上构造模型来研究上述问题。用测试用例测试程序，其结果应该看成程序语句里透露出来的信息，考虑到熵的概念和信息有关，我们选择信息论理论，构造一个熵模型。这个模型是一个框架，并在一个特殊实例上对 Dicing 方法、TARANTULA 方法、SAFL 方法进行了熵分析，得出了有意义的结果，从理论上基本肯定了文献[3]的结论。

我们今后的工作是为 TBFL 方法建立信息论模型，即认为 X，T 都是随机变量，并为(X，T)的联合分布建模，从而利用 $H(X)$、$H(T)$、$H(X|T)$等工具对 TBFL 方法进行分析。另外冗余测试是否会损害 TBFL 方法的功效，这在直观上是合理的，如同思考线索太多往往影响思路以致找不到问题的关键。但是增加测试在理论上应该有利于问题的解决，所以是否存在这样的信息论方法，当向冗余度较少的测试集里增添新的测试用例时(通常会出现更多的冗余用例)，使得该方法不仅不会对错误定位产生副作用，反而有利于寻找出软件的错误。这也是我们今后研究的另一个目标。

参 考 文 献

[1] Agrawal H, Horgan J, London S, et al. Fault location using execution slices and dataflow tests [C]// IEEE Software Reliability Engineering. 1995: 143-151

[2] Cleve H, Zeller A. Locating causes of program failures [C]// Proceedings of the 27th International Conference on Software Engineering. 2005: 342-351

[3] Hao D, Zhang L, Pan Y, et al. On similarity-awareness in testing-based fault localization [J]. Automated Software Engineering, 2008, 15(2): 207-249

[4] Haykin S. Neural networks-A comprehensive foundation [M]. Beijing: Tsinghua University Press, 2001: 484-508

[5] Kyriazis A, Mathioudakis K. Enhance of fault localization using probabilistic fusion with gas path analysis algorithms [J]. Journal of Engineering for Gas Turbines and Power, 2009, 131(5): 51601-51609

[6] Jones J A, Harrold M J, Stasko J. Visualization of test information to assist fault localization [C]// Proceeding of the 24th International Conference on Software Engineering. 2002: 467-477

[7] Jones J A, Harrold M J. Empirical evaluation of the tarantula automatic fault-localization technique [C]// Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. 2005: 273-282

[8] Liblit B, Naik M, Zheng A X, et al. Scalable statistical bug isolation [C]// Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI). 2005: 15-16

[9] Schach S R. Object-oriented classical software engineering [M]. Beijing: China Machine Press, 2007: 490-193

[10] Liu C, Yan X, Fei L, et al. SOBER: statistical model-based bug localization [C] // Proceedings of the 13the ACM SIGSOFT Symposium on Foundations of Software Engineering. 2005: 286-295

[11] Renieris M, Reiss S P. Fault localization with nearest neighbor queries [C]// Proceedings of the 18th International Conference on Automated Software Engineering. 2003: 30-39

[12] Zeller A. Isolating cause-effect chains from computer programs [C]// Proceedings of the 10th ACM SIGFOFT Symposium on Foundations of Software Engineering. 2002: 1-10

(上接第 136 页)

[6] Huang R W, Gui X L, Yu S, et al. Study of privacy preserving framework for cloud storage[J]. Computer Scienceand Information Systems, 2011, 8(3): 801-819

[7] Song D, Wagner D, Perrig A. Practical techniques for searcheson encrypted data[C]// Proc. of IEEE Symposium on Security and-Privacy'00. 2000

[8] Boneh D, Crescenzo G D, Ostrovsky R, et al. Public keyencryption with keyword search[C]// Proc. of EUROCRYP'04, volume3027 of LNCS. Springer, 2004

[9] Gentry C. Fully homomorphic encryption using ideal lattices [C]// Proceedings of the 41st ACM Symposiumon Theory of Computing(STOC09). Bethesda, Maryland, USA, 2009: 169-178

[10] Gentry C. Computing arbitrary functions of encrypted data[J]. Communications of the ACM, 2010, 53(3): 97

[11] Swaminathan A, Mao Y, Su G-M, et al. Confidentiality-preserving rank-ordered search[C]// Proc. of the Workshop on Storage Security and Survivability. 2007

[12] Boldyreva A, Chenette N, Lee Y, et al. Order-preservingsymmetric encryption[C]// Proceedings of Eurocrypt'09, volume 5479 of LNCS. Springer, 2009

[13] Wang C, Cao N, Li J, et al. Secure ranked keywordsearch over encrypted cloud data[C]// Proc. of ICDCS'10. 2010

[14] Salton G, Wong A, Yang C S. A Vector Space Model for Automatic Indexing[J]. Communications of the ACM, 1975, 18: 613-620

[15] Zobel J, Moffat A. Inverted files for text search engines [J]. ACM Computing Surverys, 2006, 38(2): 16-31

[16] Rieardo B Y, Berthier R N. ModernInformationRetrieval [M]. NewYork: ACM Press, 1999, 192: 5-10

[17] Salton G, Clement T Y. On The Construction of Effective Vocabularies for Information Retrival[C]// Proceeding of The 1973 Meeting on Programming Languages and Information Retrieval. New York, ACM, 1973: 11