

# Hierarchical clustering of large text datasets using Locality-Sensitive Hashing

Vasilii Korelin

Saint-Petersburg State University  
7-9 Universitetskaya Naberezhnaya,  
St. Petersburg, 199034, Russia  
+7 911 266 12 99  
vn.korelin@gmail.com

Ivan Blekanov

Saint-Petersburg State University  
7-9 Universitetskaya Naberezhnaya,  
St. Petersburg, 199034, Russia  
+7 921 339 53 43  
i.blekanov@gmail.com

## ABSTRACT

In this paper, we present a hierarchical clustering algorithm of the large text datasets using Locality-Sensitive Hashing (LSH). The main idea of the LSH is to “hash” items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar are. The main drawback of the conventional hierarchical algorithms is a large time complexity (e.g. Single Linkage method has time complexity of  $O(n^2)$ ) Proposed algorithm reduces the time complexity to  $O(Pn)$ . Here,  $P$  represents the maximum number of items going to the single bucket.  $P$  is a small constant as compared to  $n$  for the large number of buckets.

Clustering results of the hierarchical clustering algorithm, that uses LSH, are similar to the clustering results of the classical single linkage method. The main advantage of the hierarchical clustering algorithm, that uses LSH, is a significant increase in speed for large datasets clustering in comparison with classical algorithms.

## Categories and Subject Descriptors

H.2.8 [Database Applications]: Data mining; H.3.3 [Information Search and Retrieval]: Clustering.

## General Terms

Algorithms, Performance, Experimentation, Languages.

## Keywords

Hierarchical clustering, Locality-Sensitive Hashing, Minhashing, Shingling.

## 1. INTRODUCTION

For today clustering of the large text datasets (e.g. clustering of the web pages) is one of the urgent data mining issues. Conventional clustering algorithms allow creating clusters with some accuracy, F-measure and etc. but when it comes to the clustering of the larger datasets (high-resolution pictures, fingerprints or web pages) the vast majority of algorithms have poor speed performance [1]. For example, despite the fact that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*IWAIT*'15, Oct. 8–10, 2015, Aizu-Wakamatsu, Japan.  
Copyright 2015 University of Aizu Press.

Single Linkage algorithm allows detecting clusters in arbitrary shapes, it has a large time complexity of  $O(n^2)$  where  $n$  is number of objects. Such time complexity is inappropriate for big data clustering. To avoid the dimensional issue new clustering algorithm that uses LSH method was proposed. LSH reduces the dimensionality of high-dimensional data. It hashes input items so that similar items map to the same “buckets” with high probability (the number of buckets being much smaller than the universe of possible input items) [2].

## 2. DIMENSIONAL REDUCTION FOR MINING MASSIVE DATASETS

There are many ways to represent documents as sets for the purpose of identifying lexically similar documents. The most effective way is to construct from the document the set of short strings that appear within it (shingles). Further, for document comparison Minhashing method is used that allows comparing multidimensional sets. These conversions must be performed before applying fast hierarchical algorithm with LSH using. Below are descriptions of Shingling, Minhashing, LSH methods and new proposed algorithm. Figure 1 presents the process of finding similar items in large documents collection for every document.

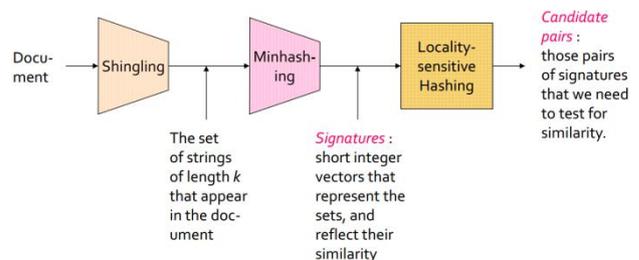


Figure 1. Dimension reduction of documents datasets

### 2.1 Shingling

A document represents a string of characters. Shingling is a process that creates sets of  $k$ -shingles. Define a  $k$ -shingle for a document to be any substring of length  $k$  found within the document [3].

For example,  $k=2$ ,  $doc="a b c a b"$ . Then after shingling next sets can be obtained:  $\{a, b\}$ ,  $\{b, c\}$ ,  $\{c, a\}$ .

Similar documents to each other will have a lot of equal shingles.

## 2.2 Jaccard Similarity

In this paper to determine the similarity of two sets Jaccard similarity is used. The Jaccard similarity of sets  $S$  and  $T$  is the ratio of the size of the intersection of  $S$  and  $T$  to the size of their union [4].

$$Sim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

Figure 2 depicts two sets  $C_1$  and  $C_2$ . There are 3 elements in their intersection and a total of eight elements that appear in  $S$  or  $T$  or both.

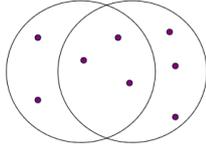


Figure 2. Two sets with Jaccard similarity 3/8

Jaccard similarity of two sets equals:

$$Sim(C_1, C_2) = \frac{3}{8}$$

Every document can be represented as set of  $k$ -shingles. Therefore the sets of documents can be represented as quite sparse boolean matrix. Rows are elements of the universal set (e.g. the set of all  $k$ -shingles). Columns correspond to the documents. 1 in row  $e$  and column  $S$  if and only if  $e$  is a member of  $S$ . Column similarity is the Jaccard similarity of the sets of their rows with 1. Figure 3 depicts the boolean matrix that represents two documents.

	$C_1$	$C_2$
$a$	1	1
$b$	1	0
$c$	0	1
$d$	0	0

Figure 3. Boolean matrix representing two documents

$Sim(C_1, C_2) = \frac{1}{3}$  – similarity of two documents.

## 2.3 Minhashing

Typical for the large number of documents boolean matrix is quite sparse. Therefore its further processing will be very time consuming. To solve this problem boolean matrix can be compressed to the signature matrix  $M$  in such a way that we can still deduce the similarity of the underlying sets from their compressed versions. This technique is called “Minhashing”.

To minhash a set represented by a column of the characteristic matrix, a permutation of the rows should be picked. The minhash function  $h(c)$  of any column is the number of the first row, in the permuted order, in which the column has a 1 [5].

Therefore the number of random permutations determines the number of the minhash functions. For example we can use 100 random permutations to create 100 signatures for every column of the boolean matrix.

Signatures can be stores in the special signature matrix  $M$ . The rows of the signature matrix are minhash values and columns

correspond to the documents. Figure 4 presents the example of the signature matrix  $M$  creation by using 3 minhash functions (3 random permutations).

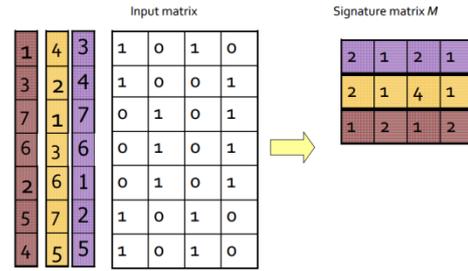


Figure 4. Minhashing

The probability (over all permutations of the rows) that  $h(C_1) = h(C_2)$  is the same as  $Sim(C_1, C_2)$ . The similarity of signatures is the fraction of the minhash functions in which they agree. Thus, the expected similarity of two signatures equals the Jaccard similarity of the columns or sets that the signatures represent. And the longer the signatures, the smaller will be the expected error [5].

This important feature allows compressing large sparse boolean matrixes to the signature matrixes with short defined number of rows with preserving similarity between rows. Thus, every document can be represented as a vector and its number of elements equals the number of minhash functions.

In spite of boolean matrix is compressed it still may be impossible to find the pairs with greatest similarity efficiently. The reason is that the number of pairs of documents may be too large, even if there are not too many documents.

## 2.4 Locality-Sensitive Hashing

**General idea:** Generate from the collection of all elements (signatures in our example) a small list of candidate pairs: pairs of elements whose similarity must be evaluated. For example, for signature matrix every column should be hashed several times and columns with equal hash values should be placed to the same bucket. Candidate pairs are those that hash at least once to the same bucket. To compare similarity of two sets threshold  $t$  should be picked ( $t < 1$ ). A pair of documents is considered to be similar only if their signatures agree in at least fraction  $t$  of the rows [6].

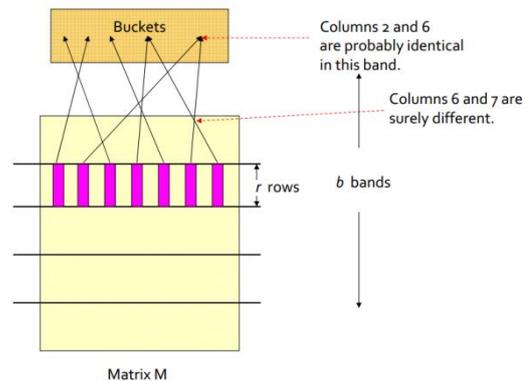


Figure 5. Hash functions for one band

An effective way to choose the hashings is to divide the signature matrix  $M$  into  $b$  bands consisting of  $r$  rows each (Figure 5). For each band, hash function takes vectors of  $r$  integers (the portion of one column within that band) and hashes them to some large number of buckets. Same hash function can be used for all the bands, but for each band there should be a separate bucket array, so columns with the same vector in different bands will not hash to the same bucket. Those columns that at least once were hashed to the same bucket are considered as candidate pairs. To catch most similar pairs, but few non similar pairs,  $b$  and  $r$  should be tuned attentively.

### 3. HIERARCHICAL CLUSTERING USING LSH

Proposed algorithm that exploits the hash tables generated by LSH. This algorithm outputs clustering results that approximate those obtained by the single linkage method [7]. The following is a detailed description of the algorithm.

#### Preconditions:

1.  $t < 1$  – threshold that determines the Jaccard similarity of 2 documents.
2.  $r$  – the initial value of the rows in each band. It should depend on number of signatures in signature matrix.
3.  $r_{min}$  – the minimum value of the rows of signatures in each band.
4.  $\Delta$  - parameter is used for  $r$  reduction.
5. Each document is view as single cluster.

#### Steps:

**Step 1:** For each band, hash vectors of  $r$  integers to the buckets. In the  $i$ -th hash table, column  $d$  (document) is stored in the bucket with index  $h_i(d)$ . However, if another column belonging to the same cluster as  $d$  has already been saved in the very bucket,  $d$  is not stored in it.

**Step 2:** For each column  $d$ , from the set of columns that enter the same bucket as  $d$  in at least one hash table, find columns whose distances from  $d$  are less than  $t$ .

**Step 3:** The pairs of clusters, each of which corresponds to a pair of columns obtained in Step 2, are connected (Figure 6).

**Step 4:** If  $r \leq r_{min}$ , algorithm terminates. Otherwise, advance to Step 5.

**Step 5:**  $r = r - \Delta$ . Advance to Step 1.

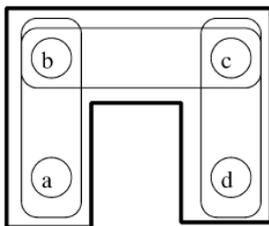


Figure 6. Merging of clusters

### 4. CLUSTERING EVALUATION

Reuters 21578, test collection of documents, was used to evaluate quality and performance of hierarchical clustering algorithm with LSH. New algorithm was compared with classical Single Linkage method. Table 1 presents that quality of clustering (accuracy and F-measure) for 1000 and 10000 number of documents.

Table 1. Comparison of the clustering quality

Algorithm	Documents count	Accuracy	F-measure
Single-Link	1000	75%	68%
	10000	79%	71%
Single-Link + LSH	1000	72%	73%
	10000	72%	74%

Clustering results show that accuracy and F-measure of two algorithms are similar.

Figure 7 presents dependence of the execution time of the number of input documents.

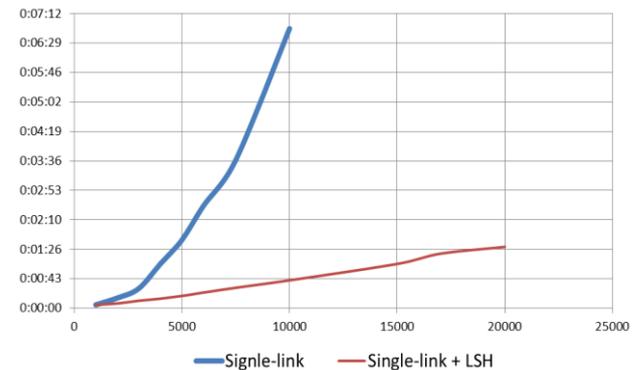


Figure 7. Comparison of the execution time

Analyzing the graphs, one can conclude that the clustering algorithm using LSH is much faster than the algorithm without LSH. Execution time of the algorithm with LSH increases linearly with an increasing number of documents.

### 5. CONCLUSIONS

In this paper we have proposed fast hierarchical clustering algorithm that uses LSH algorithm (LSH used for reducing the dimensionality of high-dimensional data). Developed algorithm is optimally suited for the massive text datasets clustering. Proposed algorithm was tested on Reuters collection of documents and showed reasonable accuracy and F-measure in comparison with classical clustering algorithms but in speed new algorithm is much superior to them. Fast developed clustering algorithm that uses LSH can be modified and used for clustering and analysis in such areas as medicine, criminalistics, sociology, etc. [2].

### 6. ACKNOWLEDGMENTS

This work was supported by the Russian Foundation for Basic Research, grant № 15-01-06105.

## 7. REFERENCES

- [1] A. Ene, S. Im, and B. Moseley. Fast clustering using MapReduce. In KDD, pages 681–689, 2011.
- [2] J. Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.
- [3] Broder, A.Z. Identifying and Filtering Near-Duplicate Documents. In proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, pp. 1-10, 2000.
- [4] Jatsada Singthongchai and Suphakit Niwattanakul, "A Method for Measuring Keywords Similarity by Applying Jaccard's, N-Gram and Vector Space," Lecture Notes on Information Theory, Vol.1, No.4, pp. 159-164, Dec. 2013. doi: 10.12720/Init.1.4.159-164.
- [5] CHUM, O., PERDOCH, M., AND MATAS, J. 2009. Geometric minhashing: Finding a (thick) needle in a haystack. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 17–24
- [6] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In VLDB, 1999.
- [7] Koga, Hisashi, Tetsuo Ishibashi, Toshinori Watanabe. 2007. Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing. *Knowledge and Information Systems* 12.1, 2007.