Multimedia Tools and Applications (2021) 80:35239–35266
https://doi.org/10.1007/s11042-020-10082-6

A survey of pre-processing techniques to improve short-text quality: a case study on hate speech detection on twitter

Usman Naseem[1] · **Imran Razzak**[2] · **Peter W. Eklund**[2]

Received: 28 April 2020 / Revised: 17 August 2020 / Accepted: 13 October 2020 /
Published online: 4 November 2020

Abstract

Pre-processing plays an essential role in disambiguating the meaning of short-texts, not only in applications that classify short-texts but also for clustering and anomaly detection. Pre-processing can have a considerable impact on overall system performance; however, it is less explored in the literature in comparison to feature extraction and classification. This paper analyzes twelve different pre-processing techniques on three pre-classified Twitter datasets on hate speech and observes their impact on the classification tasks they support. It also proposes a systematic approach to text pre-processing to apply different pre-processing techniques in order to retain features without information loss. In this paper, two different word-level feature extraction models are used, and the performance of the proposed package is compared with state-of-the-art methods. To validate gains in performance, both traditional and deep learning classifiers are used. The experimental results suggest that some pre-processing techniques impact negatively on performance, and these are identified, along with the best performing combination of pre-processing techniques.

Keywords Natural language processing · Text pre-processing · Tweet classification · Machine learning

1 Introduction

Social media platforms play a more important role in global events than ever before. Analysis of information shared on social media platforms, especially Twitter, has become a

✉ Usman Naseem
usman.naseem@sydney.edu.au

Imran Razzak
imran.razzak@deakin.edu.au

Peter W. Eklund
peter.eklund@deakin.edu.au

[1] University of Sydney, Sydney, Australia

[2] Deakin University, Geelong, Australia

significant focus for researchers in recent years. Millions of Twitter users share their opinion and views on various topics: political debate, the stock market, products, companies and so on. These opinions and views can be used to improve services, develop marketing strategies, to observe user behaviours, to anticipate emerging trends and even to identify important events [33]. Aberrant behaviour also needs to be tracked, monitored and eliminated and in this paper, the classification of "hate speech"[1] is the focus of attention.

Twitter messages are restricted to 140 characters, so the language used on Twitter is normalised to this limitation, i.e. unstructured, and at times very informal. Although many different pre-processing techniques have been applied to text classification tasks, the impact of pre-processing techniques alone, the different combinations of pre-processors, and the sequence in which they are applied, has not been systematically studied. In this article, a study of different pre-processing techniques is presented, and the results are insights into the understanding of the appropriate selection and application of pre-processing on Twitter data.

Pre-processing tweets is the process of transforming them for further tasks such as event identification, fake information detection, sentiment valance etc. Generally, people follow their own informal language rules on social media. As such, every Twitter user writes in their own style; abbreviations, non-standard punctuation and incorrect spellings are used. Tweets contain emoticons and emojis to express nuance, sentiment and opinions. Tweets often contain slang and acronyms, and they embed URLs, hashtags and user mentions. These language imperfections introduce noise that can degrade automated classification performance. According to a study conducted by Fayyad et. al. [9], noise in Twitter datasets may rise to as much as 40%, and this can impact significantly on classification performance. This being the case, one of the major challenges in dealing with noise, and the absence of structure in tweets, is by applying appropriate text pre-processing techniques in such a way that pre-processing does not deteriorate, but rather enhances classification performance. The objective is to study and understand the effects of pre-processing techniques on classification tasks and propose a method that improves otherwise low-fidelity text to improve classification performance by applying the most suitable pre-processing steps in a systematic way. This research contributes to the practice of text analytics by quantifying: (1) the effectiveness of improving the quality of text via pre-processing systematically; (2) the extent to which performance is impacted negatively when an inappropriate combination of pre-processing techniques is selected; (3) a methodology for measuring the impact of pre-processing techniques. This research focuses on proposing methods to improve the low-quality text, which helps to learn better features for the text classification task. In short, the aim of this study is when given a tweet:

```
@UnitedAirlines Cooool I'm :) with servc!
You ROCKED #urgr8 http://ow.ly.Vlbf0
```

with unique but typical tweet characteristics: an unstructured and informal nature that represents at face value a low-fidelity text as input to the classifier, namely a sequence of tokens $t_x = (t_1, t_2, ...t_k)$, where x denotes the number of a tweet and k represents the number of tokens in a tweet to be classified as label y from a set of fixed labels $y_1, y_2, ...y_k$ at output. Whereas at the input to the learning algorithm is a set of training data, n hand-labeled tweets $(T_1, y_1), ..., (T_n, y_n)$ that yields a learned classifier $f(T)$ that predicts the label (class) y of a previously unseen tweet. In the example provided the classification task might be to

[1]Hate speech is defined by Cambridge Dictionary as "public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation".

determine whether the tweet above represents positive, neutral or negative sentiment about United Airlines.

With the aim of improving the quality of low-fidelity tweets, we first examine the impact of both common and advanced pre-processing techniques and explore techniques that perform well, and which degrade classification results. Then a systematic combination of different pre-processing techniques is presented which replaces emoticons with their associated meanings, replaces abbreviations and slang, corrects spelling, imposes word segmentation on hashtag text, and expands contractions in order to utilize the otherwise hidden features in the raw unstructured tweet.

The main aim of this paper is to analyze different pre-processing both singularly and in combination. We follow earlier studies [26, 29], conducted along with recent methods that have not been explored much by researchers, such as sentiment-aware tokenization (i.e. replacing emoticons with the words they symbolise), acronym expansion and slang substitution with words that convey sentiment, spelling correction, word segmentation of text within hashtags and negations with appositive words. These pre-processors are analyzed for their impact on the selection of feature and automated classification results. The study leads to an intelligent tweet processor (a systematic method of applying pre-processing techniques without useful information loss, which in turn assists to achieve better automated classification. Proposed method; (i) removing noise and normalizing low-quality tweets, (ii) replace abbreviations and acronyms with actual words, (iii) replace emoticons and emojis with their associated meanings, (iv) spell correct and perform word segmentation on strings used in hashtags etc. We have selected N-grams and word embeddings (Word2Vec [40] and GloVe [31]) for word representation methods and classification, both traditional and deep learning classifiers are used to this end. In order to generalize the performance, we have performed a comprehensive study through 10-fold validation on three benchmark Twitter datasets that deal with labelled examples of hate speech and abusive language [7, 11, 13].

The **key contributions** of this paper can be summarized as follows: (i) multiple pre-processing techniques are applied on Twitter data and analysis performed on their effect on an automated classification task, namely identifying hate speech; (ii) an Intelligent Tweet Processor (ITP) method is proposed, a systematic combination of tweet pre-processing techniques to minimise information loss; (iii) extensive experiments on three real-world benchmark datasets are conducted that show that automated classifier performance is considerably improved when the proposed set of preprocessing techniques are adopted in correct sequence. The rest of the paper is structured as follows, Section 2 summarizes the background and related work; Section 3 describes the methodology of this research. Section 4 presents the experimental results and Section 5 its conclusions.

2 Background and related work

Text datasets contain many words and characters that do not respond well to typical methods for feature extraction, i.e. stop-words, punctuation, incorrect spellings, slang expressions, etc and their presence can have an adverse effect on the performance of an automated classification task. In the discussion below, we briefly present the different pre-processing techniques followed by literature review, where researchers analyzed the effects of various text pre-processing techniques. Further detail on text pre-processing techniques can be found in Saeed et al. [32].

2.1 Text pre-processing methods

In this section, methods and techniques related to short text pre-processing are described.

Removal of noise, URLs, hashtags & user mentions Unwanted strings and Unicode, considered as a leftover from the crawling process, contribute to noise in the data. Also, almost all tweets posted by users contains URLs that reference additional information, user mentions (`@username`) and use the hashtag symbol (`#sometrendingtopic`) to associate their tweet with some particular topic, and these hashtags can also express the sentiment. These clues give extra information, useful for human beings, but do not provide any information to machines, and can be considered as noise which needs to be handled. Researchers have presented different techniques to handle this extra information provided by users, such as in the case of URLs; a study conducted by Agarwal et al. [1] replaced them with tags whereas in another study by Khan et al. [17] removed user mentions (`@username`).

Word segmentation is the process of separating the phrases/content/keywords used in a hashtag, i.e. `#sometrendingtopic` is segmented as three words *some* + *trending* + *topic*. This step can help in understanding and classifying the content of tweets easily for machines without any human intervention. As mentioned earlier, Twitter users use hashtags in almost all tweets to associate tweets with some particular trending topic.

Replacing emoticons and emojis Twitter users use many different emoticons and emojis such as `:-)`, `;-)`, `:-(` etc, to express sentiment and opinion. So it is also important to capture this useful information to classify tweets correctly. In a study conducted by Gimpel et al. [10], these expressions and emoticons were replaced with their associated word meanings, e.g. `:-)` is replaced with *happy* and `:-(` with *sad*.

Replacing abbreviation and slang Character length limitations in Twitter restrict the use of natural language and encourage online users to use abbreviations, short words and slang in their posts online. An abbreviation can be a shortened or an acronym of a word, e.g. `MIA` stands for *missing in action* or `gr8` for *great*, `ofc` for *of course* etc. Slang is also used as an informal way of expressing thoughts or meaning which is sometimes restricted to some particular group of people or context, and is considered as informal, e.g. `attwicted` means *addicted to Twitter* and `OMG` hardly means its literal expansion *oh my God* but rather more often is simply an expression of *surprise* or *emphasis*. It is therefore crucial to handle such informal insertions in the tweets by replacing them to their actual word meaning, this results in better automated classifier performance without information loss. In a study conducted by Kouloumpis et al. [20] abbreviations and slang were converted into word meanings that were then easily understood using standard text analytical tools.

Replacing elongated characters Social media users, often intentionally, use elongated words in which they purposely write or add more characters repeatedly for emphasis, e.g. `loooovvveee`, `greeeeat`. Thus, it is important to deal with these words and change them to their base word so that an automated classifier does not treat them as different words out-of-vocabulary (OOV). In our experiments, we replaced elongated words with their original base words. Saif et al. [23] conducted a study to detect and replace elongated words and found that replacement helps to improve the classification performance.

Incorrect spelling and grammar mistakes are commonly present in tweets. Correcting spelling and grammar helps reduce the same word meaning transcribed differently. `Textblob`[2] is one the library which can be used for this purpose. Norvig's spell correction[3] method is also widely used to correct and normalize spelling.

Expanding Contractions Contractions are short-form words more colloquially written and spoken than written but widely used by online users to reduce character counts. In contraction, an apostrophe is used in the place of one or more the missing letter(s). Because we want to standardize the text for machines to process more easily, contractions and shortened words are expanded to their original root or base words. For example, words contractions such `I'm, can't, don't` and more complex expressions, such as `she'd've`, are the contractions for the words *I am, can not, do not* and *she would have* respectively. In the study conducted by Boia et al. [6], contractions were replaced with their expanded variants. If contractions are not replaced, then the tokenization step will create tokens of the word `can't` into *can* and *t* etc.

Removing Punctuation Social media users use punctuation to express sentiment and emotion, easily understood as such by humans, however not as useful for automated classification of short texts. For this reason, the removal of punctuation is common practice in pre-processing text in preparation for automated classification tasks such as sentiment analysis. However, sometimes some punctuation symbols like `!` and `?` denote sentiment. Lin et al. [21] removed punctuation in their study whereas, an alternative approach replacing a question mark or exclamation with suitable tags, e.g. `!` can often be an expression of *surprise*, and this approach is studied in Balahur [4].

Removing numbers Text corpora usually contain unwanted numerals, also useful for human beings to understand, but often a challenge for machines to disambiguate. Zhao [14] removed numbers completely in his study. However, useful information is often lost in this way, for instance if we remove numbers before transforming slang and abbreviations into word meanings. For example, words like `2moro`, `4u`, `gr8` should be first converted to actual words, *tomorrow, for you* and *great*, and then we can proceed with this pre-processing step.

Folding to lower-casing This step helps avoid different variations of the same words determined by their case. This diversity of capitalization within the corpus can cause a problem during classification and degrade performance. Folding capital letters to lower case is the most common method to handle this issue in text data. This pre-possessing technique projects all tokens in a corpus under the single feature space but also causes problems in the interpretation of some words that are also common abbreviations, e.g. `US`. The word `US` once folded to lower-case could be either a pronoun and the country name as well, so converting to lower case can be problematic [8].

Removing stop-words Present in all texts are high frequency words non-critical, words that do little to help in the classification task or contribute much to semantic meaning. For this reason it is common to remove stop words before the feature selection step. Words

[2] https://github.com/sloria/TextBlob

[3] http://norvig.com/spell-correct.html

like `a, the, is, and, am, are, on` etc. There are different stop-word libraries available such as `NLTK`[4], `scikit-learn`[5] and `spaCy`[6].

Lemmatization is simiar to stemming, namely to cut down a word to its base. However, in lemmatization inflection of words are not just chopped off, but lexical knowledge is used to transform a word into its base form. There are many libraries available which help achieve lemmatization. A few of the more famous ones are `NLTK, gensim`[7], Stanford `CoreNLP`[8], `spaCy` and `TextBlob`[9].

2.2 Related work

Text pre-processing plays a significant role in text classification tasks. Many researchers in the past have made efforts to understand the effectiveness of different pre-processing techniques, and their contribution to automated text classification tasks. Bao et al. [5] showed the effect of pre-processing techniques on the Twitter sentiment classification task. The Stanford Twitter sentiment dataset was used in their experiments. Uni-gram and bi-grams features were fed to the Liblinear[10] classifier for the classification of positive and negative classes. They showed in their study that preservation of URL features, the transformation of negation (negated words) and normalization of repeated tokens had a positive effect on classification results, whereas lemmatization and stemming impact negatively on classification performance. Saeed et al. [34–36] applied other pre-processing techniques, such as duplicate tweet removal, folding to lower case, removal of special characters, tokenization to remove white spaces, and stop-word removal and finally removal of all words consisting of less than three letters. Singh and Kumari [41] showed the impact of pre-processing on a Twitter dataset full of abbreviations, slang and acronyms for the sentiment classification task. Their study showed the importance and significance of slang and the correction of spelling mistakes. A Support Vector Machine (SVM) classifier was used in their study to measure the role pre-processing played on the performance of sentiment classification. There have been some works on the use of big data platforms in Twitter data analysis in various application domains [2, 3, 24, 42, 43].

The importance of text pre-processing is also studied by Haddi et al. [12] on the movie review dataset[11]. Their experiments show that pre-processing techniques, such as the transformation of text including expansion of abbreviations and removal of stop-words, special characters and handling of negation with the prefix ‘NOT’, i.e. `unhappy` becomes *not happy*, along with stemming, can combined to significantly improve classification performance. An SVM classifier was used in their experiments. The study conducted by Usal and Serkan [46] explored the role of text pre-processing on two different languages for sentiment classification. They employed a SVM-classifier in their studies and showed that classification performance is improved by selecting the appropriate combination of different pre-processing techniques, such as removal of stop-words, lower-casing text, tokenization

[4] https://www.nltk.org/api/nltk.html
[5] https://github.com/scikit-learn/scikit-learn
[6] https://github.com/explosion/spaCy
[7] https://radimrehurek.com/gensim/
[8] https://stanfordnlp.github.io/CoreNLP/
[9] https://textblob.readthedocs.io/en/dev/
[10] https://github.com/cjlin1/liblinear
[11] https://machinelearningmastery.com/prepare-movie-review-data-sentiment-analysis/

and stemming. They concluded that researchers should choose all possible combinations carefully because inappropriate combinations may degrade performance.

Similarly, Jianqiang and Xiaoling [15] use six different pre-processing techniques on five Twitter datasets in their study, using four different classifiers. Their experimental results show that expanding acronyms and negations improved sentiment classification, whereas the removal of stop-words, special characters and URLs had a negative impact on on sentiment classification. The role of text pre-processing to reduce the sparsity issue in Twitter sentiment classification is studied by Sail et al. [37]. Experimental results show that choosing a combination of appropriate pre-processing methods can decrease the sparsity and enhance classification results. Agarwal et al. [1] propose novel tweet pre-processing approaches in their studies. They replaced the URL, user mentions, repeated characters and negated words with different tags and removed hashtags symbols. Classification results were improved by their proposed pre-processing methods. In other studies by Saloot et al. [38] and Yamada et al. [47] in the natural language workshop focused on noise in user-generated text[12]. The noisy nature of tweets is reduced by normalizing tweets using a maximum entropy model and entity linking. Naseem et al. also highlighted in their different studies the importance of improving text quality resulting in improved sentiment classification performance results [25, 27, 28, 30].

Recently, Symeonidis et al. [44] presented the comparative analysis of different text pre-processing techniques on two datasets for Twitter sentiment analysis classification. In their work, they study the effect of each technique on four traditional machine learning-based classifiers, and one neural network-based classifier with only TF-IDF [39] (unigram) as a word representation method. Their study showed that preprocessing techniques such as removing numbers, lemmatization and expanding contractions to base words performs better, whereas removing punctuation does not contribute positively to classification. Their study also presented the interactions of a limited number of different pre-processing techniques with others and highlight the techniques which perform well when used in combination.

Despite the fact many different methods have been presented to reduce the noisy nature of short texts to improve classification performance, no work has been done on the comparison of different methods and on the recommendation of pre-processing techniques that improve the quality of the text and enhance the performance of automated classification. In this paper, this research gap is addressed, a comparison of different techniques and the recommended combination of different pre-processing techniques is presented.

3 Methodology

In this section, first we present the analysis of different pre-processing techniques evaluated to analyze the effect of pre-processing and then followed by the proposed recommended combination of different pre-processing techniques.

3.1 Individual analysis of pre-processing techniques

In this section, we investigate the commonly used pre-processing techniques individually to illustrate their impact on tweet text. Table 1 shows the selected pre-processing techniques

[12] http://noisy-text.github.io/

Table 1 Numbers associated with pre-processing techniques

number	pre-processing Technique
1	URLs, user-mentions & hashtag symbol
2	Replace Abbreviations and Slang
3	Expanding Contractions
4	Removing Numbers
5	Replace Emoticons
6	Lemmatization
7	Removing Punctuation
8	Words Segmentation
9	Lower-casing of words
10	Removing Stop-words
11	Elongated Characters
12	Incorrect Spellings

and an associated technique number. In the onward discussion, we will use both technique name or number presented in Table 1.

1. **URLs, user mentions and hashtag symbols**: Most tweets contain URLs, user mentions and hashtag symbols which users include to provide additional referential information. This extra information is considered useful for humans, but is mostly considered as noise and not much value for text analytical tasks. In our analysis, we have removed all URLs, user mentions and hashtag symbols. An example is given below:
   Before:

   ```
   This is an illustration of
   #theartoftweeting
   for the benefit of @scottmorrison
   https://tinyurl.com/y4cm2b3q
   ```

   After:

   This is an illustration of the art of tweeting for the benefit of scott morrison

2. Abbreviations and slang: as mentioned earlier, character limitation forces social media users to use different abbreviations and slang in their tweets. This is more problematic when every user writes in his own style and uses different abbreviations. Acronyms and phrases which sometimes are associated to some specific context, or to a group of people. To interpret these language imperfections, it is vital to replace them with their associated meanings, which allows a machine to understand them easily. In our experiments we used `Ekphrasis`[13] library to replace abbreviations and acronyms to replace with their associated meaning. An example is given below:
   Before:

   ```
   Comparing banking CEOs that
   don't suck, a great article on
   ```

[13] https://github.com/cbaziotis/ekphrasis

```
Bloomberg: https://tinyurl.com/y4cm2b3q
```

After:

Comparing banking Chief Executive Officers good ones, a great article on Bloomberg: https://tinyurl.com/y4cm2b3q

3. **Expanding Contractions**: Expanding contractions can be a beneficial pre-processing technique, especially before performing tokenization, because tokenization will make two different tokens of a contraction like *can't* into *can* and *t*, which is nonsense. Expanding contractions preserves information because the word *not* is an essential valence of the utterance to preserve for the classification task. In our analysis, we employed `pycontractions` 2.0.0[14] Python library to expand contractions. An example is given below:
Before:

```
I can't think of a better airline
than @SingaporeAir.
Every experience is always excellent.
```

After:

I can not think of a better airline than @SingaporeAir. Every experience is always excellent.

4. **Removing Numbers**: Numbers are important but they do not always provide information for text classification and so it is common practice to remove numbers from the corpus. However, removing numbers too soon may lose information. For instance, if we remove *8* from *gr8* then we lose useful information, and this can degrade results. Removing numbers should always be sequenced after replacing abbreviations and slang with their associated word meanings. In our analysis, we removed all the numbers. An example is given below:
Before:

```
Little man did FAB - 11
out of 13hrs sleep!!
Great flight  @SingaporeAir
```

After:

Little man did FAB - out of hrs sleep!! Great flight @SingaporeAir

5. **Replace Emoticons:** emoticons express opinion and sentiment on social media. Humans can understand the emotions and sentiments behind these emoticons, machines need to be provided with their word meanings. In order to get maximum information in our experiments, we used the `Ekphrasis` library to replace emoticons with their associated word meanings. An example is given below:
Before:

[14] https://pypi.org/project/pycontractions/

```
hey so many time changes
for UA 1534.
We going tonight or  what?
Missing In Action :(
```

After:

hey so many time changes for UA 1534. We going tonight or what? Missing In Action sad

6. **Lemmatization**: is used to replace words with their root words. In our analysis, we used `WordNet` lemmatizer[15] library to perform this step. An example is given below:
Before:

```
poorly serviced. Give
us a chance at least once.
```

After:

poor service. Give us a chance at least once.

7. **Removing punctuation**: one classic and common pre-processing technique in text classification is to remove punctuation. Punctuation is useful for humans to understand opinion and sentiments, but for machines, it does not add much to classification performance. So in our study, we removed all punctuation. An example is given below:
Before:

```
Thank you #unitedairlines for
the  free gift for
our son at  #childrensmercy
hospital in KC!.
```

After:

Thank you #unitedairlines for the free gift for our son at #childrensmercy hospital in KC

8. **Word Segmentation**: as previously mentioned, character length limitations on tweets encourage users to write in an unstructured and informal way. Social media users also run together words in their hashtag messages to express sentiment, these concatenated strings are readable and easily understood by humans but require some attention to be machine readable. In our study, we separate the remaining content/phrases after removing the hashtag symbol. An example is given below:
Before:

```
#goodvibes United Airlines
Flies  Children With
Serious Illnesses
```

[15] https://pythonprogramming.net/lemmatizing-nltk-tutorial/

```
To Santa's North Pole
```

After:

good vibes United Airlines Flies Children With Serious Illnesses To Santa's North Pole

9. **Lower-casing of words**: case-folding of all words is also a common pre-processing technique. It helps to decrease dimensionality and also helps match words with the same meaning in the corpus. An example is given below:

   Before:

```
Got to the airport early.
How do I see if I can
change flights
```

After:

Got to the airport early. how do i see if i can change flights

10. **Removing Stop words**: many frequently used words in natural language such as articles and prepositions introduce nuance to language but do not always contribute text classification. For instance, words like `the, a, am, are, on, at` etc. Removing stop words is common practice in pre-processing in text classification tasks. In our analysis we used `NLTK` stop-word library[16] to remove all stop words in the corpus. An example is given below:

    Before:

```
I thought Comcast was bad,
until I saw the bad side
of  United Airlines
```

After:

I thought Comcast bad, saw bad side United Airlines

11. **Elongated Characters**: in order to avoid the learner treating elongated words differently from their base words, characters that are repeated three times consecutively are reduced to a single character, this idea is borrowed from Kiritchenko et al. [19]. An example is given below:

    Before:

```
Gooood for you, @united.
United Airlines
brings back free snacks
```

After:

[16] https://gist.github.com/sebleier/554280

Good for you, @united. United Airlines brings back free snacks

12. **Incorrect Spelling**: is common in social media posts and messages. Sometimes users intentionally use incorrect spelling as a form of stylization, e.g. `hav` for *have*. We also study the effect of correcting spelling mistakes in this analysis. We use Norvig's spell corrector[17] in this study. An example is given below:

    Before:

    ```
    Experiencing @cathaypacific's
    First lounge in
    HKG for first tym.
    Nice dining experienc
    ```

    After:

    Experiencing @cathaypacific's First lounge in HKG for first time. Nice dining experience

3.2 Recommended combination of pre-processing techniques

Following the presentation of each of the pre-processing techniques above, we now recommend systematic combinations of pre-processing techniques. The motivation behind this activity is to improve the quality of the text, finding a combination and sequence of pre-processing techniques that perform best when compared to others on a given text classification task for which there is classification ground-truth.

Researchers usually apply four to five common data pre-processing techniques before executing a feature extraction step. However, when text quality is poor, and especially in the case of Twitter, more pre-processing techniques may need to be applied to sufficiently normalize and improve the quality of the original text so that it is fit for purpose. Replacing and normalizing spelling mistakes, contractions, abbreviations and emoticons with their actual base words are useful steps to take in automated text analysis. Selecting an appropriate sequence with the right combination of pre-processing techniques is essential to improve text quality, and to improve the resulting performance of the text classifier. Also, not all techniques perform well when combined with others; even where they perform well when used standalone, some combinations of pre-processing techniques do not interact well.

Further, not following a specific order of pre-processing techniques can result in information loss which ultimately degrades classification performance. Recommending a combination of pre-processing techniques that improves the performance text classification is not explored in previous studies. The number of possible combinations of the twelve pre-processing techniques presented is 12!; it is a difficult (and perhaps different) research question to exhaustively explore which combinations interact well with one another when combined, and which simply do not make sense when used together. However, as we have seen, at least for some pre-processing methods, there is a clear precedence relationship and so to limit the search space, our proposed method is the result of testing different combinations that interact well with other techniques and designed with the motivation of improving the quality of the transformed tweets used to train a text classifier.

[17] http://norvig.com/spell-correct.html

@UnitedAirlines Cooool I'm :) with servc! You ROCKED #urgr8 http://ow.ly/VIbf0\

Fig. 1 A Toy Example: challenges involved in raw text

To illustrate the interactions and combinations of different pre-processing techniques to improve the quality of the transformed tweets, we considered the tweet (toy example) given in Fig. 1. In addition, to achieve a better quality transformed tweet, some techniques – such as technique #1, #4, #10 and #6 – have to be applied in the same order proposed – and this reduces the possible number of combinations of the remaining pre-processing techniques to 8!. While all combinations are explored, only combinations with significant results are discussed here.

As mentioned, not only is the selection of a set of pre-processing techniques important but also their sequence must be logical. For instance, as we have seen, removing the number `8` from word `gr8` before replacing it with actual work `great` results in information loss (see toy example in Fig. 1), and this may be critical. Similarly, in Fig. 1 expanding contractions such as `I'm`, but especially negative contractions such as `couldn't` or `haven't` (not present in Fig. 1) after tokenization, can impact performance. For example, the tokenizer breaks `couldn't` into `couldn` and `t` and breaks `haven't` into `haven` and `t`. In contrast to this, if hashtag symbols are removed first followed abbreviation substitution and finally word segmentation then `#urgr8` is correctly expanded to *you are great*. Similar behaviours can be observed from sequencing other techniques where interactions between pre-processors reveal varying degrees of success. With the above-mentioned motivation in mind, we experimented with different combinations. All variations of pre-processors are explored; however, only those combinations that showed significant (and best) performance are reported in Table 2.

Based on experimental results (see Section 4.3), the best performing combination is the eighth combination presented in Table 2, graphically shown in Fig. 2. In this 12-step pre-processing sequence, the first step removes all Unicode strings, URLs, user-mentions and hashtag # symbols. Emoticons and emojis are then replaced with their associated word meanings, abbreviations and acronyms are expanded, and spelling corrected at step #2, #3 and #4, respectively. At step #5 and #6, contractions are expanded and elongated characters abbreviated respectively. In the remaining steps, all punctuation is removed, case-folding

Table 2 Different combinations of pre-processing techniques

Combination #	Names Associated for future reference	Techniques numbers take from Table 1
1	C1	1-2-3-4-5-6-7-8-9-10-11-12
2	C2	1-12-5-2-8-3-11-7-4-9-10-6
3	C3	1-8-5-2-11-12-3-7-4-9-10-6
4	C4	1-2-11-8-12-5-3-7-4-9-10-6
5	C5	1-8-11-12-5-2-3-7-4-9-10-6
6	C6	1-8-2-5-11-12-3-7-9-4-10-6
7	C7	1-2-5-12-3-11-7-9-8-4-10-6
8	Proposed	1-5-2-12-3-11-7-9-8-4-10-6

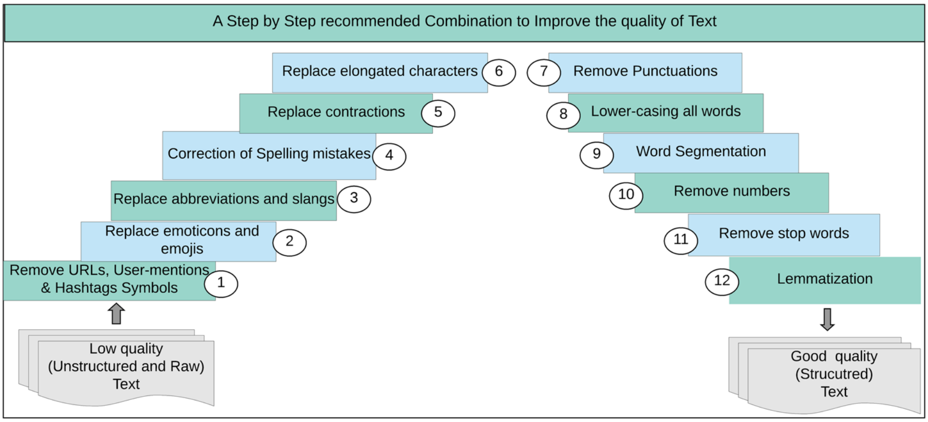

Fig. 2 A Recommended combination of pre-processing

occurs to lower-case, word segmentation is performed, numbers are removed as are stop-words. As a final step lemmatization is performed. In all steps, as mentioned above, we used the same methods and techniques we used earlier during their analysis individually.

4 Experimental analysis

In this section, the experimental settings are presented followed by the datasets used in the analysis and finally, the results are presented and discussed.

4.1 Experiment settings

In this section, we present the word representation, classifiers and evaluation metrics used in the analysis.

4.1.1 Words representation

Different word representation models are available to chose from. In this study we selected one a standard legacy word representation – TF-IDF [39] – and a more contemporary continuous word representation model – GloVe [16].

4.1.2 Classifiers

To provide a comprehensive analysis of techniques, five commonly used traditional machine learning classifiers and two deep learning-based classifiers are used to assess the effect of the different pre-processing techniques to the text classification task.

Traditional Machine learning classifiers Traditional machine learning-based classifiers such as Support Vector Machines (SVM), Naive Bayes (NB) , Logistic Regression (LR), Decision Tree (DT) and Random Forest (RF) are employed in our analysis with the TF-IDF word representation model.

Table 3 Datasets characteristics

Characteristics\ Dataset	Davidson et al.	Golbeck et al.	Waseem et al.
Total No. of tweet	25,112	19,968	15,844
No. of classes	3	2	3
No. of Words	4,60,955	4,72,744	2,08,583
Avg No. of words	13.356	13.901	10.304
No. of Emoticons	17,650	7,940	1,057
No. of Abbreviations	4,998	1,316	612
No. of Elongated Words	4,821	2,894	1,780

Deep learning classifiers : From the deep learning-based classifiers, two commonly used algorithms were used: (i) Convolutional Neural network (CNN) and (ii) Recurrent Neural network (RNN) with the GloVe word representation model. In particular, we used Kim's implementation [18] for CNN and for RNNs, we followed the implementation of Looks et al. [22] and their Tree-Long Short Term Memory (LSTM) and for bi-directional LSTM (Bi-LSTM) we used models proposed by Tai et al. [45] with default parameters, as parameter optimization was not the part of this analysis.

4.1.3 Evaluation metrics

For the evaluation of the proposed methods and comparison, we have used F1-Score metric.

4.2 Datasets

For the evaluation, the behavior of different pre-processing techniques were investigated when applied to three different Twitter datasets, related to Twitter hate speech and abusive language. Datasets statistics are given in Table 3 and briefly discussed below:

- **Golbeck et al. Dataset** Golbeck et al. [11] provided a large labeled corpus for online harassment data from Twitter. First, a list of keywords was produced for the collection of tweets that contain harassing words; then human coders were given guidelines to label the sentiment of the tweets. The first version of dataset contains 35,000 tweets with two classes (harassment or none). However, their current version of dataset contains only 19,968 tweets which are categorized into two classes (harassment and none).
- **Waseem et al. Dataset** Waseem et al. [13] provided a labeled dataset of 16,914 tweets from 136,052 collected tweets over a period of two months. Tweets were manually annotated and classified into three classes: racist, sexist, neither racist or sexist. Authors released the list of 16,907 tweets IDs and their corresponding labels. Some of the tweets were either deleted, or their visibility has been changed – Twitter itself has the ability to moderate tweets and users often delete their own tweets – so only 15,844 tweets could be found using Python's `Tweepy`[18] library, labeled into the three classes as described.
- **Davidson et al. Dataset** Davidson et al. [7] is the third labelled dataset from Twitter used in this experiment. This data focuses on differentiating between hateful and

[18]https://github.com/tweepy/tweepy

offensive language. The dataset is manually annotated by human coders into three classes: hateful, offensive, neither hateful or offensive. The total number of labelled tweets given in this dataset is 25,112.

4.3 Results and discussion

In this section, we present an analysis of the different pre-processing techniques used. Tables 4, 5 and 6 presents the results of all pre-processing techniques for three datasets. Green & Red entries denote the highest & lowest performing techniques in each row. The best performing techniques in each column are marked in bold. Technique number #0 represents the original unprocessed text, used as the baseline for the comparison of results in the study. It can be seen clearly that classification performance is inconsistent, and this reflects the varying effects of each technique on results. Below we discuss the effects of each pre-processing technique as compared to the baseline results. For the sake of simplicity, we discuss the techniques which managed to improve the performance ≥ 1, as compared to baseline.

1. **Removal of URLs, user mentions and hashtag symbols**: This pre-processing technique increases performance on two datasets. An increase in performance observed in the case of SVM (trigrams), LR (bigrams), CNN and BiLSTM classifiers on the Waseem et al. dataset and SVM (unigram and bigram), LR (unigram and bigram), DT (unigram) and all the neural network-based classifiers. No significant increase in classification performance is observed in the case of Davidson et al. The reason behind this improvement is the number of URLs, user mentions and hashtag symbols are enormous in these two datasets, when compared to the Golbeck et al. dataset. The best results achieved using this pre-processing technique is 0.695 on the Davidson et al. dataset.
2. **Replacing abbreviations and slang**: Experimental results show that classifier performance increases in all datasets when abbreviations and slang are replaced with their associated word meaning when compared to the baseline. For the Waseem et al., dataset, performance increases in the case of CNN and BiLSTM. For the Golbeck et al. dataset, increases are observed in both RNN-based classifiers whereas, and in case of the Davidson et al. dataset, a significant increase in performance is observed in all NN-based classifiers, and all traditional machine learning classifiers but results vary with different features. The prime reason for this difference in performance is that the number of abbreviations and slang expressions in the Davidson et al. dataset are greater than in the other two datasets. The best result achieved using this pre-processing technique is 0.669 on the Davidson et al. dataset.
3. **Expanding Contractions**: Expanding contractions to root words also demonstrated some improved classification performance on all three datasets. In the case of the Waseem et al. dataset, performance increased only in the case of CNN and LSTM, whereas for the Golbeck et al. dataset performance increased only for the LSTM classifier. Performance increased for all classifiers (with different features) on the third dataset. The best results achieved using this pre-processing technique is 0.644 on the Davidson et al. dataset.
4. **Removing Numbers**: This pre-processing technique outperforms the baseline results on one classifier (LSTM) in the case of the Waseem et al. and Golbeck et al. datasets. Whereas applied to the Davidson et al. dataset, it outperforms baseline in SVM-unigram and bigram, LR-trigram, DT-unigram, RF-bigram and trigram and from

Table 4 Comparison of pre-processing techniques on Waseem et al. dataset

Classifier/Technique No.		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
SVM	tfidf-uni	**0.554**	0.540	**0.554**	**0.553**	**0.554**	**0.555**	**0.559**	**0.554**	**0.554**	**0.559**	**0.556**	**0.555**	**0.554**
	tfidf-Bi	0.527	0.526	0.526	0.518	0.523	0.525	0.529	0.486	0.526	0.526	0.510	0.526	0.526
	tf-idfTri	0.500	0.510	0.499	0.501	0.500	0.501	0.500	0.441	0.499	0.505	0.480	0.501	0.499
NB	tfidf-uni	0.508	0.481	0.508	0.506	0.508	0.508	0.505	0.511	0.508	0.509	0.518	0.507	0.508
	tfidf-Bi	0.487	0.483	0.487	0.487	0.487	0.487	0.487	0.476	0.487	0.494	0.491	0.487	0.487
	tf-idfTri	0.474	0.482	0.474	0.476	0.474	0.472	0.477	0.444	0.474	0.481	0.450	0.473	0.474
LR	tfidf-uni	0.541	0.536	0.541	0.543	0.543	0.542	0.544	0.537	0.541	0.546	0.542	0.542	0.541
	tfidf-Bi	0.508	0.517	0.508	0.508	0.508	0.508	0.510	0.473	0.508	0.518	0.512	0.509	0.508
	tf-idfTri	0.490	0.495	0.490	0.494	0.491	0.490	0.489	0.447	0.490	0.497	0.475	0.490	0.490
DT	tfidf-uni	0.523	0.506	0.517	0.519	0.520	0.523	0.515	0.518	0.523	0.524	0.509	0.522	0.519
	tfidf-Bi	0.497	0.494	0.489	0.493	0.501	0.497	0.494	0.470	0.495	0.503	0.485	0.501	0.495
	tf-idfTri	0.478	0.471	0.476	0.480	0.479	0.476	0.484	0.444	0.476	0.488	0.458	0.478	0.476
RF	tfidf-uni	0.525	0.513	0.525	0.525	0.527	0.529	0.528	0.529	0.528	0.532	0.530	0.522	0.526
	tfidf-Bi	0.517	0.510	0.512	0.521	0.517	0.516	0.523	0.486	0.515	0.522	0.509	0.518	0.519
	tf-idfTri	0.500	0.498	0.500	0.496	0.503	0.501	0.496	0.445	0.497	0.510	0.474	0.502	0.498
CNN	GloVe	0.535	**0.547**	0.545	0.550	0.544	0.541	0.548	0.517	0.535	0.545	0.534	0.547	0.546
LSTM	GloVe	0.529	0.536	0.532	0.540	0.541	0.530	0.531	0.516	0.539	0.538	0.532	0.538	0.536
BiLSTM	GloVe	0.531	0.545	0.543	0.534	0.524	0.534	0.533	0.516	0.547	0.529	0.543	0.535	0.541

Table 5 Comparison of pre-processing techniques on Golbeck et al. dataset

Classifier/Technique No.		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
SVM	tfidf-uni	0.583	0.583	0.583	0.588	0.587	0.584	0.583	0.587	0.583	0.581	0.587	0.583	0.583
	tfidf-Bi	0.580	0.576	0.578	0.577	0.576	0.576	0.581	0.564	0.578	0.574	0.565	0.580	0.578
	tf-idfTri	0.558	0.563	0.557	0.559	0.557	0.557	0.563	0.553	0.557	0.566	0.543	0.557	0.557
NB	tfidf-uni	0.479	0.480	0.480	0.480	0.480	0.482	0.478	0.482	0.480	0.476	0.487	0.480	0.480
	tfidf-Bi	0.494	0.480	0.495	0.496	0.496	0.496	0.493	0.500	0.495	0.485	0.511	0.494	0.495
	tf-idfTri	0.504	0.499	0.505	0.506	0.506	0.504	0.510	0.491	0.505	0.508	0.494	0.505	0.505
LR	tfidf-uni	**0.604**	**0.605**	**0.605**	**0.606**	**0.608**	**0.606**	**0.609**	**0.602**	**0.605**	**0.607**	**0.603**	**0.606**	**0.605**
	tfidf-Bi	0.592	0.587	0.591	0.590	0.591	0.598	0.590	0.581	0.591	0.590	0.585	0.592	0.591
	tf-idfTri	0.574	0.573	0.574	0.570	0.574	0.575	0.578	0.556	0.574	0.570	0.561	0.574	0.574
DT	tfidf-uni	0.571	0.571	0.571	0.569	0.569	0.567	0.574	0.567	0.575	0.570	0.568	0.568	0.580
	tfidf-Bi	0.559	0.554	0.548	0.541	0.557	0.561	0.546	0.547	0.559	0.562	0.557	0.558	0.558
	tf-idfTri	0.539	0.521	0.538	0.528	0.535	0.539	0.543	0.529	0.538	0.540	0.524	0.534	0.535
RF	tfidf-uni	0.563	0.558	0.554	0.558	0.557	0.560	0.561	0.554	0.555	0.568	0.566	0.563	0.559
	tfidf-Bi	0.555	0.548	0.553	0.556	0.561	0.550	0.557	0.552	0.557	0.567	0.551	0.557	0.551
	tf-idfTri	0.530	0.524	0.533	0.527	0.535	0.529	0.537	0.530	0.527	0.536	0.510	0.527	0.535
CNN	GloVe	0.603	0.587	0.595	0.592	0.591	0.570	0.599	0.587	0.565	0.593	0.582	0.598	0.580
LSTM	GloVe	0.491	0.472	0.501	0.503	0.508	0.549	0.569	0.583	0.559	0.577	0.568	0.568	0.566
BiLSTM	GloVe	0.587	0.553	0.599	0.560	0.561	0.589	0.584	0.559	0.554	0.580	0.595	0.566	0.601

Table 6 Comparison of pre-processing techniques on David et al. dataset

Classifier/Technique No.		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
SVM	tfidf-uni	0.633	0.633	0.635	0.633	0.635	0.635	0.631	0.626	0.633	0.637	0.633	0.635	0.633
	tfidf-Bi	0.583	0.608	0.598	0.595	0.595	0.593	0.615	0.549	0.591	0.608	0.553	0.591	0.591
	tf-idfTri	0.454	0.485	0.470	0.474	0.475	0.468	0.474	0.392	0.468	0.473	0.407	0.463	0.468
NB	tfidf-uni	0.439	0.440	0.440	0.440	0.440	0.440	0.445	0.450	0.443	0.451	0.457	0.442	0.443
	tfidf-Bi	0.410	0.401	0.410	0.410	0.401	0.410	0.412	0.400	0.406	0.413	0.410	0.406	0.406
	tf-idfTri	0.360	0.370	0.370	0.370	0.360	0.370	0.373	0.348	0.364	0.369	0.351	0.364	0.364
LR	tfidf-uni	**0.646**	0.640	0.645	0.643	**0.644**	0.642	0.647	**0.637**	0.646	**0.648**	0.645	0.641	0.646
	tfidf-Bi	0.580	0.590	0.580	0.580	0.580	0.590	0.596	0.546	0.580	0.595	0.557	0.580	0.580
	tf-idfTri	0.440	0.471	0.450	0.460	0.470	0.460	0.460	0.378	0.458	0.460	0.402	0.457	0.458
DT	tfidf-uni	0.587	0.596	0.599	0.561	0.605	0.563	0.597	0.596	0.588	0.603	0.574	0.591	0.570
	tfidf-Bi	0.490	0.450	0.490	0.480	0.480	0.500	0.523	0.472	0.493	0.487	0.479	0.485	0.487
	tf-idfTri	0.370	0.380	0.390	0.390	0.380	0.390	0.386	0.332	0.388	0.392	0.351	0.384	0.387
RF	tfidf-uni	0.583	0.560	0.602	0.582	0.582	0.584	0.582	0.608	0.565	0.577	0.630	0.534	0.590
	tfidf-Bi	0.530	0.510	0.520	0.540	0.520	0.530	0.539	0.506	0.539	0.534	0.520	0.529	0.532
	tf-idfTri	0.410	0.410	0.420	0.420	0.410	0.420	0.426	0.381	0.417	0.426	0.366	0.416	0.419
CNN	GloVe	0.611	**0.695**	0.627	0.600	0.649	0.657	**0.671**	0.615	0.713	0.618	0.642	0.499	0.699
LSTM	GloVe	0.570	0.602	0.612	0.560	0.605	0.598	0.642	0.596	0.637	0.594	0.630	0.463	0.614
BiLSTM	GloVe	0.613	0.686	**0.669**	**0.644**	0.609	**0.684**	0.650	0.622	**0.686**	0.599	**0.674**	**0.688**	**0.676**

NN-based classifiers it outperforms the baseline in both CNN and LSTM classifiers. The best results achieved with this pre-processing technique is 0.644 against the Davidson et al. dataset.

5. **Replace Emoticons**: This pre-processing technique improves performance in only two datasets compared to the baseline. In the case of the Golbeck et al. dataset, performance increased in CNN-based classifier whereas, for the Davidson et al. dataset, performance increased in almost all tested classifiers. The reason for this is that the presence of emoticons in the Davidson et al. dataset is more common compared to the other two datasets. The best results achieved by this pre-prcessing technique is 0.684 on the Davidson et al. dataset.
6. **Lemmatization**: this pre-processing technique shows significant improvement in the Davidson et al. dataset in almost all cases. Whereas, in the Waseem et al. and Golbeck et al. datasets, performance increased only in the case of CNN (for the former) and LSTM (for the the later) classifiers. The best results achieved by this pre-procdssing technique is 0.671 on the Davidson et al. dataset.
7. **Removing punctuation**: removing punctuation did not yield any significant results when used alone and is able to beat the baseline results in only two datasets. LSTM (the Golbeck et al. dataset) and LST, RF-unigram and NB-unigram in the case of the Davidson et al. dataset. The best results achieved by this pre-processing technique is 0.637 on the Davidson et al. dataset.
8. **Word Segmentation**: separating the content of hashtagged strings improves baseline results in all datasets. In the case of the Waseem et al. dataset, results improved for RNN-based classifiers, for the Golbeck et al. dataset, only LSTM-based classifiers were able to beat the baseline scores whereas, in the Davidson et al. dataset, all NN-based classifiers outperformed the baseline results. Also results improved for SVM-trigram, LR-trigram and DT-trigram classifiers. The best results achieved using this pre-processing technique is 0.686 for the Davidson et al. dataset.
9. **Lower-casing of words**: case-folding resulted in classifier performance improvement against all datasets. For the Waseem et al. dataset, results improved in case of DT-trigram, RF-trigram and CNN classifiers. For the Golbeck et al. dataset, only LSTM and RF-bigram were able to beat the baseline results. For the Davidson et al. dataset, performance increases are observed in the case of LSTM, RF-trigram, DT-unigram and trigram, LR-bigram and trigram, NB-unigram and SVM-bigram and trigram classifiers. The best results achieved using this pre-processing technique is 0.648 on the Davidson et al. dataset.
10. **Removing stop-words**: this common pre-processing technique also outperforms the baseline results in all datasets. For the Waseem et al. dataset, BiLSTM and NB-unigram showed most improvement. NB-bigram and LSTM for the Golbeck et al. dataset and all NN-based classifiers and RF-unigram for the Davidson et al. dataset. The best results achieved using this pre-processing technique is 0.674 against the Davidson et al. dataset.
11. **Elongated character removal**: this pre-processing technique improves baseline results only in the case of two datasets (the Davidson et al. and Golbeck et al. datasets). The reason is that the presence of elongated characters is more common in these two datasets compared to the Waseem et al. dataset. The best results achieved using this pre-processing technique is 0.688 on the Davidson et al. dataset.
12. **Incorrect spelling**: correcting spelling improves results in all datasets against the baseline. For the Waseem et al. dataset, results are most improved in case of CNN and

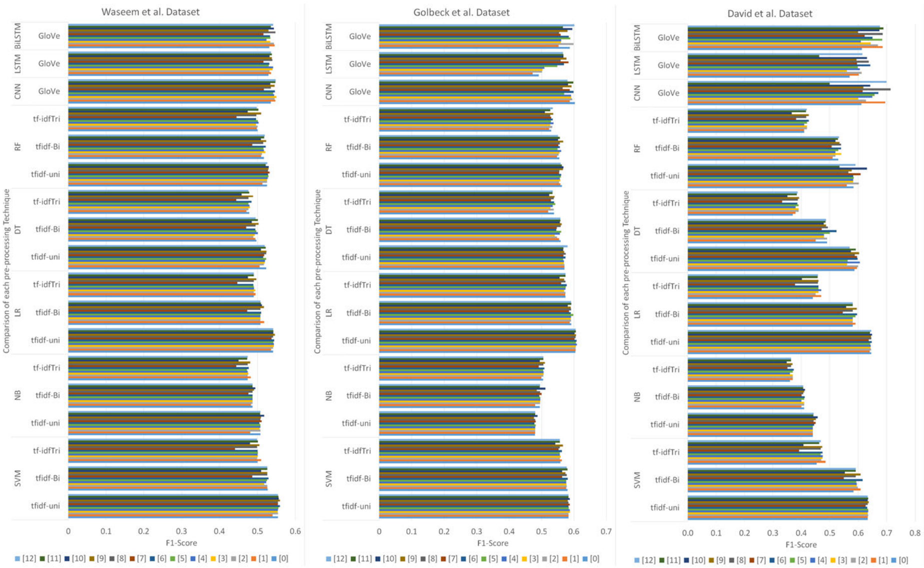

Fig. 3 Graphical Representation: Effects of pre-processing techniques

BiLSTM classifiers. For the Golbeck et al. dataset, only the RNN-based classifiers showed improvement whereas, all NN-based classifiers along with DT (trigram), LR (bigram) and SV (trigram) outperform the baseline classifier results in the Davidson et al. dataset. The best results achieved using this pre-processing technique is 0.676 for the Davidson et al. dataset.

In the previous subsection, the effects of different pre-processing techniques on three labelled Twitter hate speech and abusive language datasets is presented. According to the results, the effect of text pre-processing techniques varies depending on the different classification algorithms used. The green highlights are the best resulting classifier that outperforms the baseline in each case, whereas the red highlights are the worst performing classifier. The best results for each pre-processing technique can be seen in bold. Each technique beats baselines results in most of the classifiers in all of datasets. Results of each technique, rendered one at a time, is graphically presented in Fig. 3.

Based on these outcomes, results are divided into two (best and worst) categories according to the performance given in Table 7. We found that in the case where only one pre-processing technique is used at a time, then the best-performing techniques are lemmatization and lower-casing whereas, removing punctuation and URLs, user-mentions and hashtag symbols, has a negative impact on classification performance. In other words, when

Table 7 Best and Worst performing pre-processing techniques on all datasets

Performance	Description	Techniques
Best	High performance in most cases	Lemmatization and Lower-casing of words
Worst	Low performance in most cases	Removing Punctuation and URLs, usermentions and Hasthag symbols

the text analysis requires low pre-processing overhead, lemmatization and lower-casing are the text pre-processing techniques of choice.

4.3.1 Pre-processing sequence

As previously discussed, classification results vary depending on which text pre-processing techniques are used and the order they are applied. Tables 8, 9 and 10 show the results of our proposed combination of different pre-processing results. We compare the results of our proposed method with results against the baseline, where no text pre-processing technique is applied, and with the results of the best performing individual pre-processing techniques. It is evident from the results that the proposed pre-processing method is beneficial classification performance. This can be explained as follows.

Pre-processing improves the quality of text, by removing noise, and in so doing helps the learning algorithm extract better features. The proposed recommended combination works well because the application order of the techniques plays a significant role in improving the quality of the tweets, enhancing semantic meaning and minimising information loss. On the other hand, changing the order in which pre-processing is applied results in information loss, reducing semantic meaning, which in turn impacts negatively on the quality of the features extracted by the learner which in its turn leads to deteriorating classification results. The proposed method structures and normalizes the unstructured and informal nature of

Table 8 Comparison of Proposed Combination on classification task (Waseem et al. Dataset)

Classifier	Techniques	Baseline	Highest individual technique results	C1	C2	C3	C4	C5	C6	C7	Proposed
SVM	tfidf-uni	0.554	0.559	0.485	0.545	0.545	0.543	0.545	0.543	0.561	0.569
	tfidf-Bi	0.527	0.529	0.528	0.529	0.525	0.525	0.529	0.53	0.526	0.540
	tf-idfTri	0.500	0.510	0.545	0.508	0.508	0.512	0.508	0.508	0.494	0.570
NB	tfidf-uni	0.508	0.518	0.428	0.476	0.479	0.476	0.476	0.476	0.510	0.522
	tfidf-Bi	0.487	0.494	0.419	0.486	0.489	0.485	0.486	0.485	0.492	0.503
	tf-idfTri	0.474	0.482	0.481	0.483	0.482	0.481	0.483	0.483	0.476	0.497
LR	tfidf-uni	0.541	0.546	0.496	0.538	0.539	0.537	0.538	0.539	0.541	0.552
	tfidf-Bi	0.508	0.518	0.527	0.519	0.518	0.517	0.519	0.520	0.510	0.545
	tf-idfTri	0.490	0.497	0.542	0.498	0.498	0.497	0.498	0.496	0.484	0.558
DT	tfidf-uni	0.523	0.524	0.459	0.518	0.523	0.520	0.521	0.523	0.518	0.531
	tfidf-Bi	0.497	0.503	0.498	0.496	0.496	0.494	0.497	0.500	0.496	0.519
	tf-idfTri	0.478	0.488	0.506	0.478	0.477	0.470	0.478	0.480	0.473	0.541
RF	tfidf-uni	0.525	0.532	0.475	0.512	0.515	0.518	0.519	0.514	0.528	0.541
	tfidf-Bi	0.517	0.523	0.505	0.510	0.506	0.504	0.510	0.506	0.521	0.535
	tf-idfTri	0.500	0.510	0.524	0.498	0.499	0.495	0.497	0.500	0.492	0.539
CNN	GloVe	0.535	0.548	0.526	0.535	0.540	0.538	0.544	0.539	0.542	0.563
LSTM	GloVe	0.529	0.541	0.426	0.533	0.531	0.533	0.533	0.524	0.527	0.577
BiLSTM	GloVe	0.531	0.547	0.432	0.529	0.522	0.518	0.519	0.534	0.510	0.586

Table 9 Comparison of Proposed Combination on classification task (Golbeck et al. Dataset)

Classifier	Techniques	Baseline	Highest individual technique results	C1	C2	C3	C4	C5	C6	C7	Proposed
SVM	tfidf-uni	0.583	0.588	0.532	0.587	0.585	0.585	0.587	0.588	0.588	0.597
	tfidf-Bi	0.580	0.581	0.562	0.576	0.571	0.575	0.576	0.575	0.581	0.594
	tf-idfTri	0.558	0.566	0.577	0.563	0.559	0.569	0.563	0.567	0.561	0.596
NB	tfidf-uni	0.479	0.487	0.426	0.479	0.482	0.478	0.479	0.478	0.480	0.493
	tfidf-Bi	0.494	0.511	0.462	0.487	0.485	0.485	0.487	0.487	0.496	0.519
	tf-idfTri	0.504	0.510	0.485	0.501	0.500	0.504	0.501	0.499	0.510	0.522
LR	tfidf-uni	0.604	0.609	0.534	0.606	0.608	0.607	0.606	0.607	0.606	0.618
	tfidf-Bi	0.592	0.598	0.578	0.592	0.591	0.591	0.592	0.589	0.593	0.611
	tf-idfTri	0.574	0.578	0.601	0.575	0.568	0.574	0.575	0.575	0.572	0.619
DT	tfidf-uni	0.571	0.580	0.509	0.569	0.566	0.566	0.573	0.565	0.565	0.582
	tfidf-Bi	0.559	0.559	0.533	0.550	0.556	0.549	0.552	0.550	0.554	0.584
	tf-idfTri	0.539	0.543	0.572	0.515	0.545	0.522	0.528	0.522	0.533	0.589
RF	tfidf-uni	0.563	0.568	0.476	0.561	0.559	0.560	0.561	0.558	0.564	0.572
	tfidf-Bi	0.555	0.567	0.507	0.548	0.550	0.544	0.546	0.549	0.559	0.579
	tf-idfTri	0.530	0.537	0.554	0.520	0.529	0.526	0.522	0.529	0.528	0.568
CNN	GloVe	0.603	0.599	0.575	0.589	0.596	0.590	0.587	0.599	0.592	0.620
LSTM	GloVe	0.491	0.583	0.426	0.565	0.555	0.544	0.561	0.560	0.551	0.624
BiLSTM	GloVe	0.587	0.601	0.426	0.611	0.573	0.572	0.568	0.567	0.578	0.649

Table 10 Comparison of Proposed Combination on classification task (David et al. Dataset)

Classifier	Techniques	Baseline	Highest individual technique results	C1	C2	C3	C4	C5	C6	C7	Proposed
SVM	tfidf-uni	0.633	0.637	0.509	0.726	0.735	0.729	0.726	0.731	0.727	0.736
	tfidf-Bi	0.583	0.615	0.687	0.631	0.609	0.631	0.631	0.632	0.609	0.697
	tf-idfTri	0.454	0.485	0.471	0.491	0.486	0.487	0.491	0.488	0.478	0.731
NB	tfidf-uni	0.441	0.457	0.295	0.440	0.443	0.439	0.440	0.441	0.447	0.464
	tfidf-Bi	0.405	0.413	0.390	0.411	0.404	0.409	0.411	0.408	0.415	0.424
	tf-idfTri	0.361	0.373	0.445	0.377	0.374	0.373	0.377	0.374	0.375	0.458
LR	tfidf-uni	0.646	0.648	0.504	0.745	0.745	0.747	0.745	0.743	0.742	0.741
	tfidf-Bi	0.579	0.596	0.697	0.619	0.603	0.614	0.619	0.617	0.604	0.711
	tf-idfTri	0.439	0.470	0.723	0.479	0.479	0.478	0.479	0.477	0.459	0.733
DT	tfidf-uni	0.587	0.605	0.438	0.699	0.689	0.694	0.701	0.701	0.704	0.713
	tfidf-Bi	0.487	0.523	0.618	0.498	0.488	0.496	0.490	0.477	0.500	0.635
	tf-idfTri	0.367	0.392	0.662	0.382	0.397	0.380	0.385	0.379	0.399	0.689
RF	tfidf-uni	0.583	0.630	0.359	0.596	0.586	0.603	0.595	0.595	0.607	0.635
	tfidf-Bi	0.527	0.540	0.546	0.520	0.508	0.514	0.512	0.518	0.551	0.568
	tf-idfTri	0.408	0.426	0.609	0.424	0.417	0.427	0.420	0.423	0.431	0.627
CNN	GloVe	0.611	0.713	0.290	0.368	0.405	0.414	0.414	0.400	0.419	0.743
LSTM	GloVe	0.570	0.642	0.318	0.575	0.593	0.578	0.037	0.615	0.631	0.749
BiLSTM	GloVe	0.613	0.688	0.359	0.656	0.631	0.290	0.329	0.654	0.660	0.752

the tweet text, and is primarily responsible for the performance improvement of the classifier. Figure 4 presents the graphical comparison of our proposed method with others on all datasets.

A step-by-step sequence, when applied to the running the toy example presented earlier, is given in Table 11. It is clear from the table that by following the proposed combination of pre-processing steps the quality of the tweet text is improved. For instance; removing

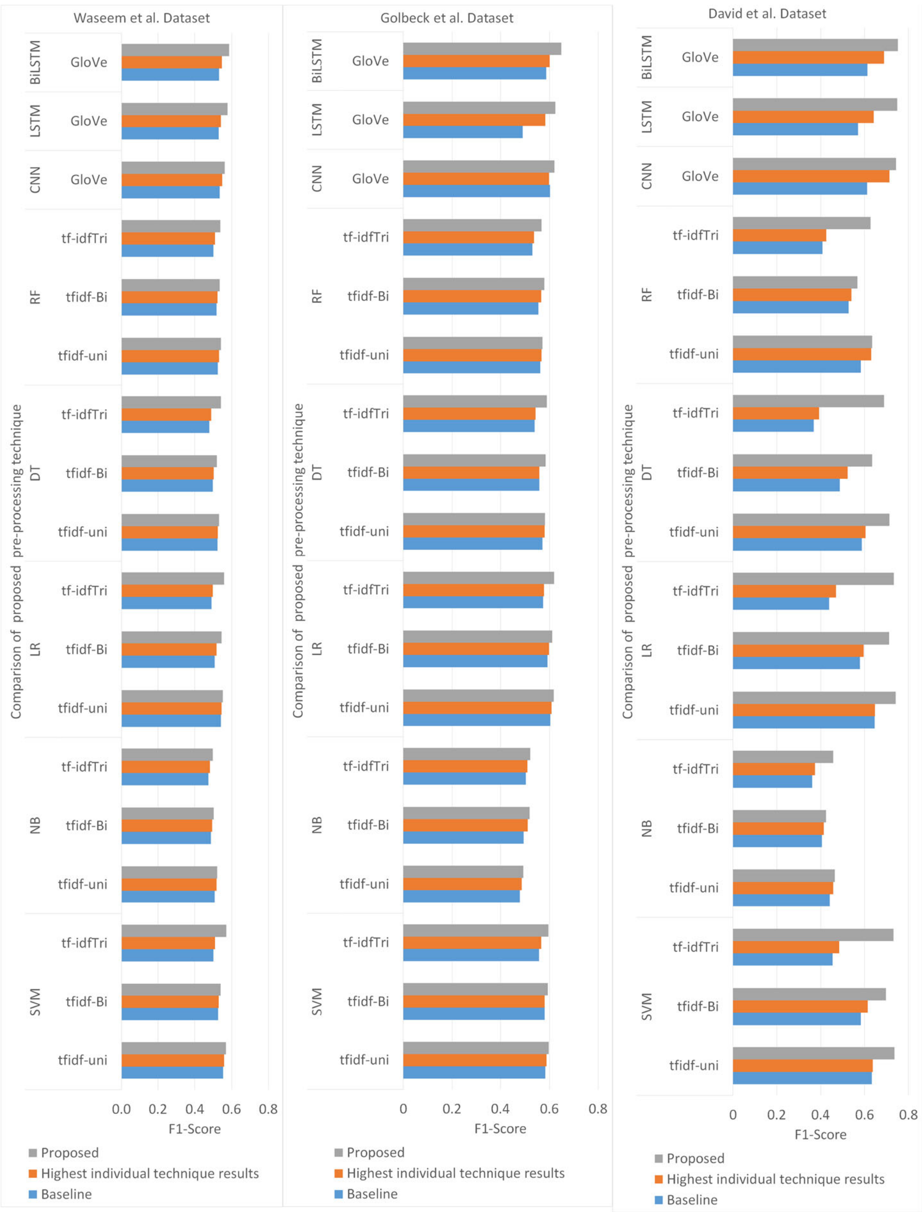

Fig. 4 Comparison of Proposed method on classification task

Table 11 A step by step working mechanism of proposed method

Tweet	`@UnitedAirlines Cooool I'm :) with servc! You ROCKED #urgr8 http://ow.ly/VIbf0`	
Steps	Recommended combination	Step by step pre-processing results
1	Removal of Unicodes, URLs, User-mentions & hashtags symbols	Cooool I'm :) with servc! You ROCKED urgr8
2	Replacing Emoticons & Emojis	Cooool I'm happy with servc! You ROCKED urgr8
3	Replacing Slangs & Abbreviations	Cooool I'm happy with servc! You ROCKED youaregreat
4	Correction of Spelling mistakes	Cooool I'm happy with service! You ROCKED youaregreat
5	Expanding Contractions	Cooool I am happy with service! You ROCKED youaregreat
6	Replacing Elongated words	Cool I am happy with service! You ROCKED youaregreat
7	Removing Punctuations	Cool I am happy with service You ROCKED youaregreat
8	Lower-casing of words	cool i am happy with service you rocked youaregreat
9	Word Segmentation	cool i am happy with service you rocked you are great
10	Removing Numbers	cool i am happy with service you rocked you are great
11	Removing Stopwords	cool happy service rocked great
12	Lemmatization	cool happy service rock great

URLs, user-mentions and hashtags, useful for humans, has in practice no impact on machine classification. Similarly, emoticons, abbreviations, spelling mistakes and other language imperfections, easily understandable for humans, are ideally eliminated by pre-processing in order to enhance machine classification.

Further, the sequence applying pre-processing techniques is crucial. For instance, if we do not follow the proposed order and randomly "remove all numbers" before replacing abbreviations and acronyms such as words `gr8`, `fi9`, `b4`, `2mro` etc into their actual semantic meaning is lost. Similarly, performing word segmentation before replacing slang and acronyms also results in information loss. For instance, if we do not first replace abbreviations and slang into their actual words from the phrases like `urgr8` and `emfi9` etc, then again we end up losing necessary and useful information. Further, expanding contractions after the tokenization step looses information and breaks contractions such as `can't` into `can` and `t` and `don't` into `don` and `t`. Similarly, this is the case with other pre-processing techniques and the sequence they are applied. The experimental analysis confirms that the proposed systematic combination is the best compared to any other combinations and addresses the challenges of improving the quality of poor quality tweet text. The proposed method improves the quality of the text, leads to better feature extraction, which in turn helps the learner produce a better classifier.

5 Conclusion

The paper has studied the effect of twelve different pre-processing techniques for tweet classification using three different labelled datasets for Twitter hate speech and abusive language. Experiments were exhaustively conducted to measure the effect of the different pre-processing on three datasets. Each pre-processing technique is evaluated with five traditional and three deep learning-based learning algorithms with various (but standard) feature extraction models. Further, the paper presents both the worst and best-performing techniques and recommends those pre-processing techniques that yield the best outcomes when used individually. Results vary with different learning algorithms, which confirms that choosing a suitable learning algorithm, a learning algorithm that is fit-for-the-purpose to the problem domain, remains a considerable factor in text classification performance. After a series of experiments with different combinations and observing the interactions of the various pre-processing techniques, a sequence of optimal pre-processing techniques that results in the best classifier performance is recommended. This research opens new opportunities to explore different techniques and methods to improve the quality of the short texts, an area that has been previously overlooked. Future work, using the same methodology, can investigate the effect of these and other pre-processing techniques in different domains, and other combinations of pre-processing techniques and their interactions.

References

1. Agarwal A, Xie B, Vovsha I, Rambow O, Rebecca J (2011) Passonneau. sentiment analysis of twitter data
2. Alomari E, Mehmood R, Katib I (2019) Road traffic event detection using twitter data, machine learning, and apache spark. In: 2019 IEEE SmartWorld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (Smart- World/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), IEEE, pp 1888–1895
3. Alotaibi S, Mehmood R, Katib I, Rana O, Albeshri A (2020) Sehaa: a big data analytics tool for healthcare symptoms and diseases detection using twitter, apache spark, and machine learning. Appl Sci 10(4):1398
4. Balahur A (2013) Sentiment analysis in social media texts. In: WASSA@NAACL-HLT
5. Bao Y, Quan C, Wang L, Ren F (2014) The role of pre-processing in twitter sentiment analysis. In: Huang D-S, Jo K-H, Ling Wang (eds) Intelligent computing methodologies. Springer International Publishing, Cham, pp 615–624
6. Boia M, Faltings B, Musat CC, Pu P (2013) A: is worth a thousand words: how people attach sentiment to emoticons and words in tweets. In: 2013 international conference on social computing, pp 345–350
7. Davidson T, Warmsley D, Macy MW, Weber I Automated hate speech detection and the problem of offensive language. arXiv:04009.2017
8. Dos Santos CN, de C. Gatti MA (2014) Deep convolutional neural networks for sentiment analysis of short texts. In: COLING
9. Fayyad UM, Piatetsky-Shapiro G, Uthurusamy R (2003) Summary from the KDD-03 panel: data mining: the next 10 years. ACM SIGKDD Explor Newsl 5(2):191–196
10. Gimpel K, Schneider N, O'Connor B, Das D, Mills D, Eisenstein J, Smith NA (2010) Part-of-speech tagging for twitter: Annotation, features, and experiments. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science
11. Golbeck J, Ashktorab Z, Banjo RO, Berlinger A, Bhagwan S, Buntain C, Cheakalos P, Geller AA, Gergory Q, Gnanasekaran RK, Gunasekaran RR, Hoffman KM, Hottle J, Jienjitlert V, Khare S, Lau

R, Martindale MJ, Naik S, Nixon HL, Ramachandran P, Rogers KM, Rogers L, Sarin MS, Shahane G, Thanki J, Vengataraman P, Wan Z, Wu DM (2017) A large labeled corpus for online harassment research. In: WebSci
12. Haddi E, Liu X, Shi Y (2013) The role of text pre-processing in sentiment analysis. In: ITQM
13. Hovy D, Waseem Z (2016) Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In: Proceedings of the student research workshop, SRW@HLT-NAACL 2016, The 2016 conference of the north american chapter of the association for computational linguistics: human language technologies, San Diego California, USA 12-17, 2016, pp 88–93
14. Jianqiang Z (2015) Pre-processing boosting twitter sentiment analysis? pp 748–753, 12
15. Jianqiang Z, Xiaolin G (2017) Comparison research on text pre-processing methods on twitter sentiment analysis. IEEE Access 5:2870–2879
16. Jianqiang Z, Xiaolin G (2018) Deep convolution neural networks for twitter sentiment analysis. IEEE Access PP:1–1, 01
17. Khan FH, Bashir S, Qamar U (2014) Tom: Twitter opinion mining framework using hybrid classification scheme. Decis Support Syst 57:245–257
18. Kim Y (2014) Convolutional neural networks for sentence classification. In: EMNLP
19. Kiritchenko S, Zhu X, Mohammad SM (2014) Sentiment analysis of short informal texts. J Artif Int Res 50(1):723–762
20. Kouloumpis E, Wilson T, Moore JD (2011) Twitter sentiment analysis: the good the bad and the omg! In: ICWSM
21. Lin C, He Y (2009) Joint sentiment/topic model for sentiment analysis. In: Proceedings of the 18th ACM conference on information and knowledge management, CIKM '09, New York, NY, USA, ACM, pp 375–384
22. Looks M, Herreshoff M, Hutchins D, Norvig P (2017) Deep learning with dynamic computation graphs. arXiv:1702.02181
23. Mohammad S, Kiritchenko S, Zhu X (2013) Nrc-canada: building the state-of-the-art in sentiment analysis of tweets. In: Second joint conference on lexical and computational semantics (*SEM), Volume 2: proceedings of the seventh international workshop on semantic evaluation (SemEval 2013), association for computational linguistics, pp 321–327
24. Naseem U (2020) Hybrid words representation for the classification of low quality text (Doctoral dissertation)
25. Naseem U, Musial K, Eklund P, Prasad M (2020) Biomedical named-entity recognition by hierarchically fusing biobert representations and deep contextual-level word-embedding. In: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–8
26. Naseem U, Khan SK, Razzak I, Hameed IA (2019) Hybrid words representation for airlines sentiment analysis. In: Australasian Joint Conference on Artificial Intelligence. Springer, Cham, pp 381–392
27. Naseem U, Musial K (2019) Dice: deep intelligent contextual embedding for twitter sentiment analysis. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp 953–958. IEEE
28. Naseem U, Razzak I, Eklund P, Musial K (2020) Towards improved deep contextual embedding for the identification of irony and sarcasm. In: 2020 International joint conference on neural networks (IJCNN), IEEE, pp 1–7
29. Naseem U, Razzak I, Hameed IA (2019) Deep context-aware embedding for abusive and hate speech detection on twitter. Aust. J. Intell. Inf. Process. Syst. 15(3):69–76
30. Naseem U, Razzak I, Musial K, Imran M (2020) Transformer based deep intelligent contextual embedding for twitter sentiment analysis. Future Gener Comp Syst 113:58–69
31. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: In EMNLP
32. Saeed Z, Abbasi RA, Maqbool O, Sadaf A, Razzak I, Daud A, Aljohani NR, Xu G (2019) What's happening around the world? a survey and framework on event detection techniques on twitter. J Grid Comput 17(2):279–312
33. Saeed Z, Abbasi RA, Razzak I (2020) Evesense: what can you sense from twitter? Adv Inform Retr 12036:491
34. Saeed Z, Abbasi RA, Razzak I, Maqbool O, Sadaf A, Xu G (2019) Enhanced heartbeat graph for emerging event detection on twitter using time series networks. Expert Syst Appl 136:115–132
35. Saeed Z, Abbasi RA, Razzak MI, Xu G (2019) Event detection in twitter stream using weighted dynamic heartbeat graph approach. arXiv:1902.08522
36. Saeed Z, Abbasi RA, Sadaf A, Razzak MI, Xu G (2018) Text stream to temporal network-a dynamic heartbeat graph to detect emerging events on twitter. In: Pacific-asia conference on knowledge discovery and data mining. Springer, New York, pp 534–545

37. Saif H, Andres MF, He Y, Alani H (2013) Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold. In: ESSEM@AI*IA
38. Saloot MA, Idris N, Mohd Shuib NL, Raj RG, Aw A (2015) Toward tweets normalization using maximum entropy. In: NUT@IJCNLP
39. Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inform Process Manag 24(5):513–523
40. Severyn A, Moschitti A (2015) Twitter sentiment analysis with deep convolutional neural networks. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, SIGIR ’15, New York, NY, USA, ACM, pp 959–962
41. Singh T, Kumari M (2016) Role of text pre-processing in twitter sentiment analysis
42. Suma S, Mehmood R, Albeshri A (2020) Automatic detection and validation of smart city events using hpc and apache spark platforms. In: Smart infrastructure and applications. Springer, p New York
43. Suma S, Mehmood R, Albugami N, Katib I, Albeshri A (2017) Enabling next generation logistics and planning for smarter societies. Procedia ComputSci 109:1122–1127
44. Symeonidis S, Effrosynidis D, Arampatzis A (2018) A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. Expert Syst Appl 110:298–310
45. Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. In: ACL
46. Uysal AK, Günal S (2014) The impact of preprocessing on text classification. Inf Process Manage 50:104–112
47. Yamada I, Takeda H, Takefuji Y (2015) Enhancing named entity recognition in twitter messages using entity linking. In: NUT@IJCNLP