A Fast Matching Method Based on Semantic Similarity for Short Texts

Jiaming Xu*, Pengcheng Liu, Gaowei Wu, Zhengya Sun,
Bo Xu, and Hongwei Hao

Institute of Automation, Chinese Academy of Sciences, 100190, Beijing, P.R. China
{jiaming.xu,pengcheng.liu,gaowei.wu,zhengya.sun,boxu,
hongwei.hao}@ia.ac.cn

Abstract. As the emergence of various social media, short texts, such as weibos and instant messages, are very prevalent on today's websites. In order to mine semantically similar information from massive data, a fast and efficient matching method for short texts has become an urgent task. However, the conventional matching methods suffer from the data sparsity in short documents. In this paper, we propose a novel matching method, referred as semantically similar hashing (SSHash). The basic idea of SSHash is to directly train a topic model from corpus rather than documents, then project texts into hash codes by using latent features. The major advantages of SSHash are that 1) SSHash alleviates the sparse problem in short texts, because we obtain the latent features from whole corpus regardless of document level; and 2) SSHash can accomplish similar matching in an interactive real time by introducing hash method. We carry out extensive experiments on real-world short texts. The results demonstrate that our method significantly outperforms baseline methods on several evaluation metrics.

Keywords: Short Text, Semantically Similar Matching, Topic Model, Hash.

1 Introduction

Short texts are prevalent on the Web, no matter in traditional Web sites, e.g. Web page titles, text advertisements and image captions, or in emerging social media, e.g. weibos, instant messages, and questions in Q&A websites [1]. Facing the massive short texts, a fast matching method for short texts has become an urgent task to mine semantically similar information for many NLP (Natural Language Processing) applications such as Machine Translation, Text Coherence Detection, etc [2]. Similarity matching can also improve the traditional search engines and user experience [3].

The approaches which improve matching performance between the query and documents can be mainly divided into two categories. The approaches in the first category attempt to do query refinement, such as spelling error correction, word splitting, phrase segmentation, acronym expansion and so on. For example, Li et al. [4] conducted spelling error correction for web search by using a Maximum Entropy

G. Zhou et al. (Eds.): NLPCC 2013, CCIS 400, pp. 299–309, 2013.

model as well as the Source Channel model. Peng et al. [5] performed automatic word stemming for web search by means of a Statistical Language model. Guo et al. [6] described a unified and discriminative model by using Conditional Random Field model for query refinement tasks. However, those works are not beyond sentence level.

In recent years, more researchers focus attention on the second category of methods, latent semantic approaches. NMF (Nonnegative Matrix Factorization) [12], PLSI (Probabilistic Latent Semantic Indexing) [14] and LDA (Latent Dirichlet Allocation) [7] are the most popular latent semantic approaches, and many extensions of those approaches have proposed. However, those approaches are rough clustering methods and time-consuming for online matching.

On the other hand, hashing methods can perform highly efficient but approximate similarity search, and have gained great success in many applications such as Content-Based Image Retrieval [8], near-duplicate document detection [9], etc. Hashing methods project high-dimensional objects to compact binary codes called fingerprints and make similar fingerprints for similar objects [10]. Nevertheless, the previous works, including topic models and hashing methods, suffer from the sever data sparsely in short texts. One popular method for short texts is to extend the short text by knowledge database, such as WordNet[1] or HowNet[2]. However, the social media data are often event-driven, temporal information. For example, typically a short text containing the word “jobs” is likely to be about employment, but right after October 5, 2011, a short text containing “job” is more likely to be related to Steve Jobs’ death [11]. A knowledge database with sufficiently good performance must to be updated timely that result in high labor and material resources for maintaining the database.

To tackle the problems above, this paper attempts to introduce latent semantic approaches and hash methods to our matching method, referred as semantically similar hashing (SSHash). SSHash is based on the following two main ideas. 1). Since hashing methods have bad performance due to the few observed words, we extend the observed features using latent features by latent semantic approaches, rather than knowledge database. 2). Since the conventional methods suffer from the sparsity problem due to the less discriminative observed features in short texts, we directly modeling the generation of features in the whole corpus, rather than the document level. Compared with conventional similar matching method, SSHash combines the merits of topic model and hashing method.

The remainder of this paper is organized as follows. Section 2 reviews the relevant research for each of our components. Section 3 describes our method SSHash and gives implementation details. Experimental results are presented in Section 4. Finally, conclusions are made in the last section.

[1] `http://wordnet.princeton.edu`: a lexical database for English.

[2] `http://www.keenage.com`: Chinese message structure base.

2 Related Works

In this section, we first introduce the two building blocks of our method: latent semantic approaches and hashing methods. Using the topic models allow us to achieve semantic similarity, while using hashing methods give our SSHash method interactive similarity matching time.

2.1 Latent Semantic Approaches

Latent semantic approaches, also known as topic models, have two approaches, non-probabilistic approach and probabilistic approach. The typical non-probabilistic approaches, such as NMF [12], and LSA [13], are all based on vector space model. Maximum likelihood estimation can be used for learning probabilistic general models such as PLSI [14] and LDA [7]. The non-probabilistic models can be reformulated as probabilistic models.

In recent years, topic models for short texts have been extensively studied. However, the early studies mainly focused on exploiting external knowledge to solve the sparsity problem of short text. For example, Phan et al. [15] learned hidden topics from large external resources to enrich the representation of short texts. These methods are overdependence on the performance of knowledge base. Researchers propose several other approaches to improve the topic models on short texts. For example, Rosen-Zvi et al. [16] expanded topic distributions from document-level to user-level, aggregate the short texts by each user into longer texts, and then train a conventional topic model. Zhao et al. [17] simply supposed that each short text only contains a single topic. Diao et al. [11] further assumed short texts published around the same time have a higher probability to belong to the same topic. However, such heuristic methods are high data-dependent, not be suitable for other datasets, like short questions or news titles, because of too many strict assumptions and sparse constraints be imposed in those approaches.

Yan et al. [1] proposed such idea that learning the latent semantic association by directly modeling the generation of word co-occurrence patterns in the whole corpus, rather than document level. With such idea, we develop our matching method SSHash based on semantic similarity for short texts to alleviate the sparse problem and improve the semantic correlation matching.

2.2 Hashing Methods

As hashing methods can perform highly efficient but similarity matching, approximate nearest neighbor search in Hamming space is widely applied in image retrieval [8], image classification [18], information retrieval [9], and so on. A notable method is Locality Sensitive Hashing (LSH), proposed by Indyk et al. [19]. LSH scheme is that hash codes of similar objects collide with high probability and the hash codes of dissimilar objects collide with low probability, such that for objects A and B:

$$\Pr[h(A)=h(B)]=sim(A,B)\ , \tag{1}$$

Where $sim(A,B)\in[0,1]$ is some similarity function. Broder, et al. [21] provided an elegant construction of LSH with Jaccard similarity coefficient:

$$sim(A,B) = J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{2}$$

Charikar [20] explored constructions of LSH functions for various other interesting similarity measures, referred as simhash. Those hash codes possesses two conflicting properties: 1). the fingerprint of a document is a "hash" of its features; 2). similar documents have similar hash values.

Manku, et al. [9] represented a detailed algorithm for simhash for similar documents matching. Manku's algorithm process is as follows: we first convert a document into a set of features, each feature tagged with its weight. Features are computed using standard IR techniques like tokenization, case folding, stop-word removal, and stemming and phrase detection. Then, we transform the features into an f -bit fingerprint where f is small, say 64.

However, Manku's algorithm for similarity matching encounters the sparsity problem in short texts. Because of the few observed features, some non-overlapping features lead to a very low similarity score despite the high semantic relatedness. One solution is to enlarge the Hamming distance to increase the recall rate. However, the larger Hamming distance, the lower precision. Fig. 1 shows an example to reveal the correspondence between similarity and Hamming distance, and Fig. 2 describes the problem that the larger Hamming distance lead to the lower precision. We can see that it is not a wise choice by enlarging Hamming distance to solve short text problem.

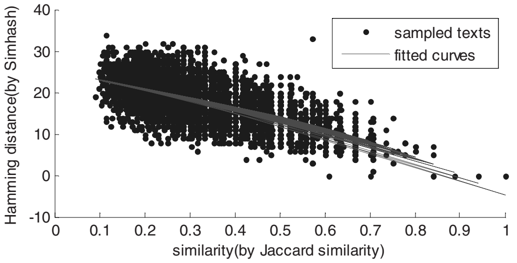


Fig. 1. The correspondence between similarity and Hamming distance (one probe query and 8359 sampled texts)

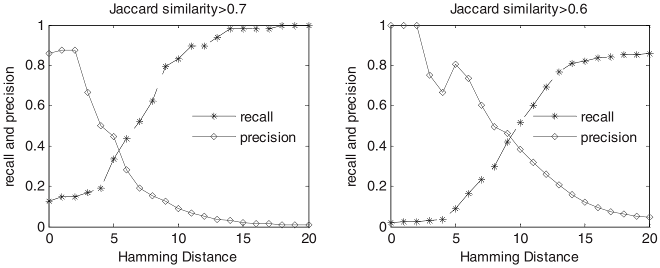


Fig. 2. Larger Hamming distance leads to the lower precision. (Respectively define the lower matching limit by Jaccard similarity as 0.7, 0.6 and 0.5)

Another approach is to refine the hash function to increase the collision probability within a same cluster. He, et al. [22] presented a k-means hashing method (KMH) to learn hashing code whose Hamming distance approximates the data similarity. In order to preserve the similarity between Euclidean and Hamming distances of k-means clusters, KMH simultaneously minimize the quantization error and the affinity error among sample x, k-means center $\boldsymbol{c}_{i(x)}$ and Hamming code $\boldsymbol{h}_{i(x)}$ by Expectation Maximization (EM) algorithm. However, KMH lead to a high computation complexity due to EM method.

In our works, SSHash solves the two problems of matching efficiency and semantic similarity simultaneously by combining the merits of latent semantic approaches and hashing methods.

3 Semantically Similar Hashing

In order to increase the recall of similar matching, we should increase the collision probability of hash codes between the similar short texts, and avoid enlarging Hamming distance. In this paper, we implement this idea by introducing biterm topic model to hash method. Fig.3 illustrates our matching method named SSHash.

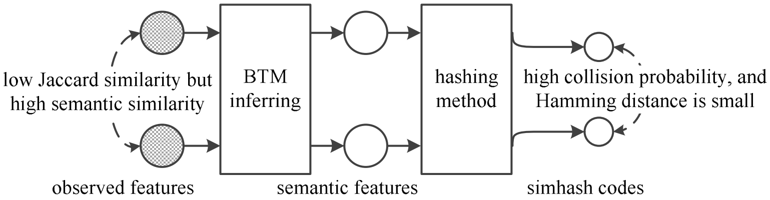

low Jaccard similarity but high semantic similarity
BTM inferring
hashing method
high collision probability, and Hamming distance is small
observed features
semantic features
simhash codes

Fig. 3. Schematic representation of SSHash

3.1 Biterm Extraction

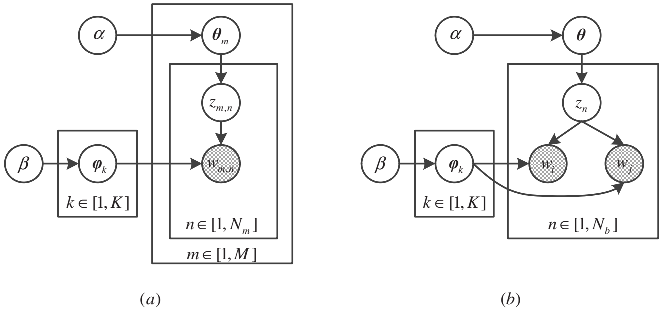
Since conventional topic models, such as PLSI and LDA, implicitly count the frequency of word co-occurrence by modeling word generation form the document level, in this paper, we follow Yan's biterm topic model (BTM) to directly model the words co-occurrence patterns based on biterms. Here, we give an example to explain what a biterm is. In a short text "A forum for the NLP researchers", we first preprocess the raw texts, such as filtering the stop words, lower case and stemming the words, and then extract the biterms with its frequency as shown in table 1. Then, we put the all biterms into corpus to train BTM.

Table 1. Biterms extracted from the example short text

Biterms	Frequency
forum, nlp	1
forum, research	1
nlp, research	1

3.2 Biterm Topic Model

BTM supposes that both words in the biterm are drawn from the same topic, and considers that the whole corpus as a mixture of topics. That is different from conventional generative models such as LDA. The difference in graphical representation is described as shown in Fig.4.



(*a*) (*b*)

Fig. 4. Graphical representation of (a) LDA and (b) BTM

We can see that BTM draws the topic assignment z_n from the corpus-level distribution $\boldsymbol{\theta}$, which are different from LDA that draws the topic assignment $z_{m,n}$ from the document-level topic distribution $\boldsymbol{\theta}_m$.

The specific generative process of BTM for the whole corpus can be assumed as follows:

1. Draw a topic distribution $\boldsymbol{\theta}$ ~Dirichlet(α) for the whole collection
2. For each topic z_n
   (a) Draw a topic-specific word distribution $\boldsymbol{\varphi}_k$ ~Dirichlet(β)
3. For each biterm b
   (a) Draw a topic assignment z_n ~Multinomial($\boldsymbol{\theta}$)
   (b) Draw two words: w_i , w_j ~Multinomial($\boldsymbol{\varphi}_k$)

And the joint probability of a biterm $b = (w_i, w_j)$ can be written as:

$$P(b) = \sum_z P(z)P(w_i \mid z)P(w_j \mid z) = \sum_z \theta_z \varphi_{i|z} \varphi_{j|z} \quad (3)$$

The parameters, $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$, can be inferred by Gibbs sampling. More details can be found in [1].

3.3 Features Representation

In order to represent the observed features in short texts by using the latent topic distribution in a document, we should infer the topic distribution in a document level.

Although BTM has no regard for document concept, the inferring process is similar with training process just considering the word distribution φ_k is constant. However, Gibbs sampling is time-consuming and not suit for online text matching because of many iterations until convergence. Here is a compromise approach to solve the dilemma problem that parameters can be estimated based on the observed frequency. BTM assumes that latent topic distribution in a document is that:

$$P(z \mid d) = \sum_b P(z \mid b) P(b \mid d) \tag{4}$$

Where $P(z \mid b)$ can be derived by the parameters estimated during the training process:

$$P(z \mid b) = \frac{P(z) P(w_i \mid z) P(w_j \mid z)}{\sum_z P(z) P(w_i \mid z) P(w_j \mid z)} \tag{5}$$

In this equation, $P(z) = \theta_z$, and $P(w_i \mid z) = \varphi_{i|z}$, and $P(b \mid d)$ can be calculated simply by the observed frequency:

$$P(b \mid d) = \frac{n_d(b)}{\sum_b n_d(b)} \tag{6}$$

Now, the semantic features in a document can be represented by the latent topic distribution $P(z \mid d)$, and each topic is a distribution over words $P(w_i \mid z)$. In order to reduce the feature dimension, we select the most likely words for each topic such as top 20. We call the selected words tokens, denoted as t to distinguish word concept w in the corpus. Note that the weights of tokens should be normalized as follow:

$$P(t_i \mid d) = \sum_z P(t_i \mid z) P(z \mid d) = \sum_z \frac{P(w_i \mid z)}{\sum_{top20} P(w \mid z)} P(z \mid d) \tag{7}$$

3.4 Fingerprinting with Hash Method

Since we have taken the tokens selected from each topic to represent the semantic features, short text are fingerprinted with hash method by using tokens with their weights. The fingerprinting process incorporated semantic information for each short text can be described as follows:

```
1. Initialize two arrays $W$ and $B$ with $f$ zeros
2. For each token $t_i$
   (a) Compute a $f$-bit hash code $h_i$
   (b) Iterate through each bit $h_{ij}$, where $j \in [1, f]$
       (i) If $h_{ij} = 1$, $W[j] = W[j] + P(t_i \mid d)$, where $P(t_i \mid d)$ is calculated in Eq.(7)
       (ii) If $h_{ij} = 0$, $W[j] = W[j] - P(t_i \mid d)$
```

3. Revisit the all bits of array W , where $j \in [1, f]$
   (a) If $W[j] \ge 0$, then set the bit to 1, $B[j] \leftarrow 1$
   (b) If $W[j] < 0$, then set the bit to 0, $B[j] \leftarrow 0$
4. Return the array B , this is a hash code fingerprinted for short text.

3.5 Similarity Matching for Online Query

When a short text is inputted by user for online matching, we should preprocess the text, as section 3.3 and 3.4, to get the fingerprint and then do approximate nearest neighbor search in Hamming space. All pairs of fingerprints found within a certain Hamming distance of each other are semantic similar texts, as shown in Fig. 5,

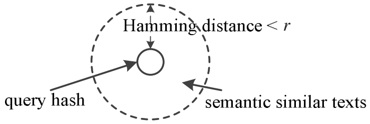


Fig. 5. Similarity matching for query

In our method, we apply block-permuted Hamming search (BPHS) and column-oriented database management system HBase[3] to speed up the online matching for large collections, the more implement details of BPHS can be found in [9].

4 Experimental Design, Comparison and Analysis

4.1 Data Set

In our experiments, we made use of real-world short text collections which are random selected from Weibo[4] between September 1st and December 13rd, 2012. Thanks for some weibos labeled by authors with hashtags to denote a specific topic, we organize those weibos with the same hashtag into a semantic similar cluster. Here, we select 20 specific topics and process the raw texts via the following steps: 1). removing hashtags and non-Chinese characters; 2). word splitting and filtering stop words; 3). removing words with documents frequency less than 5; 4). filtering out weibos with length less than 5. At last, we left 151,629 valid weibos, 34,099 words and the average document length is 14.9. The texts are randomly split into a training set containing 137,929 texts and a test set containing 13,700 texts.

4.2 Evaluation Metrics

In order to evaluate our method's performance, we measure the recall and precision of our method. Intuitively, we want to maximize the number of correct positives and

[3] `http://hbase.apache.org`: a Hadoop database, a distributed, scalable, big data store.

[4] `http://weibo.com`: a popular Chinese microblog website.

minimize the number of false positives. A correct positive is defined as a semantically similar match between a probe query and one of short text collections. To decide whether a semantically similar match, we simply test if the two weibos have the same hashtag. Recall and precision are defined as:

$$recall = \frac{\text{number of matched semantic similar texts}}{\text{total number of all semantic similar texts}} \tag{8}$$

$$precision = \frac{\text{number of matched semantic similar texts}}{\text{total number of matched texts}} \tag{9}$$

4.3 Results and Analysis

Using 64-bit hash codes, we compute a query's fingerprint and match all of the texts stored in a Hamming ball of radius from 0 to 20. Fig.6 shows the matching performance comparison by using simhash, LDA+hash and SSHash.

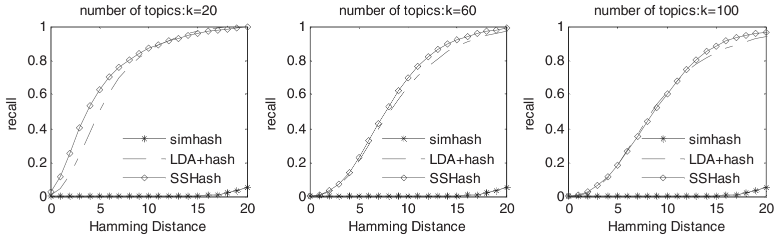


Fig. 6. Matching performance comparison with different Hamming distance (topic number: K=20, 60,100)

We can see SSHash and LDA+hash has a high recall in a little Hamming distance, and SSHash achieves higher recall rate than LDA+hash.

In Fig.7, the precision-recall curves reveal that SSHash and LDA+hash always dominate simhash, and SSHash is better than LDA+hash.

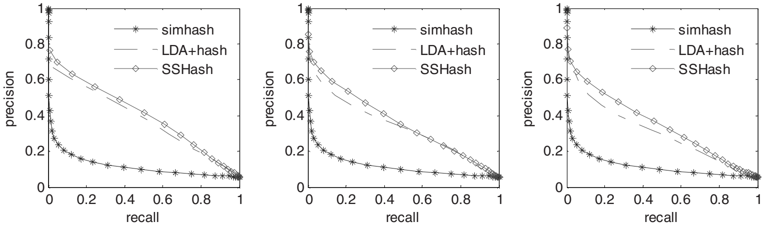


Fig. 7. Precision-Recall curves for Weibo datasets (topic number: K=20, 60,100)

From the results, we can further see that the recall is more notable when the topic number is 20. That is because that the semantic features discovered are very general when the number of topics is small. In such case, the collision probability of hash codes between the similar texts or dissimilar texts will be increased. This interpretation is verified in Fig.7. Although the recall rate is higher, the precision is lower when the number of topics is small and within a little Hamming distance. In contrast, when the number of topics is large, the semantic features discovered are very specific.

5 Conclusion

In this paper, we describe a novel method based on semantic similarity for short texts, namely semantically similar hashing (SSHash). SSHash can project the shorts text into binary hash codes with semantic information but alleviate the sparse problem due to regardless the document concept. We can find semantically similar texts in an interactive real time by using SSHash. We carried on experiments on Weibo datasets. The results demonstrated that we achieve higher recall and precision than simhash or LDA+hash applied to the real-world short texts.

For future work, there are still lots of work to do. First, we will do more analysis on computational cost. Second, the off-line training process is time-consuming, although our method can speed up the online matching. We would like to introduce online latent semantic approaches to our method.

Acknowledgements. This work is supported by the National Natural Science Foundation of China under Grant No. 61175050 and No. 61203281.

References

1. Yan, X., Guo, J., Lan, Y., Cheng, X.: A Biterm Topic Model for Short Texts. In: Proc. 22th International Conference on World Wide Web, pp. 1445–1455 (2013)
2. Weiwei, G., Diab, M.: Modeling Sentences in the Latent Space. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pp. 864–872 (2012)
3. Wan, X., Yang, J., Xiao, J.: Towards a Unified Approach to Document Similarity Search Using Manifold-Ranking of Block. Information Processing & Management 44(3), 1032–1048 (2008)
4. Mu, L., Muhua, Z., Yang, Z., Ming, Z.: Exploring Distributional Similarity Based Models for Query Spelling Correction. In: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pp. 1025–1032 (2006)
5. Fuchun, P., Ahmed, N., Xin, L., Yumao, L.: Context Sensitive Stemming for Web Search. In: The 30th Annual International ACM SIGIR Conference, pp. 23–27 (2007)
6. Jiafeng, G., Gu, X., Hang, L., Xueqi, C.: A Unified and Discriminative Model for Query Refinement. In: The 31st Annual International ACM SIGIR Conference, pp. 379–386 (2008)

7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)
8. Yan, K., Sukthankar, R., Huston, L.: Efficient Near-duplicate Detection and Sub-image Retrieval. In: Proceedings of the 12th ACM International Conference on Multimedia, pp. 869–876 (2004)
9. Manku, G.S., Jain, A., Sarma, A.D.: Detecting near-duplicates for web crawling. In: Proceedings of the 16th International Conference on World Wide Web, pp. 141–150 (2007)
10. Qixia, J., Maosong, S.: Semi-Supervised SimHash for Efficient Document Similarity Search. In: The 49th Annual Meeting of the Association for Computational Linguistics, pp. 93–101 (2011)
11. Qiming, D., Jing, J., Feida, Z., Lim, E.-P.: Finding Bursty Topics from Microblogs. In: The 50th Annual Meeting of the Association for Computational Linguistics, pp. 536–544 (2012)
12. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
13. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
14. Hofmann, T.: Probailistic Latent Semantic Indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57 (1999)
15. Phan, X.H., Nguyen, M.L., Horiguchi, S.: Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-Scale Data Collections. In: Proceedings of the 17th International Conference on World Wide Web, pp. 91–100 (2008)
16. Rosen-Zvi, M., Griffiths, T., Steyvers, M., Smyth, P.: The Author-Topic Model for Authors and Documents. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, pp. 487–494 (2004)
17. Zhao, W.X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., Li, X.: Comparing Twitter and Traditional Media Using Topic Models. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Mudoch, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 338–349. Springer, Heidelberg (2011)
18. Boiman, O., Shechtman, E., Irani, M.: In Defense of Nearest-Neighbor Based Image Classification. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2008)
19. PiotrIndyk, R.M.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, pp. 604–613 (1998)
20. Charikar, M.: Similarity Estimation Techniques from Rounding Algorithms. In: Proceedings on 34th Annual ACM Symposium on Theory of Computing, pp. 380–388 (2002)
21. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-Wise Independent Permutations. In: Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, pp. 327–336 (1998)
22. Kaiming, H., Fang, W., Jian, S.: K-means Hashing: an Affinity-Preserving Quantization Method for Learning Binary Compact Codes. In: The 24th IEEE Conference on Computer Vision and Pattern Recognition (2013)