Document Retrieval in Big Data

Feifei Pan
Computer Science Department
New York Institute of Technology
New York, USA
Email: fpan@nyit.edu

Abstract—**Nearest Neighbor Search for similar document retrieval suffers from an efficiency problem when scaled to a large dataset. In this paper, we introduce an unsupervised approach based on Locality Sensitive Hashing to alleviate its search complexity problem. The advantage of our proposed approach is that it does not need to scan all the documents for retrieving top-K Nearest Neighbors, instead, a number of hash table lookup operations are conducted to retrieve the top-K candidates. Experiments on two massive news and tweets datasets demonstrate that our approach is able to achieve over an order of speedup compared with the traditional Information Retrieval method and maintain reasonable precision.**

Keywords–Document Retrieval; Locality Sensitive Hashing; Big Data.

I. Introduction

The Nearest Neighbor Search (NNS) task [1] aims at automatically finding the top K objects (e.g., documents) which are most semantically similar to a given query object. NNS is essential to motivate the progress in many search related tasks and is fundamental to a broad range of Natural Language Processing (NLP) down-stream tasks, including name spelling correction [2], document translation pair acquisition [3], large-scale similar noun list generation [4], unsupervised mining of lexical variants from noisy texts [5], and large-scale first story detection from news and tweets [6].

Nowadays, data are being collected at unprecedented speed and scale everywhere around us: various news agencies produce thousands of news articles while Twitter generates over 500 million Tweets everyday. The traditional Information Retrieval (IR) method to tackle NNS is to represent documents in the vector space and find the candidate documents that share the highest probabilities with the query document [7]. However, it is very time consuming or even infeasiable to brute force compute the similarity score of the query document and all other documents in a large dataset. Thus, it is critical to find other solutions to deal with the search efficiency problem. In order to make it scale to big data, some researchers attempted to reduce the dimentionality of the representative vectors or add in time constraints to narrow down the search range [8]. Both of these works are able to save some computational costs, but can not solve the problem fundamentally.

Hashing has been successfully applied to several non-NLP problems including object recognition [9][10], image retrieval [11][12] and image matching [13][14]; however, it received limited attention in NLP fields. The general idea of hashing is to represent each document as a binary code (1-bit of a binary code is one digit of “0” or “1”). Its advantage is two-fold: (1). The capability to store large amount of documents in memory. For example, we can store 250 million documents with 16G memory using 64 bit for each document, while English Gigaword fifth edition [15] stores 10 million documents with 26G. (2). The time efficiency to process binary codes. For example, calculation of hamming distance between a pair of binary codes is far faster than cosine similarity over a pair of document vectors.

The paper is structured as follows: in Section I-A and II, we briefly introduce the terminologies and previous related work. In Section III, we compare our Locality Sensitive Hashing (LSH) based approach with the traditional IR method in tackling NNS task. We talk about the experiment results in Section IV and make the conclusion in Section V.

A. Background and Terminology

Here are some background information and terminologies: Bit is the basic unit of a binary code; one bit is either a digit of “0” or “1”. A binary code is a bit sequence assigned to represent an object. For example, we can represent a document as “00101100”. A hash table is a table containing all the binary codes for a set of documents while the documents with the same binary codes are located within the same bucket. Hamming Distance between two binary codes is the number of positions at which the corresponding bits are different. Hash Lookup is to find candidate neighbors in the hash table buckets within a given hamming distances from h_q, given a query q with a binary code h_q. In practice, the given hamming distance is usually set to 2. Hash Lookup Success rate is the probability to find any candidate neighbors in the buckets within a given hamming distances from h_q.

II. Related Work

Locality Sensitive Hashing (LSH) [16] is one of the notable schemes for data-independent hashing. It uses random projections to construct randomized hash functions, therefore similar data points have a higher probability to be mapped into the same bucket. To address the problem of learning similarity-preserving binary code for efficient retrieval from a large scale collection, several data dependent hash schemes were developed. Weiss et al. [17] designed Spectral Hashing (SH) which was motivated by spectral graph partitioning and it used a spectral relaxation to obtain an eigenvector solution. Liu et al. [18] utilized anchor graphs to discover the neighborhood structure inherent in the data, and Gong and Lazebnik [19] proposed an Iterative Quantization (ITQ) approach by minimizing the quantization errors. Generally speaking, data dependent hash schemes are able to learn better quality binary codes than randomized algorithms, so we adopt ITQ to hash documents to binary codes. However, in most of the data dependent hash schemes, the generated binary codes suffer from poor hash table lookup success rate problem which makes the learnt binary codes inefficient for practical use. In this paper, we aim to alleviate the search efficiency problem by taking the advantages of LSH.

III. Nearest Neighbor Search

In this section, we first show the motivation of adopting hashing techniques to NNS problem in Section III-A. Then, we introduce the details of LSH in Section III-B.

A. Traditional NNS

The most traditional way of finding Nearest Neighbors is to represent each document as a term vector, e.g., each element of the vector is the tf-idf weight of a term:

$$tf(t,d) = log(1 + f(t,d)) \tag{1}$$

$$idf(t,D) = \frac{log(|D|)}{log(|\{d \in D : t \in d\}|)} \tag{2}$$

$$tf\text{-}idf(t,d,D) = tf(t,d) \times idf(t,D) \tag{3}$$

where t is a term, d is a document, D is the whole corpus and f(t,d) is the frequency of term t in document d. Given a query document q, we used the cosine similarity metric [20] to judge the similarity of q with a document candidate c:

$$\begin{aligned} distance(q,c) = cos(\theta) &= \frac{q \cdot c}{||q||||c||} \\ &= \frac{\sum_{i=1}^{n} q_i \times c_i}{\sqrt{\sum_{i=1}^{n}(q_i)^2}\sqrt{\sum_{i=1}^{n}(c_i)^2}} \end{aligned} \tag{4}$$

The higher similarity score between q and c, the closer they are. To find out the top-K nearest neighbors for a query document q, one needs to first compute the cosine similarity scores between q and each document candidate, then pickup the K documents with the highest similarity scores. However, brute force search does not scale to big data since the computational complexity for each query is $O(n)$ and other computational costs such as similarity calculation and ranking similarity scores are not immaterial.

Hashing schemes aim to remove the curse of dimensionality and largely save computation cost. We will introduce LSH in the following section.

B. Locality Sensitive Hashing

The underlying intuition of LSH is that if two objects are close, then after a "projection" operation they will remain close together. In other words, similar data points are more likely to be mapped into the same bucket with a high collision probability.

Binary Code Learning

Given a LSH setting of M bits and L hash tables, a query data point q and a candidate data point c will collide if and only if:

$$h_{ij}(q) = h_{ij}(c), i \in [1...L], j \in [1...k] \tag{5}$$

and the hash function $h_{ij}(x)$ is defined as:

$$h_{ij}(x) = sgn(u_{ij}^T \cdot x), sgn(u) = \begin{cases} 1 & \text{u} \geq 0 \\ 0 & \text{Otherwise} \end{cases} \tag{6}$$

where u_{ij} are randomly generated vectors with components randomly selected from a Gaussian Distribution, e.g., $N(0,1)$. In this case, the probability of two points x_1 and x_2 colliding under LSH can be calculated as:

$$P_{coll} = 1 - \frac{\theta(x_1,x_2)}{\pi} \tag{7}$$

where $\theta(x_1,x_2)$ is the angle between x_1 and x_2. Given the desired probability of missing a nearest neighbor δ, we can approximate the required number of hash table L:

$$L = log_{1-P_{coll}^{M}}\delta \tag{8}$$

In this paper, given the term vector as input, we learn the binary code for each document and we try different settings for M and L to see how they influence the results.

Document Retrieval

After learning the binary codes for all documents, inverse table lookup operations are conducted to find nearest neighbors of a query document given the binary code of the query document h_q:

1) lookup the bucket that has the same binary code as h_q and retreive all the documents within the bucket. 1 hash table lookup operation is needed.
2) lookup the buckets that have Hamming Distance 1 with h_q and retreive all the documents within the buckets. $length(h_q)$ hash table lookup operation is needed.
3) lookup the buckets that have Hamming Distance 2 with h_q and retreive all the documents within the buckets. $length(h_q) \times (length(h_q) - 1)$ hash table lookup operation is needed.
4) Document candidates in the same buckets will be randomly pickup to form the top-K nearest neighbors.

We only conduct hash table lookup for the buckets which have Hamming Distance smaller than or equals to 2 with h_q. Otherwise, it requires too many hash table operations and it is not efficient for a large dataset.

IV. Experiments

A. Data

For the experiments, we use a news dataset and a tweets dataset, in order to see how the proposed methods perform for different genres. The news dataset is the English portion of the standard TDT-5 [21] dataset. It consists of $278,109$ documents from the 6-month time period since April 2003. 126 topics with an average of 51 documents per topic are annotated, and other unlabeled documents are irrelevant to them. 400 randomly selected labeled documents are used for testing. The tweets dataset is gathered through Twitter Application Programming Interface (API) from a time period between October 25th, 2012 and November 4th, 2012, filtered by hashtags '#hurricane' and '#sandy'. As a result, we collected 1.49 million tweets before, during and after Hurricane Sandy hit the northeast of the US.

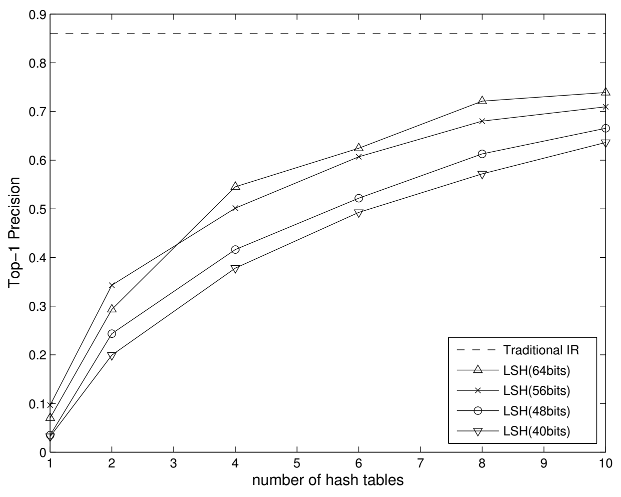

Figure 1: LSH Top-1 NNS Precision.

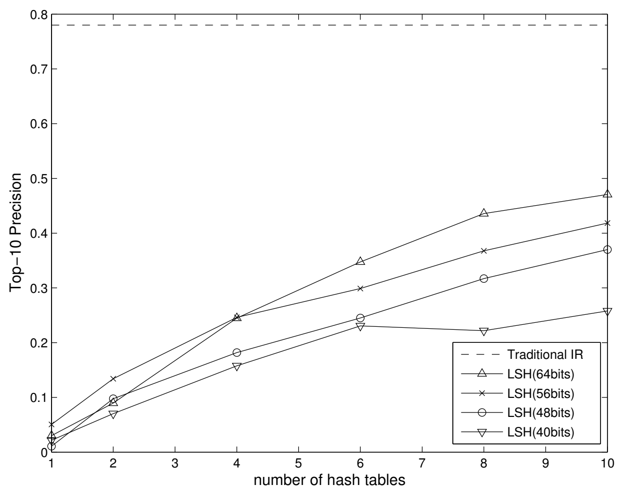

Figure 2: LSH Top-10 NNS Precision.

TABLE I: SAMPLE QUERY TWEETS AND THE CORRESPONDING TOP RETRIEVED SIMILAR TWEETS.

Query 1: @danburyweather: Mayor Bloomberg Tells NYC Residents: ‘Be Prepared To Evacuate’ Read more: http://t.co/oe2IUB2E
Mayor Bloomberg Tells NYC Residents: ‘Be Prepared To Evacuate’ http://t.co/eIXJBzmO
#Hurricane Sandy @businessinsider: Mayor Bloomberg Tells NYC Residents: ‘Be Prepared To Evacuate’ by @DinaSpector http://t.co/l76jFAAH
Watch out NYC
I can’t wait to hear Mayor Bloomberg say Zone Ah again in Spanish when talking about hurricane evacuation procedures. #Frankenstorm
For NYC residents - hurricane evacuation zone finder tool - http://t.co/mB1WSDPf
@DanSkeldonNBC40 Which storm was Bering hyped more Sandy or Irene? Do you think Cape May County residents need to evacuate for Sandy?
RT @EasternSurfMag: STAY ALERT and BE PREPARED this weekend
RT @Chels_sahagian3: Just watching the news about this hurricane is making me more and more scared:(
NYC Hurricane evacuation map
Watch there be a hurricane on Monday and Bloomberg stills makes us go to school. Jew bastard
Query 2: People return home from shelters after hurricane Sandy: More than 1800 people who were housed in shelters prior to the hurricane
People return home from shelters after hurricane Sandy http://t.co/36cSHU2r
Jamaica: More than 1800 people return home from sutlers after hurricane Sandy
At the height of #Sandy more than 1800 people were housed in Red Cross shelters
258 shelters in 16 states safely housed 11
@JeffreyYoung_HC are hospitals exempted from evacuation NYU endangered 300 lives for not evacuating prior to the storm. http://t.co/QvqQmwCG
@Tek_Roo FEMA organized with states prior to Sandys landing to get people evacuated and resources in place for rescue.
RT @nycgov: RI @NYCMayorsOffice: Mayor: we will keep shelters open until New Yorkers can safely return to their homes. #Sandy
West Deptford shelter housed 40 people during Hurricane Sandy
Thanks toRedCross258 shelters in 16 states safely hosed 11000 people. #Sandy Recovery begins today Every dollar helps http://t.co/ghhORZvz
Thousands in New York remain homeless and in shelters nearly a week after Hurricane Sandy. It seems like things are returning to normal.

We select 20 informative tweets as testing queries. For each tweet, we remove hashtags, URLs and @ information. For each news article and tweet, we apply the Stanford Tokenizer [22] for tokenization, remove stopwords based on the stop list from InQuery [23], and apply Porter Stemmer [24] for stemming.

To evaluate the system performance for new articles, we use the topic labels of documents as ground truth: if one retrieved document shares the same topic label with the query document, they are true neighbors. We evaluate the precision of the top-K candidate documents returned by each method and calculate the average precision across all queries. Since we do not have groundtruth for tweets, we will not report system performance for tweets dataset in the paper. Instead, we show system outputs given some tweet queries to demonstrate the effectiveness of our proposed method.

B. Results

Our LSH-based approach aims to alleviate the search efficiency problem of NNS. We compare the average search time of the traditional IR method and our LSH-based method. In the news dataset, LSH-based method only needs about one twentieth of the search time as the traditional IR while in tweets dataset the differences becomes even larger: LSH-based method only needs one twentieth of the time approximately. It clearly proves the superiority of our approach.

Furthermore, we compare the top-K NNS precision of the traditional IR method and our LSH-based method. Fig. 1 and Fig. 2 are the top-K NNS results for the news dataset. Generally speaking, with longer binary code length or more number of hash tables, the top-K NNS precision keeps increasing and it reaches convergence approximately in the setting of 64 bits and 10 hash tables. In Fig. 1, it is clear that when retrieving top-1 Nearest Neighbors, although LSH can not well approximate the performance of traditional IR, it is still able to produce an acceptable performance. However, in Fig. 2, LSH is surpassed by traditional IR by a large margin when retrieving top-10 Nearest Neighbors. It suggests that our approach is more reliable when K is small. Table I shows two sample query tweets and the correspoding similar tweets returned by the LSH-based approach with 64 bits and 10 hash tables setting. The returned tweets are the most tropically related results to the query tweets and it demonstrates that our approach can be adapted to tweets as well.

V. Conclusion and Future Work

In this paper, we propose an efficient LSH based solution for document retrieval in big data. Although our approach is unable to achieve similar performance as the traditional IR method, it boosts the search time over an order of magnitude. Experiments on two genres show that our approach is flexible and feasible in practical use, especially for retrieving a small number of documents in a large dataset. In the future, we plan to conduct manual annotation on the tweets dataset in order to carry out quantitative evaluations. After that, we will focus on improving the document representation to further boost the precision of the LSH based approach.

References

[1] A. Andoni, "Nearest neighbor search: the old, the new, and the impossible," in PhD Dissertation in MIT, 2009.

[2] R. Udupa and S. Kumar, "Hashing-based approaches to spelling correction of personal names," in EMNLP, 2010, pp. 1256–1265.

[3] K. Krstovski and D. A. Smith, "A minimally supervised approach for detecting and ranking document translation pairs," in The sixth ACL Workshop on Statistical Machine Translation, 2011, pp. 207–216.

[4] D. Ravichandran, P. Pantel, and E. H. Hovy, "Randomized algorithms and nlp: Using locality sensitive hash functions for high speed noun clustering," in ACL, 2005, pp. 622–629.

[5] S. Gouws, D. Hovy, and D. Metzler, "Unsupervised mining of lexical variants from noisy text," in EMNLP, 2011, pp. 82–90.

[6] S. Petrovic, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in HLT-NAACL, 2010, pp. 181–189.

[7] J. Allan, V. Lavrenko, D. Malin, and R. Swan, "Detections, bounds, and timelines: Umass and tdt-3," in In Proceedings of Topic Detection and Tracking Workshop, 2000, pp. 167–174.

[8] S. Petrovic, "Real-time event detection in massive streams," in PhD Theis at University of Edinburgh, 2012.

[9] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," in IEEE Trans. Pattern Anal. Mach. Intell., vol. 30(11), 2008, pp. 1958–1970.

[10] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in CVPR, 2008, pp. 1–8.

[11] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 12, 2009, pp. 2143–2157.

[12] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu, "Complementary hashing for approximate nearest neighbor search," in ICCV, 2011, pp. 1631–1638.

[13] S. Korman and S. Avidan, "Coherency sensitive hashing," in ICCV, 2011, pp. 1607–1614.

[14] C. Strecha, A. A. Bronstein, M. M. Bronstein, and P. Fua, "Ldahash: Improved matching with smaller descriptors." IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, 2012, pp. 66–78.

[15] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda, "English gigaword fifth edition (http://catalog.ldc.upenn.edu/ldc2011t07)," 2011.

[16] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in STOC, 1998, pp. 604–613.

[17] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in NIPS, 2008, pp. 1753–1760.

[18] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in ICML, 2011, pp. 1–8.

[19] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in CVPR, 2011, pp. 817–824.

[20] A. Singhal, "Modern information retrieval: A brief overview," in Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 24(4), 2001.

[21] K. M. S. S. David Graff, Junbo Kong, "Tdt5 multilingual text (https://catalog.ldc.upenn.edu/ldc2006t18)," 2006.

[22] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[23] J. Callan, W. Croft, and S. Harding, "The inquery retrieval system," in Proceedings of the Third International Conference on Database and Expert Systems Applications, 1992, pp. 78–83.

[24] M. F. Porter, "An algorithm for suffix stripping," Program, vol. 14(3), 1980, pp. 130–137.