一种基于潜在语义索引的“垃圾”邮件过滤方法

陈华辉
(宁波大学 计算机与自动化科学工程系，浙江 宁波 315211)

摘　要：提出了一种基于潜在语义索引(LSI)的“垃圾”邮件过滤方法，讨论了邮件概念空间的构造，该空间中邮件相似度的计算和“垃圾”邮件的过滤。

关键词：信息检索；电子邮件；潜在语义索引；过滤
中图分类号：TP391.1　TP393.098　　文献标识码：A　　文章编号：1001-3695(2000)10-0017-02

1　引言

随着因特网的越来越普及，电子邮件(E-mail)正成为一种最快捷、最经济的通信手段。但电子邮件在成为一种信息交流工具的同时，也正在成为一种商业广告手段。在收到有用信息的同时，用户也从因特网上收到各种各样广告邮件，使用户要花费大量时间和精力来处理这些所谓的“垃圾”邮件。因而需要有能自动对这类“垃圾”邮件进行过滤的处理软件。实际上，目前已有不少过滤软件，可按用户设定的过滤规则工作，如滤掉包含某些字或词的邮件，或滤掉发自某类地址的邮件等。但这类软件一般只是单纯采用信息检索中关键字匹配检索方法，来过滤包含这些词的邮件。而自然语言中词的多义性和同义性所带来的关键词检索方法的固有缺陷，使这类过滤软件的正确过滤能力大为减弱。

潜在语义索引(Latent Semantic Indexing，LSI)是向量检索方法的一种，用于文档的检索有着较好的效果[1,2]。我们将其应用于“垃圾”邮件过滤中。本文介绍这种基于潜在语义索引的“垃圾”邮件过滤方法的基本思想，先对LSI方法的基本思想作了介绍，然后讨论了电子邮件概念空间的构造，该空间中邮件相似度的计算和“垃圾”邮件的过滤。

2　LSI方法

典型的信息检索采用将要查询的词和文档中的词进行匹配的方法。但词的同义性会导致同一个概念在查询和文档中用两个不同词来表达，从而使查询不能和相关文档匹配；而词的多义性又会使不同的概念用同一词来表达，从而使查询和不相关的文档匹配。LSI方法通过引入概念空间来解决上述问题。该方法考察词出现的上下文的相似性，出现在相似上下文中的词，被认为在用法或者说含义上相近，在概念空间中也相互接近。LSI用此概念空间进行检索。

为实现上述思想，首先要构造一词语-文档矩阵[3]

$$A=[a_{ij}]_{m\times n} \tag{1}$$

其中a_{ij}为词i在文档j中出现的频率。因为一个词不会出现在每个文档中，所以A一般是一高阶稀疏矩阵。在实际应用中，根据各词语重要性的不同也可加上权值，记

$$a_{ij}=L(i,j)\times G(i) \tag{2}$$

其中$L(i,j)$为词i在文档j中的局部权重、$G(i)$为词i的全局权重。

应用奇异值分解[4](Singular Value Decomposition，SVD)对矩阵A进行分解。设$m\geq n$，$\mathrm{rank}(A)=r$，则A的奇异值分解，记成SVD(A)，定义为

$$A=U\Sigma V^T \tag{3}$$

其中$U^TU=V^TV=I_n$，对角阵$\Sigma=diag(\sigma_1,\sigma_2,\ldots,\sigma_n)$，且有，$\sigma_1\geq\sigma_2\geq\ldots\geq\sigma_r>\sigma_{r+1}=\ldots=\sigma_n=0$，U和V的列分别被称为矩阵A的左右奇异向量，Σ被称为矩阵A的奇异值标准形，Σ的对角元素被称为矩阵A的奇异值。

对SVD分解有如下定理：

设矩阵A的SVD分解由式(3)给出，且$r=rank(A)\leq p=\min(m,n)$，$U=(u_1,u_2,\ldots,u_m),V=(v_1,v_2,\ldots,v_n)$

则
$$A=\sum_{i=1}^{r}u_i\cdot\sigma_i\cdot v_i^T \tag{4}$$

若$k<r$，且记

$$A_k=\sum_{i=1}^{k}u_i\cdot\sigma_i\cdot v_i=U_k\Sigma_kV_k^T \tag{5}$$

其中

$U_k=(u_1,u_2,\ldots,u_k)$, $V_k=(v_1,v_2,\ldots,v_k)$, $\Sigma_k=diag(\sigma_1,\sigma_2,\ldots,\sigma_k)$

则
$$\min_{rank(B)=k}\|A-B\|_2=\|A-A_k\|_2=\sigma_{k+1} \tag{6}$$

也就是说，在2-范数意义下，A_k是和A最接近的k秩矩阵。

将SVD应用到LSI方法中，对由式(1)、(2)给出的词语-文档矩阵A进行SVD分解，如图1所示。

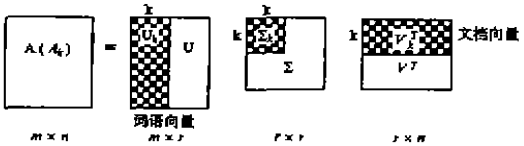

图1　矩阵A的SVD分解

图中U、V、Σ中阴影部分的乘积即表示A_k。分解后各参数可作如下解释：

A_k：最接近A的k秩矩阵

U：词语向量　U_k：压缩至k维空间的词语向量

收稿日期：2000-03-10

m：词语数 V：文档向量

V_k：压缩至k维空间的文档向量 n：文档数

Σ、Σ_k：奇异值 k：因子数 r：A的秩

A_k是对A的一个近似，且在某种意义上可以说A_k保持了A中所反映的词语和文档之间联系的内在结构(潜在语义)，但又去掉了因用词习惯或语言的多义性等带来的“噪声”。直观地说，因k比文档中总的词语数m小得多，词义上的细微区别被忽略掉了。例如，在类似文档中出现的词，在k维词语空间中也会比较接近。将此k维空间理解成概念空间，则表示了这些词在概念上是相似的或同义的。考虑词“电脑”、“计算机”、“程序”、“大象”。这里“电脑”、“计算机”是同义词，“程序”是一相关的概念，而“大象”不相关。在大多数检索系统里，若文档中没有直接出现词“电脑”，则对关于“计算机”的文档中查“电脑”和对关于“大象”的文档中查的结果是一样的，都不会命中。但对一个理想的检索系统来说，查询“电脑”应该会把关于“计算机”的文档也找出来，或把关于“程序”的文档也找出来，只是相关程度低一点，但不会把关于“大象”的文档找出来。导出的k维概念空间能表示词之间的这些有用的内在联系。大致地说，和词“电脑”在文档中同时出现的词中有许多也会出现在词“计算机”出现的文档中，如“硬件”、“软件”、“网络”、“操作系统”、“显示器”、“键盘”、“CPU”、“程序员”等，因而它们在k维空间中会有类似的表示，而“程序”的上下文语境会和“计算机”、“电脑”在某种程度上一致，而“大象”的上下文就会完全不一样。因而在k维空间表示中，“程序”和“电脑”、“计算机”更接近，而和“大象”会差较远。LSI的基本思想是把词和词的内在联系表示出来，并用来更好地进行检索。

3 邮件概念空间的构造

将邮件作为一种文档，LSI方法的思想也可应用于电子邮件过滤中。此时电子邮件的k维概念空间可按以下思想构造。

首先，为使k维概念空间反映电子邮件的语言环境，挑选一训练邮件集。对该集合中的邮件构造词语-邮件(文档)矩阵A，经SVD转换后获得电子邮件的k维概念空间，我们称之为邮件概念空间。

第二步是在初步构造的邮件概念空间基础上，加入待过滤的新邮件。

这里第一步的做法同上一节中讨论的一般LSI方法，下面对第二步作进一步讨论。为完成这一步，对每个新的邮件，均应将其表示成k维空间中的向量。设该邮件相应的m×1邮件文档向量为d，则其在k维空间中的向量可表示成

$$d^* = d^T U_k \Sigma_k^{-1} \tag{7}$$

每个新邮件的向量均应附加到V_k的列上，如图2所示，其中阴影部分表示新加的邮件文档向量。

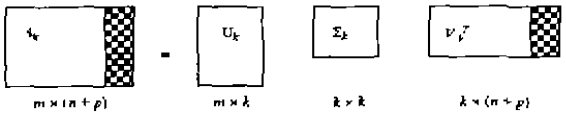

图2 在k维空间上加入p个新邮件

类似地，也可在k维邮件概念空间中加入新的词语。对每个新加的词语，设其相应的1×n词语向量为t，则在k维空间中的向量可表示成

$$t^* = tV_k\Sigma_k^{-1} \tag{8}$$

新加入的词语向量应附加到U_k的行上(如图3)。

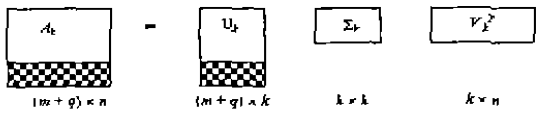

图3 在k维空间上加入q个新词语

大量新词语的加入会使k维邮件概念空间上的查询、过滤性能下降，因而要求初始的训练集应足够大，也可当新加词太多时重新进行SVD计算。

4 邮件相似度的计算

在构造的k维邮件概念空间中可比较词和词、词和邮件、邮件和邮件之间的相似度，根据相似度可进行邮件的查询、分类和过滤。下面对词之间的相似度、邮件之间的相似度两方面的计算进行讨论。

4.1 词的相似度

设词t_i和词t_j分别对应词语-邮件矩阵A的第i行和第j行，在k维邮件概念空间中，分别对应矩阵A_k的第i行和第j行。记$A_k = [\tilde{a}_{ij}]_{m,n}$，则词$t_i$和词$t_j$在k维邮件概念空间中的向量分别为

$$\vec{t}_i = (\tilde{a}_{i1}, \tilde{a}_{i2}, \ldots, \tilde{a}_{in}) \tag{9}$$

$$\vec{t}_j = (\tilde{a}_{j1}, \tilde{a}_{j2}, \ldots, \tilde{a}_{jn}) \tag{10}$$

其相似度$Sim(\vec{t}_i, \vec{t}_j)$定义为$\vec{t}_i$和$\vec{t}_j$的点积，即

$$Sim(\vec{t}_i, \vec{t}_j) = \vec{t}_i \cdot \vec{t}_j = \sum_{k=1}^{n} \tilde{a}_{ik} \cdot \tilde{a}_{jk} \tag{11}$$

而对全部m个词，其两两之间的相似度为

$$A_k \cdot A_k^T = (U_k\Sigma_k V_k^T)\cdot(U_k\Sigma_k V_k^T)^T = (U_k\Sigma_k)\cdot(U_k\Sigma_k)^T \tag{12}$$

因而$Sim(\vec{t}_i, \vec{t}_j)$的计算可由矩阵$U_k\Sigma_k$的第i行和第j行的点积得到。若将$U_k\Sigma_k$的行看成是k维空间中的向量，则这些向量之间的点积表示了词的相似度。因Σ_k是一对角阵，对k维空间中的坐标进行适当的缩放即可用U_k代替$U_k\Sigma_k$来构造词语在k维邮件概念空间中的向量而不影响各词语向量间的相似度。因而在k维邮件概念空间中，将U_k理解成词语向量。

4.2 邮件的相似度

类似词的相似度，邮件相似度为

$$A_k^T \cdot A_k = (U_k\Sigma_k V_k^T)^T \cdot (U_k\Sigma_k V_k^T) = (\Sigma_k V_k^T)^T \cdot (\Sigma_k V_k^T) \tag{13}$$

即邮件i和邮件j的相似度可由矩阵$\Sigma_k V_k^T$的第i列和第j列的点积得到。同样因Σ_k是一对角阵，因而可用V_k^T代替$\Sigma_k V_k^T$来构造邮件在k维邮件概念空间中的向量。在k维邮件概念空间中，将V_k^T理解成邮件向量。

(下转第35页)

开发环境，这里以一个典型的大规模计算型的Java程序为例来分析该环境的能力。

所分析的程序是LinpackLoop的Java版本，运行时使用了LAN中三台工作站并行处理，采用的消息传递平台是Java Parallel Virtual Machine (JPVM)平台[2]。

源程序经编译后显示如图3所示，界面左侧是源程序，右侧是任务依赖图中的某一层。三角形结点代表头任务或尾任务，圆形结点代表简单任务或循环任务，方形结点代表复合任务(此层中无复合任务)。通过鼠标双击方形结点显示该结点的下一层，而点击RETURN按钮则返回上一层。

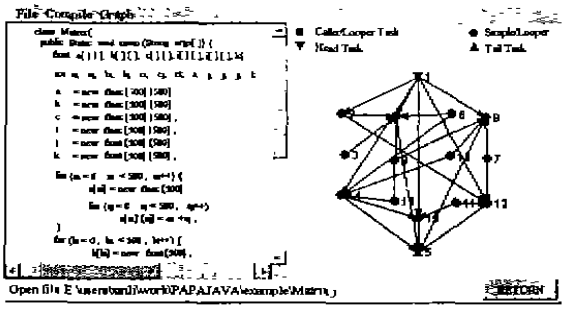

图3　任务依赖图

运行过程中任务状态的显示，我们采用了“不同状态不同颜色”的方式，即在收到消息后将消息中任务对应结点所在的层作为当前层显示出来，并对该结点的颜色作相应改变。对于处理器的分配，由于每次运行时使用的处理器都不尽相同，无法事先确定，为此我们采用动态列表的方式，即运行中动态地将首次使用的处理器加入列表作为新的一项显示，而已加入的处理器则直接将运行其上的任务显示于同一项，这样，所用的处理器及分配其上的任务便一目了然了。

图4是程序运行过程中某一时刻各任务状态及处理器分配的快照。

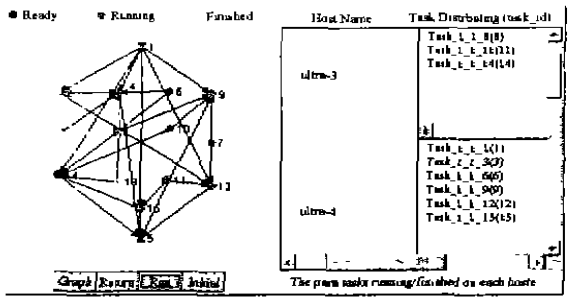

图4　处理器分配及任务分布图

最后，为了用户对并行运行的结果有较精确的认识，该系统记录了各任务的运行时间和整个程序的总运行时间，并在运行结束时提供给用户，如图5所示。

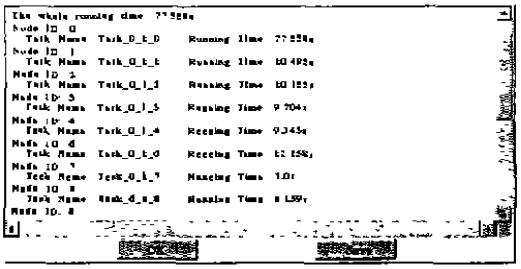

图5　运行时间报告

4　结束语

本文所实现的可视化环境在JAPS中的应用，较好地满足了用户对信息获取的需求，使得该并行系统更加易于使用。进一步的工作将包括信息显示方式上的多样化，对出错信息的完善等。

参考文献：

[1] [美]Bruce Eckel. Java编程思想[M]. 北京：机械工业出版社，1999.

[2] 杜建成，陈道蓄，谢立. JAPS：一个基于JAVA的程序自动并行化系统[J]. 中国科学(E辑)，1999，29(3)

[3] 石教英，蔡文立. 科学计算可视化算法与系统[M]. 北京：科学出版社，1996.

[4] 程景云，倪亦泉，等. 人机界面设计与开发工具[M]. 北京：电子工业出版社，1994.

(上接第18页)

5　邮件的过滤

要完成“垃圾”邮件的过滤，首先应规定一用来判定邮件是否为“垃圾”邮件的过滤向量。记过滤向量为f，则

$$f=[f_i]_{m\times 1} \tag{14}$$

其中f_i为词语i在过滤向量中的权重。在“垃圾”邮件中经常出现的词，如“免费”、“Free”、“赚钱”、“推荐”、“好机会”等，在f中将有较大的权重。要完成过滤，首先应将f在k维邮件概念空间中表示出来。和新邮件加入类似，f在k维邮件概念空间中的表示为

$$f^*=f^TU_k\Sigma_k^{-1} \tag{15}$$

这样可在k维邮件概念空间上将f^*和其它邮件向量进行相似度计算，并按相似度高低排列邮件，相似度超过某一值的邮件即可认为是“垃圾”邮件。

6　结束语

本文提出了一种基于潜在语义索引的“垃圾”邮件过滤方法。初步的实验表明，其过滤精度要高于单纯按关键字匹配方法进行的过滤。

参考文献：

[1] S Deerwester, S T. Dumais, et al. Indexing by Latent Semantic Analysis[J]. Journal of the American Society for Information Science, 1990, 41(6): 391-407.

[2] S T Dumais. Latent Semantic Indexing (LSI) and TREC-2[C]. In D. Harman, ed. The Second Text Retrieval Conference (TREC2), National Institute of Standards and Technology Special Publication, 1994. 105-116.

[3] G Salton, M J McGill. Introduction to Modern Information Retrieval[M]. NewYork: McGraw-Hill, 1993.

[4] 姜家辉. 矩阵理论基础[M]. 大连理工大学出版社，1995.