Software Engineering and Applications 软件工程与应用, 2016, 5(1), 65-74
Published Online February 2016 in Hans. http://www.hanspub.org/journal/sea
http://dx.doi.org/10.12677/sea.2016.51008

Hans汉斯

Large Scale Web Page Classification Algorithm Based on Spectral Hashing

Dandan Tian

College of Computer Science, National University of Defense Technology, Changsha Hunan
Email: 1240074883@qq.com

Received: Feb. 2nd, 2016; accepted: Feb. 22nd, 2016; published: Feb. 25th, 2016

Abstract

Nowadays, network information has been covered in all aspects of our lives, but with the development of the network, the problem of network information overload has become more and more prominent so that it is difficult for us to accurately locate the information we need in the network. The web classification can effectively improve the efficiency of web search and help us accurately locate the desired page. The current classification algorithm can handle a small amount of web pages classified, but the efficiency of large-scale web classification is not ideal. Recently, a distributed web classification is proposed. Although this method can improve the efficiency of web page classification, it does not improve classification algorithm itself. Therefore, this paper proposes a hashes and KNN method based on the design of a classification algorithm applied to large-scale web classification.

Keywords

Web Page Classification, Large Scale, Spectrum Hashing, KNN

基于谱哈希的大规模网页分类算法

田郸郸

国防科学技术大学计算机学院，湖南 长沙
Email: 1240074883@qq.com

收稿日期：2016年2月2日；录用日期：2016年2月22日；发布日期：2016年2月25日

文章引用: 田郸郸. 基于谱哈希的大规模网页分类算法[J]. 软件工程与应用, 2016, 5(1): 65-74.
http://dx.doi.org/10.12677/sea.2016.51008

摘　要

如今，网络信息已经覆盖到我们生活的方方面面，但随着网络的发展，网络信息过载的问题也越来越凸显，我们在网络中难以准确定位我们所需要的信息。将网页分类可以有效的提高网页搜索效率，帮助我们准确的定位所需网页。当前的网页分类算法可以处理少量网页分类，但对大规模网页进行分类效率不够理想。最近人们提出了分布式的网页分类方法，但这种方法虽然能够提高网页分类效率，但并没有改进分类算法本身。所以本文提出一种基于哈希散列和KNN的方法，设计一个适用于大规模网页分类的分类算法。

关键词

网页分类，大规模，谱哈希，KNN

1. 引言

随着信息时代的到来，网络上的信息资源已经覆盖到我们生活的方方面面，网页的数量也正在经历爆炸式的增长。据中国互联网络信息中心(CNNIC)提供的数据，截至 2014 年 12 月，中国网站数量为 335 万个，年增长 4.6%，中国网页数量为 1899 亿个，年增长 26.6%。这样增长量为我们的生活带来便利和高效的同时，也带来了网络信息过载的问题，使我们迷失在浩瀚的数据之中。为了能够高效、准确的定位我们所需要的网页，许多技术先后被提出，包括搜索引擎和分类处理等等。

在处理海量数据时的一个重要的方法便是把它们进行分类，顾名思义，网页分类就是根据网页所承载的信息进行分类。网页分类可以将网页按照类别存储到相应的数据库中，便于用户快速、准确的找到自己所需要的网页。网页分类可以帮助提高搜索引擎的效率以及查全率和准确率。此外，网页分类也是网络安全管理的关键技术，基于网页分类，我们可以对一些网页进行快速、准确的访问控制。这对维护社会稳定、促进国家发展具有极其重要的现实意义。

此前，网页分类是由领域专家手工完成的，虽然准确率高，但效率很低，随着网页数量的增加，手工分类早已不能满足实际的需求。于是一些基于统计和机器学习的网页分类算法被提出，包括：KNN 算法、决策树、支持向量机[1]、朴素贝叶斯概率模型[2]和神经网络[3]。这些分类算法在处理少量网页是高效的，但在处理大规模网页时，这些方法的效率便不够理想。于是又提出了基于分布式的分类方法，这种方法虽然能够满足分类效率的要求，但并没有改进分类算法本身。所以本文提出一种基于哈希散列和 KNN 的方法，设计一个适用于大规模网页分类的分类算法。

2. 相关技术介绍

2.1. 当前网页分类技术及问题分析

一般来说，网页分类通常基于两种策略：手动分类和自动分类。手动分类便是依赖于领域专家对网页进行分类，这个策略的典型例子便是雅虎(www.yahoo.com)。而自动分类便是预先构建一个分类器，训练集输入到分类器中，对分类器进行训练，分类器获得相关知识，做好分类准备。

自动网页分类器也可以进一步的分为三种类型：线性分类器、非线性分类器和统计分类器。而这些基于传统向量空间模型的分类器，其分类方法都是将网页中的特征信息表示为高维空间中的一个点，这种方法在表示网页时，向量空间的维数会达到上万维，高维向量的运算开销使得这些分类器难以应付大

规模的网页分类。而简单的减少特征项，降低特征空间维度会影响分类器的准确性。所以解决这个问题就要在不影响分类器准确性的前提下，降低原始网页特征空间的维度。谱哈希是一种理想的方法，它能够用短哈希码代表网页特征向量，同时保留原始网页特征，散列后，web 页面维度可以大幅减少。

2.2. 谱哈希

谱哈希[4]其基本思想是将高维空间向量映射至低维汉明空间(Hamming Space) [5]，并保持原空间向量相似性，使得新空间向量的汉明距离(Hamming Distance)反映原空间向量相似度的哈希算法。谱哈希将编码过程视为图分割过程，对高维数据集进行谱分析，通过放松约束条件将问题转化成拉普拉斯特征图的降维问题，从而求解得到网页的哈希编码。假设有 n 个 d 维的网页对象，将该这些表示为 $X=\left\{x\in R^d\right\}_{i=1}^n$，将所有对象数据嵌入到汉明空间后的结果为 $Y\in\{1,-1\}_{n\times r}$，则谱哈希算法可描述为下述优化问题：

$$\begin{aligned}&\min_Y \frac{1}{2}\sum_{i,j=0}^{n} W_{ij}\left\|Y_i-Y_j\right\|^2\\ &\text{s.t}\quad Y\in\{1,-1\}^{n\times r},\\ &\qquad 1^{\mathrm{T}}Y=0\\ &\qquad Y^{\mathrm{T}}Y=nI_{r\times r}\end{aligned}\tag{1}$$

(1) 式中 Y_i 表示 Y 的第 i 行，r 表示哈希编码序列的长度，$1=[1,\cdots,1]\in R^n$，W 是原始空间中的邻接矩阵，W_{ij} 表示原始空间中两样本点 X_i 与 X_j 之间的相似度。约束条件中 $1^{\mathrm{T}}Y=0$ 表示每个哈希值取 0 或 1 的概率相等，$Y^{\mathrm{T}}Y=nI_{r\times r}$ 表示不同位上的哈希编码之间不相关。引入对角矩阵 D，使得对角元素 $D_{ii}=\sum_j W_{ij}$，则目标函数化为：

$$\begin{aligned}&\min_Y \operatorname{trace}\left(Y^{\mathrm{T}}LY\right)\\ &\text{s.t}\quad Y\in\{1,-1\}^{n\times r},\\ &\qquad 1^{\mathrm{T}}Y=0,\\ &\qquad Y^{\mathrm{T}}Y=nI_{r\times r}\end{aligned}\tag{2}$$

(2) 式中 $L=D-W$，称为拉普拉斯矩阵。众所周知，这是一个 NP 难的问题，但若放宽条件 $Y_{ij}\in\{1,-1\}$，使 $Y\in R_r$，则问题变为求拉普拉斯矩阵的较小的 r 个特征值对应的特征向量(不包括 0 特征)，通过选取合适的阈值对这些特征向量进行量化即可得到对像哈希码。

2.3. TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF [6]是一种统计方法，用以评估一个词对于一个文件集或一个语料库中的其中一份文件的重要程度。词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF 的主要思想是：如果某个词或短语在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

TF-IDF 实际上是：TF * IDF，TF 词频(Term Frequency)，TF 表示词条在文档中出现的频率。其计算公式如下：

$$tf(t)=\frac{n_t}{\sum_k n_k}\tag{3}$$

(3) 式中 n_t 是 t 词在文档中的出现次数，而分母则是在文档中所有 k 个字词的出现次数之和。

IDF 逆向文件频率(Inverse Document Frequency)，是一个词语普遍重要性的度量。某一特定词语的 IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到，其公式如下：

$$idf(t) = \log \frac{|D|}{|\{j : t \in d_j\}|} \tag{4}$$

(4) 式中 $|D|$ 表示语料库中的文档总数。$\{j : t \in d_j\}$ 表示包含 t 词的文档数。

则词 t 的 TF-IDF 值计算公式如下：

$$Weight(t) = tf(t) \times idf(t) \tag{5}$$

2.4. KNN (K-Nearest Neighbor)最近邻规则分类

KNN (K-Nearest Neighbor, KNN)最近邻分类算法[7]，是一个理论上比较成熟的方法，也是最简单的机器学习算法之一。该方法的思路是：如果一个样本 d_x 在特征空间中的 k 个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则 d_x 也属于这个类别。KNN 算法中，所选择的邻居都是已经正确分类的对象。具体的算法步骤如下：

一：首先计算 d_x 与训练集中每个样本之间的相似度(本文使用余弦距离作为度量两个向量之间的相似度)，公式如下：

$$Sim(d_i, d_j) = \frac{\sum_{K=1}^{M} W_{iK} \times W_{jK}}{\sqrt{\left(\sum_{k=1}^{M} W_{ik}^2\right)\left(\sum_{K=1}^{M} W_{jk}^2\right)}} \tag{6}$$

$sim(d_i, d_j)$ 表示 d_i 和 d_j 向量之间的相似度；M 表示维度。W_{ik} 代表向量 d_i 的第 k 位。

二：对相似度计算结果排序，选出相似度高的前 K 项

三：假设前 K 项分别属于 C 个类，计算 d_x 在 C_j 类中的累计的相似度，公式如下：

$$CumSim(d_x, c_j) = \sum_{d_i \in K} Sim(d_x, d_i)\, y(d_i, c_j) - b_j \tag{7}$$

当 d_i 属于 C_j 类时 $y(d_i, c_j) = 1$，否则 $y(d_i, c_j) = 0$

四：d_x 被分类为累计相似度最高的类

3. 算法详细描述

本文所设计的算法是在原始网页已经表示成特征向量形式后，将这些高维向量转换为短的哈希码，以达到降低向量维度，减少分类运算开销的目的。利用 KNN 对紧凑哈希码进行分类。算法的详细描述如下：

Step1：从 N 个已分类的网页中，随机抽取 X 个网页作为训练集，将剩余的网页作为测试集。

Step2：对原始网页解析，去除网页中的噪音数据，抽取网页文本信息。

Step3：对网页的文本信息进行预处理，(预处理主要包括分词和词性标注、除去形容词、副词、介词、连词及感叹词等停用词)抽取特征词，生成特征词集，使用 TF-IDF 作为我们的特征词选择方法，特征选择流程的流程图如图 1 所示。(本文取文档中综合权值较高的前 5 个词加入特征集，即 n = 5)。

Step4：以特征向量的形式表示网页[8]。

Step5：利用谱哈希算法对特征向量进行降维，生成紧凑哈希码。

Step6：利用 KNN 算法对待分类网页进行分类，降维后我们使用汉明距离来度量两个散列的相似度

[9]。汉明距离公式列出如下：

$$Sim\left(h_x,h_y\right)=1-\sum_i w_i dist\left(h_{xi},h_{yi}\right) \tag{8}$$

h_x 和 h_y 表示哈希码，$dist\left(h_{xi},h_{yi}\right)$ 代表 i 位的距离，当 $h_{xi}=h_{yi}$，$dist\left(h_{xi},h_{yi}\right)=0$，否则 $dist\left(h_{xi},h_{yi}\right)=1$。

4. 算法测试

4.1. 数据集

我们从凤凰博客(http://www.ifen g.com/)爬取 2,522,182 页总共包括 9 类如下:汽车、教育、娱乐、金融、游戏、房产、新闻、体育和技术。分类的细节集如表 1 所示。

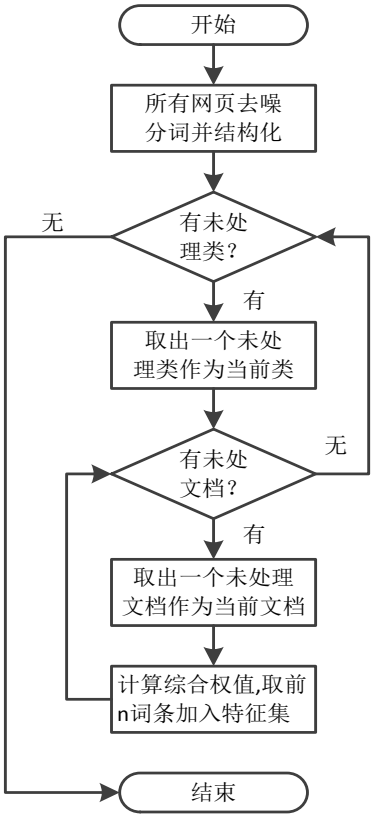

Figure 1. Feature extraction process
图 1. 特征提取流程

Table 1. Detail of dataset
表 1. 数据集分类统计表

Category	Web page Number	Category	Web page Number
Automobile	3,027,662	Education	446,539
Sports	94,146	Finance	473,222
Games	89,550	House	313,386
News	344,818	Entertainment	110,885
Technology	346,874		

4.2. 实验结果

为了证明此算法可以有效的应用于大规模网页分类，我们利用上述的数据集对算法进行测试，测试算法的分类精度、效率，并与 KNN 算法进行对比。在本文中利用宏准确率，宏召回率和宏 F1 值来评价的实验结果。

准确率是所有判断的网页中与人工分类结果吻合的网页所占的比率。其数学公式为：

$$\text{准确率}(\text{precision}) = \frac{\text{分类的正确网页数}}{\text{实际分类的网页数}} \tag{9}$$

召回率是人工分类结果应有的网页中分类系统吻合的网页所占的比率，其数学公式为：

$$\text{召回率}(\text{recall}) = \frac{\text{分类的正确网页数}}{\text{应有的网页数}} \tag{10}$$

准确率和召回率反映了分类质量的两个不同方面，二者必须综合考虑，不可偏废，因此存在一种新的评估指标，即 F1 测试值，其数学公式为：

$$\text{F1测试值} = \frac{\text{准确率} \times \text{召回率} \times 2}{\text{准确率} + \text{召回率}} \tag{11}$$

宏：计算全部类的准确率、召回率和 F1 值。

4.2.1. 实验 1 (k 值对 KNN 分类算法的影响)

在此实验中，我们将讨论 K 值对 KNN 分类算法的影响。为了节省时间，我们从每个类别中分别采样 2000 个网页。被采样的网页一半被用作训练集，其余网页作为测试集。以经过谱哈希降维后的短哈希码和原始网页向量作为输入，来计算当 K 值逐渐增大时，两种算法的宏 F1 值。其结果如图 2 和图 3。我们可以看到 K 值增大时宏 F1 值的变化。如图 3 所示，我们可以看到 F1 值是随 K 值的增加而增加的，但是 k 超过 5 时，F1 值开始逐渐减少。如图 2 所示，当 K 值增加到 100，F1 值急剧下降。因此，在我们下面的实验中，我们设定的 K 值为 5。

4.2.2. 实验 2 (训练集规模对算法的影响)

在这个实验中，我们将讨论不同规模的训练集对两种算法的影响，并测试两种算法的性能来进行比

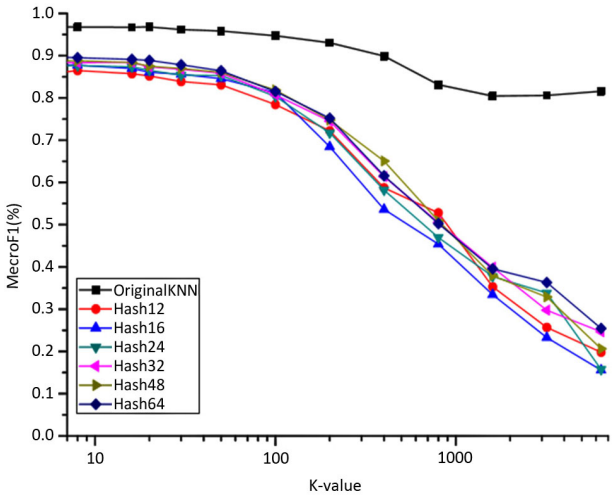

Figure 2. Classification result on choice of k value ($1 \le K \le 9000$)
图 2. 不同 K 值时的分类结果($1 \le K \le 9000$)

较。实验中我们从数据集中随机选择相应数量的网页作为训练集，分别从数据集的每个分类中选择 10,000 个网页作为测试集。分类结果的宏准确率，宏召回率，宏 F1 值和时间成本如图 4 至图 7 所示。

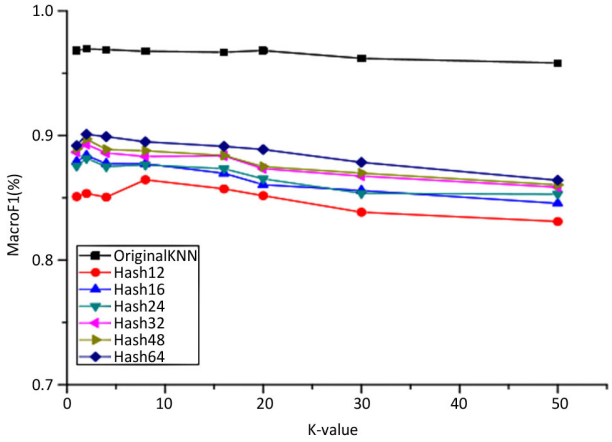

Figure 3. Classification result on choice of k value ($1 \le K \le 50$)
图 3. 不同 K 值时的分类结果($1 \le K \le 50$)

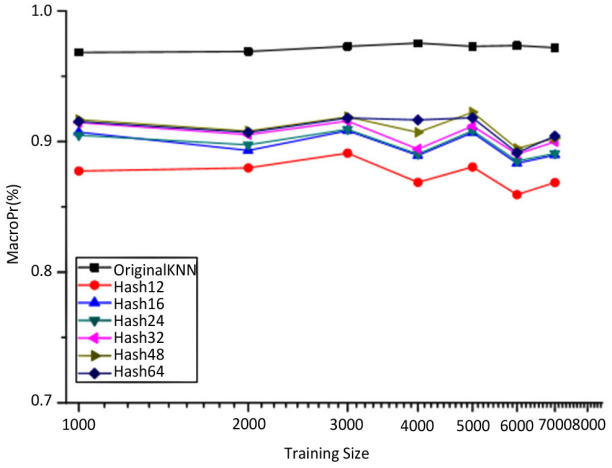

Figure 4. Macro precision on different training set
图 4. 宏准确率随训练集增大的变化

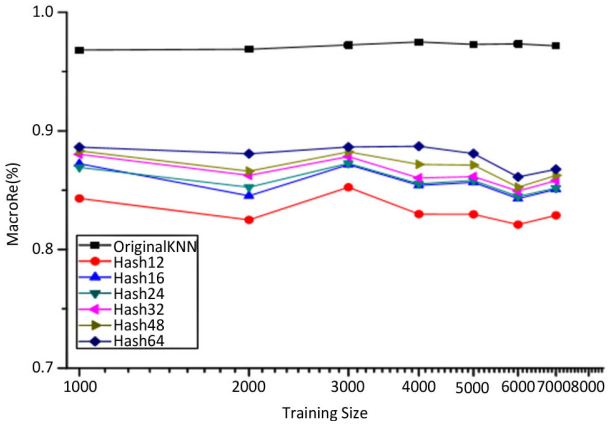

Figure 5. Macro recall on different training set
图 5. 宏召回率随训练集增大的变化

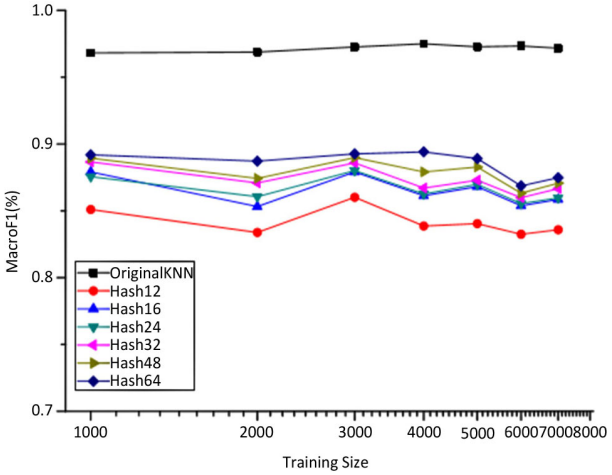

Figure 6. MacroF1 value on different training set
图 6. 宏 F1 值随训练集增大的变化

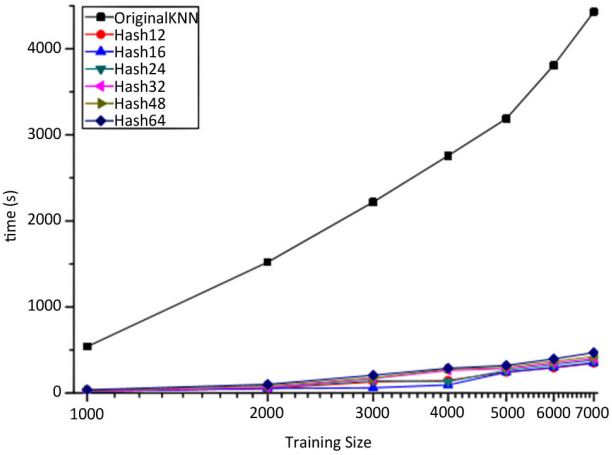

Figure 7. Time cost on different training set
图 7. 时间开销随训练集增大的变化

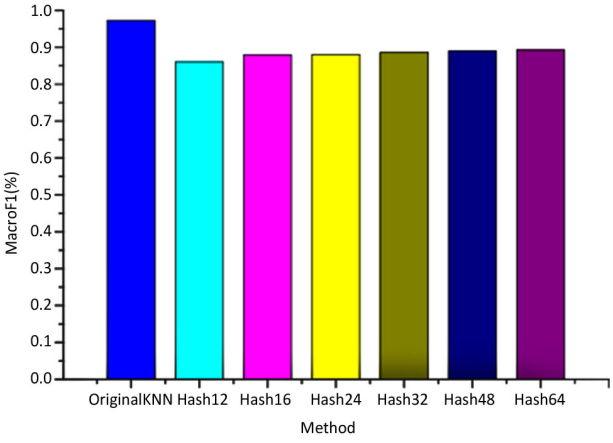

Figure 8. Comparison of MacroF1 value
图 8. 不同算法的宏 F1 对照

从图 4~6 中我们可以看出，KNN 算法的性能，无论是宏准确率、宏召回率、宏 F1 值这三个测

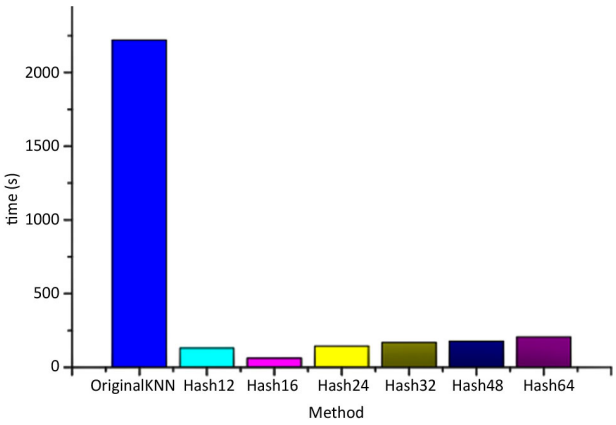

Figure 9. Comparison of time cost
图 9. 不同算法的时间成本对照

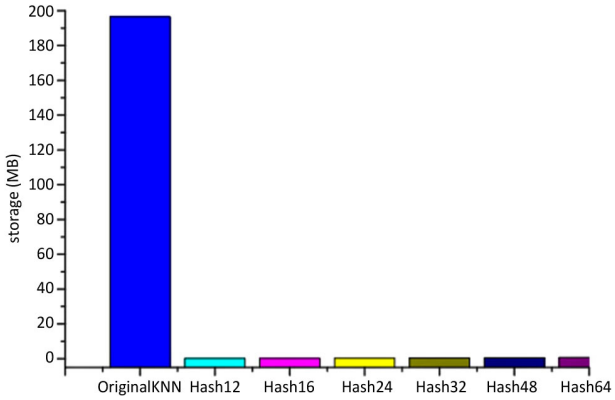

Figure 10. Storage cost of 10,000 pages
图 10. 不同算法的内存使用对照(10,000)

试指标，都比本文所设计的算法略高一点。但从图 7 中，我们可以看出 KNN 算法要比本文所设计的算法慢的多，并且 KNN 算法的时间成本随着训练集的增加成线性的增长，本文所设计的算法只是略微的有些变化。从图 7 中我们可以看出当训练集规模为 3000 时，本文所设计的算法性能最佳，所以我们选择训练集为 3000，进一步对两种算法的性能进行比较。

从图 8、图 9、图 10 中，我们可以看到 KNN 算法只比本文所设计的算法的宏 F1 值略高一点，但本文算法的时间开销、内存开销却远小于 KNN 算法。

5. 结束语

通过实验对比，本文所提出的基于谱哈希的大规模网页分类算法在分类过程中，能够有效的降低时间开销和内存开销。也说明本文所设计的算法，应用于大规模网页的分类是可行且有效的。

参考文献 (References)

[1] 贺海军, 王建芬, 周青, 等. 基于决策支持向量机的中文网页分类器[J]. 计算机工程, 2003, 29(2): 47-48.

[2] 李晋松. 基于朴素贝叶斯的网页自动分类技术研究[D]: [硕士学位论文]. 北京: 北京化工大学, 2008.

[3] 史国强. 基于 RBF 神经网络的网页分类技术研究[D]: [硕士学位论文]. 北京: 中国石油大学, 2011.

[4] Weiss, Y., Torralba, A. and Fergus, R. (2008) Spectral Hashing. *Neural Information Processing Systems*, **282**, 1753-1760.

[5] Charon, I., Cohen, G., *et al.* (2010) New Identifying Codes in the Binary Hamming Space. *European Journal of Combinatorics*, **31**, 491-501. http://dx.doi.org/10.1016/j.ejc.2009.03.032

[6] 张瑾. 基于改进 TF-IDF 算法的情报关键词提取方法[J]. 情报杂志, 2014(4): 153-155.

[7] Song, Y., Huang, J., Zhou, D., *et al.* (2007) IKNN: Informative K-Nearest Neighbor Pattern Classification. Knowledge Discovery in Databases: PKDD. Springer Berlin Heidelberg, 248-264.

[8] 许阳, 刘功申, 孟魁. 基于句中词语间关系的文本向量化算法[J]. 信息安全与通信保密, 2014(4): 84-88.

[9] 李峰, 李芳. 中文词语语义相似度计算——基于《知网》2000 [J]. 中文信息学报, 2007, 21(3): 99-105.