Optimal Incremental Clustering for Event Detection on Random Twitter Data using Bit-Locality Sensitive Hashing

[1]Abdullah K-K. A, [2]Ogunyinka P. A., [3]Sodimu S. M. & [4]Solanke O. O.
Department of Mathematical Sciences
Olabisi Onabanjo University
Ago Iwoye, Ogun State Nigeria.
E-mails: [1]uwaizabdullah9@gmail.com [2]pixelgoldprod@gmail.com , [3]princesgzy01@gmail.com [4]olasolanke@gmail.com
Phone: +2348060046592

ABSTRACT

Social network is overwhelmed by huge amount of unstructured data, ability to find specific content or events in large collections of textual documents such as Twitter data stream is important. This requires a substantial effort of information filtering to investigate the relevant topics and events. Different approaches have been used to deal with event detection or trending topics problem but suffer from high computational cost, low quality results, time and scalability. Clustering with large numbers of attributes and variables depends on effective similarity search (nearest neighbour) algorithms. In order to explicitly capture the optimality of tweets event, a method is proposed to improve clustering efficiency for large data by integration of a Bit-Locality Sensitive Hashing (BLSH) as a cluster search space reduction. This study employs Jaccard similarity and performs independent random permutation minHash on random twitter data. The data representation on k-appropriate nearest neighbour (k-ANN) represent all the data points in binary and maps the complex high-dimensional dataset to feature space for faster clustering. It relates the ANN with higher probability for data that are similar which corresponds to the smaller bit data representation. The BLSH on DBSCAN algorithms analysed the effect of representations in N-grams (uni, bi and tri-gram) to mine the concepts needed to handle event containing multiple topics/events after filtering retweets and preprocessing. Finally, an evaluation is performed on the basis of cluster quality index, volume of activity of tweets for efficiency, CPU time as well as scalability by trending the tweets events to improve computational cost.

Keywords: Bit-locality sensitive hashing, Approximate nearest neighbour, MinHash, Incremental clustering

Aims Research Journal Reference Format:
Abdullah K-K. A, Ogunyinka P. A., Sodimu S. M. & Solanke O. O. (2019): Optimal Incremental Clustering for Event Detection on Random Twitter Data using Bit-Locality Sensitive Hashing. Advances in Multidisciplinary Research Journal. Vol. 5. No. 1, Pp 39–54
Article DOI: dx.doi.org/10.22624/AIMS/V5N1P5
Available online at www.aimsjournal.net

1. INTRODUCTION

Social media is widely used as a source of information for the events detection in Twitter, hence, generates rich and timely information about real-world. Real time event detection in Twitter detects events at real time from live tweet stream as soon as an event happened. Event detection in Twitter streams has become an important task as it is used in various social and environmental events such as earthquakes, deaths of celebrities and elections etc. Researchers have used different approaches to deal with event detection problem such as topic modelling, incremental clustering based, frequency based approaches and/or term interestingness [1, 2] but suffer from high computational cost. Incremental clustering algorithms is utilised for detecting events from Twitter stream where the similarities between a tweet and event clusters are computed for identifying events. With the tremendous growth of information in social media, there is an increasing need for an efficient similarity searching method that can locate desired information with low cost.

Many researches have been done in analysing Twitter content for tracking real-world events and contain large amounts of meaningless messages and rumours [3] but can still be used to build users' social networks. This helps to understand people's reactions and opinion to events, subsequently, affect event detection performance negatively [4]. The objective of event detection is to discover new or previously unidentified events, where each event refers to a specific thing that happens at a specific time and place [5]. With the huge amount of Twitter data, mining this unstructured data with diverse number of topics, extracting its underlying topics or events has become a major challenge. The system developed by TwitInfo [6] allows a user to input event related keywords to track an event. The system finds tweets that match the user specified keywords but takes longer time in a high voluminous Twitter data. In order for Twitter data to be useful, events need to be identified with a very low latency and low computational cost. Hence, redundancy among tweets represents the same events and makes event detection becomes inevitable, therefore, poses scalability problem.

Analysing event detection in Twitter involves clustering approaches which group similar data objects together, so that the objects in each group (cluster) share the same topics or pattern of information [7]. When searching in an unstructured twitter environment, many searching methods generate a high-dimensional vector for each object, hence, conduct the Nearest Neighbours (NN) searching [8]. The nearest neighbour search and similar pair identification problems occur in various application domains. Also, traditional information searching methods rely on linear searching or tree structure but require substantial memory space or time and less efficient [9]. Using k-means for searching, builds a search tree using recursively method on each group in the dataset until its reach maximum level [10], this increases computational cost. Consequently, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) handle noise points and deal with data of any type but has high time complexity of $O(N \log N)$ and poor ability to deal with large-scale data. However, locating similar event in high dimensional data is not easy, as well as adaptation to data insertion and deletion is also an issue.

The problem represents each tweet as terms in feature vector and the closest tweet is computed by iteratively comparing tweet to one another. Comparing each tweet against one another in large documents space becomes infeasible as it grows over time. Meanwhile, clustering with a massive number of clusters, where items may be of high dimension, therefore, measure the similarity of each item to each cluster centroid is difficult in optimising the clustering task. This cannot be solved directly because it generates high dimensional vector as well as shingles problem when the dataset size is very large. Based on diversity of similarity measures, Strehl *et al.,* [11] compared the effectiveness on a number of measures in text document clustering but pose a lot of problem, therefore, the approach can only be effective on low volume Twitter data for event detection [5]. This raises a major question about which distance or similarity measures to be used for the clustering where items are with high dimensions. In order to achieve a better results, well-understood algorithm is adapt to handle large scale twitter data [12].

In order to overcome this problem, a hash function is employed to handle large scale twitter data using Locality Sensitive Hashing (LSH). LSH finds an appropriate nearest neighbour in $O(n \log n)$ time latency at low computational rate while preserving dimensionality reduction of nearest neighbour search in binary [13]. In many existing locality sensitive hashing methods**,** multiple hash functions were used at a time to find the nearest neighbours where each hash function has its own bucket set. This implies that if there are *n* hash functions and each hash function has *m* buckets. This method increases storage overhead since there are $n \times m$ buckets. In data usage, constructing hashing functions require data-independent method which does not take care of data distribution because data distribution affects the formation of hash functions. According to Kumar et al., [14], clusters constructed by Jaccard functions had no or very less noise points. Meanwhile, Jaccard similarity is proven to be successful for high dimensional sparse feature sets while it has shown that Euclidean distance is not the best distance measure for document clustering [15**].**

In this paper, a Bit-Locality Sensitive Hashing (BLSH) is employed with incremental clustering techniques for reducing the cluster search space in Twitter stream by varying the number of tweets at different time intervals with varying hashing values**.** The hash function handles large scale twitter data based on minHash distributions for Jaccard similarity. These are done by hashing each tweet using randomly chosen hash function and generate candidate pair with higher probability. Meanwhile, minHash computes each set of words in the tweets, convert them into k-bit signatures so that it becomes more compact representation. The hash codes/values provide estimate to pairwise similarity, convert tokenised text into sets of hash integers and selects the minimum values. Then, all pairs of tweet are hashed to the same bucket by at least one hash function. This eliminates dissimilar clusters and improves the efficiency of the clustering algorithms. The binary data (bit) representation on k-appropriate nearest neighbour maps the dataset into the distance measure for faster clustering. In the incremental clustering techniques, Aiello et al., [16] reported that N-grams achieve higher accuracy in detecting trending topics/events than document pivot method, so it was left out in this work. Finally, evaluations are done on random twitter stream datasets with proper noun are detected as a new event to determine accuracy of the algorithms (BLSH and DBSCAN with N-grams).

The following research questions are studied:

- Will the efficiency of BLSH improve the cluster region at a varying hash function?
- Will the different values of parameters affect the efficiency of BLSH and DBSCAN? Can it scale to large scale datasets?
- Can efficiency of incremental clustering algorithms (DBSCAN with N-grams-(uni, bi and tri-gram) affect the performance?

The rest of the work is structured as follows: Section 2 presents a brief review of related work. Section 3 introduces a BLSH based similarity searching scheme with minHash independent permutations. Section 4 describes and analyses the BLSH searching scheme and the performance on the incremental clustering algorithms. Section 5 concludes the work with recommendation on possible future work.

2. BACKGROUND OF THE STUDY AND RELATED WORK

Event detection is regarded as an application of online clustering, in which the objects to be clustered are breaking news and general topics and/ or tweets [5, 12]. Events that are unknown in Twitter are typically driven by emerging events that attract the attention of a large number of users. Event detection takes as input an unbounded stream of texts, where each text document D has a unique id, arrival time and content. It outputs clusters of text documents $[D_i \ldots D_n]$ representing events. Event detection has a problem of incremental clustering context of Twitter data stream which can be categorised into two (2) stages: Detecting burst in the number of tweets describing the topic or event and secondly grouping clusters that describe the same event.

Allan *et al*., [5] applied nearest neighbour approach (*k*-NN) for clustering news articles to discover the same event but the main problem of the approach is the representation of datasets in high d dimensional vector. However, the speed of linear search in the *k*-nearest neighbour approach has been significantly improved by an approximate nearest neighbour (ANN) search [14], although, they do not scale well with high dimension data. Monitoring and analysing rich and continuous flow of user-generated content can yield unprecedentedly valuable information from Twitter. According to Atefeh and Khreich, [17], event detection can be organised into three different approaches, these are: Term Interestingness Based (common traits), Probabilistic topic modeling and Incremental clustering. Each of these approaches have shortcomings, event detection on term interestingness capture misleading term correlations as well as measure term correlations which can be computationally prohibitive [18.19]. Subsequently, topic modeling approach incurred high computational cost [20] while incremental clustering tackle the problem of high computation cost using locality sensitive hashing.

A hash function is locality sensitive if two points that are close under the similarity distance measure D are more likely to collide. Therefore, Locality Sensitive Hashing is an indexing method that map object from a matrix domain U in d-dimensional space $\Re$ to a set of integer $\aleph$, in which nearby point is high dimensional space are hashed to the same value with higher probability.

With a distance function $d : U \times U \rightarrow \Re$

A Hash function $H = \{h : U \rightarrow \aleph\}$ $\forall$ (d_1, d_2, p_1, p_2) - sensitive for a given dataset $N \subseteq U$

Then, the following two (2) holds: For any point p, a query $q \in D$, $h \in H$,

- If $d(p,q) \leq d_1$ then $\Pr H[h(q) = h(p)] \geq p_1$ (colliding within the same radius)
- If $d(p,q) \leq d_2$ then $\Pr H[h(q) = h(p)] \leq p_2$ (colliding outside the same radius)

With the function, close objects within distance d_1 are like to have the same hash value $(p_1 > p_2)$ while far apart with d_2. In LSH, a set of points in U is preprocessed and stored into L number of buckets. For each point p $\in U$ is hashed using k independent, uniform, randomly selected hash functions $h_i(p) = (h_1(p), h_2(p), \cdots h_k(p)$ and stored in the bucket. Finding k point in $N_1 \ldots N_k$ $\forall$ the distance d_i to the query q is at approximate nearest neighbour $(1+\varepsilon)$ times the distance (d) from the ith nearest point to q.

Using BLSH provides solution to approximate nearest neighbour search (similarity search). This transforms the data item to a low dimensional representation or a short code consisting of a sequence of bits. LSH has large memory consumption, therefore, requires large number of hash tables for the entries of near neighbours [21], therefore, does not need to search the data point in the entire Twitter sets. There are several incremental clustering algorithms in event detection that uses locality sensitive hashing. The authors [22] presented K-Means Locality Sensitive Hashing (KLSH)) as a means of speeding up nearest neighbour search of large vectors but require knowing the number of clusters in the data. The work presented in [23] utilised LSH in clustering web pages in which a graph is created and each node represents a web page. Then, each edge indicates that the two pages are similar, a graph partitioning algorithm is applied to divide the web pages into different clusters. However, DBSCAN cannot handle data clusters of differing densities because its density-based definition of core points cannot address the core points of varying density clusters [24]. While the work presented in [25] uses an *N*-gram based event modeling approach that uses content analysis approaches for grouping large volume of tweets.

Meanwhile, there are lots of works on semi-supervised or supervised hashing methods which capture not only the geometry of the original data, but also the semantic relations. Also, an unsupervised method that is data-dependent hashing methods are used in spectral hashing, anchor graph hashing, kernelized locality-sensitive hashing (KLSH) to approximate the angular similarity in very high or even infinite dimensional space [26]. bBt suffers from the large variance estimation that leads to large estimation error as well as long code to accurately approximate the angular similarity. Yang *et al.,* [27] adapted hierarchical bottom-up clustering technique performed on two tasks in information retrieval for event detection that is retrospective detection and online detection The former task detects events from accumulated data and the later finds events from news feeds in real-time. Becker et al., [28] used an incremental clustering algorithm to detect events from the Twitter stream. Similarity is computed for each tweet higher than the threshold otherwise a new cluster is created. A Support Vector Machine based is used to classify between the newsworthy events and the non-events but cannot be used for large datasets.

Petrovic *et al.,* [12] adapted the online New Event Detection (NED) approach which is based on cosine similarity between documents to detect new events and focused on improving the efficiency with constant time and space using locality sensitive hashing methods but limit the search to a small number of documents. Ryan *et al*., [29] used K-Modes with LSH to significantly reduce the cluster search space with categorical clustering algorithm, however, the distances for categorical values are not so easily defined. These approaches are different from the approach use in this work because the LSH is used to speed up the clustering algorithm, hence, building graph partitioning is infeasible for very large dataset. Also, N-gram gives poor clusters but in this work only uni-gram, bi-gram and tri-gram are used which will perform better with BLSH. Finally, Slaney *et al.,* [30] used optimization parameters for LSH on a different variety of databases consisting of images to achieve the expected results. The authors concluded that it is difficult to find the best values for LSH parameters especially in large datasets.

3. THE APPROACH METHODOLOGY

This section describes the basic concepts of event detection with Bit-Locality Sensitive Hashing on incremental clustering techniques. Bit-Locality Sensitive Hashing (BLSH) is used for appropriate nearest neighbour search which involves two steps: Index construction and Object query. With hash functions, index construction with binary representation projects similar data points into the same hash bucket with a higher probability for tweets that are close to each other than those that are far apart. While object query uses a filter and refine framework to hash the data into the hash bucket through the same hash functions. In this study, tweets are grouped into several time intervals as well as varying the number of clusters to validate how the method is scaled with the large datasets. This determines the cluster search space for each incremental clustering techniques used. Consequently, computes similarity measure between a pair of tweets and/or a cluster representation however, eliminate tweets and clusters whose members are under a threshold.

- **Collection of Twitter Datasets**
  Twitter streaming API is used to collect live tweets continuously that cover a large datasets with time ranging period for twitter events. Tweets sample on February 8, 2019 from a period of 10 am to 10.30 am are used. In order to evaluate the accuracy of the pairwise similarity on the datasets, Bit-Locality Sensitive Hashing (BLSH) is tested on varying datasets D. After collection of Twitter datasets, representations of set of tweets are done before similarity matching.

- **Pre-processing and Representation**
  After collection of the datasets, pre-processing such as tokenisation, stemming and lemmatisation are done on the tweets using Natural Language ToolKit (as depicted in figure 1) so as to obtain a new version of Twitter datasets. Tokens which contain Username, stop words, and URLs are removed in the preprocessing phase. This reduces the number of tokens and hence do not contribute to clustering of related tweets event. Tokens that contain hashtags are retained, as hashtags often contain important information. All tweets are converted to English for effective clustering.

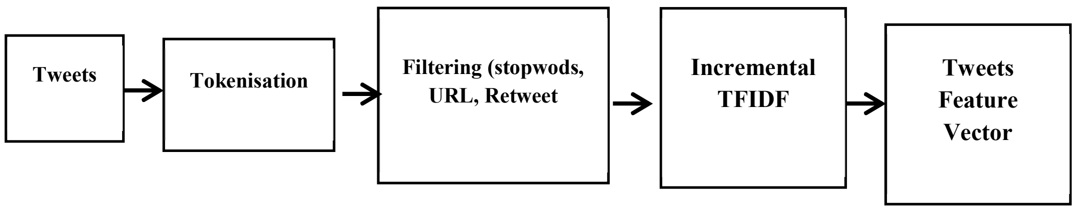


Figure 1: Preprocessing and Feature Representation

After preprocessing of Twitter datasets, representations of set of tweets are done before similarity matching. Choosing the right size of shingle is important so as to avoid repetition across the tweets and similarity problem. This can lead to false positive and and false negative as well as memory cost when shingles hashed to the bucket. Shingles of words are constructed from the tweets and this can be done by constructing a set of word in N-grams and/or in uni-gram, bi-gram and tri-gram tokens.

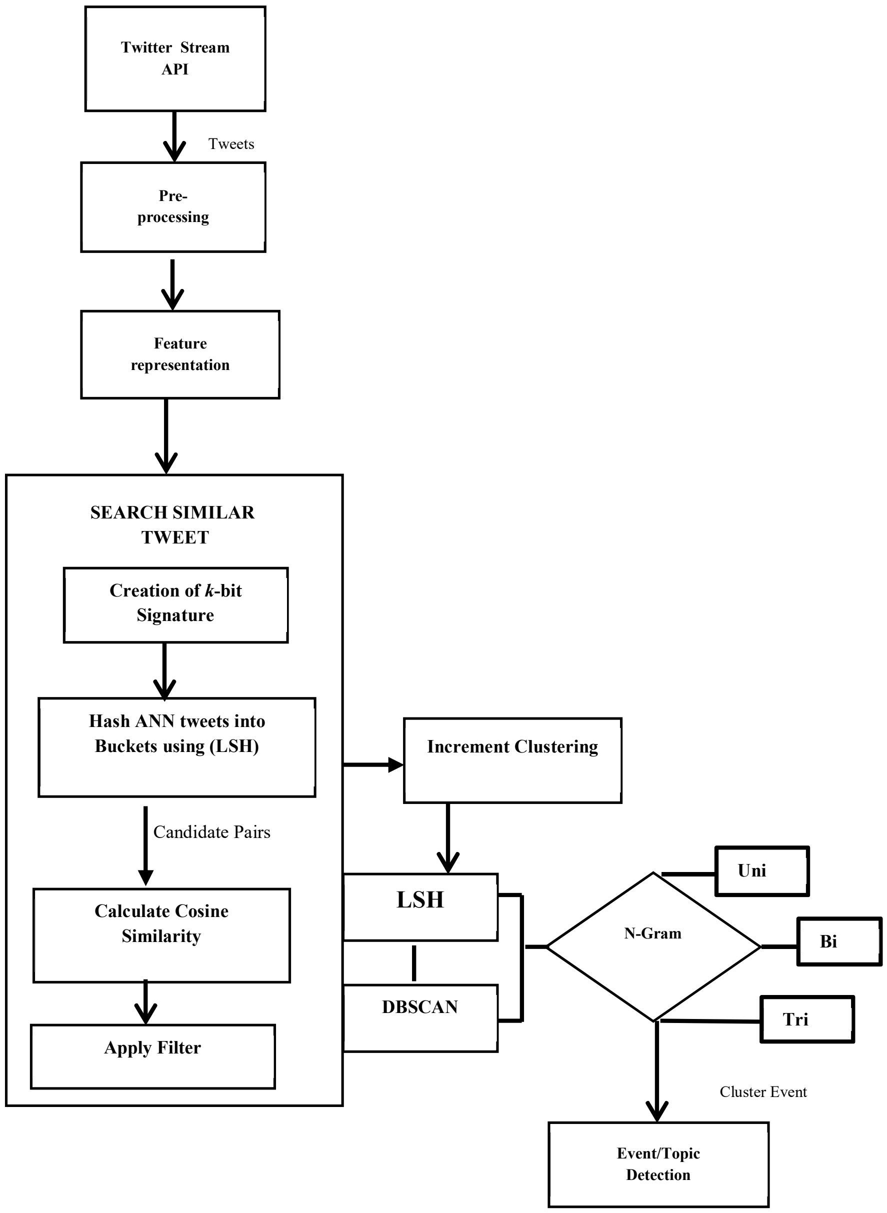


Figure 2: The Proposed Method for Event Detection

- **Feature Representation**

The N-gram is a sequence of tweet words of length n applied to construct bags of words, split the tweets into words and group them using a combination of *N*-grams. Each extracted collection of tweets compute the term frequency inverse document frequency $(tfidf_t)$ is based on the frequency of *N*-grams occurrences in some tweets at a certain time slot compared to the frequency of N grams occurrences in some previous time slots as uni, bi or tri- gram respectively. $Tfidf_t$ can be defined as:

$$tfidf = \frac{tf_i - +1}{\log\left(\frac{\sum_{j=i}^{t} tf_{i-j} + 1}{t}\right)} \qquad (1)$$

where tf_i is the frequency of n-grams occurrences in tweets at time slot i,

wf_{i-j} is the frequency of n-grams occurrences in tweets in the previous $i-j$ time slots,

and t is the number of all time slots.

- **Signature Matrices and Minhashing for BLSH**

Feature space representation is usually high dimensional and very sparse, only a small portion of features appear in a single instance of tweets. In order to reduce the memory used to store sparse vector, a signature is used which is an integer vector to represent *N* elements at an instance. A signature generates different permutation of features with fixed length k (64, 128, 256, and 512) by converting each set of terms in the tweets into signatures as represented in equation 2.

$$S_i = S_1, S_2 \ldots S_k \qquad (2)$$

The bit representation on k appropriate nearest neighbours represents all the data points in binary and maps the complex high-dimensional dataset to distance space. By independently sampling *k* d-dimensional vectors *S* from the normal tweets distribution $N \subseteq U$, the hash function can be define as $h(i) = h_1, h_2, \cdots, h_k \in H$ which consists of fixed length *k* as represented in algorithm 1. This represents a compact of each tweet and each element of the signature is a Minhash value.

Algorithm 1: MINHASH SIGNATURE GENERATION

```
Input: A vector from a single item
Input H: hash functions $h_1, h_2, \cdots, h_k \in H$
Output: Signature (S): a vector of length k
 for all i in item do
     $h_{\min}(i) = N$
 for all i in item do
     for all j in H do
         if $h_j(i) < h_{\min}(j)$ then
             $h_{\min}(j) = h_j(i)$
```

For a large set of signatures generated, it is still too costly to compare similarities for all signatures pairs. Therefore, a band is used to filter dissimilar pairs, the random permutations of the matrix (column and row) can be simulated by the use of N randomly chosen hash functions.

Thus, if there is a band from each of the two tweets map to the same bucket, this indicates a candidate pairs. Each band defines a hash function $h : \Re^{r} = \aleph$ and takes a column vector of length r map to integer, the signatures are divided into bands $(B = 100)$, each band consists of rows *r*= 50/bands. However, there are sets of buckets $(b = 5)$ to map with one set of each band to prevent overlapping between bands.

Also, the probability that a random permutation of two feature vectors produce the similar minHash values which approximates the Jaccard coefficient as similarity measures between clusters, this determines the probability that two tweets (S_1, S_2) as $\Pr(\min h(S_1) = \min h(S_2)$ presented in equation 3:

$$Sim_J = \left| \frac{S_1 \cap S_2}{S_1 \cup S_2} \right| \tag{3}$$

where $Sim_J(S_1, S_2) \in [0,1]$ is a similarity function.

All the data points in the hash buckets are adopted as candidate's pair, which is used to calculate the similarity with the tweets to find the appropriate nearest neighbours (ANN). In ANN, a set of N tweets is represented as points in a distance measure, it would effectively find the point p closest to each tweet in the dataset.

Then, computes the cosine similarity (as in equation 4) between the tweet and its k-ANN which is greater than a given threshold (T=0.75) for *N* vectors to filter/eliminate false positive and false negative on candidate pairs in each bucket. However, a bucket with one (1) tweet will be discarded.

$$\cos(\vec{S}_1, \vec{S}_2) = \frac{\vec{S}_1 \bullet \vec{S}_2}{\left|\vec{S}_1\right|\left|\vec{S}_2\right|} \tag{4}$$

$$= \frac{\sum_{i=1}^{N} tfidf\ (W, \vec{S}_1) \bullet tfidf\ (W, \vec{S}_2)}{\left|\sqrt{\sum_{i=1}^{N} tfidf_i^{\,2}(W, \vec{S}_1)}\right|\left|\sqrt{\sum_{i=1}^{N} tfidf_i^{\,2}(W, \vec{S}_2)}\right|}$$

- **BLSH with Incremental Clustering for Event Detection**

This section describes the incremental clustering used to detect trending/event topics in tweets (Proper Noun) posted in a certain length of time using incremental DBSCAN and N-grams. Clustering involves dividing a set of N data point into several non-overlapping homogenous groups and each cluster contains similar data items partition into x clusters $C_1 \ldots C_x$.

As stated above, assumes that the tweets in the same cluster describe the same event, hence, the similarity comparison is a major performance bottlenecks with large scale data clustering and poor performance in terms of efficiency. As new data comes, non-incremental clustering re-clusters all the data which decreases the efficiency and wastes computing resources. With BLSH incremental clustering, it means the percentage of changes (% of δ) in the original dataset takes into consideration the clustering accuracy. Insertion of some new data items into the already existing clusters can be defined as:

$$\% \ \delta \text{ (change in dataset)} = \frac{(\text{new data - old data})}{\text{old data}} \times 100$$

- **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**

DBSCAN predicts fixed density clusters by grouping data point in high-density regions of the feature space and handles noisy data or outlier property. DBSCAN clustering finds the area where the density exceeds the threshold and discovers new cluster of arbitrary shape depending on the radius (ε) and minimum number of point (Minpt) contained in the neighbourhood. This lowers clustering quality for datasets with various densities. Finding cluster in DBSCAN involves an arbitrary object p in D and retrieves all points ($\varepsilon -$ neighbourhood of core point) which are density reachable from p with respect to ε and Minpt.

It adds ε-neighbourhood of core point with the current cluster until no new object point can be added to the cluster. Using the BLSH to query the k-ANN of each point improve the speed of region query as well as the clustering quality. DBSCAN does not require knowing the number of clusters in the data.

Algorithm for BLSH with Incremental Clustering
Input: Live tweets (datasets), Signature length k, Time rate t , Similarity Threshold T, buckets b, point p
Output: Event topic and its tweets

```
Step1: /* BLSH Procedure */
        preprocess tweets
        build tf-idf  feature vector  FV for t in tweet N
        for each incoming tweet do
            create k-bit signature S for FV
            for each bucket $i \in b$ do
                get collision for S
                add FV with key S in $h_i(p)$
            end for
        for each bucket $i \in b$ do
        create hash table $h_i(p)$
        create random permutation vector
        end for
        repeat
            get appropriate nearest neighbor for FV from collisions
            if similarity(FV, ANN) < threshold (T)  then
                create new cluster C
                addTweetVectorToCluster(FV, $C_{ANN}$)
            else
                if FV not in ANN cluster $C_{ANN}$ then
                    addTweetVectorToCluster(FV $C_{ANN}$)
                end if
             end if
          end for
        until connection end

Step 2 : /*Filtering and elimination of dissimilar tweets*/
        Repeat
        for each bucket $i \in b$
            delete cluster c with one length <= 1 from hash Table
            for each candidates pairs in $i \in b$
            compute  cosine similarity for each tweet  in i
            delete tweets < threshold (T)
Step 3 : /*Incremental Clustering Procedure */
        Repeat
            while not end of the tweets do
                Take a tweet from the tweets
                 Validate a point p with hash table
                 if p is approximate core object then
                     Take the approximate core point p as the centre
                     Find all approximate Nearest Neighbour points and group them into a cluster
                 For all appropriate core points in N
                 if TimeRate(c) > T then
                     get all tweets in cluster c
                     select an event with highest frequency
                     display event name and its tweets
                 end if
          end
```

4. PERFORMANCE ON EVENTS DETECTION USING BLSH AND DBSCAN

For the experimental analysis, the system configuration used consists of MacOs High Sirra 10.13.4, 2.8GHz Intel Core i7, 16 GB Ram and 320 SSD. We examined different levels of incremental tweets randomly (1000, 1500, 200, 3000, 5000) and show the influence of various parameters such as similarity threshold (T=0.75) in finding approximate nearest neighbour (ANN), number of hash tables (h=5), Number of Bands= 100 with 50 rows and number of permutations (*k*=64, 128, 256 and 512) in detecting events/topics in real time.

Table 1: Data Analysis Generated for BLSH and DBSCAN with Parameters: Band=100, Row=50, Permutation=64 Bucket=5 and Similarity Threshold=0.75

No of tweet	Time of CPU (Uni) minute	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times		Time of CPU (Bi) mins	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times		Time of CPU (Tri)	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times	
		1	2	1	2		1	2	1	2		1	2	1	2
1000	3.32	Feb 17	Win 17	Feb 4	Infosec 4	3.30	Feb 35	Time 23	Feb 4	Climate 2	3.29	Feb 41	Time 33	Infosec 4	-
1500	5.11	Trump 19	Feb 15	Feb 5	Win 4	5.57	Feb 62	Trump 50	Feb 9	World 7	5.71	Feb 36	Trump 23	Feb 7	Infosec 4
2000	6,92	Feb 60	Lincoln 40	Feb 7	Win 6	6.34	Lincoln 40	Awash 39	Feb 7	Join 7	10.76	Feb 82	Trump 65	Feb 9	Win 8
3000	9.94	Trump 42	Time 31	Join 9	-	10.46	Feb 64	Trump 56	Feb 13	Time 10	10.42	Feb 122	Trump 99	Feb 14	Time 10
5000	24.4	Feb 110	Trump 103	Feb 31	Time 21	20.93	Trump 56	Feb 49	Feb 25	Time 21	15.30	Trump 78	Feb 77	Feb 30	Time 22

Table 2: Data Analysis Generated for BLSH and DBSCAN with Parameters: Band=100, Row=50, Permutation=128 Bucket=5 and Similarity Threshold=0.75

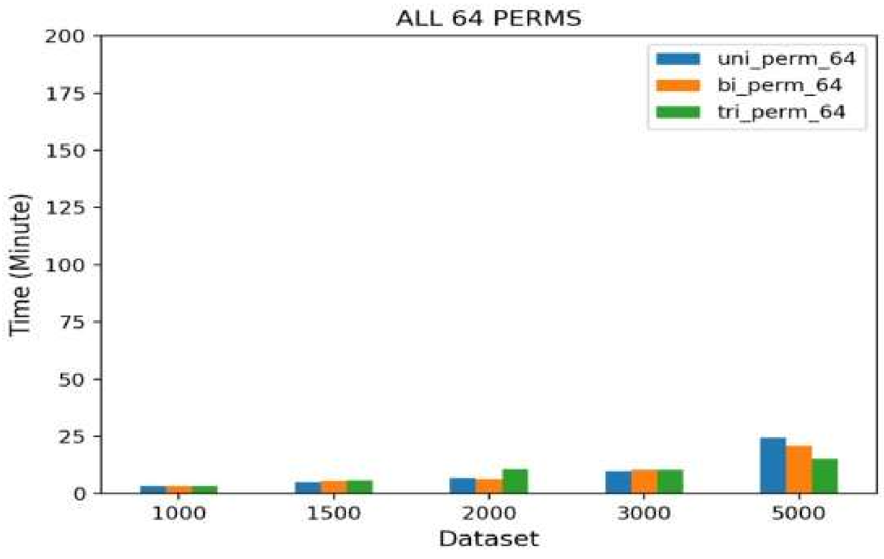
No of tweet	Time of CPU (Uni) mins	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times		Time of CPU (Bi) mins	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times		Time of CPU (Tri) min	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times	
		1	2	1	2		1	2	1	2		1	2	1	2
1000	5.67	Time 28	Feb 26	Year 3	become 2	5.49	Feb 29	Time 16	Feb 4	Climate 2	6.51	Feb 22	Trump 21	Climate 3	Feb 3
1500	8.70	Feb 56	Time 48	Feb 8	Join 5	9.11	Feb 48	Trump 35	Feb 8	World 5	9.62	Feb 64	Trump 51	Feb 7	World 6
2000	11.61	Feb 56	Time 43	Win 8	Feb 7	10.77	Feb 71	Trump 64	Feb 8	Join 7	17.37	Feb 61	Trump 46	Feb 8	Win 7
3000	17.50	Feb 69	Trump 55	Time !!	Join 10	17.42	Trump 58	Feb 57	Feb 11	Time 10	17.80	Feb 108	Trump 90	Feb 12	-
5000	34.61	Win 54	Awash 53	Feb 31	Time 21	30.37	Trump 109	Feb 111	Feb 32	Time 21	27.12	Trump 180	Feb 187	Feb 38	Time 24

Table 3: Data Analysis Generated for BLSH and DBSCAN with Parameters: Band=100, Row=50, Permutation=256 Bucket=5 and Similarity Threshold=0.75

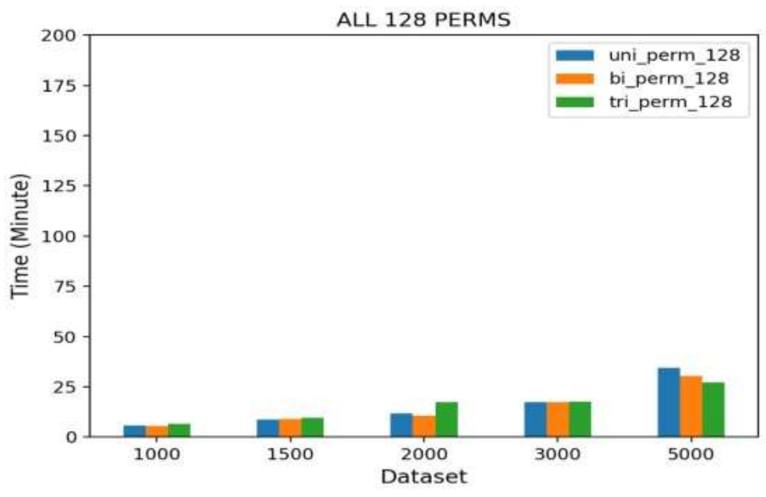
No of tweet	Time of CPU (Uni) min	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times		Time of CPU (Bi) min	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times		Time of CPU (Tri)	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times	
		1	2	1	2		1	2	1	2		1	2	1	2
1000	10.66	Feb 46	Time 35	Feb 6	Climate 3	10.85	Resa 22	Feb 17	Feb 6	Climate 3	10.24	Feb 46	Time 34	World 3	Anti 2
1500	17.79	Feb 54	Trump 46	Feb 7	World 7	16.17	Feb 62	Time 46	Feb 8	World 7	17.71	Feb 73	Trump 51	Feb 9	Win 5
2000	20,44	Feb 87	Trump 69	Feb 9	Time 9	18.98	Feb 74	Trump 69	Join 8	World 8	25.71	Feb 85	Trump 51	Feb 15	Time 8
3000	31.89	Feb 44	Trump 48	Time 11	Win 10	31.59	Feb 111	Trump 81	Feb 13	Win 12	33.08	Feb 136	Trump 112	Feb 15	Time 13
5000	169.75	Feb 114	Trump 79	Feb 30	Time 18	55.37	Feb 147	Trump 144	Feb 36	Time 20	48.26	Feb 145	Trump 167	Feb 35	Time 18

Table 4: Data Analysis Generated for BLSH and DBSCAN with Parameters: Band=100, Row=50, Permutation=512 Bucket=5 and Similarity Threshold=0.75

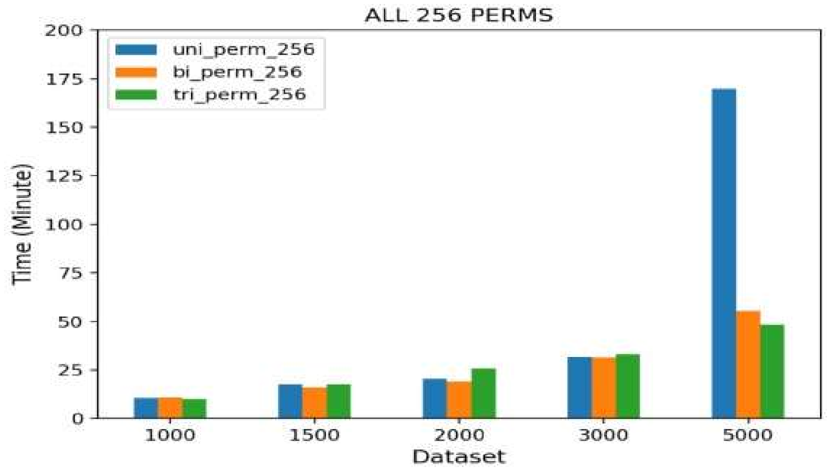
No of tweet	Time of CPU (Uni) min	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times		Time of CPU (Bi) sec	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times		Time of CPU (Tri)	BLSH Event Topic & No of Times		DBSCAN Event Topic & No of Times	
		1	2	1	2		1	2	1	2		1	2	1	2
1000	17.72	Feb 46	Time 35	Feb 5	Climate 3	18.81	Resa 22	Join 19	Feb 3	Anti 2	19.62	Feb 46	Time 35	Dewey 3	Feb 3
1500	29.87	Feb 52	Time 43	Feb 7	Win 6	31.60	Feb 25	Time 18	Feb 6	Climate 3	36.28	Feb 64	Trump 51	World 6	Feb 7
2000	40.36	Feb 74	Trump 64	Time 9	Feb 7	36.10	Feb 74	Trump 69	Join 7	Time 8	52.77	Feb 54	Trump 55	Win 10	Time 8
3000	68.06	Feb 122	Trump 113	Feb 14	Time 14	64.78	Feb 83	Trump 77	Win 11	Feb 10	70.34	Feb 91	Trump 82	Nt 11	Time 11
5000	101.16	Feb 145	Trump 115	Feb 32	Time 20	105.85	Feb 178	Trump 172	Feb 36	Time 23	103.82	Feb 144	Trump 118	Feb 30	Time 17



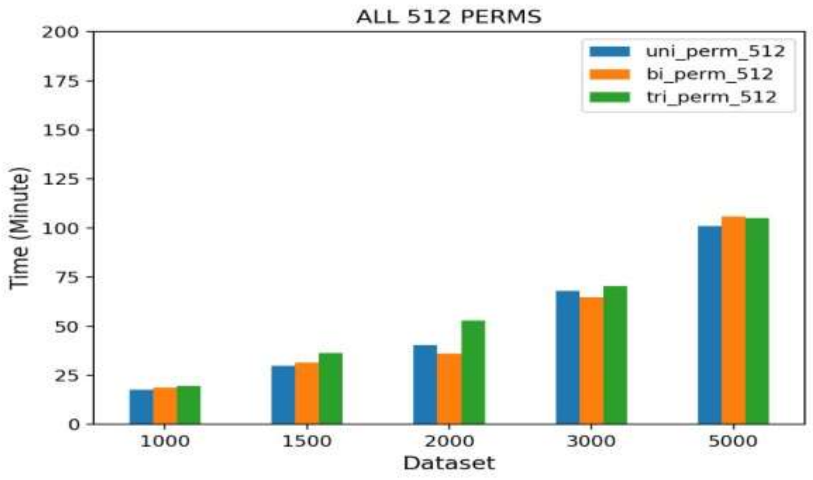
(a)



(b)



(c)



(d)

Figure 3a-d: Shows time taken and effect of N-grams

Finally, comparison on the average computation times to find a nearest neighbour from buckets is determined on Table 1-4 and figure 3a-d show the illustration of the performance of BLSH approach in detecting different events in uni-gram, bi-gram and tri-gram as well as DBSCAN on each permutation. However, DBSCAN reduced the number of events/topic generated compared to BLSH only in N-gram because it re-clusters the tweets. But as the the number of tweets increases, computational time and the permutation increase, but the bigram is better compared to unigram and trigram.

5. CONCLUSION AND RECOMMENDATION

It was found that it is difficult to determine the best values for the BLSH parameters. We explored a number of solutions to find an appropriate set of hashing functions that fit the constraints of not colliding dissimilar items into the same bucket. Various experiments were conducted in this research work to compute the best values for the BLSH parameters. In this work, DBSCAN clustering improves the efficiency for large datasets by integration of BLSH as a cluster search space reduction. Hence, reduced error bound in terms of cluster quality compared to using BLSH. In future work, in order to get better result a Graphical Processing Unit (GPU) is preferred to increase CPU time and the number of dataset. Also, the method can be extend to other clustering techniques such as K-means and as well using word embedding as vector representation and improve semantic level.

REFERENCES

1. Lau, D.J. H., Collier, N. and Baldwin, T. 2012. On-line trend analysis with topic models: Twitter trends detection topic model online. Proceedings of the International Conference on Computational Linguistics, COLING, pp. 1519–1534.
2. Hasan, M., Orgun, M. A and Schwitter, R. 2017. A survey on real-time event detection from the Twitter data stream. Journal of Information Science, 0165551517698564.
3. Castillo, C., Mendoza, M., and Poblete, B. 2011. Information credibility on Twitter. In Proceedings of the 20th International Conference on World Wide Web, WWW '11, ACM, New York, NY, pp. 675–684.
4. Kaplan, A. M. and M. HAENLEIN. 2011. The early bird catches the news: Nine things you should know about micro-blogging. Business Horizons, 54(2): 105–113.
5. Allan, V. Lavrenko, and H. Jin. 2000. First story detection in TDT is hard. In Proc. of CIKM,.
6. Marcus, A., Bernstein, M. S., Badar, O. , Karger, D. R., Madden, S. and Miller, R. C. 2011 "TwitInfo: Aggregating and visualizing microblogs for event exploration," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '11. New York, NY, USA: ACM, , pp. 227–236.
7. Philbin, J. , Chum, O. , Isard, M. , Sivic, J. and Zisserman, A. 2007.. "Object retrieval with large vocabularies and fast spatial matching;' Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition
8. Lv, Q., Josephson, W. , Wang, Z. , Charikar, M. and Li. K. 2003 Integrating semantics-based access mechanisums with P2P file systems. In Proc. Of the the Third International Conference on Peerto Peer Computing (P2P), Linkping, Sweden, September pp. 1-3.
9. Bohm, C., Berchtold, S. and Keim, D. A. 2001. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Comput. Surv. 33(3), pp 322–373.
10. Nister, D. and Stewenius, H. 2006. Scalable Recognition with a Vocabulary Tree," Proc. CVPR , vol. 5.
11. Strehl, A. Ghosh, J. and Mooney, R. 2000. Impact of similarity measures on web-page clustering. In AAAI- July 2000: Workshop on Artificial Intelligence for Web Search,
12. Petrovic, M. Osborne, and V. Lavrenko. 2010. Streaming first story detection with application to Twitter. In Proc. Of NAACL
13. Indyk, P. and Motwani, R. 1998. Approximate nearest neighbours: towards removing the curse of dimensionality. In STOC,
14. Kumar. A. P., Agrawal J. and Mishra N. 2012. Analysis of Different Similarity Measure Functions and their Impacts on Shared Nearest Neighbor Clustering Approach. International Journal of Computer Applications (0975 – 8887) Volume 40(16) pp. 1-5.
15. Wang, J. Feng, J. and Li, G. 2010 Efficient trie-based string similarity joins with edit-distance constraints," Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 1219–1230.
16. Aiello, L.M., Petkos, G., Martin, C., Corney, D., Papadopoulos, S., Skraba, R., Göker, A., Kompatsiaris, I., Jaimes, A., 2013. Sensing trending topics in twitter. IEEE Trans. Multimedia 15, pp 1268–1282
17. Atefeh, F. and Khreich, W. 2015. A survey of techniques for event detection in Twitter," Computational Intelligence, Wiley Online Library. vol. 31, no. 1, pp. 132–164.
18. Li, C. Sun, A. and Datta, A. 2012. "Twevent: Segment-based event detection from tweets," in Proceedings of the ACM International Conference on Information and Knowledge Management, ser. CIKM '12. ACM, 2012, pp. 155–164.
19. Stilo, G. and Velardi, P. 2015. Efficient temporal mining of micro-blog texts and its application to event discovery. Data Mining and Knowledge Discovery, Springer.pp. 1–31.
20. Lau, J. H. Collier, N. and Baldwin, T. 201. 2On-line trend analysis with topic models: Twitter trends detection topic model online." in Proceedings of the International Conference on Computational Linguistics, COLING, pp. 1519–1534.

21. Buhler, J. 2001. Efficient large-scale sequence comparison by locality-sensitive hashing. Bioinformatics. 5(17), 419–428.
22. Paul eve, L., Jegou, H. and Amsaleg, L. 2010. Locality sensitive hashing: A comparison of hash function types and querying mechanisms," Pattern Recognition Letters, vol. 3 (1), pp. 1 348-1358.
23. Haveliwala, T. Gionis, A. and Indyk, P. 2000. Scalable Techniques for Clustering the Web. Third International Workshop on the Web andDatabases (WebDB 2000), pp. 1 29-1 34,
24. Gu, H., Xie, X. , Lv, Q. , Ruan Y. and L. Shang. E 2011. Tree: Effective and efficient event modeling for real-time online social media. In Proc. Web Intelligence and Intelligent Agent Technology, WI-IAT 2011, IEEE/WIC/ACM International Conference, 2011. 1: pp. 300–307.
25. Broder, S. Glassman, M. Manasse, G. Z. 1997. Syntactic clustering of the Web", WWW6, pp. 391-404,
26. Liu, W., Wang, J. Ji, R. Jiang, Y. and Chang, S. 2012. Supervised hashing with kernels. In CVPR, pp 2074–2081.
27. Yang, Y. Pierce T. and Carbonell, J. 1998. A study of retrospective and on-line event detection," in Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, pp. 28-36.
28. Becker, H., Naaman, M. and Gravano, L. 2011. Beyond trending topics: Real-world event identification on Twitter. in Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media, ICWSM,.
29. Ryan McConville, Xin Cao, Weiru Liu, Paul Miller. 2016. Accelerating Large Scale Centroid-based Clustering with Locality Sensitive Hashing. 32nd IEEE International Conference on Data Engineering (ICDE) pp. 649-660
30. Slaney, M., Lifshits, Y. and He J. 2012. Optimal Parameters for Locality-Sensitive Hashing Proceedings of the IEEE | 0018-9219/$31.00 _2012 IEEE Vol. 100 (9), September pp. 2604-2623