Multimedia Tools and Applications
https://doi.org/10.1007/s11042-022-12006-y

Instant message summarization with Emoji unicode characterset support

Darren Ramsook[1] · **Patrick Hosein**[2] · **Nicholas Hosein**[3]

Received: 10 March 2021 / Revised: 6 August 2021 / Accepted: 3 January 2022 /
Published online: 29 January 2022

Abstract
Summarization techniques have traditionally achieved good performance results when summarizing sentences and documents. However, their application to instant messaging notifications have not been thoroughly examined. This research outlines a model for summarizing instant messages, through the use of text and the Emoji Unicode Characterset, for applications where screen space is limited (e.g, in smartwatches and smart bracelets). The proposed model uses a Greedy N-gram token replacement method. This method produced high quality results and was evaluated using human participants. We found that there is a decrease in the time taken to read the summarized message when compared with the original message.

Keywords Textual summarization · Instant message · Emoji · Machine learning · Natural language processing

1 Introduction

Natural Language Processing (NLP) is one of the main computational fields in modern day artificial intelligence. It is an area supported by Data Science, Computer Science and Linguistics and mainly seeks to close the gap between computing systems and human language. NLP has made its way to most digital goods and services in modern day applications. NLP

✉ Darren Ramsook
darrenramsook@outlook.com

Patrick Hosein
patrick.hosein@sta.uwi.edu

Nicholas Hosein
nhosein@ucdavis.edu

[1] Department of Electronic and Electrical Engineering, Trinity College, Dublin, Ireland

[2] Department of Computer Science, The University of the West Indies, St. Augustine, Trinidad

[3] Department of Electrical and Computer Engineering, University of California, Davis, CA, USA

techniques are used in Search Engine systems [4], Machine Translation [11], Speech Recognition [6, 13] and Sentiment Analysis [10, 14] to name a few. The ongoing symbiotic trend with regards to the increased electronic information consumption and the production of data by end-users using these electronic systems means that the field of NLP is a thriving area of continued research and development with new resources being created daily.

Textual data is constantly being shared between users or stored in the cloud depending on the application. Every day 5.5 billion cellular text messages are exchanged [3], 500 million tweets are created per day as of 2013 [21] and the average instant messaging service consumer sends 72 messages per day [20].

An important and challenging area of NLP is the use of textual summarization. The end goal of summarization techniques is, given an original sequence of words/characters (typically a sentence or paragraph), produce a condensed version of the original sequence that satisfies a set of linguistic characteristics. A similarity score between the original and shortened sequence can be used to quantify performance. Textual summarization can be split into two main categories, Extractive and Abstractive Summarization. Both types of summarization were key points in this research and will be explored thoroughly in a later section. Textual summarization is mainly implemented in the fields of customer review mining [7], headline generation [2, 18] and extraction of the main points of large documents like blogs or research papers [1].

Simultaneously, with the rise of increased use of social media services which are heavily dependent on text, there has been an increase in the use of a set of unicode characters classified as Emojis. Emojis are pictographic representations, that are typically displayed as one character, of a given phrase or word and can be interpreted differently given the context of usage. Emojis represent a wide array of items, from traditional emoticons to locations. It was found that Emojis infer deeper meanings into conversations [9] in addition to emotions.

Notification bars on mobile phones typically only convey part of a received message. The same can be said about smart watches. If a lengthy message is received, the message is normally split and concatenated with "..." depending on the screen width of the device. This can be viewed as tedious for the end user as it requires them to further expand or open the message fully to view its contents. This problem can be exacerbated as this problem is repeated throughout the day. This paper explores a solution to the above problem by using textual summarization techniques to condense incoming text messages into shorter versions through the use of emojis.

The remainder of this paper is organized as follows. Section 2 serves as a high level definition of the problem, Section 3 looks at existing research, firstly around the Unicode Emoji Characterset in modern day communication and then existing Summarization techniques, Section 4 looks at the proposed method with an Example, Section 5 looks at the results obtained with subjective human testing and Section 6 discusses the results.

2 Problem definition

Consider a scenario where an end user of a mobile instant messaging service, such as WhatsApp or iMessage, is in a high focus situation such as driving an automobile or attending a university lecture. In such scenarios, the time spent on the task at hand by the end user is critical. This end user may also have a smart wearable such as a bracelet or smartwatch that has a digital display to which message notifications are sent.

Now consider that a message is sent to the end-user from one of their contacts. The end-user, upon receiving the notification, will have to read the notification on their smart wearable device and then make a personal judgement call on whether or not they have to open the full message on their mobile phone. This personal judgement call is assumed to be dependent on two main factors: the contact sending the message and the amount of the message that is visible on the screen of the smart wearable device. Typically the length of the message may be too long to be condensed into the display of the wearable and so the end user will default to opening the message fully to read and interpret the message. If the message is not important, this process is inefficient in terms of the high focus task at hand. Had the end-user known the context of the message, they would have opened the entire message at a later time. This then calls for a method that minimizes the read time and interpretation time for a given message.

To express this mathematically, let an incoming message be denoted by a vector $\bar{X}$ so that each word is a element token separated by a white space character. We assume that the vector $\bar{X}$ is of size N so that $\bar{X} = [X_0, X_1, \ldots, X_N]$. It is assumed that read time and interpret time is proportional to the length of the message and the context of the message. Therefore, the problem then becomes to create a summarized representation of the original sentence, $\bar{Y}$, so that some length and similarity constraints are considered. The vector $\bar{Y}$ of size M is given by $\bar{Y} = [Y_0, Y_1, \ldots, Y_M]$ The individual tokens of $\bar{X}$ and $\bar{Y}$ can be represented using either a one hot encoding method of the entire vocabulary, or a N-dimensional context vector of that word. Regardless of the representation, we would like $M \ll N$. We would like to maximize the similarity between both sentences, $\mathsf{Sim}(\bar{X}, \bar{Y})$, so that the original context is not lost in the summarized version. In addition to a reduction in the required screen space, the decreased message length will lead to decreased reading and interpretation times.

3 Previous research

3.1 Textual summarization

The research area of textual summarization has been thriving in recent years due to the increased use of social media networks and other internet based services. The area of summarization can be split into two main areas, Extractive and Abstractive summarization. Extractive summarization (ES) is the condensing of an original text sequence so that the summary consists only of extracted content of the original text [22]. It is any means of extracting certain words, phrases or paragraphs from the original text. Early techniques of ES involve assigning certain scores associated with each input sentence or document. Extractive Summarization methods can also be divided into unsupervised and supervised methods. Some common word level features [12] that are used in summarization are:

Keyword Features: These are used to identify important areas of a sentence. Traditionally portions of text that use keywords are most likely to be included in the final summary.
Title Word Features: These utilize the notion that phrases or sentences in a document that contains the title words have a higher chance to contribute to the summary.
Cue phrase features: Cue phrases (e.g., “because”, “summary”) are used to extract as it is assumed that these phrases indicate overall document flow.

While research into summarization with Emojis are in its infancy, there is continuous work into textual based summarization. Previous work by [16] proposed an ES method that

uses frequency and position features of the words along with the triangular membership function to generate scores. The sentences with the higher scores are included in the summary. An ES method proposed by [8] uses word vector embedding of the sentences of the abstractly summarized and original text in training a neural network. The neural network assigns a score to each sentence of the original text and then a selection of the highest scoring sentences are used in creating an Extractive summary.

3.2 Abstractive summarization

Abstractive summarization (AS) is a technique in which a summary is generated through novel words, phrases or sentences rather then extracting specific parts of sentences as in Extractive summarization. This generation is done by creating an internal representation of a source document, by analysing features from the source document through deep analysis and by reasoning about the semantic and context.

Current advances of Abstractive summarization are due to Deep Neural Networks. An LSTM-CNN (Long Short Term-Convolutional Neural Network) approach was proposed by [17]. This network uses a CNN to encode words in a sentence and then uses a LSTM network to decode the sequence of encodings into a summary. Promising results was demonstrated through the use of generative adversarial networks (GANs) by [15] which uses an LSTM encoder-decoder model with attention.

Generation of long and short Abstractive summaries were also evaluated by [23]. This study used various model architectures as well as human judgement in evaluating Abstractive summaries. They found that traditional summarization scores do not correlate well with human judgement. While the models achieve higher ROGUE (Recall-Oriented Understudy for Gisting Evaluation) scores, they do not surpass the baseline when evaluated by humans.

4 Proposed method of summarization

This section details the proposed method of summarizing instant messages using a method that is based on Dense Embedding Representation of N-Gram words and its relation to Emojis and their keywords.

4.1 *N*-gram token replacement method

The overall flow of this method is defined in Fig. 1. This process outlines some key steps to be followed. Given an input sentence, the sentence is split up into tokens where each word is a token. These tokens are then parsed and extracted based on if they are critical in adding context to the sentence. The set of tokens is then used to create a set of N-gram representations of the extracted tokens. Each of these tokens are then compared to another collection of tokens referring to a set of Emojis and the Emoji with the minimal Mean Squared Error (MSE) is selected. This entire process is discussed in more detail later. For the remainder of this paper, this method will be referred to as Method 1.

Let $\bar{X}$ be an input sentence to be summarized, with an example shown below. This example will be used as a basis for explaining the overall summarization process.

$$\bar{X} = \text{“}can\ we\ go\ to\ the\ beach\ take\ my\ car\text{”} \tag{1}$$

This input sentence is then converted to a set of tokens defined by the whitespace character. Let this be represented by $\bar{X}_{Token}$ as follows:

$$\bar{X}_{\text{Token}} = [\text{“}can''\text{, “}we''\text{, “}go''\text{, “}to''\text{, “}the''\text{, “}beach''\text{, “}take''\text{, “}my''\text{, “}car''] \tag{2}$$

Upon completing tokenization, the first core problem can be seen. It involves a method of extracting the important words from a given sentence. To do this, a Part-of-Speech Tagger was used.

A pre-trained greedy average perceptron tagger was used as the part-of-speech tagger at this stage of the model. The selected tagger, from the NLTK package, was trained on a large corpus of data from the Wall Street Journal dataset. While the NLTK package is more computationally expensive, it is more accurate than other publicly available pre-trained perceptron taggers. We can now apply the average perceptron tagger to the sequence of tokens $\bar{X}_{\text{Token}}$. For each token now there should be a corresponding part of speech associated with that token. Let this be represented by $\bar{X}_{\text{POS}}$. For each element of $\bar{X}_{\text{POS}}$, we now have the original word at index 0 and the tagged part of speech at the index 1. The complete tagset of words can be found in the NLTK documentation.

$$\begin{aligned}\bar{X}_{\text{POS}} = [&(\text{‘}can'\text{, ‘}MD'), (\text{‘}we'\text{, ‘}PRP'), (\text{‘}go'\text{, ‘}VB'), (\text{‘}to'\text{, ‘}TO'),\\ &(\text{‘}the'\text{, ‘}DT'), (\text{‘}beach'\text{, ‘}NN'), (\text{‘}take'\text{, ‘}VB'),\\ &(\text{‘}my'\text{, ‘}PRP'), (\text{‘}car'\text{, ‘}NN')]\end{aligned} \tag{3}$$

We then assume that specific words with appropriate parts of speech convey the core meaning of the original sentence. The words selected are all forms of Nouns, all forms of Verbs, all forms of Adjectives and all Adverbs. After examining the tags of $\bar{X}_{\text{POS}}$, words that do not meet this criteria are dropped from the sentence. Let the new representation be $\bar{X}_{\text{Ext}}$

$$\bar{X}_{\text{Ext}} = [\text{‘}go'\text{, ‘}beach'\text{, ‘}take'\text{, ‘}car'] \tag{4}$$

The next step is creating an equivalent unigram, bigram and trigram representation. This will be referred to as an N-gram representation from now on. This is done so that words with similar meaning that are next to each will have a chance to be represented together. An example of this would be considering the words “fast”, “car” as a single representation of “fast car”. Doing this then creates a collection of N-gram tokens as defined by Gram.

$$\begin{aligned}\text{Gram} = [&[(\text{‘}go'), (\text{‘}beach'), (\text{‘}take'), (\text{‘}car')], \text{– unigram representation}\\ &[(\text{‘}go'\text{, ‘}beach'), (\text{‘}beach'\text{, ‘}take'), (\text{‘}take'\text{, ‘}car')]\text{– bigram representation}\\ &[(\text{‘}go'\text{, ‘}beach'\text{, ‘}take'), (\text{‘}beach'\text{, ‘}take'\text{, ‘}car')]]\text{– trigram representation}\end{aligned} \tag{5}$$

We now have a collection of N-gram tokens, as defined by Gram. The problem then becomes creating a numerical representation of these tokens. To create this representation, a pre-trained model of Word2Vec [5] was used. Let the Word2Vec representation of a given word w be represented by:

$$\text{Vector Representaion} = word2Vec[w] \tag{6}$$

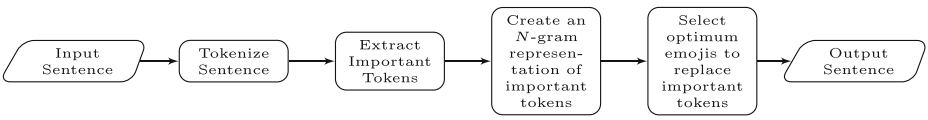

Fig. 1 Flow diagram of the N-gram token replacement Method

Table 1 Example Emoji and Keywords before word2Vec Representation

Emoji	Keywords
U+1F600	keyword1, keyword2
U+1F603	keyword3, keyword4, keyword5
U+1F604	keyword6, keyword1, keyword7

A unique result of using word2Vec is that the vector representation of different words can be algebraically formulated together to generate a unique result. We can then exploit this and create unique vector representations for each token within each N-gram representation. Therefore the function tokenVec is applied to each index of each N-gram representation.

$$\text{tokenVec}(token_{index}) = \sum_{all\ j} word2Vec[w_j] \tag{7}$$

where j is the jth element in the token. Applying this operation to the first element of the unigram representation would be equivalent to: $word2Vec[`go']$. while the first element of the bigram representation would be:

$$word2Vec[`go'] + word2Vec[`beach'] \tag{8}$$

while the first element of the trigram representation would be

$$word2Vec[`go'] + word2Vec[`beach'] + word2Vec[`take'] \tag{9}$$

We now have complete vector representations for all N-gram tokens. The next step would be creating a vector representation for the Emoji characterset so that algebraic methods can be applied with the vector representations of the tokens and the Emoji vector representation.

4.2 Vector representation of Emojis

A Kaggle dataset [19], by the name of EmojiNet, consists of keywords and labels for each Emoji other than the ones provided by the official Unicode website. The scraped dataset from the Unicode website and this EmojiNet dataset was joined on the basis of the Emoji Unicode. The keywords from EmojiNet and the description from the Unicode website were joined and a definite set of words for each Emoji representation was created. All duplicates within these words were removed so that each emoji had a concise set of words describing them. Table 1 shows an example of this merged dataset with the each Emoji having a set of keywords.

Each of these set of keywords was then converted to word2Vec representation so that each Emoji has an array of 300 dimensions as its rows and the number of keywords as its columns for that given emoji. Table 2 shows the same emojis but with a vectorized keyword representation instead. This method of representation was used for the remainder of this study.

We now have a collection of the vectorized N-gram representations and also a collection of vectors describing emojis as detailed in Table 2. For each vectorized token in the N-gram representations, the mean squared error between that token and each keyword from the vectorized keywords is calculated and averaged over the entire array for that emoji. This process is done for all emojis. The resultant array is then sorted in ascending order and only the $N/2$ remaining errors are averaged to create an error score for that emoji. This process is illustrated in Algorithm 1

Table 2 Example Emoji and Keywords after word2Vec Representation

Emoji	Vectorized Keywords
U+1F600	Array of Dimension 300x2
U+1F603	Array of Dimension 300x3
U+1F604	Array of Dimension 300x3

Algorithm 1 Error calculation.

Result: err
VectorizedToken;
N = Number of Keywords in Emoji;
vectorizedKeywords = word2Vec represenation of keywords;
errVector = empty array;
for *i in range(0,N)* **do**
  errVector.append(mean(square(VectorizedToker - vectorizedKeywords[:,i])))
end
errVector = sortAscending(errVector);
err = mean(errVector[:,N/2]);
return err;

The emoji with the minimum error for that corresponding keyword is then chosen as the prospective replacement for that token. We then have a collection of error values as shown by $\hat{E}$, a collection of prospective emoji representations (Fig. 2), and a collection of previous N-gram representations, given by $Gram$. $\hat{E}$ is then updated again by averaging the error value by the number of words in each token used to create that error and given in (10).

$$\begin{aligned}\hat{E} = &[[(0.0188), (0.182), (0.0353), (0.0)], \text{– unigram errors}\\ &[(0.01765), (0.183), (0.0082)]\text{– bigram errors}\\ &[(0.0232), (0.175)]]\text{– trigram errors}\end{aligned} \tag{10}$$

The problem then becomes replacing tokens with the prospective emoji based on the error values. In this scenario we utilize a greedy approach, where $\hat{E}$ is scanned and the the token with the minimal error value is replaced with the prospective emoji. All tokens are

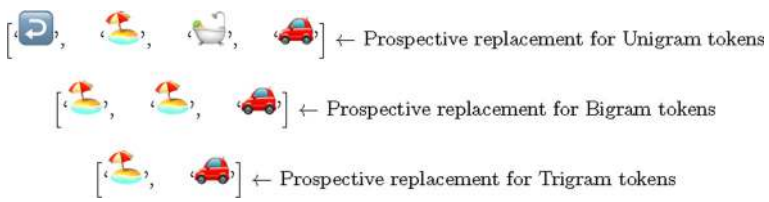

Fig. 2 Prospective Emojis based for the N-gram Model

then updated to exclude any tokens that are related to the replaced token. Therefore $Gram$ and $\hat{E}$ will then become:

$$\begin{aligned} \text{Gram} = & [[('go'), ('beach'), ('take')], \text{– unigram tokens} \\ & [('go','beach'), ('beach','take')]\text{– bigram tokens} \\ & [('go','beach','take')]]\text{– trigram tokens} \end{aligned} \tag{11}$$

$$\begin{aligned} \hat{E} = & [[(0.0188), (0.182), (0.0353)], \text{– unigram errors} \\ & [(0.01765), (0.183)]\text{– bigram errors} \\ & [(0.0232)]]\text{– trigram errors} \end{aligned} \tag{12}$$

This process is repeated until some threshold is met so that a certain proportion ρ of the sentence is converted to an Emoji representation. The replaced tokens and non replaced tokens are then reconstructed to form a final sentence. Figure 3 shows this final reconstruction, with $\rho = 0.5$, meaning that only half of the tokens will be converted into an emoji representation.

5 Testing

The proposed model does not have a set of labelled output to compare against and so a subjective test using Human participants was used. The key values measured are the time taken to read a summarized message and the time taken to interpret the context of the message.

A sample of 10 people were selected to look at a combination of 10 sentences. The 10 sentences consisted of 5 un-summarized messages and 5 summarized messages using the proposed method. The times taken to read and interpret each sentence was then measured. Let read time be denoted by T_r and interpret time be T_i. The detailed methodology was as follows:

1. The collection of 10 sentences is shuffled.
2. The participant simultaneously starts the timer and begins reading the sentence.
3. The participant notes the elapsed time upon completion of reading the sentence.
4. The participant notes the elapsed time upon successfully interpreting the context of the sentence.
5. Repeat the above process for each of the participants.

6 Discussion of results and conclusions

The times across all participants for reading and summarizing the sentences were averaged based on the sentences. This is shown in Table 3. The average read time for the unsummarized sentences was 143.8 ms. It was noted that for the full unsummarized sentences, all participants noted that the read time and interpretation time are the same.

The read times of the summarized sentences had an average of 64.2ms. The average interpretation times had an average of 108.4ms. A Wilcoxon signed-rank test was used to

Fig. 3 Final summarized version of example

Table 3 Summary of Read Time and Interpretation Time

Sentence	Read Time/ms	Interpretation Time/ms
UnSummarized1	175	Same as Read Time
UnSummarized2	134	Same as Read Time
UnSummarized3	115	Same as Read Time
UnSummarized4	141	Same as Read Time
UnSummarized5	154	Same as Read Time
Summarized1	54	119
Summarized2	79	110
Summarized3	65	120
Summarized4	59	70
Summarized5	64	123

compare the read times for unsummarized and summarized sentences as well as to compare the interpret times for these sentences. Based on the outcomes of this statistical test, it was found that the difference in read times are statistically significant. However, it was found that the difference in the Interpretation times between unsummarized and summarized sentences were not statistically significant.

This research explored the concept of textual summarizing utilizing the Emoji Unicode Characterset and its application to instant messaging services. The problem was defined as shortening an input vector to a summarized representation so that the length of the summarized representation is less than the input vector, and that the similarity between both input and output are maximized.

A core concept of the proposed method was the vector embedding representation of Emojis. It was found that representing the Emojis as an algebraic sum of their respective keywords was quite efficient for this task. The keywords were extracted through web scraping the official Unicode website and using the EmojiNet database. A part of speech tagger was also used in the form of a pre-trained greedy average perceptron tagger, from the NLTK package. The method of selecting emoji replacement for N-gram tokens was a greedy approach.

Human testing on the proposed model was also conducted, with 10 participants evaluating 10 sentences, 5 of which are summarized and 5 unsummarized. Their read and interpret times for each sentence were noted and aggregated. It was found that the read time for sentences summarized less than the read time of unsummarized sentences. This was further confirmed through the use of a Wilcoxon rank sum test.

The interpretation times for both the unsummarized and summarized sentences were also examined. It was found that the interpretation times, while being less on average for the summarized version, were not statistically different. It was found that the true location shift is equal to zero at an $\alpha = 0.05$ level, i.e. the medians of the interpretation time for the unsummarized and summarized sentences are similar.

Future work in this area should focus on exploring various optimization methods of selecting Emoji replacements for N-gram tokens for the proposed model. Investigation of a global approach may have better results in terms of minimizing the error but may drive computational time upwards. The difference in interpretation times between the unsummarized and summarized sentences should also be further investigated.

Funding No funds, grants, or other support was received.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Altmami NI, Menai MEB (2020) Automatic summarization of scientific articles. a survey. J King Saud Univ - Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2020.04.020
2. Colmenares CA, Litvak M, Mantrach A, Silvestri F (2015) Heads: Headline generation as sequence prediction using an abstract feature-rich space. Proceedings of the 2015 conference of the North American Chapter of the association for computational linguistics: Human Language Technologies. https://doi.org/10.3115/v1/n15-1014
3. CTIA (2019) 2019 ctia annual survey highlights. https://www.ctia.org/news/2019-annual-survey-highlights
4. Devlin J, Chang MW, Lee K, Toutanova K (2018) BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805
5. Google (2013) Word2vec model. https://code.google.com/archive/p/word2vec/
6. Haridas AV, Marimuthu R, Sivakumar VG (2018) A critical review and analysis on techniques of speech recognition: The road ahead. Int J Knowl-based Intell Eng Syst 22(1):39–57. https://doi.org/10.3233/kes-180374
7. Hu M, Liu B (2004) Mining and summarizing customer reviews. Proceedings of the 2004 ACM SIGKDD international conference on knowledge discovery and data mining - KDD '04. https://doi.org/10.1145/1014052.1014073
8. Jain A, Bhatia D, Thakur MK (2017) Extractive text summarization using word vector embedding. In: 2017 International conference on machine learning and data science (MLDS), pp 51–55. https://doi.org/10.1109/MLDS.2017.12
9. Kelly R, Watts L (2015) Characterising the inventive appropriation of emoji as relationally meaningful in mediated close personal relationships. In: Experiences of technology appropriation: unanticipated users, usage, circumstances, and design, experiences of technology appropriation: unanticipated users, usage, circumstances, and design; Conference date: 20-09-2015 Through 20-09-2015
10. Liu B (2012) Sentiment analysis and opinion mining. Synth Lect Hum Lang Technol 5(1):1–167. https://doi.org/10.2200/s00416ed1v01y201204hlt016
11. Madankar M, Chandak M, Chavhan N (2016) Information retrieval system and machine translation: a review. Procedia Comput Sci 78:845–850. https://doi.org/10.1016/j.procs.2016.02.071
12. Moratanch N, Chitrakala S (2017) A survey on extractive text summarization. In: 2017 International conference on computer, communication and signal processing (ICCCSP), pp 1–6. https://doi.org/10.1109/ICCCSP.2017.7944061
13. Nassif AB, Shahin I, Attili I, Azzeh M, Shaalan K (2019) Speech recognition using deep neural networks: a systematic review. IEEE Access 7:19143–19165
14. Pang B, Lee L (2008) Opinion mining and sentiment analysis. Found Trends® Inf Retrieval 2(1–2):1–135. https://doi.org/10.1561/1500000011
15. Rekabdar B, Mousas C, Gupta B (2019) Generative adversarial network with policy gradient for text summarization. In: 2019 IEEE 13th international conference on semantic computing (ICSC), pp 204–207. https://doi.org/10.1109/ICOSC.2019.8665583
16. Sharaff A, Khaire AS, Sharma D (2019) Analysing fuzzy based approach for extractive text summarization. In: 2019 International conference on intelligent computing and control systems (ICCS), pp 906–910. https://doi.org/10.1109/ICCS45141.2019.9065722
17. Song S, Huang H, Ruan T (2019) Abstractive text summarization using lstm-cnn based deep learning. Multimed Tools Appl 78(1):857–875
18. Tan J, Wan X, Xiao J (2017) From neural sentence summarization to headline generation. A coarse-to-fine approach. Proceedings of the twenty-sixth international joint conference on artificial intelligence. https://doi.org/10.24963/ijcai.2017/574
19. Tatman R (2017) Emojinet. https://www.kaggle.com/rtatman/emojinet
20. Twilio (2016) Global mobile messaging consumer report 2016. Twilio. https://www.twilio.com/learn/commerce-communications/how-consumers-use-messaging

21. Twitter (2016) New tweets per second record, and how!. https://blog.twitter.com/engineering/en_us/a/2013/new-tweets-per-second-record-and-how.html
22. Widyassari AP, Rustad S, Shidik GF, Noersasongko E, Syukur A, Affandy A, Setiadi DRIM (2020) Review of automatic text summarization techniques & methods. J King Saud Univ- Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2020.05.006, http://www.sciencedirect.com/science/article/pii/S1319157820303712
23. Zhang S, Celikyilmaz A, Gao J, Bansal M (2020) Emailsum: Abstractive email thread summarization. In: Proceedings of the 59th annual meeting of the association for computational linguistics