




ARTICLE

Identification and summarisation of events from Twitter using clustering algorithms and deep neural network

Kunal Chakma , Anupam Jamatia  and Dwijen Rudrapal 

Computer Science and Engineering Department, National Institute of Technology, Agartala, Tripura, India

Corresponding author: Anupam Jamatia; Email: anupamjamatia@gmail.com

(Received 25 January 2023; revised 11 June 2024; accepted 12 June 2024)

Abstract

The proliferation of social networks has caused an increase in the amount of textual content generated by users. The voluminous nature of such content poses a challenge to users, necessitating the development of technological solutions for automatic summarisation. This paper presents a two-stage framework for generating abstractive summaries from a collection of Twitter texts. In the first stage of the framework, event detection is carried out through clustering, followed by event summarisation in the second stage. Our approach involves generating contextualised vector representations of tweets and applying various clustering techniques to the vectors. The quality of the resulting clusters is evaluated, and the best clusters are selected for the summarisation task based on this evaluation. In contrast to previous studies, we experimented with various clustering techniques as a preprocessing step to obtain better event representations. For the summarisation task, we utilised pre-trained models of three state-of-the-art deep neural network architectures and evaluated their performance on abstractive summarisation of the event clusters. Summaries are generated from clusters that contain (a) *unranked tweets*, (b) *all ranked tweets*, and (c) *the top 10 ranked tweets*. Of these three sets of clusters, we obtained the best ROUGE scores from the *top 10 ranked tweets*. From the summaries generated from the clusters containing the top ten tweets, we obtained ROUGE-1 F score of 48%, ROUGE-2 F score of 37%, ROUGE-L F score of 44%, and ROUGE-SU F score of 33% which suggests that if relevant tweets are at the top of a cluster, and then better summaries are generated.

Keywords: Abstractive summarisation; Twitter; BERT; transformer; pointer-generator

1. Introduction

The task of gathering information by reading large volumes of documents can be a challenging endeavour for humans. A document summary allows the reader to quickly and accurately identify the fundamental components of a document.^a This, in turn, enables readers to determine the document's relevance and interest and make an informed decision on whether to read the document in its entirety. As noted in Manuel and Moreno (2014), creating a summary involves two essential elements: comprehending the source text and producing a concise and brief version of it. These elements necessitate extralinguistic skills and knowledge of the document on the part of the human summariser. Therefore, it is imperative to develop techniques to automatically generate summary from documents. Automatic text summarisation (ATS) algorithms and techniques are an approximation of those created by human summarisers. An ATS system is deemed effective when the summaries generated are similar to those produced by humans in terms of content,

^a<https://www.sciencedirect.com/topics/computer-science/text-summarization>

form, readability, and conciseness. Despite the fact that humans create better summaries, ATS is more accessible, can be distributed more quickly, and is at a lower cost in terms of time.

Summarisation proves to be a valuable technique when dealing with documents of considerable length. However, this article aims to delve into the realm of summarisation specifically concerning Twitter^b texts. Twitter serves as a microblogging platform where users can post messages known as “tweets” that are restricted by a character limit. Initially, when Twitter was launched in 2006, the character limit was set to 140, but in late 2017, it was increased to 280 characters. The limit of 280 characters still persists even after the conversion of Twitter to X. However, the character limit has been increased to 25,000 for premium users when Twitter became X. Twitter users attempt to convey their opinions on a particular topic or report on an event by encapsulating as much information as possible within the given limit. Consequently, Twitter has enabled the rapid dissemination of news through its platform. The pressing question now arises, “*Why summarise tweets?*”. As per data from Internet Live Stats, the average number of tweets published every second is 6,000, which amounts to 350,000 per minute, 500 million per day, and approximately 200 billion per year. This extensive volume of content generated on Twitter can become overwhelming to consume. While not all users may find the voluminous content useful, organisations, for instance, involved in conducting opinion polls during a political campaign, would only require tweets relevant to their task. Going through each tweet to comprehend the opinions would not be feasible for such an organisation. Therefore, automatically generated summaries of the opinions extracted from the tweet collection would be a desirable outcome.

According to the survey conducted by Hasan *et al.* (2017), an event on Twitter can be defined as a discussion on a particular subject matter that is driven by users’ interest shortly after its occurrence or in anticipation of it. Consequently, tweets can be viewed as brief texts that are generated in response to events that take place over a specific period of time. As a result, ATS is deemed essential for generating summaries of such events on Twitter. Twitter has evolved into an extensive collection of real-time information that encompasses a variety of topics. As the volume of tweets continues to surge, the need for efficient and effective summarisation techniques becomes increasingly paramount. Abstractive summarisation of Tweets is a solution to the need for distilling the essence of multiple tweets into coherent summaries that capture the main themes and sentiments of the conversation. Unlike extractive summarisation, which simply selects and concatenates existing sentences from the source text, abstractive summarisation creates new sentences that may not be identical to the parts/sentences of the original text. This approach has great potential for improving the accessibility and usability of the vast amount of information available on Twitter. However, the journey towards achieving a robust and comprehensive abstractive summarisation of tweets is fraught with challenges. Present methodologies often encounter significant difficulties in effectively encapsulating the subtleties and intricacies that are inherent in tweets. The conciseness, informal language, use of emojis, slang, and hashtags, as well as the frequent absence of grammatical structure, all present distinctive hindrances for automated summarisation systems. Additionally, the rapid and dynamic nature of Twitter conversations, where the context and sentiment can be altered in a matter of moments with new information, requires adaptability and real-time processing capabilities, which many existing techniques lack. Due to the difficulties stated above, the ATS of a single tweet is not feasible as it is already too short. Therefore, ATS in tweets is rather a multi-document summarisation (Manuel and Moreno 2014) task, where the objective is to identify the central topic of discussion from a given collection of tweets.

Recently, neural network-based solutions proposed by researchers such as Nallapati *et al.* (2016), Cheng and Lapata (2016), Zhou *et al.* (2018) for ATS from well-formed documents containing texts from books, news articles, and web pages have shown groundbreaking results with the development of state-of-the-art solutions. However, these solutions were not designed for summerisation of tweets. Although there are studies (Section 2) on ATS from microblogs such

^bTwitter was officially renamed to “X” on July 23, 2023. For brevity, we have used “Twitter/twitter” throughout this article.

as Twitter, the majority of them concentrate on creating clusters using statistical methods such as TF-IDF or non-contextualised vectors such as Word2Vec (Mikolov *et al.* 2013). These studies typically employ either one or two clustering techniques as a preprocessing step to generate summaries, without evaluating the impact of the clustering techniques deployed for the summarisation task. In contrast, we investigate the use of contextualised vectors obtained from the 5W1H semantic segmentation of tweets. Unlike previous approaches, we also used four different clustering approaches and evaluated their influence on summary creation.

This paper reports on the adoption of a pipeline approach for ATS of tweets using pre-trained models from three state-of-the-art neural network architectures based on *Pointer-generator with coverage mechanism* (See, Liu, and Manning 2017), and two *transformer* (Vaswani *et al.* 2017)-based systems: Presumm (Liu and Lapata 2019), and BART (Lewis *et al.* 2020). The major contributions of our work are as follows:

- Creation of a Twitter corpus on three general topics: *Demonetization*, *Me too movement in India*, and *US Presidential elections of 2016*.
- Semantic segmentation of tweets based on the 5W1H concept.
- Applying different clustering techniques on semantically segmented tweets for event detection.
- Evaluation of the clusters generated as a preprocessing step for summarisation.
- Selection of the most appropriate clusters to generate summaries.
- Generate abstractive summaries from the event clusters with three neural network architecture and compare the generated summaries.

The rest of the paper is organised as follows. Section 2 discusses related work. Section 3 describes the workings of the proposed implementation. Section 4 presents our experiments. Results and analysis are discussed in Section 5. Section 6 presents the time complexity analysis of clustering and summarisation. The paper is concluded in Section 7.

2. Related works

Earlier, there have been several tweet summarisation techniques such as Lexrank (Erkan and Radev 2004), LSA (Gong and Liu 2001), Luhn (Luhn 1958), MEAD (Radev, Hovy, and McKeown 2002), SumBasic (Nenkova and Vanderwende 2005), SumDSDR (He *et al.* 2012), and COWTS (Rudra *et al.* 2015). The major limitations of these algorithms are that they consider a single statistical feature that is used to assign a score to each tweet for a summary generation. Sharifi *et al.* (2010) introduce a Phrase Reinforcement (PR) algorithm that uses word usage patterns and retweets to identify key phrases in Twitter posts, generating concise summaries based on frequency and proximity. The algorithm uses a graph-based approach, weighting phrases based on frequency and proximity to the topic phrase. The paper reported a ROUGE-1 F1 score of 0.30. Inouye and Kalita (2011) introduced Hybrid TF-IDF as a statistical measure to generate twitter summaries. The authors considered a single tweet as a document. However, when computing the term frequencies (TF), they considered the entire collection of tweets as the document. Therefore, the TF component of the TF-IDF formula uses the entire collection of tweets, while the IDF component treats each tweet as a separate document. The authors normalise the weight of a tweet by dividing the weight by a normalisation factor so that the algorithm is not biased towards longer tweets. The tweets are initially grouped into k clusters using cosine similarity. Each cluster is then summarised by selecting the most weighted tweet, determined by the Hybrid TF-IDF weighting. The authors adopted two cluster-based approaches to generate multiple summaries of tweets for a Twitter event: k-means++ (Arthur and Vassilvitskii 2007) and bisecting k-means (Zhao and Karypis 2001) clustering algorithm. The clusters contain the feature vectors of each post in a

cluster computed by using Hybrid TF-IDF weighing of words. Tweets with the maximum weights are selected as the most important ones to be included in the summary with a maximum of four tweets. Their proposed approach reported an average ROUGE-1 F1 score of 0.252. The work presented by Beverungen and Kalita (2011) normalises the tweet as proposed by Kaufmann (2010) and expands the tweet terms proposed by Perez-Tellez *et al.* (2010). Their approach further optimises the clustering process utilising the gap statistic mechanism of Tibshirani *et al.* (2001). They evaluated their approach with *analysis of variance* (ANNOVA) as a statistical significance test. The evaluation indicates that while the normalisation of posts alone does not affect the summaries, enhancing clustering and term expansions considerably enhances the quality of the summaries. Shou *et al.* (2013) first perform the clustering of tweets and then select some representative tweets from each group. Then, the arrangement of these tweets is carried out using the graph-based LexRank (Erkan and Radev 2004) algorithm. They evaluated their work with ROUGE-1 F1 score but did not specify the obtained score. Another cluster-based approach (SPUR) (Yang *et al.* 2012) proposed batch summarisation of the stream of tweets by dividing the input stream into clusters based on a time window of one hour. For the proposed work, the authors gathered 2,100 hours of Twitter message streams from June to September in 2010 that contributed a total data size of 5.9 GB. Therefore, the clusters are formed by equal-sized batches of one-hour windows, which are then compressed by replacing individual words with frequently used phrases. The frequently used phrases are then extracted from the relevant tweets of the event which are then ranked by their utility values and are finally included in the summary. The authors evaluated their proposed approach by measuring the false positive rates. “Sumblr” (Wang *et al.* 2015) is a summarisation framework that summarises tweet streams with timeline generation. Clusters are generated from the event-related stream of tweets where the clusters hold information by two data structures called tweet cluster vector (TCV) and pyramidal time frame (PTF) (Aggarwal *et al.* 2003). The k-means clustering algorithm is used to generate the initial clusters with a small number of tweets. Then the TCVs are initialised accordingly with the initial cluster statistics. Every incoming new tweet on its arrival either goes to an existing cluster or forms a new cluster based on the cosine similarity of the tweet with the centroid of the cluster. After that, the summary is generated. They evaluated their work using ROUGE-1 F1 score with different *step sizes* between 4,000 and 20,000. They obtained an average ROUGE-1 score of 0.425.

OnSes (Niu *et al.* 2016) is a pipeline framework that implements a tweet summarisation system based on a neural network. They used 2,400,591 short texts and summary pairs provided by Hu *et al.* (2015) to train the model for summarisation. Additionally, they have used 18,126 short texts in total crawled from Twitter. Word vectors are created using Word2Vec (Mikolov *et al.* 2013), and then the word vectors are clustered using the k-means algorithm. The relevance between tweets in a group is then ranked using the BM25 (Amati 2009) ranking algorithm. The top-ranked tweets within a cluster are then selected to generate the summary. They obtained ROUGE-1 of 0.2283, ROUGE-2 of 0.1067, and ROUGE-L of 0.2143. Doğan and Kaya (2019) proposed a deep learning-based event summarisation system for thousands of hotel and movie reviews, and subject hashtags collected from social networks. In this study, a semantic context is created using the Word2Vec model in the text. A 2-output negative–positive model was formed by training on a GRU (Gated Recurrent Unit) neural network. The aim is to generate summary text for the user by gathering comments under a label on Twitter. The model also generated abstract text by associating the notion of sense. The study collected comments from twitter and applied sentiment classification to determine the meaning of most comments. The classified comments are then summarised with the LSA algorithm. However, the authors did not report on the performance evaluation of the generated summaries.

He *et al.* (2020) introduce TWEETSUM, an event-oriented dataset for social summarisation. The authors present the performance of extractive summarisation methods on TWEETSUM to establish baselines and report a ROUGE-1 score of 0.44886. This dataset addresses the need for social summarisation systems to quickly grasp the core information from the vast amount

of short and noisy messages produced on social media during hot events. The dataset includes 12 real-world hot events with 44,034 tweets and 11,240 users, along with expert summaries and annotation quality evaluation. It also incorporates additional social signals such as user relations, hashtags, and user profiles, establishing a user relation network for each event.

Rajaby Faghihi *et al.* (2022) introduce CrisisLTLSum, the largest dataset of local crisis event timelines, which is crucial for tasks such as timeline extraction and abstractive summarisation in the context of emergency response using social media data. The dataset contains 1,000 crisis event timelines across four domains: wildfires, local fires, traffic, and storms. The authors use a semi-automated approach to collect data from the public Twitter stream. Initial experiments show a significant performance gap between strong baselines and human performance on these tasks. Their best model achieves 0.4705, 0.2540, and 0.3590 in ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

Bilal *et al.* (2022) introduce Microblog Opinion Summarisation (MOS) that involves summarising opinions expressed in microblogging platforms, specifically Twitter. The dataset is based on COVID-19 (Chen, Lerman, and Ferrara 2020) and UK Elections (Bilal *et al.* 2021)-related tweets. On the COVID-19 tweets, they achieved 0.2154, 0.0374, and 0.1454 in ROUGE-1, ROUGE-2, and ROUGE-L, respectively. Whereas, on the UK Elections tweets, they achieved 0.2050, 0.0342, and 0.1363 in ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

Olabisi *et al.* (2022) introduce the DivSumm dataset, which consists of diverse dialect tweets and extractive and abstractive summaries written by humans and analyses the extent of dialect diversity in both human and system-generated summaries. DivSumm consists of input-summary pairs on a set of 25 topics of tweets written in three different dialects (African-American (AA) English, Hispanic English, and White English). The dataset includes corresponding human-generated extractive and abstractive summaries. The authors explore the diversity-preserving qualities of summarisation algorithms using three approaches: Vanilla, Cluster-Heuristic (Cluster-H), and Cluster-Automatic (Cluster-A). The Vanilla approach uses a single set of randomised documents from all dialect groups as input to the summarisation model without any preprocessing. The Cluster-H approach partitions the input documents into group-based subsets before generating separate group summaries, which are then combined into a single document to generate a new combined summary. An attribute-agnostic approach based on automatic clustering (Cluster-A) is used to make summaries when the sensitive group attribute cannot be reliably observed or inferred. This is done by grouping the data into clusters and joining the resulting summaries into a single summary. For the abstraction summary generation, the authors used the BART (Lewis *et al.* 2020) model that achieved 0.17 and 0.15 in ROUGE-1 and ROUGE-L, respectively.

Shen *et al.* (2023) focus on the challenging task of abstractive summarisation for long documents. While their work is primarily focused on long documents, it offers insights into abstractive summarisation techniques that may be applicable to event summarisation from Twitter. The article employs the concept of unbalanced optimal transport for text alignment, which can potentially be adapted for aligning and summarising Twitter data. They achieved 0.4887, 0.2034, and 0.4461 in ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

Karimi *et al.* (2023) focus on improving event detection by incorporating external information from news streams. Their approach could be valuable in the context of event summarisation, as it highlights the importance of leveraging additional data sources to enhance the quality of event summaries.

Our work is similar to the pipeline framework of OnSes (Niu *et al.* 2016) which includes first clustering the tweets and second, generating summaries from each cluster. However, the significant difference in our approach is the implementation of a contextualised vector representation of the tweets instead of non-contextualised vector representations such as Word2Vec. The previous studies explore only one clustering technique and do not provide much insight into the effects of choosing a particular clustering technique on the summarisation task. In contrast,

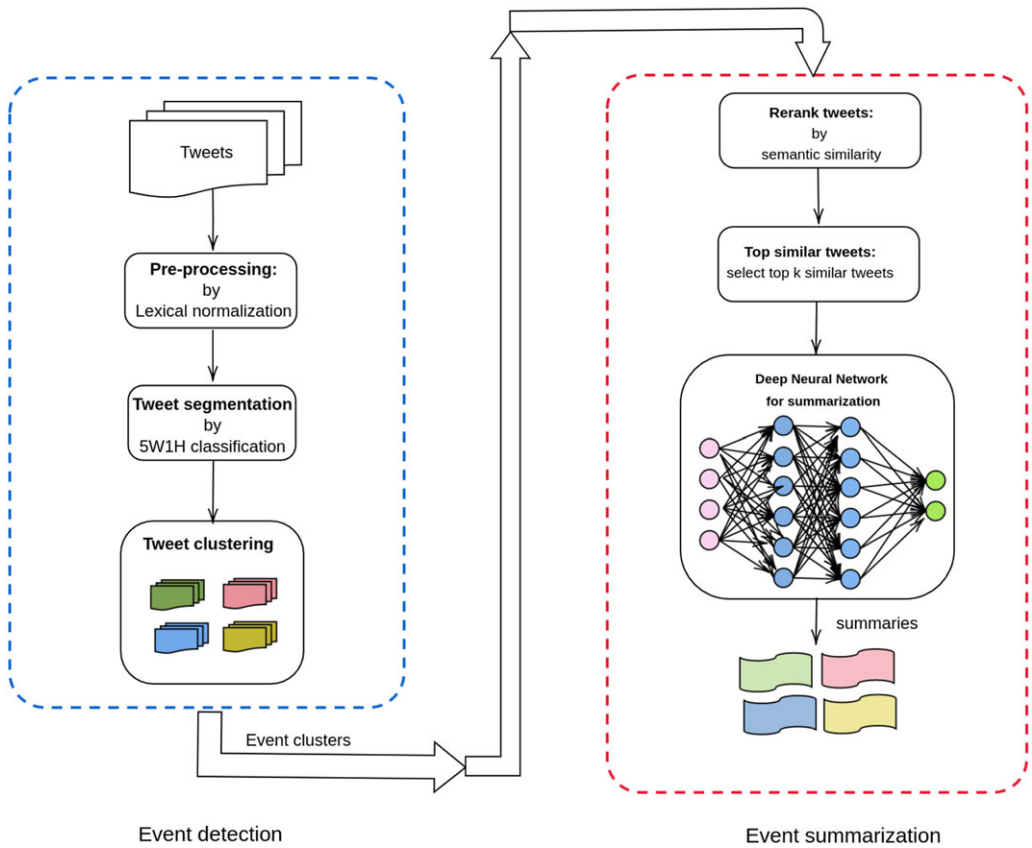


Figure 1. Framework of the proposed system.

we explore several clustering techniques, evaluate the outcome of each clustering technique, and then select only those clusters for the summarisation task that describe the topic (sub-topic) or event (sub-event). For the summarisation task, we explore three state-of-the-art neural network architectures:

- *Pointer-Generator with coverage mechanism* (See *et al.* 2017)
- *Presumm: Text summarisation with pre-trained Encoders* (Liu and Lapata 2019), a BERT (Devlin *et al.* 2019)-based network,
- *BART: Bidirectional and Autoregressive Transformers.* (Lewis *et al.* 2020).

Despite recent advancements in these architectures, exemplified by the work of Moirangthem and Lee (2020), Liu and Liu (2021), liang *et al.* (2021), and Aghajanyan *et al.* (2020), we intentionally excluded them from the summarisation task. This decision was made to determine the individual impact of the three architectures in their fundamental forms. With several experimental set-ups, we compare the summary generated by each of the three architectures: See *et al.* (2017), Liu and Lapata (2019), and Lewis *et al.* (2020) and report their performance in Twitter abstractive summary generation task.

3. System overview

There are three major components of our framework: (1) *5W1H segmentation*, (2) *event detection by clustering*, and (3) *event summarisation* as shown in Fig. 1. Event detection is done first

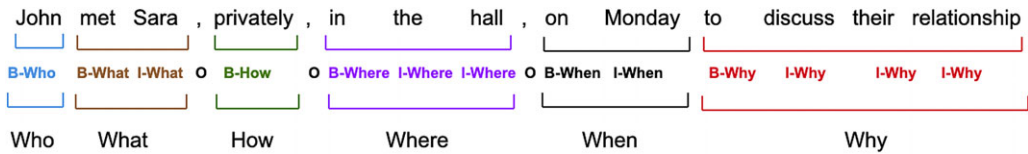


Figure 2. 5W1H BIO sequence labelling example.

by segmenting every tweet into the 5W1H (*Who*, *What*, *When*, *Where*, *Why*, and *How*) semantic components. On the basis of the semantic similarity of the components, the tweets are then clustered together to represent an event (and/or subevents). The generated clusters are then fed to a neural network (pointer, transformer, or BART) to generate abstractive summaries. We describe the components of our proposed approach in the following subsections.

3.1 5W1H semantic segmentation

In the initial stage, we segment the tweets by labelling them with the 5W1H components: *who*, *what*, *when*, *where*, *why*, and *how*. The 5W1H components are considered essential for covering a piece of information and therefore form the basis for information extraction. In journalism, a report is considered complete only if it answers the question of *Who* did *what* to *whom*, *when*, *where*, *why*, and *how*. For example, the given sentence “*John met Sara privately, in the hall, on Monday to discuss their relationship*” describes the event “*someone meeting someone.*” We, therefore, model the tweets as a sequence of 5W1H (*Who*, *What*, *When*, *Where*, *Why*, and *How*) semantic components. Let $w = \{w_1, w_2, \dots, w_n\}$ be the sequence of words in a tweet and A be the attribute to which w is to be mapped. Therefore, a tweet with the 5W1H components may be considered as (w, A) , where, A is the tuple $\langle WHO, WHAT, WHEN, WHERE, WHY, HOW \rangle$ in 5W1H. Let $w = \text{“John met Sara privately, in the hall, on Monday to discuss their relationship.”}$ In this example sentence, the 5W1H components will be identified as:

$$\langle \text{John} \rangle_{[WHO]} \langle \text{met Sara} \rangle_{[WHAT]} \langle \text{privately} \rangle_{[HOW]}, \langle \text{in the hall} \rangle_{[WHERE]}, \\ \langle \text{on Monday} \rangle_{[WHEN]} \langle \text{to discuss their relationship} \rangle_{[WHY]}$$

The set of words contained in the text w is represented by $\psi_A(w)$ which is classified as the attribute A where $A \in 5W1H$. Therefore, the 5W1H model of tweets can be represented as:

$$\psi_{5W1H}(w) = \bigcup_{A \in 5W1H} \psi_A(w) \quad (1)$$

Equation (1) suggests that a tweet is a sequence of words where the words are grouped into the 5W1H components. Due to its short length, a tweet may or may not have all of the 5W1H components present. Therefore, in equation (1), the 5W1H model of the tweets is the union (\bigcup) of the set ψ_A containing the words w where A could be any of the 5W1H components. Therefore, we model 5W1H as a sequence labelling task. Since each 5W1H component is either a single word or a phrase, we adopted the BIO encoding scheme for labelling the tweets, where a “B” indicates the beginning of the 5W1H component, an “I” indicates that a word is inside of a 5W1H component, and an “O” indicates that the word is outside of 5W1H (does not represent any of the 5W1H components). The BIO sequence labelling strategy is shown in Fig. 2.

For this purpose, we used a deep neural network system implemented by Chakma, Das, and Debbarma (2019) that gave an F-1 score of 88.21% with an error of 11.79% on the dataset. Since there is an error of 11.79%, we corrected them by manually relabelling the misclassified tweets. In some cases, the deep learning system did not correctly label the BIO-based tag sequences of the 5W1H components. The following example illustrates the misclassification scenario.

Example Tweet:

the way trump will give his victory speech will define his presidency

The above tweet has two main verbs: *give* and *define*. This represents two events (“giving” and “defining”). If we consider the “giving” event, then the 5W1H sequence tagging/labelling is done as:

$\langle the \rangle_{B-How} \langle way \rangle_{I-How} \langle trump \rangle_{B-Who} \langle will \rangle_O \langle give \rangle_{verb} \langle his \rangle_{B-What} \langle victory \rangle_{I-What}$
 $\langle speech \rangle_{I-What} \langle will \rangle_O \langle define \rangle_O \langle his \rangle_O \langle presidency \rangle_O$

Whereas, if we consider the “defining” event, then the 5W1H sequence tagging/labelling is done as;

$\langle the \rangle_{B-Who} \langle way \rangle_{I-Who} \langle trump \rangle_{I-Who} \langle will \rangle_{I-Who} \langle give \rangle_{I-Who} \langle his \rangle_{I-Who} \langle victory \rangle_{I-Who}$
 $\langle speech \rangle_{I-Who} \langle will \rangle_O \langle define \rangle_{verb} \langle his \rangle_{B-What} \langle presidency \rangle_{I-What}$

In the case of the “defining” event, the *Who* component covers “trump” which is labelled as $\langle trump \rangle_{I-Who}$. However, the system misclassified it as $\langle trump \rangle_{B-Who}$. Therefore, it is necessary to manually correct this type of misclassification. We then convert the 5W1H-labelled tweets into a structural vector representation e^{5w1h} shown in Fig. 3. Then a contextualised vector $e^{contextual}$ is obtained by concatenating e^{5w1h} with embeddings of *tweets* and embeddings of *verbs*:

$$e^{contextual} = \left[e^{tweet} \circ e^{5w1h} \circ e^{verb} \right], \quad (2)$$

where e^{tweet} is the word embeddings of the tweet, e^{verb} are the verb embeddings, and e^{5w1h} is the 5W1H embeddings. It is not necessary that all 5W1H components are always present in a tweet. For example, in the sentence “Donald Trump won the elections,” there is no “When,” “Where,” “Why,” and “How” components present. Therefore, to generate fixed-sized embeddings, the embeddings are padded with zeros (0’s). The input to BERT is given in the format $\langle [CLS] \text{ tweet } [SEP] \text{ 5W1H } [SEP] \text{ verb } [SEP] \rangle$ (Fig. 3), where the sequence $[SEP] \text{ 5W1H } [SEP]$ is the *Who*, *What*, *Where*, *When*, *Why*, and *How* encoding of the sentence. A “verb indicator” embedding is then concatenated into the 5W1H encoding to distinguish the verb tokens from the nonverb ones. The sequences $([e^{tweet}, e^{5w1h}, e^{verb}])$ are then fed into the BERT network to generate contextual embeddings $e^{contextual}$. Contextual embeddings are obtained by aggregating (concatenating) them on the last axis, resulting in sentence encoding of 1,536 dimensions.

3.2 Event clustering

We group the contextual embeddings generated by BERT into clusters where each cluster corresponds to a possible event (or sub-event). For clustering contextual embeddings, we applied four clustering techniques such as k-means (Jin and Han 2017), *Hierarchical Density-Based Spatial Clustering of Applications with Noise* (HDBSCAN) (Campello, Moulavi, and Sander 2013), *Hierarchical Agglomerative Clustering* (HAC) (Zepeda-Mendoza and Resendis-Antonio 2013), and *Affinity Propagation* (AP) (Frey and Dueck 2007). k-means requires the number of clusters to be specified before starting the algorithm. Since arbitrarily selecting k is not a viable approach; we, therefore, selected the number of clusters k based on quality metrics such as *Silhouette* (Rousseeuw 1987) for the given dataset. *Silhouette score*^c is a method for measuring the similarity of a data object within the same cluster to that of other clusters. The *Silhouette score* varies between -1

^cshorturl.at/pxETW

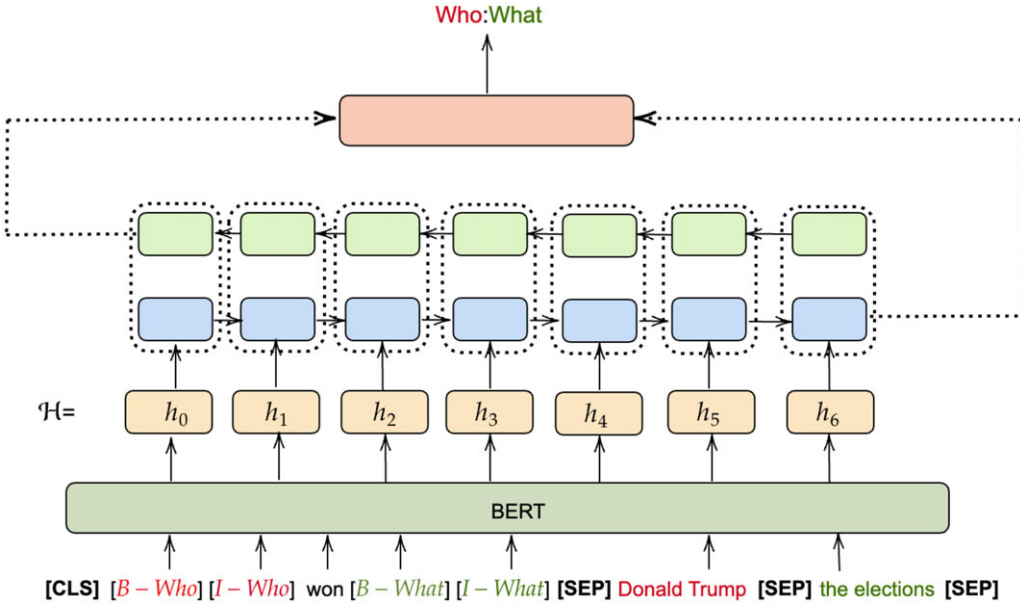


Figure 3. Contextualised vector generation.

and +1. Higher values indicate higher similarity of a data object within its own cluster and poor similarity with neighbouring clusters. The *Silhouette score* for a sample s is measured as:

$$s = \frac{b - a}{\max(a, b)} \quad (3)$$

where an increase in *Silhouette score* for k represents the suggested number of clusters. We used the scikit-learn^d API for implementing k-means clustering.

HDBSCAN is an extension of the DBSCAN (Ester *et al.* 1996) algorithm that converts DBSCAN into a hierarchical clustering algorithm. HDBSCAN views data as a weighted graph with vertices and edges, with weights equal to the mutual reachability distance between the points. Using the weighted graph, HDBSCAN extracts flat clustering based on the stability of the clusters. The algorithm goes through the following steps: (1) *Transform the space according to the density/sparsity of the data distribution*, (2) *Build the minimum spanning tree of the distance weighted graph*, (3) *Construct a cluster hierarchy of connected components*, (4) *Condense the cluster hierarchy based on the minimum cluster size*, and (5) *Extract the stable clusters from the condensed tree*. The algorithm considers dense data distribution as “islands” and sparse noise as “sea.” The objective is to find the islands of higher density from a sea of sparser noise. That means making “sea” points more distant from each other and from the “island.” Campello *et al.* (2013) defines *core distance* as the distance of a point to the k th nearest neighbour which is denoted as $\text{core}_k(x)$ for parameter k for a point x . The distance metric between points is measured with *mutual reachability distance* which is represented as:

$$d_{\text{reach-}k}(a, b) = \max \{ \text{core}_k(a), \text{core}_k(b), d(a, b) \} \quad (4)$$

where $d(a, b)$ is the original metric distance between a and b . All the low core distance points remain at the same distance from each other, but the larger core distance points are pushed away to

^d<https://scikit-learn.org/stable/modules/clustering.html#k-means>

be at least their core distance away from any other point. This process lowers the sea level, thereby spreading sparse sea points out while leaving land untouched. However, it depends on the choice of k where larger k values interpret more points as being in the sea. The algorithm then constructs a *minimum spanning tree* with the data points as vertices and an edge between any two points with a weight equal to the mutual reachability distance of those points. The minimum spanning tree is then converted into a hierarchy of connected components. This is most simply accomplished by sorting the tree's edges by distance (in increasing order) and then iterating through, establishing a new merged cluster for each edge. The initial phase of cluster extraction involves the condensation of the intricate and extensive cluster hierarchy into a more concise tree, wherein each node contains a slightly augmented dataset. Upon examination of the aforementioned hierarchy, it is often observed that a cluster split emerges as a result of one or two points branching off from a cluster. It is critical to recognise that this split should be viewed as a persistent, single cluster that undergoes a loss of points, rather than the formation of two new clusters. To solidify this concept, it is essential to establish a *minimum cluster size*, which serves as a parameter for HDBSCAN. Once this *minimum cluster size* is determined, we can proceed to traverse the hierarchy and inquire at each split whether one of the newly formed clusters contains fewer points than the *minimum cluster size*. If this is the case, we declare it as “*points falling out of a cluster*” and attribute the larger cluster with the parent's cluster identity, while recording the points that “*fell out*” and the corresponding distance value. Conversely, if the split leads to two clusters, each at least as significant as the *minimum cluster size*, we acknowledge it as a genuine cluster split and allow it to persist in the tree. By following this process and traversing the entire hierarchy, we obtain a considerably smaller tree with only a few nodes, each of which contains information regarding the cluster's size at that particular node and the subsequent decrease in size as the distance varies. We used the python^e-based implementation of HDBSCAN for our purpose.

HAC comes under the family of hierarchical clustering algorithms that build nested clusters by merging them successively with a bottom-up approach. The merging could be done by either one of the four approaches: *ward*, *complete linkage*, *average linkage*, and *single linkage*. Ward minimises the sum of squared differences within all clusters. Complete linkage minimises the maximum distance between observations of pairs of clusters. Average linkage minimises the average of the distances between all observations of pairs of clusters. Single linkage minimises the distance between the closest observations of pairs of clusters. We used the scikit-learn^f API for implementing HAC clustering with linkage parameters set to *ward* and *complete linkage*. For *ward*, we set the *affinity* parameter to “euclidean” and for *complete linkage* to “cosine,” respectively. We did not observe a significant difference in the generated clusters by both methods. In scikit-learn, HAC requires a threshold value to be set for either one of the two parameters: *n_clusters* and *distance_threshold* where the former represents the number of clusters to be generated, whereas the latter represents the linkage distance threshold above which clusters will not be merged. Selecting a distance threshold is difficult. Therefore, we specified the number of clusters k based on two evaluation parameters: *silhouette score* and *dendrogram cutoff*. Both *Silhouette score* and *dendrogram cutoff* suggested $k = 10$ for the given contextual embeddings.

In AP, clusters are created by sending messages between pairs of samples until convergence. A concept called “exemplar” is used which is the most representative of other samples in a given dataset. The messages actually represent the suitability of one sample to be the exemplar of the other. This is then updated based on the values received from the others. This updating process continues iteratively until convergence, and then the final exemplars are chosen thus giving the final cluster. The messages fall into two categories as per scikit-learn API:^g (a) the accumulated evidence called *responsibility* denoted as $r(i, k)$ such that sample k should be the exemplar for a

^e<https://hdbscan.readthedocs.io/en/latest/index.html>

^f<https://bit.ly/3dVJrQ4>

^g<https://bit.ly/3eN47K7>

sample i and (b) *availability* denoted as $a(i, k)$ which is the accumulated evidence that sample i should choose sample k as its exemplar. The basic idea is that exemplars are chosen by samples if they are (1) similar enough to many samples and (2) chosen by many samples to be representative of themselves. The responsibility of a sample to be the exemplar of sample is given by:

$$r(i, k) \leftarrow s(i, k) - \max [a(i, k') + s(i, k') \forall k' \neq k] \quad (5)$$

where $s(i, k)$ is the similarity between samples i and k . The availability of sample k to be the exemplar of sample i is given by:

$$a(i, k) \leftarrow \min \left[0, r(k, k) + \sum_{i' s.t. i' \neq i, k} r(i', k) \right] \quad (6)$$

Initially, all values for r and a are set to zero, and the calculation of each iterates until convergence.

3.3 Event summarisation

There are two broad categories of text summarisation: (1) *extractive* and *abstractive*. Extractive summarisation is the task of generating a summary of a document by identifying the most important sentences (or phrases) and then concatenating them without the inclusion of any new words. Abstractive summarisation is the task of generating a summary of a document by paraphrasing the sentences (or phrases) and producing newer phrases. For our work, we explore the abstractive summarisation approach. Recently, the natural language processing (NLP) domain has observed significant growth in neural network-based solutions for the text summarisation task. Encoder-decoder (Sutskever, Vinyals, and Le 2014) networks are generally used for the summarisation task where the encoder could be a Recurrent Neural Network (RNN) (Sherstinsky 2020) or a transformer (Vaswani *et al.* 2017) network and the decoder could be autoregressive or non-autoregressive. In the autoregressive approach, the decoder can make current predictions with knowledge of previous predictions. For the abstractive summarisation task, we explored three neural network architectures:

- Pointer-generator (Vinyals, Fortunato, and Jaitly 2015)-based network (See *et al.* 2017) (we denote it as P-Gen)
- Presumm (Liu and Lapata 2019), a transformer-based implementation.
- BART (Lewis *et al.* 2020), a combination of Bidirectional and Autoregressive Transformers.

The above-mentioned three architectures have shown promising results in abstractive summarisation tasks on state-of-the-art datasets (Hermann *et al.* 2015). Therefore, we used the pre-trained models of *P-Gen*, *Presumm*, and *BART* on our dataset and performed several experiments to evaluate the generated summaries.

3.3.1 Pointer-generator (P-Gen)

P-Gen is an encoder-decoder network with Long Short Term Memory (LSTM) at the encoder side and a pointer-generator network at the decoder side. It is a hybrid of the sequence to sequence (Nallapati *et al.* 2016) attention and pointer network. P-Gen also exploits the concept of coverage (Tu *et al.* 2016) mechanism which is added to the pointer-generator network. Here, we reuse the definition of coverage vector and the other parameters defined in P-Gen. Coverage is a vector c^t which is the sum of the attention distributions of all previous decoder time steps represented as:

$$c^t = \sum_{t'}^{t-1} a^{t'} \quad (7)$$

where $a^{t'}$ is the attention distribution. The coverage vector is basically a modification of the original attention vector of Bahdanau *et al.* (2015). The modified attention vector is represented as:

$$e_i^t = v^T \tanh(W_h h_i + W_s S_t + w_c c_i^t + b_{attn}) \quad (8)$$

where v^T , $W_h h_i$, $W_s S_t$, $w_c c_i^t$, and b_{attn} are learnable parameters. P-Gen also define *coverage loss* which penalises the decoder for repeatedly attending the same locations. The *coverage loss* is represented as:

$$covloss_t = \sum \min(a_i^t, c_i^t) \quad (9)$$

The final composite loss function in P-Gen is defined as:

$$loss_t = -\log P(w_t^*) + \lambda \sum \min(a_i^t, c_i^t) \quad (10)$$

where λ is some hyperparameter.

3.3.2 Presumm

It is an encoder-decoder network with a transformer network on the encoder and decoder side. Presumm presents a two-stage approach where the extractive summarisation is performed in the first stage and then the abstractive summarisation in the second stage. The encoder is therefore fine-tuned twice. Since Presumm is based on the BERT architecture, special tokens [CLS] and [SEP] are inserted between each sentence to generate positional embeddings. Interval segment embeddings are generated to differentiate each sentence. For example, assuming a document D consisting of sentences $[sent_1, sent_2, sent_3, sent_4]$, embeddings can be generated as $[E_A, E_B, E_C, E_D]$. For the extractive summarisation, vector t_i of the i^{th} [CLS] token from the top layer is used to represent $sent_i$. Several transformer layers are stacked together to obtain document-level features for extracting summaries:

$$\tilde{h}^l = LN(h^{l-1} + MHAtt(h^{l-1})) \quad (11)$$

$$\tilde{h}^l = LN(h^l + FFN(h^l)) \quad (12)$$

where LN is the layer normalisation operation (Ba, Kiros, and Hinton 2016), $MHAtt$ is the multi-head attention (Vaswani *et al.* 2017), $h^0 = PosEmb(T)$, T denotes the sentence vectors, and function $PosEmb$ adds positional embeddings to T . Sigmoid is used at the final output layer, which is represented as:

$$\hat{y}_i = \sigma(W_o h_i^L + b_o) \quad (13)$$

where h_i^L is the vector for $sent_i$ from the top layer (the L^{th} layer) of the transformer. For abstractive summarisation, the presumm uses the pre-trained encoder of the extractive summariser and a six-layer transformer for the decoder.

3.3.3 BART

In BART, the sequences are encoded as denoising autoencoders. The training data for BART contain “corrupted” or “noisy” text, which will be mapped to the original or clean text. The training format is similar to the denoising autoencoders. Noising schemes include *Token Masking*, *Token Deletion*, *Text Infilling*, *Sentence Permutation*, and *Document Rotation*. The first part of BART uses the BERT’s bidirectional encoder to find the best representation of the input sequence. For each sequence of text in the input, the BERT encoder outputs an embedding vector for each token in the sequence, plus an additional vector containing sentence-level information. After obtaining the tokens and sentence-level representations of the input text sequence, the decoder must interpret

them to match the output destination. Therefore, causal or autoregressive models that look only at past data to predict the future are practical. The decoder section of conventional transformers and the GPT-1 type shared a similar construction. Only learning from previous tokens can affect the computation of current tokens because GPT stacks twelve of these decoders sequentially. The GPT decoder uses feedforward layers with masked multi-head self-recognition blocks, just like the original transformer decoder. The model is initially pre-trained on tokens t looking back to k tokens in the past to compute the current token. To enable the model to “learn the language,” this is done unsupervised on a sizable text corpus:

$$L_1(T) = \sum_i \log P(t_i | t_{i-k}, \dots, t_{i-1}; \theta) \quad (14)$$

The model is then fine-tuned in a supervised manner to maximise the likelihood of a label y given feature vectors $x_1 \dots x_n$:

$$L_2(C) = \sum_{x,y} \log P(y | x_1, \dots, x_n) \quad (15)$$

Equations (14) and (15) are combined to get the objective in equation (16), where the lambda is a learned weight parameter that limits the impact of language modelling:

$$L_3(C) = L_2(C) + \lambda L_1(C) \quad (16)$$

4. Experiments

In this section, we discuss the various clustering and summarisation experiments performed with the pre-trained models of P-Gen, Presumm, and BART.

4.1 Dataset

We collected tweets on three different topics using twitter4j^h API: *Demonetization*,ⁱ *US Elections 2016*,^j and *Me Too India Movement*.^k For *Demonetization*, we crawled around 14,940 tweets during two different time periods: one between 22nd and 23rd November 2016; and the other between 11th and 21st April 2017. The intention for the first period was to extract tweets related to the *Demonetization* topic after its announcement by Govt. of India on 8th November 2016. The second period was intended to extract tweets that discuss the impact of demonetisation on the new financial year 2017–2018 that began on 1st April 2017. For *US Elections 2016*, we crawled 38,984 tweets on the day of the elections, that is, 8th November 2016, and on the final declaration of the electoral results on 17th January 2017. The tweets were collected with hashtags such as “#USElections2016,” “#ElectionNight,” “#DonaldTrump,” “#HillaryClinton,” “#DonaldTrumpWins,” and “#USElections2016Results.” Apart from hashtags, we also used terms such as “Donald Trump,” “Trump,” “Hillary Clinton,” “Hillary,” and “Clinton.” For the *Me Too* topic, we crawled a total of 248,160 tweets with queries such as “#MeToo,” “#MeTooCampaign,” “#MeTooControversy,” and “#MeTooIndia” from 11th to 24th October 2018.

We performed some preprocessing tasks by lower-casing the tweets, removing Non-English tweets, and *retweets*. We observed that approximately 80% of the collected tweets contained *retweets* as well as non-English tweets. Therefore, after preprocessing, we finalised a total of 16,375 tweets: 3,484 on “*Demonetization*,” 6,558 on “*Me Too India Movement*,” and 6,333 on “*US*

^h<https://github.com/Twitter4J/Twitter4J>

ⁱshorturl.at/mqrAN

^jshorturl.at/bktP9

^kshorturl.at/clmCX

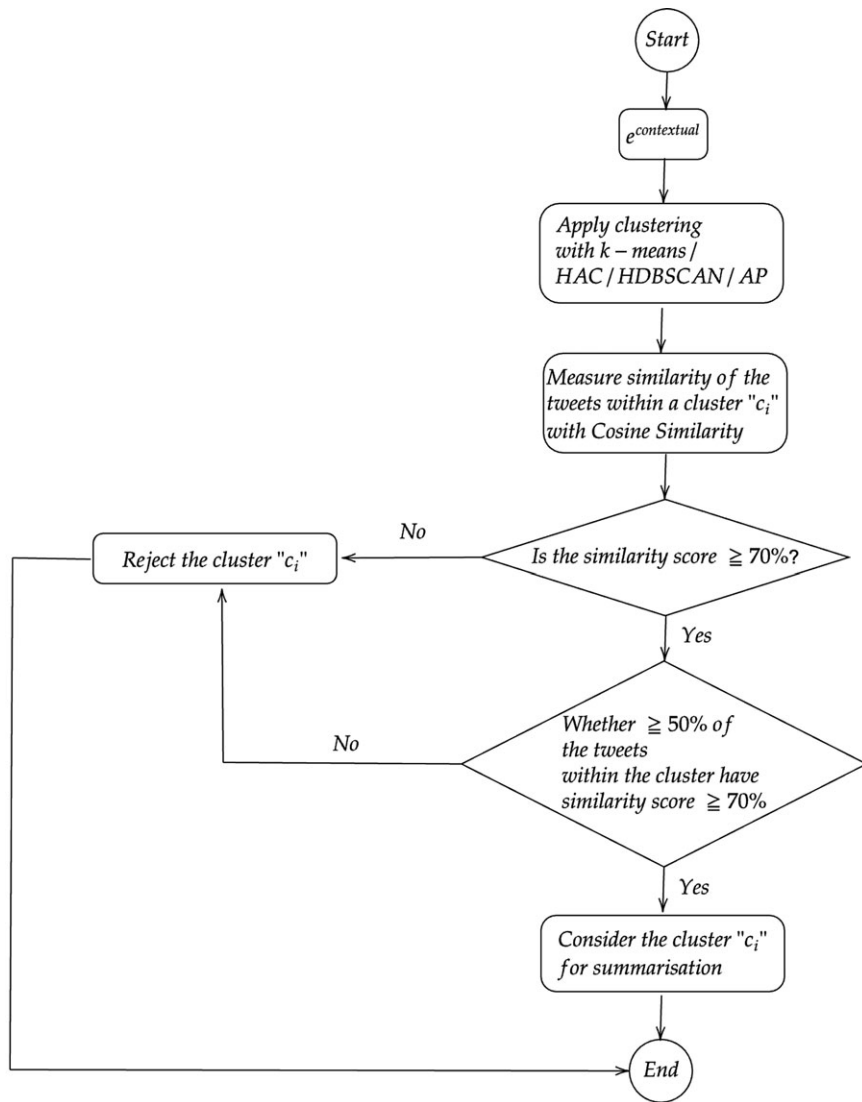


Figure 4. Cluster selection process for the summarisation task.

Elections 2016.” Due to data privacy reasons, the corpus is not available publicly. The dataset will be available on request.

4.2 Cluster selection

In this subsection, we discuss the criteria for the selection of clusters for the summarisation task. This procedure is shown in Fig. 4. We feed the contextual embeddings $e^{\text{contextual}}$ to four clustering algorithms (*k-means*, *HAC*, *HDBSCAN*, and *AP*). For *k-means* and *HAC* clustering, we choose the number of clusters k based on the suggested *Silhouette score*. As shown in Fig. 5(a) and (b), we set the number of clusters $k = 10$ for both the *k-means* and *HAC* clustering algorithms. Therefore, both *k-means* and *HAC* generated ten clusters. For *k-means*, we set the following parameters: $n_clusters = best_k$, $n_jobs = -1$, and $random_state = seed$. The parameter $best_k = 10$ is chosen

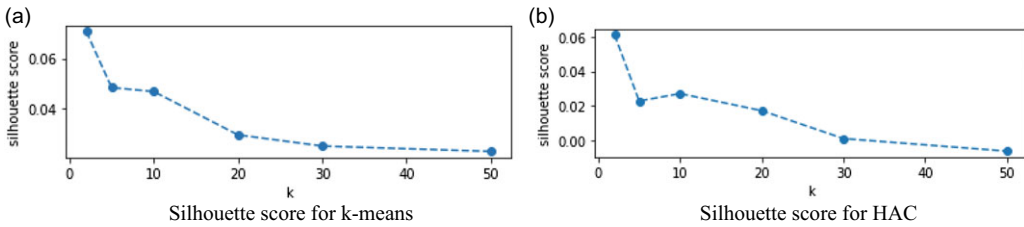


Figure 5. Selection of best value for number of clusters k for k-means and HAC.

based on the *Silhouette Score* iterations between 2 and 50. The parameter n_jobs refers to the number of parallel threads to use for the computation. A value of “-1” means that all available processors will be used for the computation. The parameter $random_state = seed$ determines a random number generation for centroid initialisation. This is done to produce the same results across different calls. We use the popular *seed* value of 42 for the generation of random numbers. For HAC, we set the following parameters: *affinity* = ‘euclidean’ or ‘cosine’, *linkage* = ‘ward’ or ‘complete_linkage’, and $n_clusters = 10$. For HDBSCAN, we set the standard parameter settings: $min_cluster_size = n$,¹ *prediction_data* = *True*, *core_dist_n_jobs* = -1, and *memory* = ‘data’. For AP clustering, we use the default parameter settings such as *damping* = 0.5, *preference* = *None*, and *affinity* = ‘euclidean’.

Cluster formation is shown in Fig. 6 and the population distribution in Fig. 7. As can be seen in Fig. 6(a) and (b) and Fig. 7(a) and (b), both the k-means and the HAC generated ten clusters, respectively. AP generated a total of 934 clusters (Figs. 6(c) and 7(c)). It produced cluster sizes between 2 and 115 tweets. The average size of the clusters produced by AP is 17.53 tweets. HDBSCAN (Figs. 6(d) and 7(d)) was unable to group 10,057 (i.e., 61.4% of 16,375) tweets and labelled them “-1”. For the remaining 6,318 tweets, HDBSCAN generated 2,356 clusters. It produced a minimum cluster size of two tweets and a maximum of twenty-three tweets, which makes it insignificant for generating summaries. We therefore ignored HDBSCAN generated clusters because 61.4% of the tweets were not considered in any cluster (labelled “-1”). Therefore, excluding HDBSCAN, we measured the cosine similarity of tweets in each cluster (c_i) produced by k-means, HAC, and AP approaches. As shown in Fig. 4, we set the minimum threshold of the similarity score to 70%. If more than or equal to 50% of the tweets in a cluster fall below the minimum similarity threshold of 70%, the cluster and the corresponding clustering method (i.e., k-means/HAC/AP) are not considered. Based on observations of cosine similarity scores, only AP clusters with a cluster size of a minimum of ten tweets are considered for the summarisation task. The first few tweets from the dataset labelled by k-means, HAC, HDBSCAN, and AP are shown in Table 1. The table shows the results of applying different clustering approaches to the dataset. For example, k-means and HAC generated ten clusters with labels 0–9. HDBSCAN could not generate clusters for more than 60% of the data and labelled them with -1. HDBSCAN generated clusters only for the remaining 40% of the data with cluster labels between 0 and 2,355. AP generated 934 clusters with labels ranging from 0 to 933.

4.3 Reference summary creation

As there is no available dataset for abstractive summarisation on tweets, we had to build our own reference summaries (gold summaries). Since the number of clusters produced by AP is large (a total of 934 clusters out of which we considered only those having a minimum of 10 tweets per

¹We experimented with the different values of $min_cluster_size$ with a default value of $n = 5$ up to $n = 10$ but did not observe a significant difference.

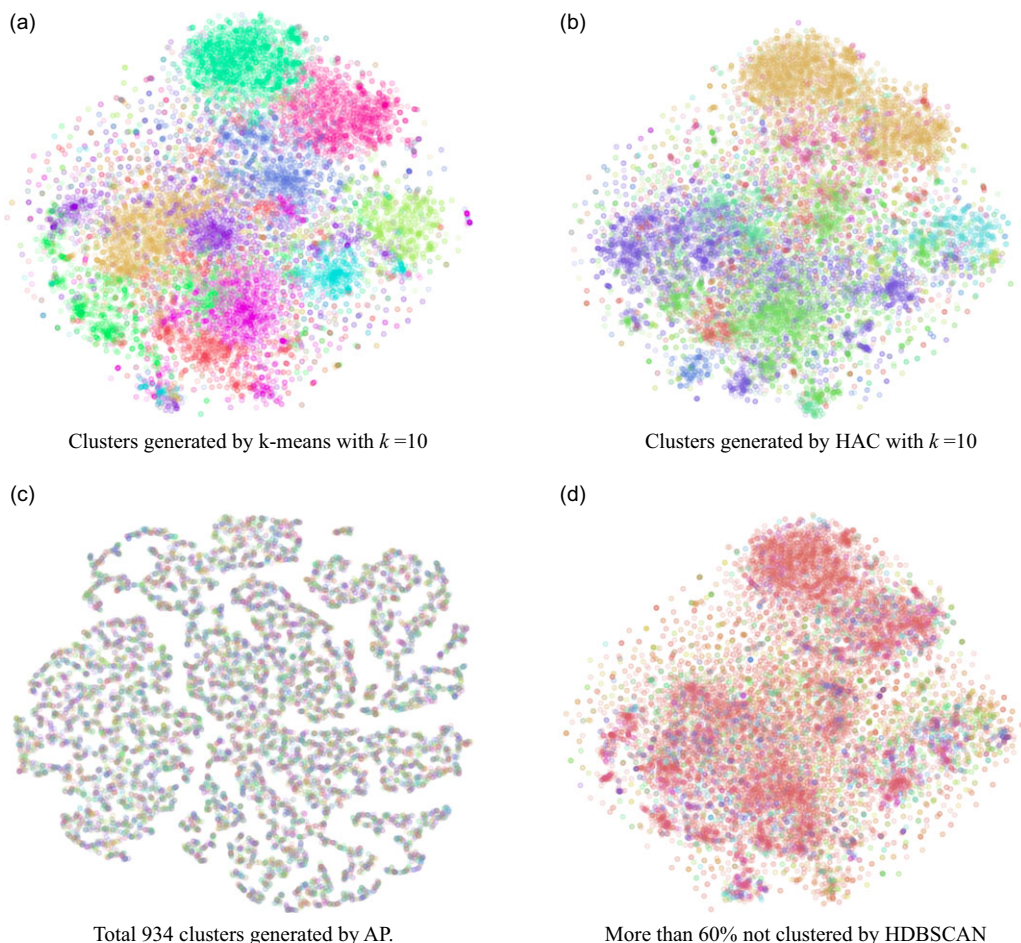


Figure 6. Visualisation of generated clusters.

cluster) as shown in Fig. 7, manually evaluating the clusters for generating reference summaries is a laborious task. Therefore, we adopted the approach shown in Fig. 8 to generate reference summaries. In the first step, we applied four extractive summarisation techniques such as Lexrank (Erkan and Radev 2004), LSA (Gong and Liu 2001), Luhn (Luhn 1958), and Textrank (Mihalcea and Tarau 2004) to obtain four candidate sets (s_1 , s_2 , s_3 , and s_4). Then in the second step, we asked two annotators to select the references from the candidate sets based on two criteria. The criteria for selecting references are as follows: (1) *the reference is found in at least two candidate sets* and (2) *it is relevant to the event/topic*. We measured the similarity of the candidate sets in terms of the ROUGE (Lin 2004) scores shown in Table 2. From Table 2, it is observed that LSA, Luhn, and Textrank extracted similar tweets compared to the Lexrank technique. The relevance of a tweet in a candidate set is measured according to the following procedure. We obtained the top n -grams (unigrams, bigrams, and trigrams) from each set of candidates and formulated queries based on the n -grams to rearrange the tweets. The distribution of n -grams in a particular cluster is shown in Fig. 9. Formally, we can represent this as an information retrieval (IR) problem described below.

Let q be a query of n -grams given over a set of documents D where the IR task is to rank the documents in D so that the documents most relevant to q appear at the top. In our case, each

Table 1. A snapshot of the cluster of tweets labelled by k-means, HAC, HDBSCAN, and AP

Tweet	Cluster labels			
	k-means	HAC	HDBSCAN	AP
@amitanatverlal and he is shamelessly showing no sign of supporting #metoo	3	0	−1	193
empowering women worldwide: https://t.co/hiawhf3c4l #genderequality #feminism #metoo	1	4	1,834	0
meanwhile rakhi sawant is also trying to recollect some interesting incidents to participate for #metoo	1	4	855	13
@iowahawkblog offended by spelt ?? #metoo! and any other grasses for that matter	3	0	−1	300
trump mocks the #metoo movement in nonsense rant at pennsylvania rally	0	1	1,446	1
@hvgoenka sir; this gives us confidence that rpg won't ever be tagged in #metoo	3	0	−1	42
we as a community need a closure; we have been silenced for long; the government needs to apologise #metoo #metooindia #metookashmir	3	0	1227	14
breaking #metoo: #lasithmalinga named by @chinmayi for sexual assault	3	0	2,041	179
till self motive (work) complete it's #sweetu and once self motive completed its #metoo	1	4	−1	303
i liked a @youtube video https://t.co/oabbpgbwtw #metoo movement and mj akbar case bebak ep 3 with abhisar sharma and dhruv rathee	1	7	2,047	86

Table 2. Average ROUGE F-1 score for candidate summary similarity of Lexrank with other methods

Method	ROUGE-1	ROUGE-2	ROUGE-L
Luhn	0.60008	0.46778	0.56358
LSA	0.21643	0.02337	0.17602
Textrank	0.61768	0.48855	0.58366

tweet within a candidate set is a document and the particular set is the document pool which is represented as:

$$D = \bigcup_{k=1 \dots n} \tau_k \tag{17}$$

where τ_k is the top k tweets retrieved. When $k = n$, all the tweets from a cluster are retrieved. To obtain the ranking scores, we used the Sentence BERT (SBERT) (Reimers and Gurevych 2019) system, which measures the semantic similarity of the tweets in a given cluster. In SBERT, the cosine similarity between the query and the tweet is measured by computing the cosine of the query vector u with the tweet vector v . Both vectors u and v have the same dimensions. To rank the tweets, *Spearman's rank correlation* is computed between the cosine similarity of the query

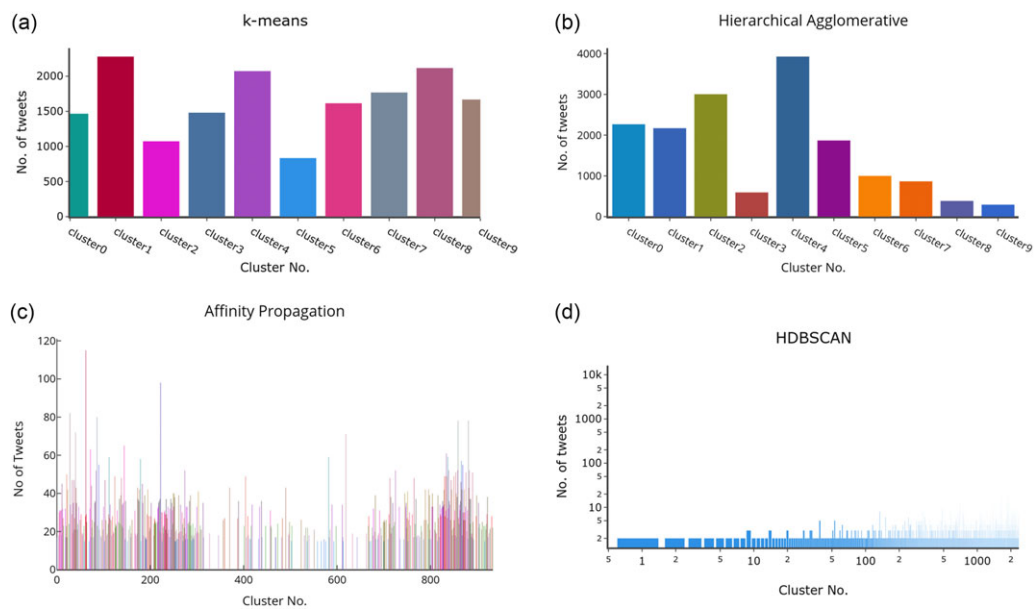


Figure 7. Population distribution of clusters by k-means, HAC, AP, and HDBSCAN. (a) Ten clusters generated by k-means clustering with the distribution of 1,466, 2,280, 1,073, 1,481, 2,073, 834, 1,615, 1,768, 2,117, and 1,668 tweets. (b) Ten clusters generated by HAC clustering with the distribution of 2,267, 2,170, 3,004, 594, 3,928, 1,868, 100, 866, 385, and 293 tweets. (c) A total 934 clusters were generated by AP with minimum and maximum cluster size of 2 and 115 tweets, respectively. (d) HDBSCAN failed to cluster 10,057 tweets which is 61.4% of the dataset. The remaining 38.6% are clustered into 2,356 clusters with length between 2 and 23 tweets.

and the tweets which is defined as:

$$\rho = 1 - \frac{\sum_{i=1}^n (u_i - v_i)^2}{n(n^2 - 1)} \tag{18}$$

where u_i and v_i are the corresponding ranks of u and v , for $i = 0, \dots, n - 1$. For example, in a particular cluster, the top trigram “mj akbar resigns” forms the query $q = \text{“mj akbar resigns.”}$ The query is then issued to SBERT to retrieve the top k tweets from the given cluster. The annotators then pick the tweets from each candidate set based on *informativeness*, *fluency*, and *succinctness* of the contents and write the summaries with up to three sentences. We measured the agreement ratio between the annotators with standard ROUGE metrics using *ROUGE-1.5.5* and obtained average F ROUGE-1 of 0.828, ROUGE-2 of 0.795, and ROUGE-L of 0.825. The summaries written by the two annotators are compared and manually aligned (through a consensus) to generate the final reference summaries.

4.4 P-Gen set-up

For the P-gen set-up, we used the pre-trained model *pre-trained_model_tf1.2.1^m* which is trained on the CNN/Daily Mail dataset. This model uses a vocabulary of 50K words with 21,50,1265 parameters. This model has 256-dimensional hidden states and 128-dimensional word embeddings. We set the parameters *max_enc_steps* and *max_dec_steps* to 400 and 10 tokens, respectively, to restrict the length of the source text and the summary, respectively. For beam search decoding, we set *beam_size* to 4.

^mshorturl.at/aeOTW

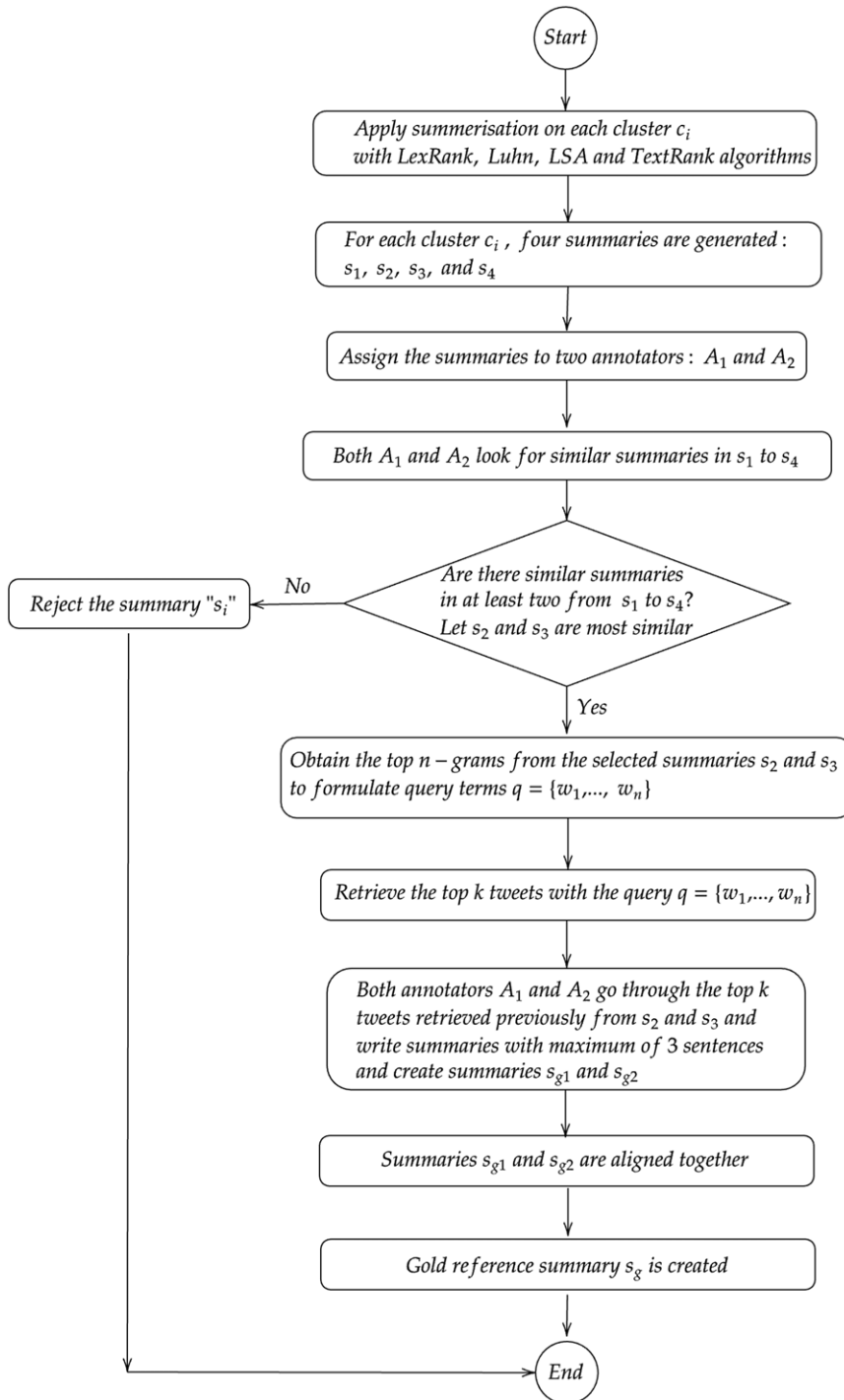


Figure 8. Gold reference summary creation process for the summarisation task.

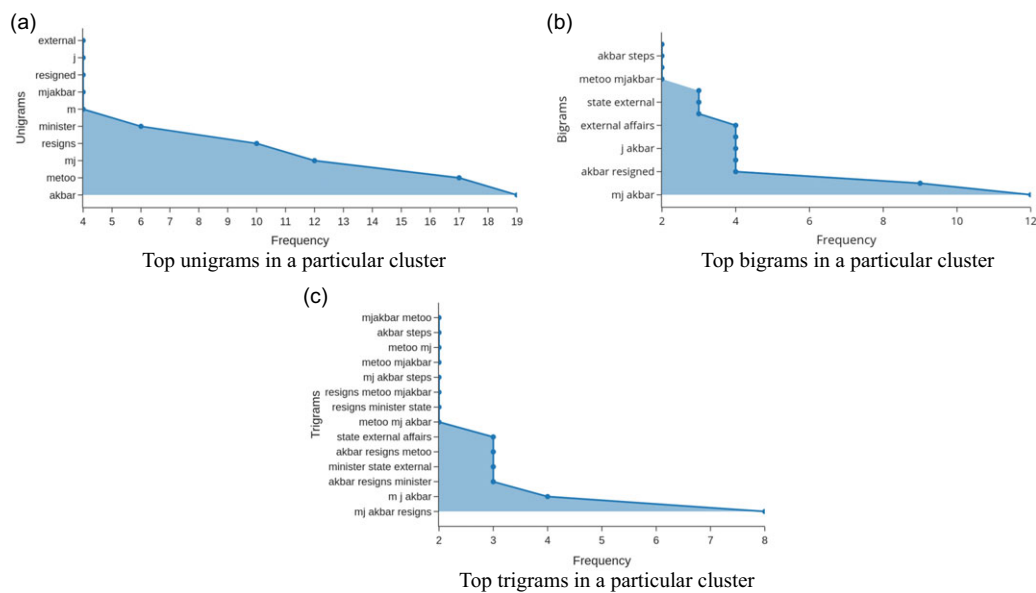


Figure 9. n-gram distribution in a particular cluster.

4.5 Presumm set-up

From Presumm, we used the *bertext_cnndm_transformer*ⁿ pre-trained model for the abstractive summarisation task. This model is also trained on the CNN/Daily Mail dataset. Presumm uses *Trigram blocking* mechanism to prevent redundancy of phrases/sentences in the generated summary. The transformer decoder has 768 hidden units with a hidden size of 2,048 for all the feedforward layers. The original model adopted a two-stage fine-tuning approach. The model first undergoes fine-tuning using an extractive BERT model. This involves training the model to identify the most important sentences or phrases in the text that should be included in a summary. Essentially, the model learns to choose relevant content for summarisation from the larger body of text. After fine-tuning with extractive BERT, the model undergoes fine-tuning with an abstractive BERT model. In this stage, the model is trained to generate new concise summaries using its learnt knowledge of the source text. The abstractive model works to create summaries that are not just extracts from the original text but are rewritten and rephrased to be coherent and fluent. In our case, we fine-tuned only at the second stage. The following settings were applied during the training process: - `drop_out = 0.2`: A dropout rate of 20% was established to help prevent overfitting by randomly disabling a portion of the neurones during training. This regularisation technique improves the generalisability of the model. - `beam_size = 5`: A beam size of 5 was chosen for the beam search algorithm. This setting allows the model to consider multiple possible paths during the generation process, potentially improving the quality and diversity of the output. - `lr = 0.02`: A learning rate of 0.02 was used to control the speed of updates to the model's parameters. This rate determines how quickly the model learns and adjusts its weights during training. - `train_steps = 15000`: A total of 15,000 training steps were specified to control the duration of the training process. This influences the performance and training time of the model. - `batch_size = 200`: A batch size of 200 was selected to indicate the number of samples processed in each training iteration. This choice affects the efficiency of training and the stability of the model's learning process.

ⁿshorturl.at/crtI4

4.6 BART set-up

For the BART set-up, the *bart-large-cnn*^o pre-trained model was utilised. Fine-tuning of the model was conducted with the following settings: - TOTAL_NUM_UPDATES = 20000: This specifies the total number of training iterations applied during the fine-tuning process. The model's learning is influenced by this number, impacting the extent of its training. - UPDATE_FREQ = 4: Updates were applied after every four training steps. This approach allows the model's learning to stabilise by accumulating gradients over multiple steps before applying them. - WARMUP_UPDATES = 500: During fine-tuning, 500 warm-up updates were conducted. This involved gradually increasing the learning rate from a lower initial value to its target value, which can help prevent sudden large parameter changes at the start of training. - LR = 3e-05: A learning rate of 0.00003 was set. This controlled the speed of parameter updates, allowing for careful adjustments to the model's weights and preventing overly rapid changes. - BEAM_SIZE = 4: A beam size of 4 was used in beam search, an algorithm for decoding that explores multiple possible sequences simultaneously. This helps to improve the quality and diversity of the output by considering multiple paths during summary generation. Overall, these settings guide the fine-tuning process to enhance the performance of the pre-trained BART model on our specific summarisation task.

5. Results and analysis

5.1 Cluster evaluation

On careful analysis, it is observed that the clusters generated by k-means and HAC contain tweets that discuss on multiple sub-events (or sub-topics) present under the same cluster. For example, cluster-1 under k-means contains 2,280 tweets mostly related to the “*Me Too Movement*.” However, the same cluster contains tweets that represent latent sub-topics or sub-events such as “*Donald Trump mocking the Me Too Movement*” and “*Resignation of minister M.J. Akbar due to Me Too allegations*.” All these tweets have been found to be under the same cluster by both k-means and HAC clustering. In contrast, AP clustering generated 934 clusters with cluster lengths between 2 and 115 tweets. Therefore, with AP clustering approach, it was possible to detect the latent sub-topics or sub-events present under the top-level events on our dataset. For example, for the top-level event of “*Me too movement*,” the AP clustering detected the sub-events: “*Donald Trump mocking the Me Too movement*” and “*Resignation of minister M.J. Akbar due to Me Too allegations*.” To evaluate the quality of the clusters, we measured the semantic similarity of the top ten tweets with SBERT (since we chose 10 as the minimum threshold length of a cluster under AP). The similarity score in the AP clustering ranges from 0.53 to 0.90, indicating appropriate grouping of similar tweets related to an event or subevent. Therefore, these observations suggest that AP generated the most relevant clusters that are further considered to generate the summaries. Fig. 10 displays the degree of similarity between ten tweets in a cluster generated by AP.

5.2 Summary evaluation

We evaluated the performances of P-Gen, Presumm, and BART architectures with standard ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU scores by using *ROUGE-1.5.5*. The ROUGE scores are shown in Fig. 11. To evaluate the summaries, we performed three sets of experiments on the clusters: *Unranked*, *Ranked*, and *Top 10 ranked*. *Unranked* is the cluster of tweets without any changes in the ordering of the tweets. *Ranked* is the cluster of ranked tweets based on their relevance to the event/topic. *Top 10 ranked* is the cluster of tweets containing only the top ten ranked tweets based on their relevance to the event/topic. These three sets are described in more detail in the following sections.

^o<https://rb.gy/aiz1pb>

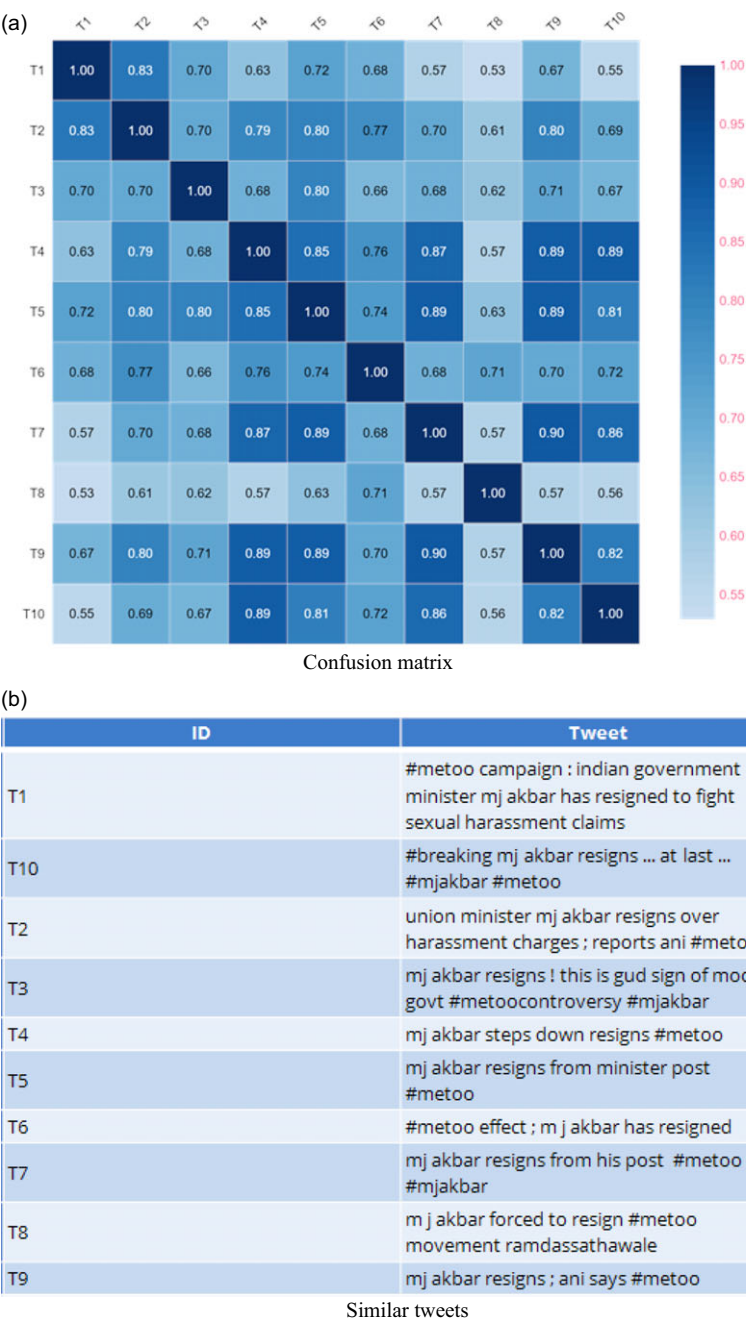


Figure 10. Similarity of tweets under a particular cluster by AP.

5.2.1 Unranked

For clusters with unranked tweets (clusters without changes in the ordering of the tweets), the three architectures produced summaries with very low ROUGE scores. The ROUGE-1 F scores for the three architectures are 0.07, 0.12, and 0.16, respectively (Fig. 11). As stated above, the gold summaries were prepared based on the top ten ranked tweets in each cluster. Therefore, the

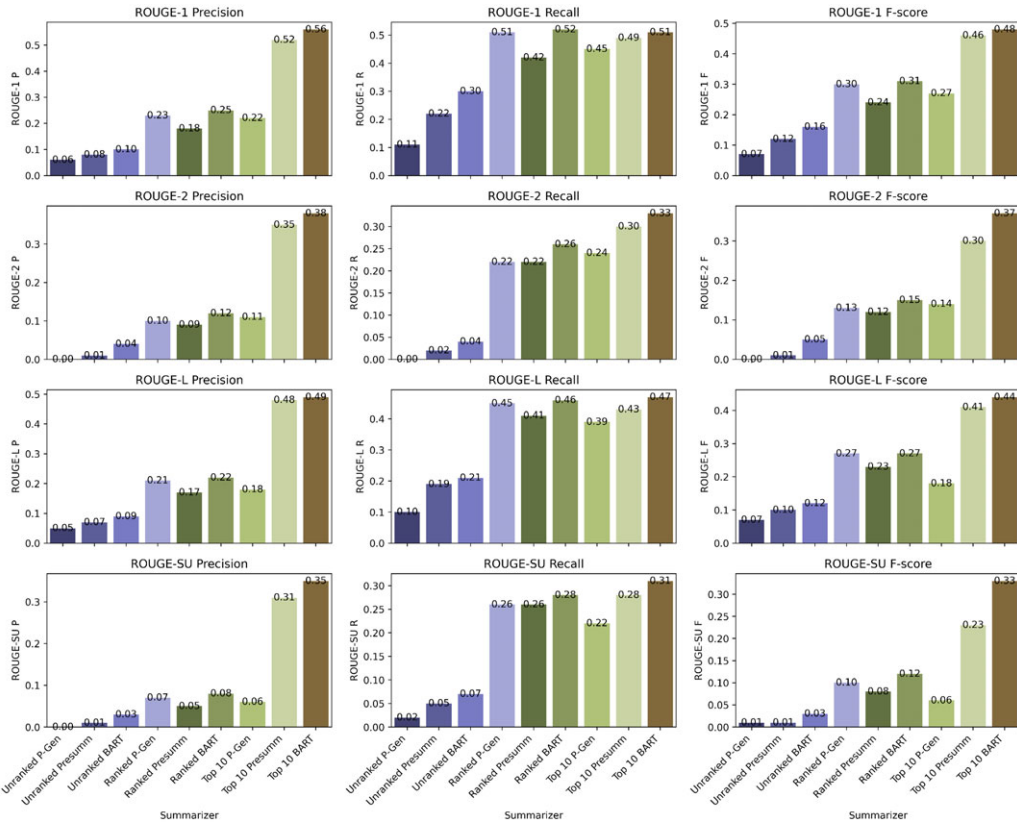


Figure 11. ROUGE Scores—Precision, Recall, and F-score for ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU.

summaries did not match most of the n-grams of the gold summaries, resulting in very low scores. These observations also suggest that the three deep learning architectures perform poorly when the document length is large. The reason for the poor performance is that the summarisers pick the earlier tweets that appear in a cluster.

5.2.2 Ranked

A significant improvement in ROUGE scores is observed when tweets are ranked within a group based on query q . In Fig. 11, the rows Ranked P-Gen, Ranked Presumm, and Ranked BART are the ROUGE scores for all the three architectures, respectively. For the ranked cluster, BART performed slightly better than the pointer-generator with coverage mechanism P-Gen. The transformer-based Presumm scored the lowest. BART beat Presumm by 0.01 ROUGE-1 F and 0.02 ROUGE-2 F. For the ROUGE-L score, both P-Gen and BART scored 0.27. Presumm scored the least with 0.23 in ROUGE-L F. These differences are very marginal among the three approaches, but a significant improvement over the unranked scores. These observations suggest that the position of the tweets in a cluster affects the summary generation by all three approaches.

5.2.3 Top 10 ranked

Our final evaluation is based on generating summaries from the clusters that contain the top ten tweets ranked. We observed a significant improvement in the performance of BART and Presumm when the clusters with the top ten relevant tweets were considered. A possible reason could be

that the gold summaries were prepared based on these top ten relevant tweets, which resulted in high similarity with the n-grams of the generated summaries. The last three rows of Fig. 11 show the ROUGE scores for the summaries generated from the top ten ranked tweets in the clusters. ROUGE-1 F score for P-Gen dropped from 0.30 (Ranked P-Gen) to 0.27 (top 10 P-Gen), whereas for Presumm and BART, there is a significant improvement from 0.24 (Ranked Presumm) to 0.46 (top 10 Presumm), and from 0.31 (Ranked BART) to 0.48 (top 10 BART) respectively. Similar are the cases for ROUGE-2, ROUGE-L, and ROUGE-SU.

From the results, we observe that for abstractive summarisation task, in addition to good ROUGE scores, the number of noble n-grams produced by the summariser are also important. We observe that transformer-based BART and Presumm comparatively perform better than the pointer-based P-Gen models in terms of producing noble n-grams.

A one-way ANOVA test was conducted to compare the results of ROUGE scores given in Fig. 11 in order to confirm if some approach is statistically better. By considering the significance level (α) set at 0.05, it was observed that the p -value (0.000002574) is less than α , thus leading to the rejection of the null hypothesis. Furthermore, it was noticed that the averages of certain groups are not considered equal; in other words, the difference between the averages of some groups is statistically significant. A p -value of 0.0000025742 was obtained, with $p(x \leq F) = 0.999997$, indicating a low chance of type 1 error (rejecting a correct null hypothesis, H_0), at only 0.000002574. A smaller p -value strongly supports the alternate hypothesis, H_1 . The test statistic F was calculated as 9.899524, which falls outside the 95% region of acceptance: $[\infty; 2.3053]$. Finally, it has been observed that the following pairs are found to be significantly different when using Tukey's HSD: (Unranked P-Gen, top 10 Presumm), (Unranked P-Gen, top 10 BART), (Unranked Presumm, top 10 Presumm), (Unranked Presumm, top 10 BART), (Unranked BART, top 10 Presumm), (Unranked BART, top 10 BART), (Ranked P-Gen, top 10 BART), (Ranked Presumm, top 10 BART), (Ranked BART, top 10 BART), and (top 10 P-Gen, top 10 BART).

6. Time complexity analysis

6.1 Clustering time complexity

It can be challenging to determine the precise time complexity of the k-means algorithm for the detection of events from tweets, as there are a variety of factors that can affect the calculation, such as the number of tweets, the number of features extracted from tweets, the number of clusters (k), the convergence criteria, and the initialisation method used. The k-means algorithm is iterative in nature, and its time complexity is usually expressed in terms of the number of iterations (T), the number of data points (N), the dimension of the data (D), and the number of clusters (k). Therefore, the overall time complexity of the k-means algorithm is $O(T \times N \times k \times D)$. In HAC, the algorithm first computes similarities or distances between tweets, which has a time complexity of $O(N^2)$. This is because HAC needs to compare each tweet with every other tweet. HAC initialises each tweet as a cluster that has a time complexity of $O(N)$. In each iteration, HAC merges the two closest clusters for $(N - 1)$ iterations. This requires an update in the distances between the newly formed cluster and the remaining clusters. The time complexity for these updates depends on the distance metric and the data structure used to store the distances. If a priority queue or a suitable data structure is used, then each update might take $O(\log N)$ time, leading to an overall complexity of $O(N \log N)$ for agglomeration. Once all the clusters are merged into a single cluster, we have our final result. This step involves $O(N)$ operations. The most influential element of the process is usually the agglomeration step. Therefore, the total time complexity of HAC for the detection of tweet events is likely to be approximately $O(N^2 + N \log N)$, depending on the exact implementation and the distance measure used. Estimating the time complexity of AP for event detection from tweets can be a difficult task, as it depends on a variety of elements, such as the magnitude of the input data, the parameters selected, and the implementation specifics. AP iteratively updates message passing values between data points to determine the best exemplars (cluster

centres) for the data points. The main steps involved in the algorithm are to update the messages and update the responsibility/availability matrix with a time complexity of $O(N^2)$, where N is the number of data points. Following the message update step, the algorithm chooses exemplars based on responsibility and availability values. This step has a complexity of $O(N^2)$. The number of iterations the algorithm needs can vary depending on the data and how it converges. In the worst case, the number of iterations could increase in relation to the number of data points, resulting in an overall time complexity of $O(N^3)$.

For smaller datasets, k-means might perform faster due to its linear dependence on the number of data points. As the dataset grows, the quadratic dependence of AP on the number of data points can make it significantly slower than k-means. In general, the HAC time complexity has a clear dependence on the number of tweets and the chosen linkage criterion. On the other hand, the complexity of AP is more dependent on the convergence properties of the data and the algorithm parameters. Additionally, AP can discover clusters with irregular shapes, while k-means assumes that clusters are spherical and equally sized. The choice between these algorithms should also consider the quality of the clustering results and the suitability of the algorithm for the underlying data distribution and problem. On our dataset, both the k-means and HAC clustering algorithms did not generate appropriate clusters that represent the top-level events and the corresponding subevents. For example, the event “US Presidential Elections” has several sub-events such as (a) *before the election*, (b) *on the day of the election*, and (c) *after the declaration of the election results*. Both k-means and HAC clustering did not generate quality clusters to represent these subevents. This is because most of the clusters generated by k-means and HAC contain tweets representing almost all of the subevents. This has a significant impact on the summarisation task because a cluster consisting of multiple subevents may not be suitable for the appropriate summary generation.

6.2 Time complexity of summarisation

Comparing the time complexity of various text summarisation models can prove to be a difficult task, as it depends on various factors, including model architecture, input size, and hardware. The time complexity may also be subject to variations depending on how the model is implemented and optimised. The pointer network with coverage (P-Gen) has been introduced for the purpose of abstractive text summarisation. It is important to note that the time complexity of this model is primarily influenced by the length of the input document as well as the length of the output summary. In terms of complexity, it can be approximated as $O(n)$, where n represents the length of the input document. Presumm employs pre-trained encoder models, such as BERT, for both extractive and abstractive summarisation. The computational efficiency of these models is primarily based on the length of the input document and the number of tokens processed by the model. With regard to BERT-based models, computational efficiency is conventionally characterised by a time complexity of $O(L)$, where L denotes the number of tokens present in the input. BART is a model that utilises sequence-to-sequence technology for abstractive text summarisation. As with other models of this type, the level of complexity is linked to the lengths of both the input document and the output summary. Given its utilisation of multiple attention mechanisms and transformer layers, BART’s complexity tends to be greater than that of simpler models. In fact, its complexity can reach a level higher than $O(n)$, sometimes reaching $O(n^2)$ or worse for long documents, depending on the particular implementation and model size.

7. Conclusion

Studies on ATS from microblogs such as tweets have been in existence for a long time. Earlier approaches are mostly based on constructing graphs and generating clusters, where the

summarisation is based on statistical approaches. On the contrary, our approach is based on semantics that uses contextualised word vectors. In this paper, we presented a pipeline-based framework that involves clustering in the first phase and then summarisation in the second phase. We generate contextual embeddings based on the 5W1H semantic components to generate semantic segments of the tweets. Unlike previous studies, we explored several clustering techniques, such as k-means, HAC, HDBSCAN, and AP clustering, to cluster tweets targeting the same topic or event. Among these four clustering techniques, we chose the clusters generated by AP for the summarisation task based on the quality of the clusters. We measured the quality of the clusters based on the cosine similarity scores of the tweets within each cluster. Our experiments show that the selection of the clustering technique plays an important role in representing the events and sub-events which affects the generated summaries. This suggests that the adoption of a particular clustering technique as a preprocessing step helps in better representation of events and sub-events. For the summarisation task, we adopted three state-of-the-art neural network architectures (P-Gen, Presumm, and BART) and compared the summaries generated by them on our dataset. The results of our experiments show that the three neural network architectures do not produce quality summaries when tweets are arbitrarily placed (unranked). We also observed that better summaries are produced when tweets are ranked in a cluster. Both the transformer-based models: Presumm and BART performed better than the pointer network P-Gen. The present limitation of this pipeline approach is the time complexity associated with the clustering algorithms. Applying clustering before summarisation may not be a feasible solution for summarisation of tweets in real time. The application of clustering as a preprocessing step is useful only when the notion of real-time processing of tweets is not a concern. The other limitation of our approach is the use of pre-trained models of P-Gen, Presumm, and BART, which have been trained on formal texts rather than tweets. In the future, we intend to train the three neural network architectures on tweets and apply those models for the summarisation of tweets. Furthermore, we intend to apply our approach to a larger dataset, including multilingual tweets, and perform further analysis.

Acknowledgement. We express our sincere gratitude to anonymous reviewers for their invaluable contributions to improving the readability and quality of the article. We thank them for their time and expertise in advancing the knowledge in this domain.

References

- Aggarwal C.C., Han J., Wang J. and Yu P.S. (2003). A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29, VLDB'03*, Berlin, Germany. VLDB Endowment, pp. 81–92.
- Aghajanyan A., Shrivastava A., Gupta A., Goyal N., Zettlemoyer L. and Gupta S. (2020). Better fine-tuning by reducing representational collapse. *CoRR*, abs/2008.03156.
- Amati G. (2009). Bm25. In *Encyclopedia of Database Systems*. Boston, MA: Springer US, pp. 257–260.
- Arthur D. and Vassilvitskii S. (2007). k-means++: the advantages of careful seeding. In *SODA'07*, New Orleans, Louisiana. ACM-SIAM, pp. 1027–1035.
- Ba J.L., Kiros J.R. and Hinton G.E. (2016). Layer normalization. arXiv preprint [arXiv:1607.06450](https://arxiv.org/abs/1607.06450).
- Bahdanau D., Cho K. and Bengio Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, May 7–9, 2015, San Diego, CA, USA. ICLR, pp. 1–15, Conference Track Proceedings, pp. 1–15, San Diego, CA, USA. ICLR.
- Beverungen G. and Kalita J.K. (2011). Evaluating methods for summarizing twitter posts. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM)*, Hong Kong. ACM, pp. 1–6.
- Bilal I.M., Wang B., Liakata M., Procter R. and Tsakalidis A. (2021). Evaluation of thematic coherence in microblogs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 6800–6814, Online.
- Bilal I.M., Wang B., Tsakalidis A., Nguyen D., Procter R. and Liakata M. (2022). Template-based abstractive microblog opinion summarization. *Transactions of the Association for Computational Linguistics* **10**, 1229–1248.

- Campello R.J.G.B., Moulavi D. and Sander J.** (2013). Density-based clustering based on hierarchical density estimates. In Pei J., Tseng V. S., Cao L., Motoda H. and Xu G. (eds), *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer, pp. 160–172.
- Chakma K., Das A. and Debbarma S.** (2019). Deep semantic role labeling for tweets using 5w1h: who, what, when, where, why and how. *Computación y Sistemas* 23(3), 751–763.
- Chen E., Lerman K. and Ferrara E.** (2020). Tracking social media discourse about the covid-19 pandemic: development of a public coronavirus twitter data set. *JMIR Public Health and Surveillance* 6(2), e19273.
- Cheng J. and Lapata M.** (2016). Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany. Association for Computational Linguistics, pp. 484–494.
- Devlin J., Chang M.-W., Lee K. and Toutanova K.** (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 4171–4186.
- Doğan E. and Kaya B.** (2019). Text summarization in social networks by using deep learning. In *2019 1st International Informatics and Software Engineering Conference (UBMYK)*, Ankara, Turkey. IEEE, pp. 1–5.
- Erkan G. and Radev D.R.** (2004). Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22(1), 457–479.
- Ester M., Kriegel H.-P., Sander J. and Xu X.** (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96*, Portland, Oregon. AAAI Press, pp. 226–231.
- Frey B.J. and Dueck D.** (2007). Clustering by passing messages between data points. *Science* 315(5814), 972–976.
- Gong Y. and Liu X.** (2001). Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'01*, New York, NY, USA. Association for Computing Machinery, pp. 19–25.
- 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* 44(4), 443–463.
- He R., Zhao L. and Liu H.** (2020). TWEETSUM: event oriented social summarization dataset. In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online), pp. 5731–5736. International Committee on Computational Linguistics.
- He Z., Chen C., Bu J., Wang C., Zhang L., Cai D. and He X.** (2012). Document summarization based on data reconstruction. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, Sheraton Centre, Toronto. AAAI Press, pp. 620–626.
- Hermann K. M., Kocisky T., Grefenstette E., Espeholt L., Kay W., Suleyman M. and Blunsom P.** (2015). Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, 28, pp. 1693–1701.
- Hu B., Chen Q. and Zhu F.** (2015). LCSTS: a large scale Chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal. Association for Computational Linguistics, pp. 1967–1972.
- Inouye D.I. and Kalita J.K.** (2011). Comparing twitter summarization algorithms for multiple post summaries. In *IEEE International Conference on Social Computing (SocialCom)*, pp. 298–306.
- Jin X. and Han J.** (2017). K-means clustering. In *Encyclopedia of Machine Learning and Data Mining*. Boston, MA: Springer, pp. 695–697.
- Karimi S., Shakery A. and Verma R.M.** (2023). Enhancement of twitter event detection using news streams. *Natural Language Engineering* 29(2), 181–200.
- Kaufmann M.** (2010). Syntactic normalization of twitter messages. In *Final Paper. IIT Kharagpur. ICON*, pp. 1–7.
- Lewis M., Liu Y., Goyal N., Ghazvininejad M., Mohamed A., Levy O., Stoyanov V. and Zettlemoyer L.** (2020). BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 7871–7880, Online.
- Liang x., Wu L., Li J., Wang Y., Meng Q., Qin T., Chen W., Zhang M. and Liu T.-Y.** (2021). R-drop: regularized dropout for neural networks. In Ranzato M., Beygelzimer A., Dauphin Y., Liang P. and Vaughan J.W. (eds), *Advances in Neural Information Processing Systems*, 34. Curran Associates, Inc., pp. 10890–10905.
- Lin C.-Y.** (2004). ROUGE: a package for automatic evaluation of summaries. In *Text Summarization Branches Out*, Barcelona, Spain. Association for Computational Linguistics, pp. 74–81.
- Liu Y. and Lapata M.** (2019). Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics, pp. 3730–3740.
- Liu Y. and Liu P.** (2021). SimCLS: a simple framework for contrastive learning of abstractive summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, pp. 1065–1072, Online.
- Luhn H.P.** (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development* 2(2), 159–165.

- Manuel J. and Moreno T.** (2014). *Automatic Text Summarization*. 1st edn. Hoboken, NJ: Wiley.
- Mihalcea R. and Tarau P.** (2004). TextRank: bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain. Association for Computational Linguistics, pp. 404–411.
- Mikolov T., Sutskever I., Chen K., Corrado G. and Dean J.** (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13*, Red Hook, NY, USA. Curran Associates Inc., pp. 3111–3119.
- Moirangthem D.S. and Lee M.** (2020). Abstractive summarization of long texts by representing multiple compositionality with temporal hierarchical pointer generator network. *Neural Networks* **124**, 1–11.
- Nallapati R., Zhou B., dos Santos C., Güçleüre C. and Xiang B.** (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, Berlin, Germany. Association for Computational Linguistics, pp. 280–290.
- Nenkova A. and Vanderwende L.** (2005). The impact of frequency on summarization. In MSR-TR-2005. Redmond, Washington: Microsoft Research, MSR, pp. 1–9.
- Niu J., Zhao Q., Wang L., Chen H., Atiquzzaman M. and Peng F.** (2016). Onses: a novel online short text summarization based on bm25 and neural network. In *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, D.C., USA. IEEE, pp. 1–6.
- Olabisi O., Hudson A., Jetter A. and Agrawal A.** (2022). Analyzing the dialect diversity in multi-document summaries. In *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea. International Committee on Computational Linguistics, pp. 6208–6221.
- Perez-Tellez F., Pinto D., Cardiff J. and Rosso P.** (2010). On the difficulty of clustering company tweets. In *Proceedings of the 2nd International Workshop on Search and Mining User-Generated Contents, SMUC'10*, New York, NY, USA. Association for Computing Machinery, pp. 95–102.
- Radev D.R., Hovy E. and McKeown K.** (2002). Introduction to the special issue on summarization. *Computational Linguistics* **28**(4), 399–408.
- Rajaby Faghihi H., Alhafni B., Zhang K., Ran S., Tetreault J. and Jaimes A.** (2022). Crisistlsum: a benchmark for local crisis event timeline extraction and summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Online and Abu Dhabi, United Arab Emirates*. Association for Computational Linguistics.
- Reimers N. and Gurevych I.** (2019). Sentence-BERT: sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics, pp. 3982–3992.
- Rousseeuw P.J.** (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* **20**, 53–65.
- Rudra K., Ghosh S., Ganguly N., Goyal P. and Ghosh S.** (2015). Extracting situational information from microblogs during disaster events: a classification-summarization approach. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM'15*, New York, NY, USA. Association for Computing Machinery, pp. 583–592.
- See A., Liu P. and Manning C.** (2017). Get to the point: summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics, pp. 1073–1083.
- Sharifi B., Hutton M.-A. and Kalita J.** (2010). Summarizing microblogs automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT'10*, USA. Association for Computational Linguistics, pp. 685–688.
- Shen X., Lam W., Ma S. and Wang H.** (2023). Joint learning of text alignment and abstractive summarization for long documents via unbalanced optimal transport. *Natural Language Engineering* **30**(3), 525–553.
- Sherstinsky A.** (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena* **404**, 132306.
- Shou L., Wang Z., Chen K. and Chen G.** (2013). Sumbl: continuous summarization of evolving tweet streams. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'13*, New York, NY, USA. Association for Computing Machinery, pp. 533–542.
- Sutskever I., Vinyals O. and Le Q.V.** (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. Montreal, Canada: Curran Associates, Inc., pp. 3104–3112.
- Tibshirani R., Walther G. and Hastie T.** (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **63**(2), 411–423.
- Tu Z., Lu Z., Liu Y., Liu X. and Li H.** (2016). Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany. Association for Computational Linguistics, pp. 76–85.
- Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L. and Polosukhin I.** (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*. abs/1706.03762:5998–6008.

- Vinyals O., Fortunato M. and Jaitly N.** (2015). Pointer networks. In Cortes C., Lawrence N.D., Lee D.D., Sugiyama M. and Garnett R. (eds), *Advances in Neural Information Processing Systems* 28. Montreal, Canada: Curran Associates, Inc., pp. 2692–2700.
- Wang Z., Shou L., Chen K., Chen G. and Mehrotra S.** (2015). On summarization and timeline generation for evolutionary tweet streams. *IEEE Transactions on Knowledge and Data Engineering* 27(5), 1301–1315.
- Yang X., Ghoting A., Ruan Y. and Srinivasan P.** (2012). A framework for summarizing and analyzing twitter feeds. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'12*, New York, NY, USA. Association for Computing Machinery, pp. 370–378.
- Zepeda-Mendoza M.L. and Resendis-Antonio O.** (2013). Hierarchical agglomerative clustering. In *Encyclopedia of Systems Biology*. New York, NY: Springer New York, pp. 886–887.
- Zhao Y. and Karypis G.** (2001). Criterion functions for document clustering: experiments and analysis. Technical report, University of Minnesota Digital Conservancy.
- Zhou Q., Yang N., Wei F., Huang S., Zhou M. and Zhao T.** (2018). Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics, pp. 654–663.