Event Detection through Lexical Chain Based Semantic Similarity Algorithm

Swati Jain1, Suraj Prakash Narayan1, Nalini Meena1, Rupesh Kumar Dewang1, Utkarsh Bhartiya1, and Varun Kumar1, Arvind Mewada1*
1Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology, Allahabad, Prayagraj-211004, India
*arvindmewada@mnnit.ac.in

Abstract. Twitter is a platform where millions of people tend to tweet about new events happening in their lives. Celebrities tweeting about the new product endorsement, Politicians tweeting about their views towards a policy or people, Natural calamities occurring are tweeted instantly. Studying this data can provide us with useful information. In this paper, we have proposed an Event detection using a lexical chain based semantic similarity algorithm, for detecting Events from Twitter streams. Lexical chains have been used to preserve the Lexical cohesion in a text. The Twitter data set was collected using “tweepy” API, then pre-processing was done, steps like tokenization, stop word removal, and stemming is carried out and stored the tweets in a text file. Then lexical chains were built using the tweets in the file. The formation of the key graph, with each node as a lexical chain, was carried out. Then the clustering algorithm ‘SCAN’ was used to cluster the lexical chains, the formed clusters represent the keywords of an event. The last summarization step was carried out for each cluster representing an event.

1. Introduction
Twitter is one of the most popular social networking sites currently, and over 500 million tweets are written by users every day and has more than 330 million monthly active users. All this data says a lot about twitter’s popularity, and it generates a huge amount of data frequently. People on Twitter express their opinions, events surrounding them, social information, emotional reactions, etc. An event on Twitter is characterized by a huge number of tweets regarding the event within a time frame while engaging a considerable no of participants. Observations show that ‘Twitter’ is often the first medium to break events such as earthquakes, disease outbreaks, suicides, etc. This paper attempts to detect events using Twitter alone. People often want to know the events happening right now or recently in the past. The question that many people ask every day. People are more interested in those events that happen locally [3, 15]. Corporations are interested in selling their products to favourable customers [4, 14]. Event detection can answer these questions. Therefore, our project aims to do event detection using the tweets collected from ‘tweepy’ a python Twitter API. We also attempt to preserve the lexical cohesion among the words in a text, using the concept of lexical chains which are nothing but a list of words from a text connected by cohesion or relatedness.

2. RELATED WORK
Event detection has attracted significant research in information retrieval, machine learning, and social media modelling [6]. Along a similar concept of lexical chains work by Rupesh Kumar Dewang, Anil Kumar Singh [1]. International Journal of Engineering and Technology (IJET). It was related to spam review detection through lexical chain based semantic similarity algorithm for negative reviews. They preserved the semantic relation of the words in a text by using the concept of lexical chains and using them as features instead of keywords. The feature vector consisting of lexical chains and tweets fed into supervised learning algorithms gave an extremely high accuracy of 99.5% in SVM.
S. Sangeetha, R.S. Thakur, Michael Arock [2] used lexical chains to detect events from text documents. Their architecture followed three steps namely natural language pre-processing, lexical chain construction, and event detection. They also tried to preserve the lexical cohesion in the text.

Mario Jarmasz, Stan Szpakowicz [5] used Roget’s Thesaurus to automate the construction of lexical chains. Paula Hatch, Nicola Stokes, Joe Carthy [7] work on topic detection using lexical chaining. They used lexical chaining to improve the overall performance by allowing them to exploit the context of a word and to disambiguate its senses.

3. Formatting the text
In this section, we give a detailed explanation of the methodology used.

A. Dataset
The Twitter data set was collected using ‘tweepy’ a python API for Twitter, live streaming of tweets was done for data collection.

B. Pre-processing
It was carried out using the python NLTK toolkit. Steps like tokenization, stop word removal, and stemming was carried out and the resultant tweets were stored in a text file.
1. Stop words: They are commonly used words.
   Examples: ‘a’, ‘an’, ‘the’, ‘or’, ‘but’, ‘how’ etc.
2. Stemming: Stemming is a process of reducing words to their base or stem.
   Example: Stemming operation on words ‘cats’, ‘catlike’, and ‘catty’ reduce them to ‘cat’.
3. Tokenization: means chopping into pieces. Tokenizing a sentence means separating words from it while in a word it means separating characters of the word.

C. Building lexical chains
Often, lexical cohesion occurs not simply between pairs of words but over a succession of several nearby related words that can span the entire text or a part of it. These sequences of related words are called lexical chains. Lexical chains may not end at sentence boundaries. They can connect a pair of adjacent words or range over an entire text. Consider an example highlighting the five parameters to determine lexical cohesion:
Examples [8]
1. Reiteration with the identity of reference.
   (a) Lara bit into a mango.
   (b) Unfortunately, the mango was not ripe enough.
2. Reiteration without the identity of reference.
   a. Lara ate some mangoes.
   b. She likes mangoes very much.
3. Reiteration through superordinate.
   a. Lara ate a mango.
   b. She likes fruit.
4. Systematic semantic relation
   a. Lara likes green bananas.
   b. She does not like yellow ones.
5. Non-systematic semantic relation
   a. Lara spent three hours in the park today.
   b. She was plucking flowers.

The initial three relations are the simple repetition of the same word in the same sense (e.g., peach and peach), (e.g., peach and fruit). The last two relations include collocations i.e., semantic connections between words that regularly co-occur (e.g., green and red), (e.g., peach and fruit). We have used the ELKB (Electronic Lexical Chain Base) tool for the generation of lexical chains from the text file. The ELKB is based on Roget's Thesaurus. We have applied five types of thesaurus relations that suggest the inclusion of a candidate word in a lexical chain. The five thesaurus relations used are:
1. Inclusion in the same Head.
2. Inclusion in two different Heads linked by a Cross-reference.
3. Inclusion in References to the same Index Entry.
4. Inclusion in the same Head Group.
5. Inclusion in two different Heads linked to a Common third head by a Cross-reference.

The most important and widely used relations are relations 1 and 2. See Table 1.

A lexical chain is a sequence of related words in writing, spanning short (adjacent words or sentences) or long distances (entire text). A lexical chain is independent of the grammatical structure of the text and in effect, it is a list of words that captures a portion of the cohesive structure of the text. The main motive to use the lexical chain was to exploit the semantic relatedness of the words in any sensible text. Unlike the traditional methods like the bag-of-words model or the tf-idf model that work on keywords in a text disregarding the cohesiveness of words in it.

---

**Algorithm-1 Construction of lexical chains**

**Input:** Keywords (TW)  
**Output:** set S of lexical chains

1. \( TW = \{TW_1, TW_2, \ldots, TW_n\} \) and \( S \leftarrow NULL \)
2. \textbf{For} \( t \) in \( TW \), \textbf{do}:
3. \( K \leftarrow \text{keywords} (TW) \)
4. \textbf{For} \( k \) in \( K \)
5. \textbf{If} \( k \) has a repetition of the same word relation or inclusion of the same paragraph
6. \( LC_i \leftarrow LC_i + k \)
7. \textbf{Else If} No chain \( (LC_i) \) is constructed
8. \textbf{then} Create a new chain \((LC_n)\) and \( LC_n \leftarrow LC_n + k \) and \( S \leftarrow S + LC_n \)
9. \textbf{Else} go to step 5
10. \textbf{End If}
11. \textbf{End If}
12. \textbf{End For}
13. \textbf{End For}

---

**D. Formation of the feature matrix**

For constructing the feature matrix, we used TF-Itf (Term frequency-inverse tweet frequency). If we have ‘M’ lexical chains and ‘N’ tweets, the feature matrix will be of N×M dimensions. For \( LC_i \) i.e., \( i^{th} \) lexical chain and \( j^{th} \) tweet, we need to calculate TF-Itf and put it in corresponding \( j, i \) cell of the matrix. After filling all the cells, we get the feature matrix. See figure 4.

1. **The TF-Itf model for the formation of the feature matrix**

Term frequency (TF) is calculated for tweets. Hence, defined the name of the algorithm is Term Frequency-Inverse Tweet Frequency (TF-Itf). The Classical tf-idf Concept is modified and is based on the lexical chain. The total number of tweets is ‘t’ and the total number of lexical chains generated is ‘c’. This generated feature vector matrix is of dimension \( t \times c \).

a.) Term frequency (TF)
The graph below represents a semantic score of lexical chains constructed using ELKB. Chains with lower scores were removed as they were weekly semantically related. score(chain) = length X homogeneity.

Length = no. of all members in the chain. Homogeneity = 1 – (distinct members)/length.x

In TF-ItF, it measures how frequently the words of a lexical chain appears in a tweet. Let us consider a lexical chain LCj and a tweet Ti where 0<i< (number of tweets) and 0<j< (number of lexical chains) formed. The frequency for LCj and Ti is given by the following formula:

$$TF = \log(1 + \text{sum}(i,j)) \quad (1)$$

where sum (i, j) is the sum of the frequency of each word of LCi in a tweet Tj.

b.) Inverse Tweet Frequency (ItF)

The ItF is calculated by the following formula: $Itf = \frac{1}{\log(1+t)} \quad (2)$

Where ‘t’ is the total number of tweets.

Example:

“Zuckerberg senior Facebook execute repeatedly congress parliament Facebook advertise want to happen Dr Dena Grayson Facebook Zuckerberg senior Facebook execute repeatedly congress parliament content clear UK parliament host intern parliament.”

LC1 = [parliament, parliament, parliament, parliament, congress, congress, congress]

Lexical chain LC1 each word count in the tweet example:

('parliament',4), ('parliament',4), ('parliament',4), ('parliament',4), ('congress', 2), ('congress', 2), ('congress', 2)

Total no of count = 22

TF-tweet= log10(1+count)

TF-tweet = 1.36

Itf (no. of tweets in which lexical chain appears) =1 Itf = 1/log10(1+ tf), ItF = 1/log10(1+1) = 3.32 TF-ItF = 1.36 x 3.32 = 4.51

Now say that in the feature matrix this Lexical chain appears on the jth column and this tweet appears in the ith row, then the value to be put in the i, j cell of the matrix = 2.78
E. Formation of Key graph
We calculated the semantic scores of each lexical chain and removed the ones that had scores below a threshold. The rest of the lexical chains are assumed to be the nodes of the graph. We created a graph variable which is a dictionary with key-list pair values. So, nodes LC\textsubscript{i} as the key will have nodes LC\textsubscript{j} in the list on value only if both lexical chains have TF-\textit{ItF} scores in any one of the tweets is greater than 0.001. Now edge list is formed for the first graph where an edge is formed between two lexical chains only if that lexical chain pairs appear in the same tweet.

F. Formation of clusters
After forming the key graph, where each node is a lexical chain in the graph, we ran a clustering algorithm on the graph to cluster the same nodes together, to identify distinct events in the form of separate clusters. We used SCAN (Structural Clustering Algorithm for networks) for clustering the nodes. SCAN calculates the Structural similarity of surrounding nodes with the current node and uses a threshold ‘\(\varepsilon\)’ to either include it in the cluster or not. This is how the SCAN algorithm works on a high level, but there is more complexity to it which is out of the scope of this paper, for finding out the clusters of lexical chains from the key graph based on direct connectivity of lexical chains. Therefore, at the end of this phase, we have clusters of lexical chains that consist of keywords that can together represent the keywords of an event.

G. Summarization of Events
Each cluster of lexical chains may be considered as a synthetic document. Let us call it a key document for our purposes, now it can be used to produce a summarization of events. Documents in the original corpus (tweets) which are like this key document can be clustered, thus retrieving a cluster of topical documents. We used cosine similarity between two documents to discover document clusters for key documents. After clustering of original documents, we have applied a summarization algorithm in which each cluster of documents will be given as input and produces the summary of each cluster as output. Hence, the results will be displayed as a summary of each cluster of tweets where the summary represents the representative tweet that belonged to that cluster. See Figure 5.

The following key graph was obtained.

Figure 2. This is a PCA plot and the axes shown here are two principle components. This is the plot of a key graph. Here LC\textsubscript{i} represents \(i\)th lexical chain. Two chains have an edge between them only if they both appear in the same tweet.
Figure 3. This is a PCA plot and the axes shown here are two Principal Components. This plot is the result of the SCAN algorithm on the graph from figure 2. LC\textsubscript{i} represents \textit{i}th Cluster of lexical chains. These Clusters represent events.

|   | LC1 | LC2 | LC3 | LC4 | LC5 | LC6 | LC7 | LC8 | LC9 | LC10 |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|
| T1| 0.0 | 0.0 | 0.21| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0  |
| T2| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0  |
| T3| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0  |
| T4| 0.0 | 0.0 | 0.21| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0  |
| T5| 0.08| 0.217| 0.307| 0.0 | 0.42| 0.42| 0.0 | 0.0 | 0.0 | 0.0  |
| T6| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0  |
| T7| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0  |
| T8| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0  |
| T9| 0.0 | 0.517| 0.307| 0.0 | 0.42| 0.42| 0.0 | 0.0 | 0.0 | 0.0  |
| T10| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0   |

Figure 4. The following matrix is the subset of the feature vector matrix formed from lexical chains (LC) and tweets (T). The i,j cell represents the TF-Itf score of i\textsuperscript{th} tweet in j\textsuperscript{th} lexical chain.

Figure 5. Summarization was obtained on pre-processed tweets. In summarization basically, the most representative tweets were selected from each group of tweets that formed as a result of measuring cosine similarity between the tweets and the synthetic document. The synthetic document is nothing, but the clusters obtained, see (Figure 3).
4. RESULTS
The following screenshot shows the summary obtained from a cluster. The clusters formed can be termed as a synthetic document, now each cluster formed is measured against each tweet for similarity, i.e., we find out cosine similarity of each synthetic document and tweet, we do it for every tweet and synthetic document and the tweets having the highest similarity with some synthetic documents are group into one and termed as an event, as all the tweets in that group represent an event. The summarization is obtained by selecting the most representative tweet from the groups formed.

| Table 1. The below table shows the lexical chains Generated from the pre-processed tweet. |
|---------------------------------------------------------------|
| LC1 | parliament | parliament | parliament | parliament | Congress | Congress | Congress |
| LC2 | intern     | intern     | Intern     | senior     | senior   | intern   | rank     |
| LC3 | shop       | shop       | Break      | trump      | break    | trump    | shop     |
| LC4 | offer      | offer      | Pleas      | Cover      | Pleas    | Cover    | Pleas    |
| LC5 | Document   | document   | Support    | Weight     | Speak    | Support  | Weight   |
| LC6 | twitter    | twitter    | Twitter    | strong     | strong   | strong   | Twitter  |
| LC7 | Person     | Person     | Watch      | Danger     | Person   | Person   | Watch    |
| LC8 | Power      | Chief      | Power      | Chief      | Power    | Biggest  | Biggest  |
| LC9 | progress   | progress   | progress   | progress   | Profit   | Upset    | Profit   |
| LC10 | Insight   | Insight    | insight    | insight    | insight  | insight  | insight  |
| LC11 | Matter     | Matter     | Import     | Import     | Import   | Import   | Import   |
| LC12 | Gentler    | Fine       | Gentler    | Fine       | Easiest  | Easiest  | Fine     |
| LC13 | Share      | Share      | Share      | share      | Leader   | Leader   | Parallel |
| LC14 | Social     | Social     | Social     | Social     |         |         |         |
| LC15 | hottest    | hottest    | Hottest    | hottest    |         |         |         |
| LC16 | bigot      | bigot      | Bigot      | bigot      |         |         |         |
| LC17 | vector     | vector     | Vector     | vector     |         |         |         |
| LC18 | traffic    | system     | Traffic    | traffic    |         |         |         |
| LC19 | elect      | elect      | User       | user       |         |         |         |
| LC20 | attack     | attack     | Blast      | blast      |         |         |         |
| LC21 | host       | host       | Veteran    | veteran    |         |         |         |
| LC22 | club       | start      | Club       | start      |         |         |         |
| LC23 | annoy      | annoy      | Annoy      |           |         |         |         |
| LC24 | album      | album      | Review     |           |         |         |         |
| LC25 | Worth      | worth      |           |           |         |         |         |

The lexical cohesion can easily be seen among the formed chains:

For Example:

LC1 [parliament → parliament (Repetition)]; [congress → congress (Repetition)]
LC21 [attack → attack (Repetition)]; [attack → blast (Collocation)]
5. Conclusion
We used Lexical chains as a replacement to keywords used in previous works, as it takes care of semantic relations among words i.e., lexical cohesion among the words in a tweet. Also, it gives better features for a tweet than a bag of words model or tf-idf model (term frequency-inverse document frequency) which used tf-idf weights for keywords, we have modified it to suit our work and used tf-itf (term frequency-inverse tweet frequency) for weighing lexical chains. Here, we have worked upon a more optimal graph clustering algorithm for clustering of events when compared to the traditional event detection algorithms i.e., event detection using the co-occurrence of keywords.

Apart from the SCAN algorithm used for clustering, we used a lexical chain based semantic similarity algorithm for performing clustering in initial stages i.e., building lexical chains is also a kind of clustering based on semantic similarity of the words. In this algorithm, clustering is performed in such a way that all the keywords in all the tweets which are semantically related to each other fall into one chain. Here nodes of the graph are represented by lexical chains instead of keywords as in previous works done on event detection. Therefore, we can gain insight into events or topics that are extremely popular among their customers and hence, able to generate additional profits by doing event detection more efficiently by the cohesive structure of the text.

Hence, we conclude that by using lexical chains as features instead of keywords we were able to detect events more efficiently while taking care of the semantic relatedness of among the words in tweets. It was not possible with the keywords approach as it does not exploit the various forms of cohesion among the word in a meaningful text.

6. References
[1]. Rupesh Kumar Dewang et al. *International Journal of Engineering and Technology (IJET)*
[2]. S. S., Thakur R.S., Arock M. (2010) *Event Detection Using Lexical Chain*. In Loftsson H., Rögnvaldsson E., Helgadóttir S. (eds) Advances in Natural Language Processing. NLP 2010. Lecture Notes in Computer Science, vol 6233. Springer, Berlin, Heidelberg
[3]. Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, *Who, where, when and what: discover Spatio-temporal topics for Twitter users*, in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013, pp. 605-613.
[4]. A. Farzindar, *Industrial perspectives on social networks*, in EACL 2012 Workshop on Semantic Analysis in Social Media, vol. 2, 2012.
[5]. Jarmasz M., Szpakowicz S. (2003) *Not as Easy as It Seems: Automating the Construction of Lexical Chains Using Roget’s Thesaurus*. In: Xiang Y., Chaib-Draa B. (eds) Advances in Artificial Intelligence. AI 2003. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 2671. Springer, Berlin, Heidelberg.
[6]. *Event Detection and Tracking in Social Streams* by Hassan Sayyadi, Mathew Hurst, and Alexey Maykov (Proceedings of the Third International ICWSM Conference 2009).
[7]. Paula Hatch, Nicola Stokes, Joe Carthy, Nicola Stokes, Paula Hatch, Joe Carthy, *Topic Detection, a new application for lexical chaining?* In the Proceedings of BCS IRSG Colloquium 2000, pp. 94-103, 2000.
[8]. https://www.slideshare.net/uspansayuti/lexical-cohesion-28663231
[9]. Ellison, Nicole B. *Social network sites: Definition, history, and scholarship*. Journal of Computer-Mediated Communication 13.1 (2007): pp.210-230.
[10]. Kwak, H., Lee, C., Park, H., Moon, S.: *What is Twitter, a social network, or a news media? Categories and subject descriptors*. Most 112(2), 591-600 (2010).
[11]. A. Java, X. Song, T. Finin, and B. Tseng, *why we twitter understanding microblogging usage and communities*, in Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis. ACM, 2007, pp. 56-65.
[12]. X. Wang, M. S. Gerber, and D. E. Brown, *Automatic crime prediction using events extracted from Twitter posts*, in Social Computing, Behavioural-Cultural Modelling, and Prediction. Springer, 2012, pp. 231-238.
[13]. NALLAPATI, R., A. FENG, F. PENG, and J. ALLAN. 2004. Event threading within news topics. In Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM04, ACM, New York, NY, pp. 446-453.

[14]. Dewang, Rupesh Kumar, and Anil Kumar Singh. "State-of-art approaches for review spammer detection: a survey." Journal of Intelligent Information Systems 50.2 (2018): 231-264.

[15]. Sabeeh, Ahmed, and Rupesh Kumar Dewang. "Comparison, classification and survey of aspect-based sentiment analysis." International Conference on Advanced Informatics for Computing Research. Springer, Singapore, 2018.