LWCS: A Large-scale Web Page Classification System Based on Anchor Graph Hashing

Yi Zheng, Chengcheng Sun, Chengzhang Zhu, Xv Lan, Xiang Fu and Weihong Han
College of Computer
National University of Defense Technology
Changsha, Hunan, China,
justice131@163.com

***Abstract*—Nowadays, while we are enjoying the convenience brought by such a huge repository of online web information, we may come across difficulties in finding the web pages we want related to particular information we are searching for. Hence, it is essential to classify web documents to facilitate the search and retrieval of pages. Existing algorithms work well with a small quantity of web pages, whereas, they become slow and even non-effective while dealing with a large scale of web pages. Recently, some of these algorithms were adapted to distributed platforms which boosted their classification speeds effectively. However, due to high dimensions of web page features, the parallel classifiers were still trained with limited capacity training sets. In addition, these methods didn't improve the classification itself, merely boosted by high computing performance of distributed platforms. So oriented to large-scale web page classification, we propose to integrate anchor graph hashing with K-Nearest Neighbour(KNN) classifier to reduce the pages' original feature dimensions. The hash value of each page is used for training and classification instead of the original vectors. Experimental comparison with the original KNN on a large dataset demonstrates the efficacy of our proposed method.**

***Keywords*—large-scale web page classification; Chinese web page classification; automatic classification; anchor graph hashing; k-nearest neighbours;**

I. Introduction

With the explosive growth of information technology, the number of web pages increases at an amazing speed. According to the China Internet Network Information Center (CNNIC) [1], until Dec.2014, the Chinese web pages have reached 189 billions, which are 26.6 percent more than that of 2013. While we are enjoying the convenience and benefits it brings, we may find ourselves lost in the vast data oceans. Moreover, these pages vary to a great extent in both their information contents and qualities [2]. Both the huge amount and organization of these web pages don't allow for an easy search. So it is essential to arrange these pages into some orders so that we can locate the information we need in an acceptable period of time. For this purpose, many techniques including search engines and classification were proposed to deal with such problems. Among these techniques, the web page classification seems an efficient one.

As its name implies, web page classification is to classify web pages into categories according to the information they convey. It can help us to improve the quality of our search engines, provide better site navigation as well as maintain and extend our web directory sites automatically. In addition, the web page classification of online pages is the key technology of network security management. Based on the classified categories, we are able to implement fast and fine network access controls which are vital to our national security and stability. For example, it is possible to protect our teenagers from harmful or unhealthy information such as pornographic websites by denying their network access to sites of such categories.

Earlier, the web page classification was done by domain experts manually. But very soon, manual classification didn't meet practical demands. Then some of the approaches used in text classification were adopted to web page classification. These methods are based on statistical and machine learning algorithms like KNN approach [3], decision trees [4],Baynesian probabilistic models [5], support vector mechanics [6] and neural networks [7]. They work well with a small quantity of web pages, however, they become slow and even noneffective while dealing with a large scale of web pages. Therefore, some new approaches based on parallelization [8] [9] [10] were proposed recently. Web pages are parallelly processed in different clusters so as to boost processing speeds. Such methods are capable to satisfy the speed demands, whereas their vulnerability lies in their ignorance of large-scale web pages diversity and heterogeneousness. Only a few pages are used for classifier training which may lead to loss of the classification accuracy. Moreover, these methods didn't improve the classification itself, merely boosted by high computing performance of distributed platforms. From this point of view, we propose to adopt a new hash approach named anchor graph hashing [11] to build a fast and accurate K-Nearest Neighbor classifier which is suitable for Large-scale web page classification.

In the remainder of this paper, we review background knowledge and several relevant work in Section II, present our system design in Section III, show experimental results in Section IV, and conclude the work in Section V.

II. Literature Survey

A. Techniques Used for Web Page Classification

Broadly speaking, two strategies are commonly used for webpage classification, i.e., manual classification and automatic classification. Manual classification relies on domain experts and one typical example of this strategy is Yahoo(www.yahoo.com) . As an alternative, automatic classification depends on an pre-built classifier which takes place of domain experts. These automatic classifiers usually distinguish different web page categories by their contents, links or structures. In general, the automatic classification consists of two steps. First, the training set was inputed into a classifier for training. Then after being trained, the classifier acquired related knowledge and got ready for classification.

Note that automatic classifiers can be further categorized into three types, i.e., linear classifiers, nonlinear classifiers and statistical classifiers [12]. Linear classifiers (e.g., perceptron algorithm) are fit for simple and small-scacle web pages classification. Nonlinear classifiers (e.g., support vector machine) and statistical classifiers (e.g., KNN) are suitable for classifying more complex web pages. However, the three traditional types of automatic classifiers are powerless to deal with large-scale web pages because of their slow processing speeds. Therefore, they were adapted to work on distributed platforms (e.g., hadoop) soon [8] [9] [10]. With the aid of the high performance computing ability, the classification speed was boosted effectively. Nevertheless, cause of the high dimensions of web page feature space, the capacity of training set used to train the classifiers was limited. Moreover, large-scale web pages are made up of huge different and heterogeneous pages, it is hard to classify the pages accurately with a limited training set. Hence, it is urgent to reduce the original web page feature space dimensions if we want to enlarge our training set and get better classification. Then hashing is an ideal approach. It is capable to represent the web page features with short hash codes while preserving original features as possible. After being hashed, the web pages' dimensions can be reduced sharply and lower memory will be used.

B. Anchor Graph Hashing

Among the hashing methods, anchor graph hashing is an excellent one both in its speed and accuracy. Its basic idea is to embed the data in Hamming space such that the neighbors in the original data space remain neighbors in the Hamming space [11]. Similar to Spectral Hashing [13], it tries to seek an r-bit Hamming embedding $Y \in \{1,\text{-}1\}^{n\times r}$ for n points in the database by minimizing

$$\begin{aligned} &\min_{Y} \frac{1}{2}\sum_{i,j=1}^{n} \|Y_i - Y_j\|^2 A_{ij} = tr(Y^\top L Y) \\ s.t \qquad & Y \in \{1,\text{-}1\}^{n\times r}, 1^\top Y = 0, Y^\top Y = nI_{r\times r} \end{aligned} \tag{1}$$

where Y_i is the i^{th} row of Y representing the r-bit code for point x_i, A is the $n \times n$ similarity matrix, and $D = diag(A\mathbf{1})$ with $\mathbf{1} = [1, ..., 1]^\top \in \mathbb{R}^n$. Different from extract neighborhood graph construction, it is not that extract but feasible with large scale. It reduces the graph constructing cost from $O(dn^2)$, e.g. KNN graph, to linear $O(n)$ by approximating the true adjacency matrix A with anchors. The anchors are a small set of $m(m << n)$ points which are centers $U = \{u_j \in \mathbb{R}^d\}_{j=1}^{m}$ obtained from clustering. Then the truncated similarities Z_{ij}'s between all n data points and anchors are defined as,

$$Z_{ij} = \begin{cases} \frac{exp(-D^2(x_i,u_j)/t)}{\sum_{j' \in \langle i\rangle} exp(-D^2(x_i,u_{j'})/t)} & ,\forall j \in \langle i\rangle \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\langle i\rangle \in [1 : m]$ denotes the indices of $s(s{<<}m)$ nearest anchors of point x_i in U according to a distance function $D()$, and t denotes the bandwidth parameter. Derived from random walks across data points and anchors, the Anchor Graph provides a powerful approximation to the adjacency matrix A as $A^{'} = ZV^{-1}Z^\top$ where $V = diag(Z^\top \mathbf{1}) \in \mathbb{R}^{m\times m}$ [14]. In addition, the eigenvectors of $A^{'}$ is easy to solve by utilizing its low-rank property. Specifically, we solve the eigenvalue system of a small $m \times m$ matrix $M = V^{-1/2}Z^\top ZV^{-1/2}$, resulting in $r(< m)$ eigenvector-eigenvalue pairs $\{(p_k, \sigma_k)\}_{k=1}^{r}$ where $1 > \sigma_1 \geq ... \geq \sigma_r > 0$. After expressing $P = [p_1, ..., p_r] \in \mathbb{R}^{m\times r}$ (P is column-orthonormal) and $\sum = diag(\sigma_1, ..., \sigma_r) \in \mathbb{R}^{r\times r}$, we obtain the desired spectral embedding matrix Y as $Y = \sqrt{n}ZV^{-1/2}P\sum^{-1/2} = ZW$ which satisfies $\mathbf{1}^\top Y = 0$ and $Y^\top Y = nI_{r\times r}$.

III. LWCS

A. Architecture

The core architecture of LWCS, our large-scale web page classification system based on anchor graph hashing, is illustrated in Figure 1. LWCS mainly consists of 5 modules i.e., web page crawler, preprocessor, feature selector, anchor graph hashing and KNN classifier. To begin with, the web page crawler crawls online web pages from the Internet. Then, contents of these crawled pages are parsed and denoised. After that, the NLPIR tokenizer will segment these denoised texts into phrases. The next step is to select feature phrases according to the tf-idf value of each phrase. The original pages are expressed as vectors with Vector Space Model. In the following step, these high dimension vectors will be replaced by short hash codes. Finally, the compact hash codes are inputed into KNN classifier to get the classification results. All results are stored in a distributed database, providing service to other systems.

B. Web Page Crawler

We deeply customize an open source crawler named heritrix as our web page crawler. Heritrix is a java-based program and famous for its good scalability. The original way of loading seeds to heritrix, from a xml or a text file, is difficult to manage and unfit for dynamic appending. So we adapte herirtix to load seeds from databases first. Then in oder to avoid reduplicative crawling, an effective filter to record, detect and avoid repetitive urls is built. In addition to these, we reimplemente its work queue to achieve continuous crawl.

C. Preprocessor

The raw web pages crawled from the Internet are full of noise data. Therefore, it is necessary to eliminate noises before the next process. Our htmlparser parses all the contents in raw pages first and then the denoising part starts denoising work. After the denosing process, NLPIR tokenizer(developed by the Chinese Academy of Sciences) is utilized to segment pure contents into phrases, ready for feature selection.

D. Feature Selector

We use term frequency-inverse document frequency (tf-idf) as our feature selection method. The phrases are descendingly sorted according to their calculated tf-idf values. Then we take the top 500 phrases of each category as our feature phrases and expresse the original pages as vectors.

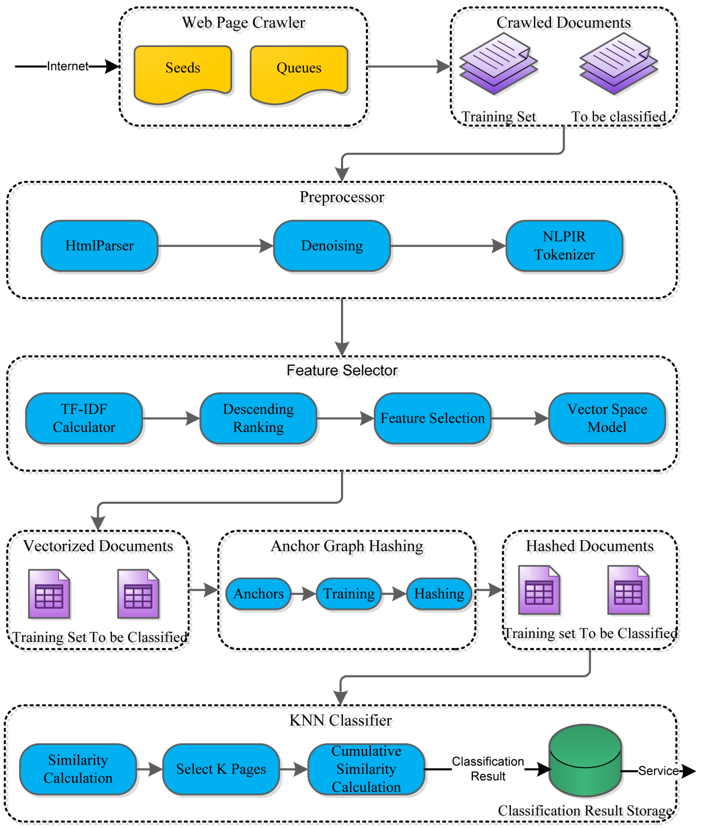

Fig. 1. LWCS framework architecture.

E. *Anchor Graph Hashing*

First, K-means clustering is performed on 13500 points to obtain 500 cluster centers that act as our anchor points with 10 iterations. Then the hasher is trained with the above 13500 points and 500 anchor points. Finally, the vectors we get from last step are inputed into the hasher and corresponding hash codes are generated.

F. *KNN Classifier*

The computation of KNN mainly lies in the caculation of similarity between the page to be classified and each page in training set. In this paper, we use cosine distance as the measurement of similarity between two vectors. The specific formula is as follows:

$$Sim(d_x, d_y) = \frac{\sum_{k=1}^{n} W_{xk} W_{yk}}{\sqrt{(\sum_{k=1}^{n} W_{xk}^2)(\sum_{k=1}^{n} W_{yk}^2)}}. \qquad (3)$$

where d_x and d_y are vectors; $sim(d_x, d_y)$ donates the similarity between vector d_x and d_y; n is their dimensionality; W_{xk} represents the k^{th} member of vector d_x. Analogously, we use hamming distance as the similarity measurement between two hashes. Formula of Hamming distance is listed as follows:

$$Sim(h_x, h_y) = 1 - \sum_{i} w_i dist(h_{xi}, h_{yi}). \qquad (4)$$

where h_x and h_y are hash codes; $sim(h_x, h_y)$ donates the similarity between hash code h_x and h_y; w_i is weight of the i^{th} bit. $dist(h_{xi}, h_{yi})$ represents distance of the i^{th} bit, when h_{xi} equals h_{yi} then $dist(h_{xi}, h_{yi}) = 0$, otherwise $dist(h_{xi}, h_{yi}) = 1$. After similarity calculation, pages in training set are descendingly sorted according to their similarity with the unclassified one. Then the top k pages are selected to compute the accumulative similarity between the page to be classified and each category. Finally, the unclassified page is classified into the category which gets the largest similarity values.

IV. Experiments

A. *DataSets*

The existing benchmark datasets only contain a small number of labeled Chinese web pages which can't meet our demands. So we crawl a large amount of Chinese web pages from Ifeng(http://www.ifeng.com/), a top-ranked large scale portal websites as our dataset. Its channel categories are of certain authority for they are classified manually. Specifically, we crawled 2522182 pages in total including 9 categories as follows: automobile, education , entertainment, finance, games, house, news, sports and technology. Distribution details of our datasets are shown in table1. To run our LWCS, we randomly choose a specified number of pages as our training and test set from this dataset.

TABLE I. Detail of Dataset

Category	Web page Number	Category	Web page Number
automobile	302762	education	446539
sports	94146	finance	473222
games	89550	house	313386
news	344818	entertainment	110885
technology	346874		

B. *Results*

In order to demonstrate the efficacy of LWCS, we evaluate it on the dataset above. Moreover, its performance is compared against the original KNN on accuracy, time and storage usage. Note that we adopt single-layer anchor graph hashing as our hash method in all the experiments below. Besides, the macro precision, macro recall rate and macro F1 value are used to evaluate the experimental results.

1) Experiment 1: In this experiment, we discuss the effect of k value of KNN on the classification results. To save time, we sample 2000 pages respectively from each category in dataset. Half of the sampled pages are used as training set and the remain as the test set. From the classification results in Fig. 2 and Fig. 3, we can see the macro F1 value varys with k value. Specifically, the macro F1 value increases slightly at the beginning which can be clearly seen in Fig. 3. However, when k exceeds 5, the macro F1 value starts to decrease gradually as Fig. 3 shows. When it comes to 100, the Macro F1 declines sharply. So in our following experiments, we set the turning point 5 as our k value.

2) Experiment 2: In experiment 2, we analyse how the training set size affects the classification results first and then compare our hash based methods's performance against the method based on original vectors. Before that, we randomly

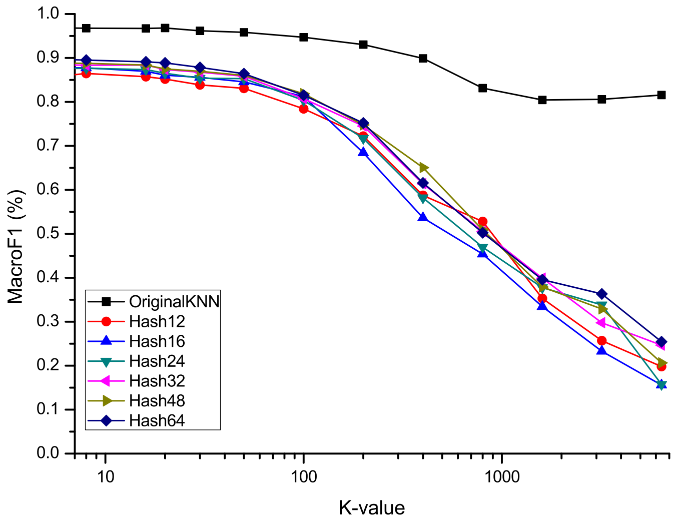

Fig. 2. Classification result on choice of k value ($1 \leq k \leq 9000$) .

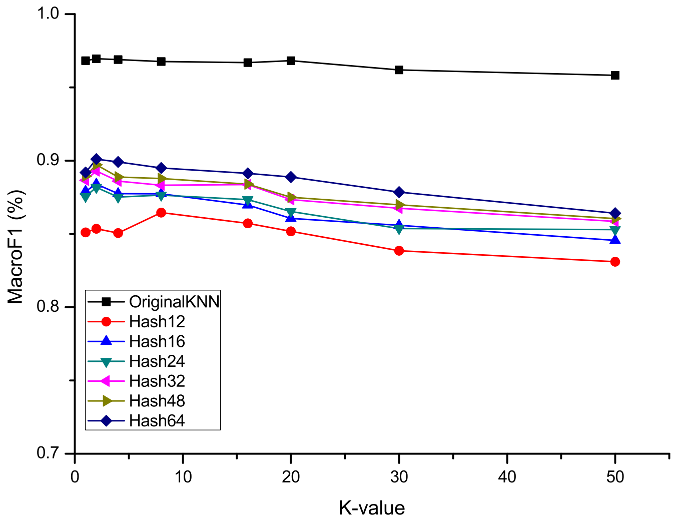

Fig. 3. Classification result on choice of k value ($1 \leq k \leq 50$) .

choose a corresponding number of pages from each category in dataset as our training set and use 1000 pages for each category respectively in the remaining as the test set. Besides, our KNN classifier takes 5 as our k value. The classification macro F1 value and time cost are shown in Fig. 4 and Fig. 5 respectively. From the first three figures, we can see all the three evaluation indexes vary slightly with training set size. Specifically the KNN classifier based on original vectors performs a bit better than that based on our hash methods both in precision and recall rate. However, it is much slower than our hash methods which is clearly shown in Fig. 5. Moreover, the time cost of the original method increases linearly, whereas our method just increases slightly. Note that the time here refers to the time cost on classifying all the 9000 test pages (1000 pages in each category). As for our hash methods, the longer the hash code is, the better performance it achieves. It is interesting to find out that nearly all our hash methods achieve their best performance at training set size 3000. Besides, the time cost of our methods at 3000 is relatively low. So in order to demonstrate the efficacy of our proposed method, we set our training size to 3000 to compare the performance further.

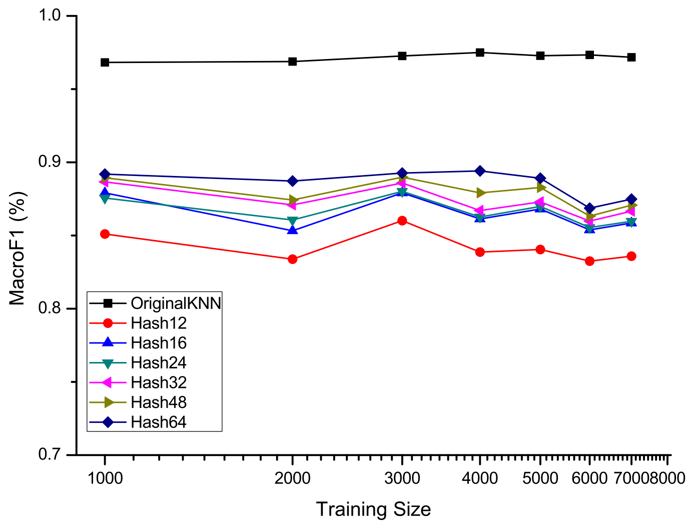

Fig. 4. Macro F1 value on different training set.

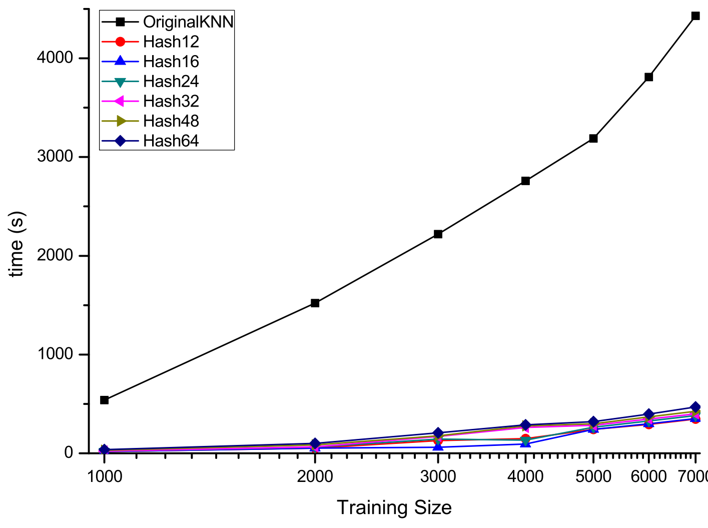

Fig. 5. Time cost on different training set.

From the bar chart Fig. 6 and Fig. 7, we can clearly see that the original method is just a bit better than our methods on F1 value, whereas our methods are much better than it on time cost. Specifically, our method is more than 10 times faster than the original one. Moreover, our method needs much less storage. As shown in Fig. 8, the storage cost of 10000 pages using original vectors is hundreds of times more than our method, cause of its high dimensionality. All results above demonstrates our method's efficacy that our method is much faster and low storage usage while retaining the accuracy.

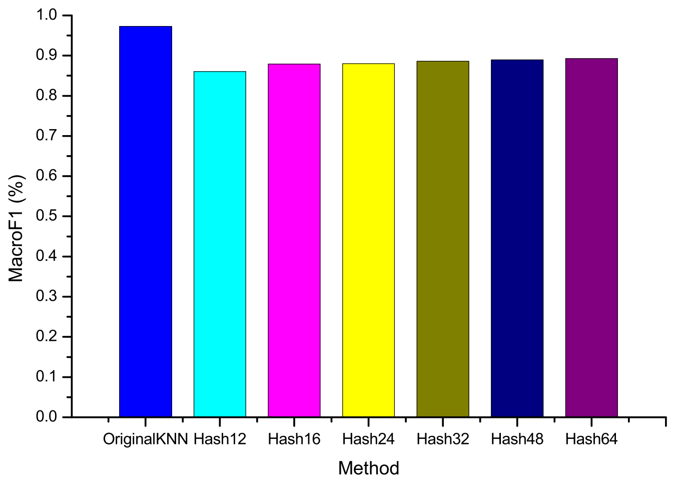

Fig. 6. Comparison of macro F1 value .

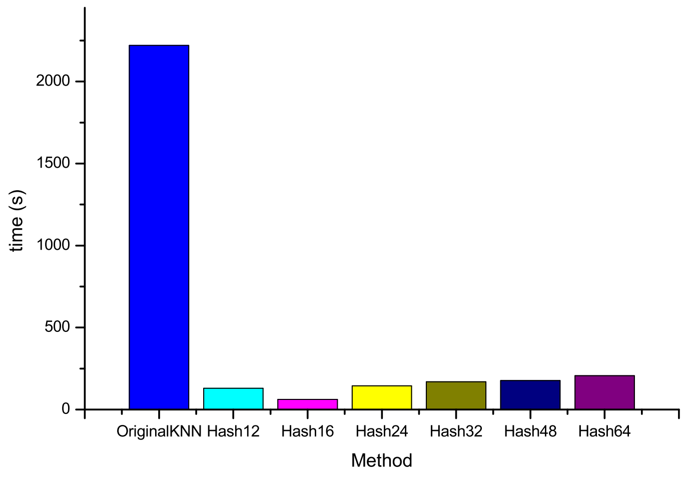

Fig. 7. Comparison of time cost .

V. Conclusion

We have developed a fast and low storage usage classification system named LWCS, which is suitable for large scale Chinese web page classification. We described its architecture, modules and related techniques. Experimental comparison with the original KNN classifier demonstrates LWCS's efficacy. It nearly retains the accuracy but is much faster and low storage usage. In the future, we would like to extend it to web page classification of other language e.g., English and improve its accuracy if possible.

Acknowledgments

This work was supported by National High Technology Research and Development Plan of China (863 plan)(No.2012AA012600, 2012AA01A401, 2012AA01A402), White List System based on Service Discovery.

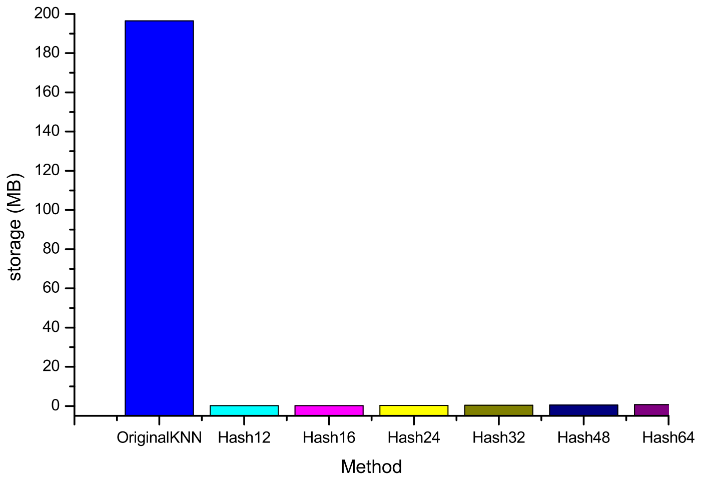

Fig. 8. Storage cost of 10000 pages.

References

[1] C. I. N. I. Center, "35th statistical report on internet development in china," *IEEE National Convention*, 2014.

[2] A. Prakash, A. Kranthi, and K. Ravi, "Web page classification based on document structure," *IEEE National Convention*, 2002.

[3] K. O.-W. and J. Lee, "Text categorization based on k-nearest neighbor approach for web site classification." *Information Processing & Management An International Journal*, vol. 39, no. 1, p. 2544, 2003.

[4] W. Cao and N. Z. Zhang, "A c4.5 decision tree based algorithm for web pages categorization," *Computer Systems & Applications*, vol. 19, no. 10, pp. 195–198, 2010.

[5] Z. L. . Y. Tan, "A chinese web page classification algorithm based on combination of bayes classifier and clustering," *Future Communication Technology*, 2014.

[6] Z. Pan and H. Chen, "Research on chinese web page svm classifer based on information gain," *Journal of Shanghai Normal University*, 2013.

[7] F. Shen, X. Luo, and Y. Chen, "Text classification dimension reduction algorithm for chinese web page based on deep learning," in *Cyberspace Technology (CCT 2013), International Conference on*, 2013, pp. 451 – 456.

[8] X. J. Xiang, "Parallel text categorization of massive text based on hadoop," *Computer Science*, vol. 38, no. 10, pp. 184–188, 2011.

[9] Y. Liu and Z. Wang, "Research on classification of large-scale text on gpu platform," *Computer Engineering & Applications*, vol. 48, no. 8, pp. 141–141, 2012.

[10] H. U. Hong-Yu and G. H. Feng, "Research on large scale text categorization based on support vector machine," *Microcomputer Information*, 2012.

[11] W. Liu, J. Wang, and S. F. Chang, "Hashing with graphs." *Icml*, pp. 1–8, 2011.

[12] M. Y. Q. X. H. Xuanjing, "A fast algorithm for large scale web page classification." *Computer Applications and Software*, pp. 260–263, 2012.

[13] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing." *Proc Nips*, vol. 282, no. 3, p. 490512, 2008.

[14] W. Liu, J. He, and S. F. Chang, "Large graph construction for scalable semi-supervised learning," in *International Conference on Machine Learning*, 2010, pp. 679–686.

[15] D. J. Hand, H. Mannila, and P. Smyth, "Principles of data mining." *Drug Safety*, vol. 30, no. 7, pp. 621–622(2), 2007.

[16] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," *Lecture Notes in Computer Science*, pp. 986–996, 2003.