Data-Dependent Locality Sensitive Hashing

Hongtao Xie[1], Zhineng Chen[2], Yizhi Liu[3], Jianlong Tan[1], and Li Guo[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences,
National Engineering Laboratory for Information Security Technologies,
Beijing, China, 100093
[2] Interactive Digital Media Technology Research Center, Institute of Automation,
Chinese Academy of Sciences, Beijing, China, 100190
[3] School of Computer Science and Engineering, Hunan University of Science and Technology,
Xiangtan, China, 411201
{xiehongtao,tanjianlong,guoli}@iie.ac.cn,
zhineng.chen@ia.ac.cn, liuyizhi928@gmail.com

Abstract. Locality sensitive hashing (LSH) is the most popular algorithm for approximate nearest neighbor (ANN) search. As LSH partitions vector space uniformly and the distribution of vectors is usually non-uniform, it poorly fits real dataset and has limited performance. In this paper, we propose a new data-dependent LSH algorithm, which has two-level structures to perform ANN search in high dimensional spaces. In the first level, we first train a number of cluster centers, then use the cluster centers to divide the dataset into many clusters and the vectors in each cluster has near uniform distribution. In the second level, we construct LSH tables for each cluster. Given a query, we first determine a few clusters that it belongs to with high probability, and then perform ANN search in the corresponding LSH tables. Experimental results on the reference datasets show that the search speed can be increased by 48 times compared to E2LSH, while keeping high search precision.

Keywords: Locality sensitive hashing, approximate nearest neighbor.

1 Introduction

Nearest neighbor search in high-dimensional space is the core problem in database management, data mining and computer vision [1]. Traditional tree-based indexing methods can return accurate results, but they are time consuming for data with high dimensionalities. It has been shown that when the dimensionality exceeds 10, existing tree-based indexing structures are slower than the linear-scan [2], which is known as "dimensionality curse". To solve this problem, hash-based methods are proposed [3] for approximate nearest neighbor (ANN) search.

Among all the hash-based algorithms, locality sensitive hashing (LSH) [4] is one of the most widely used ANN search methods. It first takes a number of random projection functions to group or collect items close to each other into the same buckets with a high probability. In order to perform similarity query, LSH hashes the query item into some buckets and uses the data items within those buckets as potential candidates for the final results. Moreover, the items in the buckets are ranked according

W.T. Ooi et al. (Eds.): PCM 2014, LNCS 8879, pp. 284–293, 2014.

to the exact distance to the query item to compute the nearest neighbor. The final ranking computation among the candidates is called *short-list* search. LSH is an algorithm based on random projection, so it partitions the vector space uniformly. In contrast, the distribution of the items in vector space is usually non-uniform. Consequently, the distribution of randomly projected values is far from being uniform. In such a situation, querying will cost a lot of time for many disturbance points are probed and it severely limits its search performance.

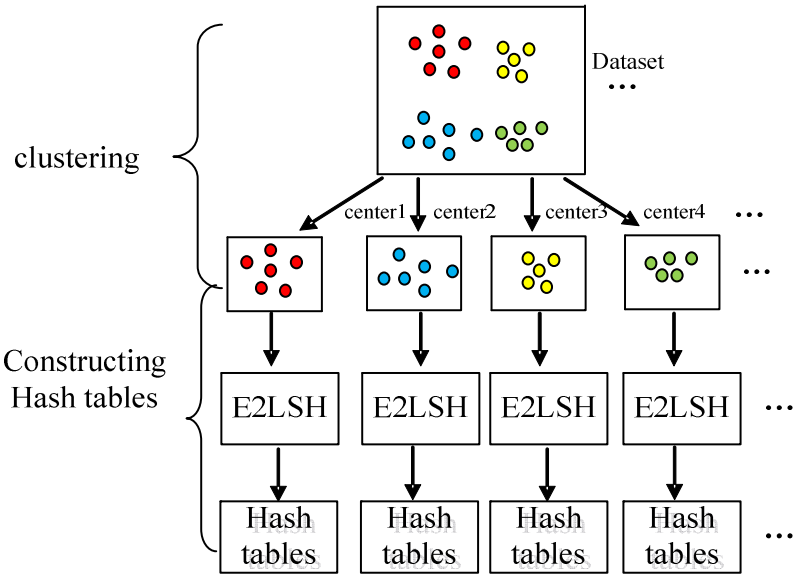


Fig. 1. The framework of our two-level index. In the first level, we group the dataset into clusters though the centers that are trained beforehand. During the second level, we apply E2LSH [4] to each cluster.

To solve aforementioned problem, improved hash algorithms have been proposed [5][6][7]. These methods use more candidates, estimate optimal parameters or use improved hash functions. As they only consider the average runtime or quality of the search process over randomly sampled hash functions, which may result in large deviations in runtime cost or the quality of k-nearest neighbor results [8], they still cannot solve the problem well. In this paper, we present a novel data-dependent LSH algorithm, which is composed of two-level structures. During the first level, we first train a number of centers through K-means algorithm [9] on the training set and then divide the dataset into a number of clusters which are corresponding to these centers. This step gets the similar points together. Within each cluster, the distribution of items is more uniform than that of the items in the whole dataset. During the second level, we apply E2LSH technique to each cluster to construct LSH tables. The uniform distribution of points within clusters results in the uniform distribution of point indices in the corresponding hash tables. The framework of proposed data-dependent LSH algorithm is illustrated in Figure 1.

To further improve the search speed, we propose a novel search pruning method, based on a new distance metric which utilizes information about the relative positions and sizes of all clusters. Given a query, we first compute its distances to all cluster centers and take a few clusters as the candidates. Then, we apply E2LSH search algorithm to the candidate clusters and get the nearest neighbors to the query. The experiments conducted on benchmark datasets show that our algorithm can obviously improve the search speed while keeping high precision.

The rest of this paper is organized as follows. Section 2 gives a review of previous works. Section 3 presents the proposed data-dependent LSH method. Section 4 shows the experimental comparisons. In section 5, we give the conclusion and future work.

2 Literature Review

As the basic LSH algorithm cannot deal with non-uniformly distributed dataset, many techniques have been proposed for improvement. These methods are classed into data independent and data dependent.

Data independent methods do not use the base dataset for building index. Lv *et al.* propose multi-probe LSH [12], which systematically probes the buckets near the query points in a query-dependent manner, instead of only probing the bucket that contains the query point. It can obtain a higher recall ratio with fewer hash tables, but may result in larger selectivity from additional probes. Bawa *et al.* propose LSH-forest [13], which avoids tuning of the parameter by representing the hash table as a prefix tree and the parameter is computed based on the depth of the corresponding prefix-tree leaf node. Dong *et al.* [14] construct a statistical quality and runtime model using a small sample dataset, and then compute parameters that can result in a good balance between high recall ratio and low selectivity. Pan *et al.* propose Bi-level LSH [8]. In the first level, it uses a RP-tree to partition the dataset into sub-groups with bounded aspect ratios and is used to compute well-separated clusters. During the second level, it builds a single LSH hash table for each sub-group along with a hierarchical structure based on space-filling curves. Bi-level LSH is similar to our method in this paper, but as it uses RP-tree to partition the dataset randomly, it has limited performance.

Data dependent methods use the base dataset for building index. Babenko et al. propose inverted multi-index [15], which generalizes the inverted index by replacing the standard quantization within inverted indices with product quantization. In some methods, data are represented by binary codes to reduce memory usage and computation times, as the calculations in Hamming space can be executed by efficient bitwise XOR operation. Among these schemes, spectral hashing [6] is a representative method. It defines a hard criterion for a good code that is related to graph partitioning and uses a spectral relaxation to obtain an eigenvector solution. Using binary codes can improve the efficiency of indexing [16]. However, we only focus on using pure floating-point vector.

3 Data-Dependent Locality Sensitive Hashing

In this section, we first elaborate the details of our data-dependent LSH framework and propose a distance metric. Then we present a new search algorithm for data-dependent LSH.

3.1 Constructing Data-Dependent LSH Index

As LSH cannot deal with non-uniform distribution dataset, it has limited search efficiency. K-means is a popular clustering algorithm and can partition n items into k clusters, in which each item belongs to the cluster with the nearest mean [9]. To make the distribution of the items in the hash tables much more uniform, we use K-means algorithm to divide the dataset into several clusters. Then, we apply E2LSH method to each cluster to build the two-level index.

The procedure of constructing data-dependent LSH hash tables is given in Algorithm 1. We first apply the K-means algorithm to the training dataset D_T^n, and get a number of cluster centers $\mu_j (j = 1, 2 \cdots k)$. Then we assign the points in the dataset D^n to these centers and partition the dataset into k clusters. Finally, we adopt E2LSH algorithm [4] to each cluster for building LSH tables.

Algorithm 1. Building data-dependent LSH

Input: A set of data items $\{x_i\} \in R^n$ (n is the dimension of each item.)

Output: A set of E2LSH structures $\{T_i\}$. Each E2LSH structure is composed of multiple LSH tables, and there are many hash buckets in each table. The indices of the points are preserved in corresponding buckets.

Notations: D^n, the index dataset; $x_i \in D^n$ $(i = 1, 2 \cdots M)$; D_T^n, the training set; $\mu_j (j = 1, 2 \cdots k)$, the centers gotten by K-means algorithm on D_T^n; $A_j (j = 1, 2 \cdots k)$, the clusters which D^n is grouped into.

Operation:

1. $\mu_j (j = 1, 2 \cdots k) \leftarrow$ apply K-means algorithm to D_T^n
2. for $1 \le i \le M$ do
3. $\quad A_j \leftarrow \arg\min_j \| x_i - \mu_j \|^2$
4. end for
5. for $1 \le j \le k$ do
6. $\quad T_j \leftarrow$ apply E2LSH to A_j
7. end for

Because we apply K-means algorithm to partition the dataset, the distribution of data items in our hash tables is more uniform than that in original E2LSH hash tables.

3.2 Search Algorithm for Data-Dependent LSH Index

In LSH-based methods, *short-list* search is the main bottleneck of improving search speed [8]. To reduce the time cost of *short-list* search, we present a novel pruning algorithm for our data-dependent LSH.

In the first level, we partition the dataset into a number of clusters. Intuitively, these clusters have different sizes, and the relative distances between the cluster centers are also different. Due to influence of the size of different clusters and the relative distances, we incorporate these factors into the computation of distance and propose the distance metric $D_i^{'}$ as follows:

$$D_i^{'} = \frac{d_i}{d_1} \times \frac{s_1}{s_i} \cdot \tag{1}$$

We first compute the distances $d_i^{'}$ between the query q and all cluster centers μ_i and rank the distances in ascending order and get the sequence d_i . d_1 is the nearest center to q and s_1 is the size of corresponding cluster. In this paper, we use the amount of the items in each cluster as its size metric.

In order to perform similarity search, we first get the sequence $D_i^{'}$ as mentioned above and rank $D_i^{'}$ in ascending order to get the ordered sequence D_i . Then, we take the top m clusters $A_1, A_2, \cdots A_m$ as the candidate clusters to be probed. We apply E2LSH search algorithm to $A_1, A_2, \cdots A_m$ and perform *short-list* search to get the nearest neighbors of the query q . The procedure of the search algorithm is depicted in algorithm 2.

Algorithm 2. Search algorithm for Data-dependent LSH

Input: A query point q.

Output: the nearest neighbor of q.

Notations: $L_{D_i}(i = 1, 2, \cdots m)$, the clusters corresponding to D_i ; $m(1 \le m < k)$, the number of probed clusters; s_i , the size of corresponding cluster.

Operation:

1. for $1 \le i \le k$ do
2. $\quad d_i^{'} = \| q - \mu_i \|_2$
3. end for
4. $d_1 d_2 \cdots d_k \leftarrow$ rank $d_i^{'}(i = 1, 2 \cdots k)$ in ascending order
5. for $1 \le i \le k$ do
6. $\quad D_i^{'} = \frac{d_i}{d_1} \times \frac{s_1}{s_i}$
7. end for
8. $D_1, D_2, \cdots D_k \leftarrow$ rank $D_1^{'}, D_2^{'}, \cdots D_k^{'}$ in ascending order
9. $a_1, a_2 \cdots \leftarrow$ apply E2LSH search algorithm to $L_{D_i}(i = 1, 2 \cdots m)$.
10. $n = \arg\min_i \| q - a_i \|_2$
11. return a_n

In practice, algorithm 2 is a pruning strategy. As algorithm 2 takes advantage of the information about the relative positions and sizes of all clusters, we can improve the search speed while keeping high precision, as illustrated in the next section.

4 Experimental Results

In this section, we first study the influence of different parameters. Then, we conduct multiple comparison experiments between E2LSH [4], Bi-level LSH [8] and our data-dependent LSH.

4.1 Experimental Setup

In the experiments, we use the publicly available BigANN set [10]. BigANN set is composed of four data packages including ANN_SIFT10K, ANN_SIFT1M, ANN_GIST1M and ANN_SIFT1B. We use ANN_SIFT1M, which contains 1M SIFT descriptors as index data, 100k SIFT descriptors as training set and 10k SIFT descriptors as query data, as described in Table 1. The training set is used to learn the cluster centers.

Our method is single-threaded programmed and executed on a server which has 64G main memory, Intel Xeon E5-2620*2(2.00 GHz, 7.2GT/s, 15M cache, 6cores). We will draw the conclusion in term of the comparison of accuracy and efficiency.

Table 1. Description of training set, index set and query set

Dataset	Size
training set	100,000
query set	10,000
base set	1,000,000

For experimental evaluation, we use speedup to compare our two-level index with E2LSH [4] and Bi-level LSH [8]. The speedup is defined as follows:

$$speedup = \frac{linear\_time}{ann\_time}, \tag{2}$$

where *linear_time* is time cost of linear search and *ann_time* is time cost of our method, E2LSH or Bi-level LSH. Precision is another principle to measure the performance of an index algorithm. The precision is defined as follows:

$$precision = \frac{count(AR \cap LR)}{count(LR)}, \tag{3}$$

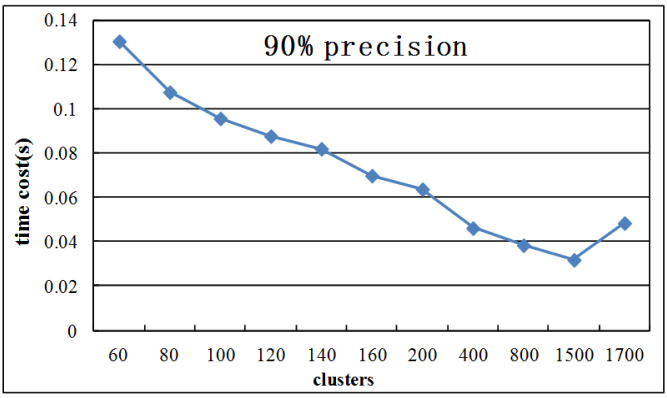
where *AR* is the ANN search result, *LR* is the linear search result. The function *count* is used to count the number of the result set. For each query, we calculate its precision and we take the mean value over all queries.

4.2 Influences of Parameters

The performance of our data-dependent LSH is influenced by two parameters *i.e.* numbers of clusters in building index and number of probed clusters in search.

To make out how these two parameters affect the performance of our method, we plot two diagrams and two tables which show how the time cost and the number of probed clusters changes along with the number of clusters in search precision of 90% and 96%, respectively.

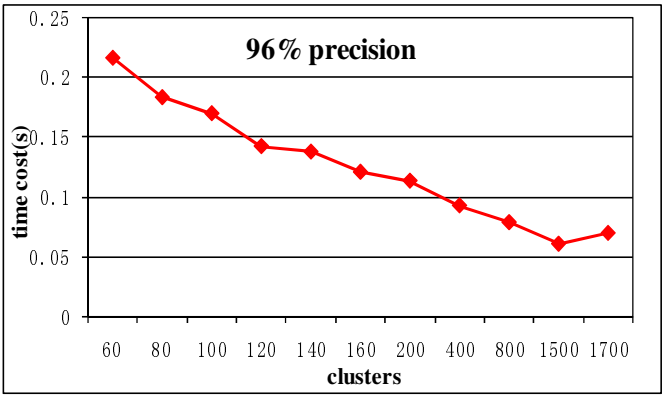
From Figure 2, we can see that to reach the same accuracy, the time cost declines when the number of clusters increases at the beginning, *e.g.* the number is less than 1500. But the time cost begins to increase when the number of clusters is larger than



(a)

clusters	60	80	100	120	140	160	200	400	800	1500	1700
probes	2	3	3	3	4	4	5	6	8	11	23

(b)



(c)

clusters	60	80	100	120	140	160	200	400	800	1500	1700
probes	4	5	5	6	6	7	8	12	17	23	34

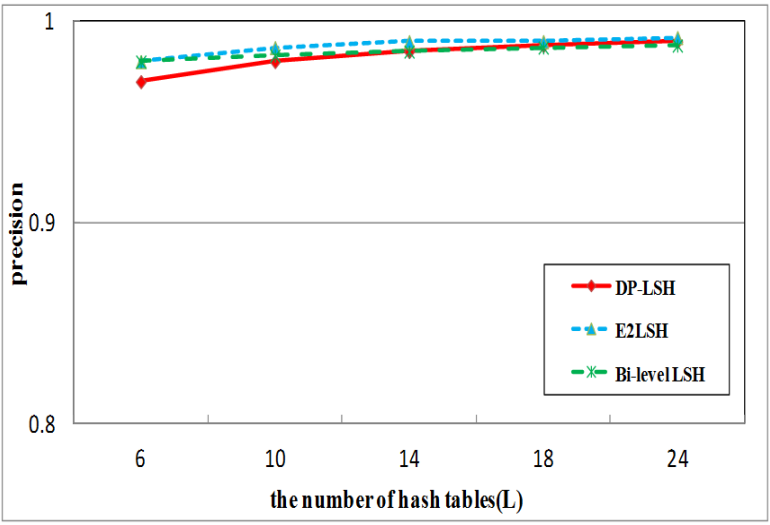
(d)

Fig. 2. (a), (c) The time cost changes along with the number of the clusters in precision of 90% and 96% respectively; (b), (d) the number of probed clusters with respect to different number of clusters in precision of 90% and 96% respectively

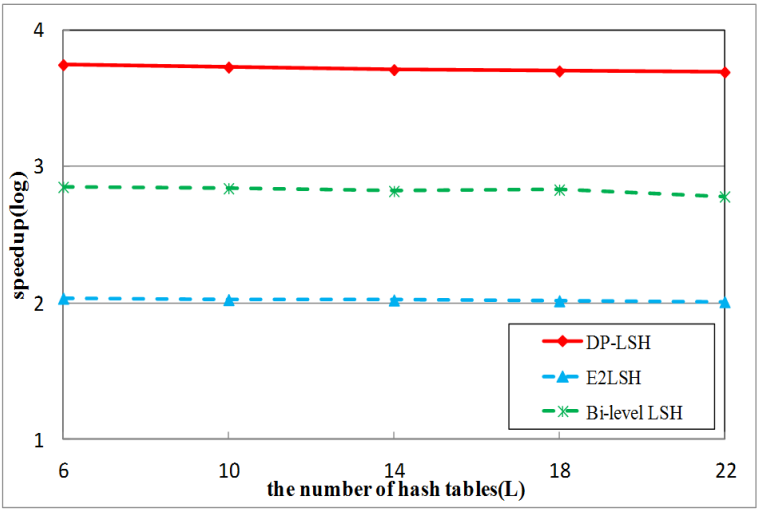
1500. The reason is that, in the beginning, with the increment of cluster number, the reduced amount of items in probed clusters is more than the increased amount of items in additional probed clusters to reach the same precision. But, the situation is reverse when cluster number is too large. Moreover, along with the increment of the number of clusters, the time cost of conducting step 1 to step 3 in algorithm 2 is also increased. So we choose 1500 clusters. As shown in Figure 2 (d), to reach high precision, we may choose 23 probed clusters.

4.3 Comparison of Overall Performance

To verify the performance of our data-dependent LSH (DP-LSH), we conduct comparative experiments among DP-LSH, E2LSH [4] and Bi-level LSH [8]. DP-LSH divides the dataset into 1500 clusters in the first level and the number of probed clusters is set to be 23. In the experiments, we compare the precision and speedup using different number of hash tables to testify that our algorithm can get better speedup compared to E2LSH and Bi-level LSH in all cases. Experimental results are shown in Figure 3.



(a) precision



(b) speedup

Fig. 3. (a) Search precision changes along with the number of hash tables; (b) speedup ($\log^{10}$) changes along with the number of hash tables

From Figure 3(a), we can see that the precision of DP-LSH, LSH and Bi-level LSH is varying during 0.97-1, and the precision of DP-LSH is about 1% lower than E2LSH when the number of hash tables is less than 14. Nevertheless, the precision of DP-LSH is almost equal to E2LSH and Bi-level LSH when the number of hash tables is larger than 14. So these three methods have very similar search precision.

From Figure 3(b), we can see that the speed of DP-LSH is about 48-50 times faster than E2LSH, using different number of hash tables. This efficiency superiority of DP-LSH is stable even when the number of hash tables changes. Besides, DP-LSH is much faster than Bi-level LSH. This is because that DP-LSH has a more uniform partition of dataset than Bi-level LSH, which results in better search efficiency. To our knowledge, this is the best performance of the state-of-the-art hash-based indexing methods.

Based on the above comparisons, we prove that DP-LSH can significantly improve the search speed while keeping high search precision. Moreover, the improved performance is stable.

5 Conclusions and Future Work

In this paper, we propose a new two-level hashing index structure and corresponding search algorithm. The goal is to deal with the problem that the LSH method loses efficiency when indexing non-uniform distribution datasets. Our index has two-level structure to perform ANN search in high dimensional spaces. In the first level, we divide the dataset into many clusters, to ensure the items in each cluster has near uniform distribution. In the second level, we construct LSH tables for each cluster. Besides, we also propose a novel search pruning method. In brief, the proposed method is simple and cheap, and has been proven by exhaustive experiments.

In future work, we will test our algorithm in more large scale datasets. We also need to design efficient out-of-core algorithm to handle many very large datasets (e.g. >10 billion).

Acknowledgement. This work is supported by the "Strategic Priority Research Program" of the Chinese Academy of Sciences (XDA06031000), National Nature Science Foundation of China (61303171, 61303175), Hunan province university innovation platform open fund project (14K037), National High Technology Research and Development Program (2011AA01A103).

References

1. Wan, J., Tang, S., Zhang, Y., Huang, L., Li, J.: Data Driven Multi-Index Hashing. In: IEEE International Conference on Image Processing (2013)
2. Zezula, P., Amato, G., Dohnal, V., et al.: Similarity Search: The metric space approach. Advances in Database Systems (2006)
3. Adonis, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Symposium on Foundations of Computer Science (2006)

4. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Symposium on Computational Geometry (2004)
5. Jegou, H., Amsaleg, L., Schmid, C., Gro, P.: Query adaptative locality sensitive hashing. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (2008)
6. Weiss, Y., Torralba, A., Fergus, R.: Spectral Hashing. In: Advances in Neural Information Processing Systems (2008)
7. Heo, J.-P., Lee, Y.: Spherical Hashing. In: IEEE Conference on Computer Vision and Pattern Recognition (2012)
8. Pan, J., Manocha, D.: Bi-level Locality Sensitive Hashing for k-Nearest Neighbor Computation. In: Very Large Data Base (2010)
9. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
10. Jegou, H., Douze, M., et al.: Product Quantization for Nearest Neighbor Search. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(1), 117–128 (2011)
11. Pauleve, L., Jegou, H., Amsaleg, L.: Locality sensitive hashing: A comparison of hash function types and querying mechanisms. Elsevier B.V. (2010)
12. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe LSH: efficient indexing for high-dimensional similarity search. In: Very Large Data Base (2007)
13. Bawa, M., Condie, T., Ganesan, P.: LSH forest: self-tuning indexes for similarity search. In: International Conference on World Wide Web (2005)
14. Dong, W., Wang, Z., Josephson, W., Charikar, M., Li, K.: Modeling lsh for performance tuning. In: Conference on Information and Knowledge Management (2008)
15. Babenko, A., Lempitsky, V.: The inverted multi-index. In: IEEE Conference on Computer Vision and Pattern Recognition (2012)
16. Xie, H., Zhang, Y., Tan, J., Guo, L., Li, J.: Contextual Query Expansion for Image Retrieval. IEEE Trans. on Multimedia 16(4) (June 2014)