applied sciences

MDPI

Article

Enhancement of Text Analysis Using Context-Aware Normalization of Social Media Informal Text

Jebran Khan and Sungchang Lee *

School of Electronics and Information Engineering, Korea Aerospace University, Deogyang-gu, Goyang-si 412-791, Korea; jebran.khan@kau.kr

* Correspondence: sclee@kau.ac.kr; Tel.: +82-2-300-0127

Abstract: We proposed an application and data variations-independent, generic social media Textual Variations Handler (TVH) to deal with a wide range of noise in textual data generated in various social media (SM) applications for enhanced text analysis. The aim is to build an effective hybrid normalization technique that ensures the use of useful information of the noisy text in its intended form instead of filtering them out to analyze SM text better. The proposed TVH performs context-aware text normalization based on intended meaning to avoid the wrong word substitution. We integrate the TVH with state-of-the-art (SOTA) deep-learning-based text analysis methods to enhance their performance for noisy SM text data. The proposed scheme shows promising improvement in the text analysis of informal SM text in terms of precision, recall, accuracy, and F1-score in simulation.

Keywords: social media; noisy text; informal text; LSTM; BERT; text normalization; text analysis

check for updates

Citation: Khan, J.; Lee, S. Enhancement of Text Analysis Using Context-Aware Normalization of Social Media Informal Text. *Appl. Sci.* **2021**, *11*, 8172. https://doi.org/10.3390/app11178172

Academic Editor: Luis Javier Garcia Villalba

Received: 14 June 2021
Accepted: 23 August 2021
Published: 3 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The influence of technology and ease of access to online platforms have mapped human social society to an online social society. Online social networks (OSN) have gained more popularity, and people share their experiences, opinions, and views on these OSN, usually in the form of text. This social media (SM) text is regarded as one of the key sources of information in text analysis applications for decision-making. These applications include customer reviews [1], weather reports [2], newspaper headlines [1], novels [3,4], emails [4,5], tweets [6], reviews [7] and blogs [8]. However, the quality of the text generated on SM is usually inappropriate for accurate analysis due to its informal nature. The informal text on SM contains numerous variations and inconsistencies due to the author's limited vocabulary in a specific language, cognitive and typographic spelling errors, the limit of message length, word shortening and abbreviations, use of slang, and emotional expressions.

The variations in SM informal text result in out-of-vocabulary (OOV) words, which are considered noise, and may result in inaccurate text classification [9,10]. Accuracy is critical in most text analysis applications such as review classification and ranking, cyberbully identification and control, spam detection, opinion mining, and SM analytics. For accurate analysis, the SM text needs to be tamed before applying it to the text analysis models. Most of the traditional text analysis models fail to sense such noisy words present in the text because of their lexical invalidity, which results in the loss of sentiment information. Another commonly used method to handle textual noise is to extend the lexicon by manually/automatically adding the most frequent OOV words to the list of valid words. This method is time- and resource-consuming and is highly application- and data-dependent. Text normalization is a pre-processing step before applying the text to the text analysis applications to resolve information loss and resource consumption issues.

Text normalization has been used for improvement of SM text. Normalization is the process of translating noisy and non-standard OOV words into their standard lexical

representation. Due to the massive growth of informal texts on SM, the problem of mapping them to their correct representation has received its due attention from the research community. The spell-error correction algorithm is one of the most basic and straightforward text normalization methods, where the OOV words are considered misspelled words. These algorithms normalize the noisy misspelled OOV words to their standard lexical counterpart using spell-checking algorithms to enhance the performance of the traditional text analysis methods [11,12].

Text normalization is not a "one size fits all" task of substituting OOV words with its valid replacement [13]. Besides the correction of the misspelled words, a normalization algorithm needs to handle a wide range of OOV by sensing the error patterns, identifying the error types, and activating the appropriate correction methods. Some common variations in SM text includes word enlargement (great → greaaaaaaaaaaaaaatttt), user-defined abbreviating (Good Morning → GM), word shortening (Good Night → gd nyt), phonetic substitution (wait → w8), dialectal/informal word usage (are not → aint) , words deletion (Where are you? → where?), punctuation omission (don't → dont), and censor avoidance (fuck → f***) [14]. These textual variations occur due to users' behavior of time and space-saving or emotional expression during the informal communication on the SN. The spell-error correction techniques alone cannot handle such textual variation, and they may insert the wrong substitution due to its high diversity and personalized text generation behavior.

In the social media text these textual variations can occur simultaneously i.e., different types of OOV words can occur in the same text. Therefore, there is a need to provide an ensemble technique to handle a wide range of textual variations simultaneously. Many works have suggested different techniques to handle different types of textual variations; however, these techniques are not applied simultaneously to the same text.

The aim of this paper is to provide a TVH for translating the OOV words into their actual intended words to preserve the valuable sentiment information present in these OOV words. The proposed approach is an ensemble of the previous text-normalization techniques to enhance the performance of text classification for informal communication. The proposed scheme is designed to rank the suggested candidates for the OOV words based on context, to select the best possible substitution. We applied the proposed TVH as a pre-processing to the SM text before text analysis and observe the effects on the performance of the text analysis algorithms. The main contribution of this work is as follows:

The remaining sections of the paper are organized as follows. In Section 2, the related work is discussed. Section 3 discusses the methodology and description of the techniques used in this work. In Section 4, we have discussed the dataset, results, and compared the performance of the proposed scheme with the existing methods. Section 5 is the conclusion of this work.

2. Background and Related Work

Data presentation is an important task to retrieve accurate information from unsupervised text generated on SM. In most sentiment analysis approaches, pre-processing is applied for noise reduction in the SM text [15]. The most common pre-processing operations applied include POS tagging, stemming, short-word removal, stop-word removal, removing URLs, stemming, mentions substitutions, and acronym expansions The pre-processing of the SM text reduces noise in the data, and it improves the prediction accuracy and processing time of the text classification.

Haddi et al. [16] investigated the effects of text pre-processing on the performance of text analysis in movie reviews. The experimental results significantly improve movie reviews' sentiment classification accuracy by using appropriate feature selection and representation after the text pre-processing. Tajinder et al. [17] explored the effect of slang word pre-processing on Twitter sentiment analysis. They relied on the relationship of slang words with other co-existing words to observe the significance and opinion translation of the slang words due to other words. Saif et al. [18] empirically explored the influence of

various stop-word removal methods on tweet classification by applying various stop-word removal methods to different Twitter datasets. They evaluated the impact of stop-word removal based on the variations in data sparsity, feature space size, and classification accuracy. They observed that the use of the pre-compiled stop-word list affects performance negatively. Saif et al. [19] observed a significant reduction in feature space due to text pre-processing. They reduced the word vocabulary size by 62% after pre-processing.

However, these studies do not discuss the effects of pre-processing on the performance of Twitter text classification. Bao et al. [20], and Jianqiang [21] investigated the effects of removing URLs, negation, stop words, repeated letters and numbers, acronym expansion, stemming, and lemmatization on the accuracy of Twitter sentiment classifiers. The experimental results show an increase in the accuracy of Twitter sentiment classifiers with negation replacement, acronym expansion, and repeated letter removal. However, it hardly changes with the removal of URLs, numbers, and stop words. Zhao et al. [15] provided a deep analysis of the impact of text pre-processing on Twitter text analysis considering various features models, different machine-learning classifiers on multiple Tweets datasets, in the multi-class classification task.

The conventional pre-processing steps such as removing URLs, stop words, short words, stemming, and lemmatization only ensure the removal of unwanted and non-informative text from the SM text, and it groups similar words. However, it does not ensure complete cleaning and standardization of SM text. We use different pre-processing approaches to acquire clean processable data from noisy SM text and to handle noisy SM text efficiently for text analysis. These approaches include noisy text filtering [22], lexicon creation [23,24], data-driven approaches [25], and text normalization [26].

The traditional text analysis method adopted the approach of filtering out noisy data prior to information retrieval from the SM text [22]. Xu et al. [27] and Kumar et al. [28] provided methods to clean textual data. They provided an integrated method for textual data cleaning by combining different data cleaning techniques. Chu et al. [29] presented an overview of the challenges of different textual data cleaning methods. The approach of text filtering out dwindles the size of textual data by removing the un-interpretable and non-sentiment text, leaving only the standard text (valid words). However, this may result in losing useful sentiment information due to filtering out the deliberately generated OOV words.

Several manual lexicons have been prepared successfully for text analysis to overcome the problem of sentiment information loss. The General Inquirer has a manually labeled resource of about 3600 words [30], developed for content analysis in behavioral sciences. Hu et al. [23] used a list of about 6800 manually labeled terms for opinion mining from customer reviews. Wilson et al. [31] developed an MPQA lexicon of about 8000 words with context-aware sentiment labels. Mohammad et al. built the NRC Word-Emotion Association Lexicon of about 14,000 words with sentiment and emotion labels [32,33]. The labels were compiled manually by crowdsourcing through Mechanical Turk. Shamsudin et al. [34] created a manual lexicon named "full form dictionary" to replace the OOV words by their valid substitutions. It is humanly impossible to manually create a general-purpose sentiment lexicon of valid and invalid (noisy) words for SM text. The method of manual lexicon creation is highly time and resource-consuming, and it is not possible to cover all variants of the enormous word vocabulary.

Manual lexicon creation is costly, semi-supervised, and automatic methods are presented to create language resources for sentiment analysis. Brian et al. [35] used noun-phrase co-occurrence statistics to semi-automatically generate a semantic lexicon. Amati et al. [36] used the divergence of term frequencies in a set of opinionated and relevant documents to select the subjective-term candidates for automatic generation of the term-opinion lexicon. Kiritchenko et al. [25] described the process for automatic tweet-specific lexicon creation. They provided a set of positive (30) and negative (47) hashtags to crawl keyword-specific tweets using Twitter API and marked the tweets as positive and negative based on the occurrence of the hashtagged seed words. Kaity et al. [37]

provided a semi-automatic multilingual framework for domain-based sentiment word extraction. They used lexicon-based, corpus-based, and human-based word tagging approaches. Li et al. [38] presented an effective method for automatic construction of a depression-domain lexicon. They used Word2Vec, a semantic relationship graph, and the label propagation algorithms. Tan [39] presented a robust frame-based approach to create a lexicon for domain-specific text analysis automatically. The proposed model was proven better in handling OOV targets. Other useful studies about the feasibility of using automatic lexicon creation and methods for automatic lexicon construction are presented in [40–43]. The automatic or data-driven lexicon creation method can be used to handle noise in SM text by developing a lexicon of the noisy words; however, it is strictly domain-specific and application-dependent.

Generally, text normalization is used to avoid the limitations of information loss and application dependencies in text analysis applications. Normalization is translating noisy SM text into its canonical (standard) form before text analysis for better results. Text normalization is proved to be an important pre-processing step for noise-handling in natural language processing (NLP) applications, such as sentiment analysis [26], machine translation [44], text accessibility [45], spam filtering [46], and text to speech synthesis [47] etc.

The baseline algorithm for text normalization is spelling-error correction, where the OOV words are considered to be misspelled words. Spelling correction is a method to correct spelling errors due to deliberate abbreviating or typing mistakes. The spelling correction technique was used by [11,12] in their work for converting (normalizing) noisy words into meaningful replacements. The techniques that are used for spelling correction are edit distance [48,49], similarity key [48,49], rule-based [48,49], N-gram [49], probabilistic [49], neural network [50], and auto-encoders [51].

Spelling correction is a well-established domain, with advances in pattern-matching techniques and the development of n-gram analysis techniques that have improved over the last two decades [52]. However, the scope of problems introduced by user-generated content in online SM platforms exceeds the range of simple spelling correction. Other problems include rapidly changing OOV slangs, phonetic spelling, punctuation errors and omissions, misspelling for verbal effect and other intentional misspelling, short-forms and acronyms, and recognition of OOV named entities [53]. These algorithms perform poorly, as they miss user behavior of intentionally generating OOV words on SM.

In the last decade, researchers have actively worked for improvement in SM text normalization using different techniques. Han and Baldwin [54] used linear SVM and morpho-phonemic similarity for OOV identification and standardization. Chrupala et al. [55] proposed a model for SM text normalization based on Conditional Random Field (CRF) and Recurrent Neural Network (RNN) for learning sequence of editing operations and neural text embedding, respectively. Liu et al. [56,57] investigated the human perspective of word normalization, including enhanced letter transformation, visual priming, and string/phonetic similarity. Sprout et al. [58] developed a taxonomy for OOV words based on four distinct types of texts to propose a more generalized approach for text normalization which can handle new texts. However, this approach is still data/domain-dependent. A fully automated and statistical approach based on n-grams was developed in [59] for text normalization. A rule-based adaptive approach was proposed in [60,61] for SM text normalization by selecting the normalization method based on the type of error in the OOV words. Doshi et al. [62] proposed a text normalization method based on phonetic and string similarity. A Hidden Markov Model (HMM) approach was adopted in [63], to model all correction possibilities for each word in the corpus, weighted with their occurrence probabilities for ranking based on the nature and type of the SMS text. In [64,65] a probabilistic approach was proposed based on the Trie data structure for text normalization. This approach was reported to perform better than the HMM-based approach [64].

Normalization is applied to SM text applications as a pre-processing step in many works to improve the accuracy of the text analysis applications. Alexandra [66] applied rule-based text normalization to the Twitter data for accurate sentiment classification.

Sharma et al. [26] proposed a normalization model for code-mixed multilingual text to investigate its impact on sentiment analysis of Hindi–English mixed tweets. Monika and Vineet [67] applied the character-level embedding method with deep CNN to normalize SM text for sentiment analysis of Twitter data. Arnold et al. [68] used a Genetic algorithm trained Bayesian network to normalize noisy features in SM text for enhanced spam filtering. Other more recent studies on text normalization for enhanced sentiment analysis in different languages are presented in [69–71].

These normalization approaches can overcome the information loss and domain dependency problem; however, these approaches are not generalized. They mostly ignore the context of the intended words by substituting the OOV words with the nearest replacements. Moreover, most of the existing techniques normalize isolated word problems. However, in informal conversation, dialectal usage and word deletion are widespread, which can only be handled by sentence-level normalization.

3. Proposed Sentiment Analysis Method

In this paper, we proposed a new strategy for text analysis of informal SM textual data to use the useful information in the noisy SM text. We proposed a new method to handle textual variations in SM text by context-aware substitution of OOV words with its valid replacements, named TVH. The proposed TVH can handle a wide range of textual variations which are very common in SM text. We integrate text classification methods with the proposed TVH to enhance its performance of in the case of noisy SM text. Figure 1 shows the block diagram of the methodology of our proposed text analysis scheme.

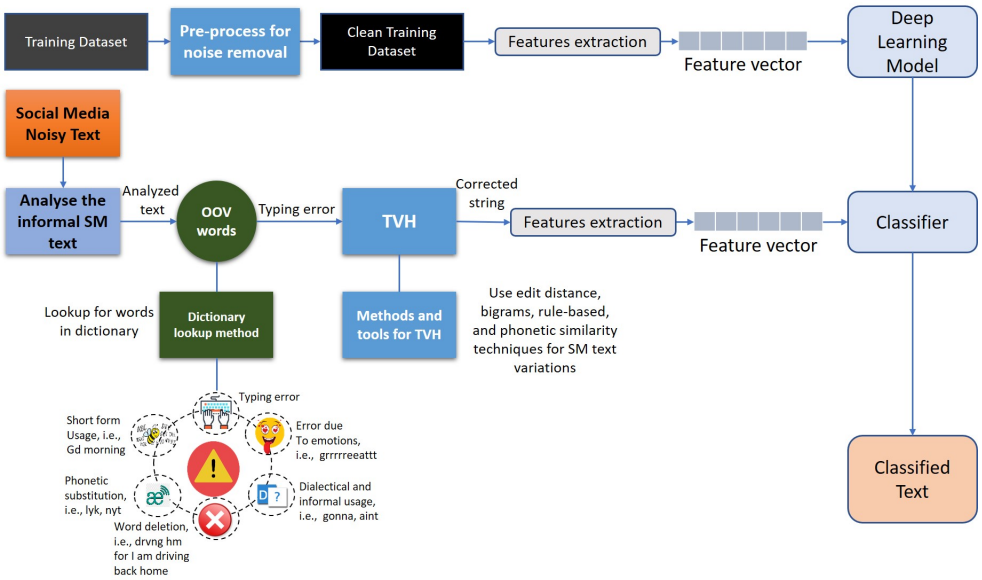


Figure 1. Block diagram of the proposed text analysis scheme for informal social networks text.

In text analysis applications based on machine/deep-learning techniques the ideal case is to have clean data for accurate classification. However, the social network data contains a vast range of variations which distort the performance of the text classification. It is inevitable to either clean/normalize the SM text or embed the intentionally generated OOV words in the lexicon for accurate classification. Both approaches have limitations: the former approach is time- and resource-consuming due to huge volume of training data required for model training whereas the latter method falls short in coverage of different intentionally generated variations due to users' different behaviors of text generation. In this work, we adopted the approach to train the text analysis model with clean SM text and normalize the input SM text before feeding it to the classifier as shown in Figure 1.

Algorithm 1 shows the stepwise pseudocode of the proposed text analysis model. Load clean and labeled SM text as training data, and feed noisy SM text as test/input to

the system. Pass the training data to from the Preprocess(TD) function. The Preprocess(TD) function clean the text from stopwords, punctuations, URLs, tags and mentions. Define a deep-learning network architecture for text analysis by setting its parameters. The actual values and options which we used in this work are discussed in Section 3.2. Train the text analysis model using the preprocessed clean labeled SM text. We used different training datasets for different applications such as sentiment analysis, cyberbullying identification, spam detection and reviews analysis. Preprocess the test/input data using the function Preprocess(TsD). Pass the preprocessed input data from the proposed TVH(PTsD) function. The TVH(PTsD) function normalizes the SM text by substituting the OOV words with their context-appropriate replacements as shown in Figure 2 and Algorithm 2.

Algorithm 1: Step-wise approach for text analysis of informal SM text

Output: Text_Category
Input: textDataTrain; Load clean and labeled SM text as training data (70%);
textDataValidate; clean SM text (15%);
textDataTest; Noisy SM text (15%);
Preprocess the training text data (*TD*) with function ***Preprocess(TD)***;
Create a deep-learning network architecture by setting network parameters as following;
layers = *Settings from Table 1 or Table 2*;
options = *Options from Table 1 or Table 2*;
net = ***trainNetwork(XTrain, YTrain, layers, options)***;
while *D* **do**
    Preprocess the test data (*TsD*) with function *PTsD* = ***Preprocess(TsD)***;
    if *ErrorDetect(PTsD, dict)* **then**
        Normalize the text using Normalized_text = ***TVH(PTsD)*** as shown in Algorithm 2;
    else
        Normalized_text = PTsD
    end
    Text_Category = classify(net, Normalized_text);
end

Algorithm 2 shows the pseudocode for the proposed TVH. The function TVH(PTsD) handles multiple types of most commonly occurring (un)intentional textual variations in SM, which results in OOV words. These variations include word enlargement, abbreviations, word shortening, spelling errors, phonetic substitution, dialectal usage, and word deletion. We used rule-based approaches, discussed in Section 3.1.2, to handle these textual variations, which gives a list of valid words which are the closest to the OOV words. We adopted two types of approaches to handle these textual variations i.e., word-to-word substitution (WtWS) and word-to-phrase substitution (WtPS). The WtWS was adopted to deal with OOV words which result from character-level errors, whereas the WtPS is used for word-level errors such as word deletion, word abbreviations etc.

We ranked the list of valid words based on the context of the sentence to select the most appropriate substitution. For context estimation we adopted an n-gram-based approach.

Algorithm 3 shows the stepwise approach for spell-checking process. The block diagram of the spell-checker is shown in Figure 3. In this method we aim to deal with three types of normalizations, i.e., word enlargement, word shortening and typo spelling errors. The input text is passed from the spell-checker, where WEP(D[i], dict) resolves the word enlargement problem by removing the consecutive repeated characters. The WSP(D[i], Sdict) function is used to resolve the problem of word shortening. We made a new list of valid words *Sdict* by removing all the vowels from the valid words in the original dictionary leaving only starting vowels. The function WSP(D[i], Sdict) uses the edit-distance algorithm to find the nearest neighbors for the misspelled words in the new

dictionary *Sdict*. The nearest neighbors are then compared with the actual dictionary to extract and return its corresponding word as a candidate.

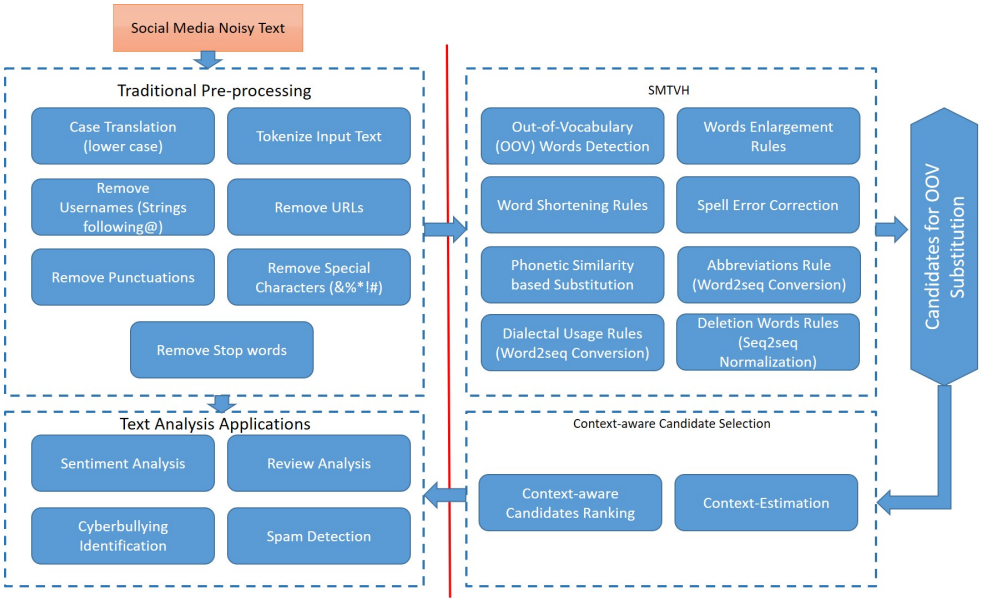


Figure 2. Block diagram for the proposed SM textual variations handler (TVH) and its integration with current text analysis.

Algorithm 2: Proposed social media textual variations handler (TVH) algorithm

Result: Normalized_text
Preprocessed test data (*PTsD*);
while *i*<*n* **do**
  if WtPS(PTsD) then
    User-generated abbreviations problem $Cand_Normalized = UGAP(PTsD)$;
    Dialectal/informal usage problem $Cand_Normalized = DIUP(PTsD)$;
    Word-deletion problem $Cand_Normalized = WDP(PTsD)$;
  else
    Resolve word enlargement problem, word shortening, and typo spelling errors as discussed in Algorithm 3 *Cand_Normalized* = ***SpellCorrection****(PTsD,* ***dict)***;
    Cand_Normalized = Resolve phonetic-based words substitution using Metaphone algorithm [72];
  end
  Normalized_text= ***CandidateRanking(Cand_Normalized, tri-gram(dict))***
end

The typo spelling correction TypoCorrection(D[i], dict) is a combination of two widely used string-proximity estimation methods, i.e., edit distance and bi-grams. The edit distance provides a list of correct words from the dictionary which are the nearest neighbors of the misspelled word. The bi-grams of each word, in the list of nearest neighbors, and the misspelled word are matched to return the nearest neighbor with maximum bi-grams matched with the misspelled word. The word returned is the correct replacement for the misspelled word.

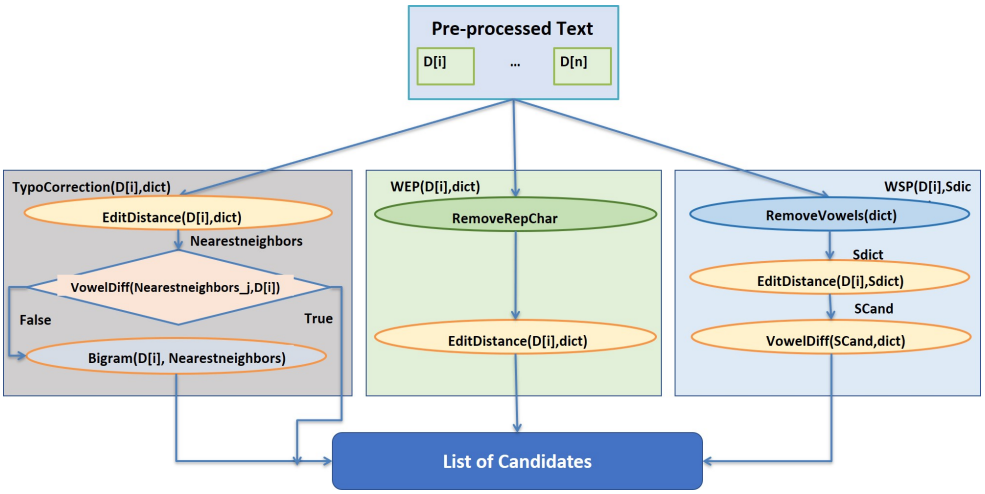


Figure 3. Block Diagram for spell-checking process.

Algorithm 3: Algorithm for enhanced spell-checking

Result: *Cand_Normalized* ;
Input: Preprocessed test data (*D*);
while *i*<*n* **do**
  Resolve word enlargement problem *Cand_Normalized* = ***WEP(D[i], dict)***;
  Resolve word shortening problem *Cand_Normalized* = ***WSP(D[i], Sdict)***;
  Resolve typo spelling errors *Cand_Normalized* = ***TypoCorrection(D[i], dict)***;
  function TYPOCORRECTION(PTSD, DICT);
  (*Nearestneighbors*) = ***EditDistance(D[i], dict)***;
  if VowelDiff(Nearestneighbors_j,D[i]) then
    Cand_Normalized = (*Nearestneighbor_j*);
  else
    Cand_Normalized = ***Bigram (D[i], Nearestneighbors)***;
  end
  end function;
end

The algorithm is repeated until the last word D[n] in the document, where *n* is the number of words in the document.

3.1. Methods Used for Text Normalization

The TVH task consists of two sub-tasks; OOV word detection and OOV word normalization.

3.1.1. Out-of-Vocabulary (OOV) Words Detection

OOV word detection is the process of examining the linguistic validity of the target words in the lexicon of a language. OOV words are detected if linguistically invalid words are found in a document. Generally, a dictionary lookup method is used to detect OOV words.

The dictionary lookup is a direct method, in which each word in the input document is examined/compared directly against the list of valid words, known as the lexicon/dictionary. If there exist words in the input document which are not found in the lexicon, an OOV word is detected. The errored words are stored as a list with proper index, and the TVH algorithm is revoked, by passing the list of invalid words and dictionary, to suggest a context-aware replacement for the OOV words from the dictionary.

3.1.2. Social Media Text Normalization

Context-aware normalization is the process of replacing the OOV words with its lexically valid substitute which is the most likely to be intended in the target context. In this work we used a combination of rule-based, statistical, probabilistic methods, and machine-learning methods, depending on the type of errors, to normalize the OOV words which are frequently found in SM text. We have divided this problem in two classes, i.e., word-to-sentence substitution (WtPS) and word-to-word substitution (WtWS).

3.1.3. Word-Level Substitute Prediction (WLSP)

In this type of normalization method, the information about words is only user-defined. We divided the WLSP into two categories based on their output i.e., word-to-phrase/sequence substitution (WtPS) and word-to-word substitution (WtWS). These categories and their respective normalization techniques are discussed in Sections 3.1.4 and 3.1.5, respectively.

3.1.4. Word-to-Phrase Substitution (WtPS)

User-Generated Abbreviations Problem

The list of standard abbreviations is part of the dictionary in a language. In SM it is very frequent to generate abbreviations from very commonly occurring phrases or words pairs, such as "Good Morning → GM". These abbreviations are user-generated, which is not standard. We collected the 1365 most commonly used abbreviations from multiple web sources to make a list of new abbreviations which are not included in the lexicon, to resolve the problem of SM user-generated abbreviations. These abbreviations include tbh → to be honest, tldr → too long didn't read, dyk → did you know, etc. These abbreviations are very difficult to identify/correct using existing normalization methods.

Dialectal/Informal Usage Problem

Dialectal usage is replacement of multiple words with one or more OOV words where each word is a combination of words such as "are not → aint". Such errors cannot be corrected by the above word-level normalization methods. This problem can be resolved by a word-to-sequence approach.

Word-Deletion Problem

The problem of word deletion is similar to that of character deletion-based errors. This can be a word-to-sequence transformation or a sequence-to-sequence translation. In SM text to reduce the message length, some words, which are not needed for other users to understand the message, are omitted such as "Where are you? → Where?". However, for machine understanding, these words may be necessary. The LSTM technique is well suited for both word-to-sequence and sequence-to-sequence normalization problems.

We used the LSTM-based word-to-sequence model for converting the abbreviated word to its intended sequence. We created an LSTM model in MATLAB with the same settings as discussed in Section 3.2.1.

3.1.5. Word-to-Word Substitution (WtWS)

Word Enlargement

In informal SM text, word enlargement is commonly used to express enhanced emotional intensity, such as "Good Morning → goooooooooooood morrrrrrninggg" etc. As a matter of fact, in English it is very unlikely for a character to repeat more than once. We have reduced the character repetition to single characters where a similar character occurs more than two times consecutively. We left the problem of double character occurrence to be resolved by simple spell-checkers if the suggested word is still not a valid word.

Word Shortening Problem

In modern online conversation there are two types of short-word usage, i.e., vowel removal and using half a word. In English language consonants are the sound producer whereas the vowels are used to rise, or lower the sounds of phonemes in the language. In informal communication on SM there is a trend of writing mostly using consonants and excluding most vowels from words, except the starting vowel, to reduce the message length. In this work we process the OOV words to mine consonant only words, such as "Good Night → gd nght" for restoration by adding vowels at the appropriate places.

Spelling-Error Correction

We used edit-distance method to find out the nearest neighbors. The edit distance works on the principle of minimum editing operations. In the case of spelling-error correction the edit distance of two words is the minimum number of editing operations required to replace one word by another. The editing operations can be insertion, deletion, substitution or transposition of characters. In this method a list of words which have minimum edit distance from the misspelled word is extracted from the lexicon. This list is then passed from a bi-gram matching method.

Bi-grams are sub-strings of every 2 adjacent characters in a string. In this method the target words, the misspelled word and words obtained from minimum edit-distance method, are broken down into sequences of 2 characters at a step of 1 character until the end of the word. The bi-grams of the misspelled word are checked against the corresponding bi-grams of each word in the list. Each word in the list obtained from the edit-distance method are checked with the misspelled word to see their difference. If the difference are only vowels, that word is the correct replacement. Otherwise, each element in the list is scored as the number of its bi-gram matching with the misspelled word. The word with the maximum bi-gram score is considered to be the correct word and the misspelled word is replaced by this correct word.

We used and enhanced version of [11] for word enlargement, word shortening and spelling-error problems. The enhanced algorithm is explained in Section 3 Algorithm 3.

Phonetic Similarity

In SM text, a user can replace characters or words such as phone, ate, to, and too etc. by its phonetically similar characters such as fone, 8, 2, and 2, respectively to reduce the number of characters in SM conversation/posts. Such deliberate character substitution results in phonetic similarity-based OOV words. The phonetic similarity-based errors can also be cognitive errors, where users do not know the spelling of a word e.g., compare → Kompare, what → wat etc. Phonetic similarity-based noise in SM text is handled by the phonetic hashing method. We used the Metaphone algorithm [72] to generate the normalized candidates based on phonetic similarity of words using the Soundex algorithm [73].

3.1.6. Sentence-Level Substitute Prediction (SLSP)

Sentence-level normalization is also referred to as context-aware. After the candidate generation they are ranked with respect to their relevance in the intended context. At this stage we have a list of n candidates for each OOV word. This sub-module is to re-rank the list of candidates to find the best context-aware substitute for the target OOV word. We used the n-gram-based conditional probabilities dictionary for context understanding. We added the n-gram (tri-gram) probability to the features of the language model (LM) of each candidate substitute as context information. Each tri-gram contains a combination of the candidate, a previous word, and the next word of the target OOV word. We searched for the tri-grams, in the tri-gram dataset [74], and sentences corpus from Natural Language Corpus Data: Beautiful Data (http://norvig.com/ngrams/, accessed on 22 March 2021), which are similar to each tri-gram of OOV word substituted by the suggested word. The suggested

word with highest number of occurrences in the extracted tri-grams is considered to be the correct substitution.

3.2. Methods Used for Text Analysis

Text analysis is the process to computationally identify, classify and categorize the opinions, behavior, and experiences articulated in a piece of text. It is data-mining technique to measure author opinion disposition leveraging the tools and concepts of natural language processing (NLP), data sciences and data mining.

We input the preprocessed and normalized textual data to the text analysis module. For simulation we used two state-of-the-art text analysis techniques, i.e., long short-term memory (LSTM) and bidirectional encoder representations from transformers (BERT).

3.2.1. Long Short-Term Memory (LSTM)

LSTM is a deep-learning algorithm based on recurrent neural network (RNN) architecture. An LSTM network can learn long-term dependencies between time steps of sequence data. Theoretically, the traditional RNNs can keep track of arbitrary long-term dependencies in the input sequences. However, in practice, RNNs are not capable of learning these dependencies. The RNNs work well in simple cases where the gap between the relevant information and the target location is small. Moreover, in classic RNNs, there is a problem of vanishing gradient in the back-propagation while training. The LSTM was able to learn long-term dependencies and partially resolve the issue of vanishing gradient.

The LSTM is well suited to classify, process, and predict from sequences in data such as sequence in time series data, and textual data (naturally sequential) etc. Any text is a sequence of words (sequence of characters) which may have dependencies between them. The LSTM network is advised to be used for the classification of sequence data where long-term dependencies are used for accurate classification. The detailed architecture of LSTM and implementation method is given on Github (https://colah.github.io/posts/2015-08-Understanding-LSTMs/, accessed on 16 February 2021) and Mathworks (https://www.mathworks.com/help/textanalytics/ug/classify-text-data-using-deep-learning.html accessed on 30 November 2020), respectively. The settings and options of our LSTM model is provided in Table 1.

Table 1. Settings and options of our LSTM model.

	Attribute	**Value**
Settings	Sequence Input	Sequence input with 1 dimensions
	Word Embedding Layer	Word embedding layer with 100 dimensions
	LSTM	LSTM with 180 hidden units
	Fully Connected	N fully connected layer, N is numClasses
	Softmax layer	softmax
	Classification Output	crossentropyex
	numClasses (N)	Number of categories in training dataset
Options	Training Solver	Adam Solver
	mini-batch size	16
	Shuffle	Shuffle the data every epoch
	Gradient Threshold	1
	Initial Learning Rate	0.01
	Validation data	yes
	Verbose	"false" to suppress verbose output

3.2.2. Bidirectional Encoder Representations from Transformers (BERT)

BERT [75] is a state-of-the-art pre-trained encoder stack. BERT is regarded for achieving high performance in many NLP tasks. The strength of this model is its highly pragmatic approach, and its training on huge datasets such as Wikipedia and BookCorpus. The datasets consist of more than 10,000 books from various genres. BERT is made using trans-

formers, which enhances the performance of the model. The keystone of the transformers is the self-attention mechanism. BERT transformer uses bidirectional self-attention, where each token can attend to context from both directions [75]. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. BERT uses WordPiece embedding [76] with a vocabulary of 30,000 tokens. The WordPiece model is a data-driven approach, where the words to be processed are first broken into sub-word units (wordpieces) of the most frequent sequences of words in the vocabulary. The WordPiece algorithms ensure normalization of many OOV words by its segmentation mechanism to allow only the valid vocabulary sequences (characters combination) [77]. Besides its ability to normalize some of the OOV words, there are some OOV words/sequences which cannot be handled by only using BERT. We have used the BERT implemented in MATLAB 2021b. The basic architecture and implementations are available on Toward Data Science (https://towardsdatascience.com/bert-for-dummies-step-by-step-tutorial-fb90890ffe03, accessed on 30 March 2021) and Github (https://github.com/matlab-deep-learning/transformer-models, accessed on 10 April 2021), respectively. We have used the BERT base uncased model which consists of 12 layers, 768 hidden, N fully connected layers. The model is trained on lower-case English text. The settings and options of our BERT model is provided in Table 2.

Table 2. Settings and options of our BERT model.

	Attribute	**Value**
Settings	Features input layer	768 features
	Number of Layers	12 layers
	Fully Connected	N fully connected layer
	Softmax layer	softmax
	Classification Output	crossentropyex
	num Classes	Number of categories in training dataset
Options	Training Solver	Adam Solver
	mini-batch size	64
	Shuffle	Shuffle the data every epoch
	Validation data	yes
	Verbose	"false" to suppress verbose output

In our model, we first train a text classifier on the cleaned training dataset. In the next step we passed the preprocessed, normalized test data from the classifier to classify the text into their respective categories.

4. Results and Analysis

4.1. Dataset Properties

In this work we used different datasets to evaluate the performance of our proposed scheme in different text analysis applications. These applications include sentiment analysis, cyberbullying classification, spam detection, movie reviews classification, news reports categorization and mobile app reviews classification. The details of the datasets used are shown in Table 3. The data set contained comparable ratios of each class of sentences. However, most datasets are unbalanced. We divided the dataset into training, validation, and test sets by ratio of 70%, 15%, and 15% respectively.

Table 3. Descriptions of the datasets used.

Dataset	Description	(Un)balanced	Application	Classes	Fraction of Textual Variations
Sentiment [78]	13,871 Tweets dataset	Unbalanced	Tweets polarity	Positive, Negative, Neutral	High
Cyberbullying [79]	13,040 comments from social media	Unbalanced	Cyberbullying Classification	Racism, Abuse, Sexism, Opposition,	Medium
Spam [80]	5572 spam and other comments	Unbalanced	Spam emails classification	Spam, ham	Low
Movies reviews [81]	50,000 movie reviews dataset	Balanced	Reviews classification	Positive, negative	Medium
Topics Classification [82]	1,000,000 news articles 4 different topics	Balanced	Topics classification of news articles	Sports, Entertainment, Medical, and Politics	Medium
Covid apps Reviews [7]	23,092 reviews of COVID-19 apps on App-store	Unbalanced	Reviews classification	Positive, Negative, Neutral	Medium

4.2. Experimental Tools and Environment

The simulation was conducted on an LG system (LG Electronics Nanjing Displays Company Ltd., Nanjing, China), with Corei5, 2.3 GHz processor, 8 GB RAM, with operating system Windows 10Pro64-bit (Microsoft, Albuquerque, NM, USA). For simulation we used MATLAB 2019b and MATLAB 2021b (Mathworks, Inc., Natick, MA, USA). We used the deep-learning toolbox, machine-learning statistical toolbox, and BERT toolbox.

4.3. Simulation Results

We simulated the proposed text analysis pipeline to evaluate and compare its performance with the existing text analysis methods. From simulations, we observed that the proposed text analysis pipeline performs better than the previous methods in the case of noisy/informal SN data.

For performance evaluation of the proposed sentiment analysis strategy, we used accuracy, average precision (AP), average recall (AR), F1-score, and Accuracy. The average precision (AP) is the sum of precision (P_{ci}) of all classes divided by number of classes (N) in the dataset as shown in Equation (1).

$$AP = \frac{\sum_{ci=1}^{N} P_{ci}}{N} \tag{1}$$

The average recall (AR) is the mean of the recall (R_{ci}) for all classes, as shown in Equation (2).

$$AR = \frac{\sum_{ci=1}^{N} R_{ci}}{N} \tag{2}$$

where P_{ci} and R_{ci} are the precision and recall of class ci, respectively. The precision is defined as the ratio of true prediction of class ci to the number of all predictions of class ci.

$$P_{ci} = \frac{TP_{ci}}{PR_{ci}} \tag{3}$$

The recall is defined as the number of true predictions of class ci divided by the total number of occurrences of class ci in the dataset.

$$R_{ci} = \frac{TP_{ci}}{T_{ci}} \tag{4}$$

TP_{ci}, PR_{ci}, and T_{ci} are the true predictions, total predictions, and total number of elements of class ci, respectively.

In textual communication the text can be noisy or clean; however, the SM text is usually very noisy, due to its informal nature. We conducted simulations of four different cases for analysis of SM text. These cases include clean training and test (ideal case), noisy training and test (real case), clean training and noisy test (very poor performance), and clean training and preprocessed (TVH) test (proposed method and can be realized). For simulation we used the spam dataset. For the first case i.e., clean training and test, we used manually normalized (cleaned) dataset. We have introduced some errors manually in the spam dataset to make it noisy. In the second case we have used the noisy training and test data, while as a third case we used the manually normalized training data and the noisy test data for simulating. In the last case we trained the text analysis model using the manually normalized data and test data normalized using the proposed TVH. In Table 4, the results of different possible scenarios is shown.

From simulation we observed that the results of our proposed text analysis pipeline are the closest to the ideal case. The advantage of our proposed method is its independence of the wide range of textual variations. In the existing methods, classification is done by learning from data. The existing techniques (noisy training and text) works well for the text with a narrow range of variations and unique texts. However, in SM text, there is

wide range of textual variations and hence this method is not suitable for SM informal text analysis.

Table 4. Simulation of different simulation scenarios of SN text analysis.

Algorithm	**LSTM**				**BERT**			
Scenario	**Precision**	**Recall**	**F-Score**	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Accuracy**
Clean training and test	84.01	95.58	89.94	93.65	97.46	98.23	97.84	99.04
Noisy training and test	76.11	87.70	81.49	88.50	89.04	89.99	89.51	95.33
Clean training and Noisy test	70.36	79.76	74.77	84.55	82.23	83.79	83.00	92.34
Clean training and TVH test	86.39	93.46	89.79	95.55	94.59	96.39	95.48	97.96

We also observed an improvement in performance of the text normalization while tackling the wide range of OOV words with context consideration. For performance evaluation of spell-checking we used precision (P), recall (R), and F1-scores. The precision and recall are calculated using Equations (5) and (6), respectively.

$$P = \frac{TC}{TC + WC} \tag{5}$$

$$R = \frac{TC}{TC + NC} \tag{6}$$

where TC, WC, and NC are true corrections, wrong corrections, and no corrections. We consider that for any misspelled word there exists a valid dictionary replacement word; therefore, we considered that no corrections NC as false negatives.

Table 5 shows the comparison of the proposed TVH with the current normalization methods. From simulation we observed that the proposed normalization method outperformed the existing methods, due to the hybrid nature of noise handling and context awareness. The proposed TVH method is simulated on the spell-check dataset [83], and other manually collected data such as list of dialects, and list of abbreviated data. Moreover, we have compared the performance of the proposed TVH with the deep-learning (RNN)-based normalization method. The performance of the proposed scheme is better due to the standardizing of the textual variations instead of learning these variations from data because there is wide range of possible textual variations in SM text, and it is not possible for the deep-learning methods to learn all the possible variations. The proposed TVH is better due to its hybrid nature, and it can handle different types of textual variations simultaneously.

Table 5. Comparison of the proposed TVH with other SM text normalization techniques.

Algorithm	**Precision**	**Recall**	**F-Score**	**Accuracy**
Textblob	50.81	51.73	51.27	53.65
Norvig	69.59	68.70	63.83	69.98
Hunspell	62.83	73.06	67.56	75.57
Jamspell	67.17	77.21	71.84	81.44
Manual Lexicon	58.80	67.58	62.89	68.02
Spell-checker	60.39	69.74	64.73	71.77
Probabilistic	67.32	80.70	73.41	79.64
Deep-Learning-based	74.99	83.65	79.09	88.02
Proposed Spell-Checker	63.74	75.58	69.16	75.36
Proposed hybrid method (TVH)	81.14	88.06	84.46	91.74

We observed that the performance of the overall pipeline is dependent on the performance of the proposed TVH, as it handles the noise by automatically correcting it before it is put into the text classifier. If the performance of the TVH is poor, it will replace the OOV words with the wrong suggestion, which may result in wrong classification. The existing schemes performed well in normalization of word-to-word substitution, without giving much regard to the context. One of the main advantages of the proposed method is that it normalizes text with respect to the context of the words/phrases. The proposed scheme also outperformed the existing methods when it comes to word-to-phrase- and sentence-level normalization.

The performance of the text analysis is directly related to the fraction of noise in the data. As shown in Table 6, the lower the variation in text, the higher the performance of the text analysis. We also observed from the simulation that when the fraction of variations in data is low, the improvement in performance with the proposed text analysis pipeline is also low as there exists less margin of textual variation correction and hence less improvement in performance. Like the other methods, the proposed TVH is also not an ideal normalization method. From simulations, we observed that the proposed TVH can also sometimes degrade the performance of the text analysis pipeline in the cases of text with little variation. The performance of the BERT in the case of spam (low variations) classification was degraded when the test data were normalized by TVH due to the possibility of introducing more textual variations by wrong substitutions.

Table 6. Effects of the proposed model with respect to textual variations.

		Without TVH				**With TVH**				
	Textual Variations Level	**Precision**	**Recall**	**F-Score**	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Accuracy**	**Improvement**
LSTM	Less Variations (Spam)	74.30	80.67	77.35	91.62	86.39	93.46	89.79	95.55	3.93%
	High Variations (Tweets)	54.76	52.48	53.59	64.04	74.19	70.63	72.37	77.40	13.36%
BERT	Less Variations (Spam)	96.40	97.66	97.03	98.68	94.59	96.39	95.48	97.96	−0.72%
	High Variations (Tweets)	61.83	56.94	59.28	68.94	77.82	74.91	76.34	80.87	11.93%

We performed extensive simulation of the proposed text analysis pipeline on text in various text analysis applications. These applications are mentioned in the previous Section 4.1, along with the descriptions of the datasets used for simulation of each application. Table 7 shows the effect of the proposed TVH on the performance of the two state-of-the-art (SOTA) text analysis methods. In the simulation we trained the text analysis model using clean training, and validation data. We have checked and cleaned the training, and validation data manually. We normalized the noisy test data using the proposed TVH, and then passed the normalized test data from the text classifier. We observed significant improvement in the performance of these existing method when integrated with our proposed TVH.

From the experiments it was found that the average accuracy of our system in text normalization is about 91%, and the average improvement in the accuracy of our proposed text analysis pipeline is about 6% for LSTM and 4% in the case of BERT. The improvement in the case of BERT was lower than the LSTM-based text classification method; however, the overall performance of the BERT-based text classification was far better than the LSTM-based text classification method. The overall result depends on the performance of the spell-checker as well as the type of text. The maximum improvement in the accuracy was up to 12% when the proposed TVH was applied to the very noisy text.

The performance of the proposed system is improved in terms of accuracy; however, the combination of normalization with sentiment analysis introduced a processing time overhead. However, this integration is inevitable for more accurate prediction, without losing useful information and process automation (noise detection and correction).

Table 7. Performance of the proposed TVH in the classification of text in various applications.

		Without TVH				With TVH			
	Application	**Precision**	**Recall**	**F-Score**	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Accuracy**
LSTM	Sentiment Analysis	54.76	52.48	53.59	64.04	73.04	69.67	71.31	76.59
	Cyberbullying Identification	81.01	74.09	77.40	77.35	83.05	80.64	81.83	82.04
	SPAM	74.30	80.67	77.35	91.62	86.39	93.46	89.79	95.55
	Reviews	76.05	72.26	74.33	72.69	85.04	84.05	84.54	84.05
	Topic Modeling	71.02	69.03	70.01	68.97	76.89	76.43	76.66	76.44
	Covid Apps Reviews	59.88	58.70	59.28	80.31	71.73	65.01	68.21	87.06
BERT	Sentiment Analysis	61.83	56.94	59.28	68.94	77.82	74.91	76.34	80.87
	Cyberbullying Identification	86.65	88.96	87.80	86.91	92.90	91.17	92.03	91.36
	SPAM	96.40	97.66	97.03	98.68	94.59	96.39	95.48	97.96
	Reviews	87.04	87.80	87.42	87.33	90.95	90.91	90.93	90.91
	Topic Modeling	87.33	86.29	86.81	86.78	88.71	87.75	88.23	88.51
	Covid Apps Reviews	66.93	74.79	70.64	88.40	83.18	73.96	78.30	90.63

5. Conclusions

In this paper, we simulated a new method for text analysis of informal/abbreviated text data using the deep-learning-based text classification model integrated with the proposed TVH. We used two state-of-the-art text analysis models, i.e., LSTM-based text classification, and BERT-based text classification. We proposed a new context-aware generic normalization method to handle noise in SM text without losing useful information in this text. The proposed TVH handles a wide range of SM textual variations. These variations include word-to-phrase/sequence transformation, and word-to-word transformation. The word-to-sequence normalization is used to tackle user-generated abbreviations, dialectal usage, and word-deletion problems in the SM text. The word-to-word normalization include typos, word enlargement, short words and phonetic similarity-based OOV words handling.

The performance of the text analysis is inversely proportional to the fraction of noise in the data. The more the noise in the data the poorer the text analysis performed. BERT performs better than the LSTM-based text analysis in all cases. However, there is room for improvement in both cases, depending on the fraction of noise. From simulations, we observed that the proposed TVH can be used as a pre-processing tool to improve the performance of the text classification significantly. In our experiments we observed that the average improvement is higher in the case of noisier data.

We compare the performance of the proposed text analysis pipeline with the existing state-of-the-art text analysis methods to observe the performance when noise is removed from the SM text. We observed from simulation that the performance of the text analysis methods is improved when integrated with the proposed normalization scheme. The average improvement in the performance of the proposed text analysis pipeline was up to 6% where the maximum improvement was about 12% in the case of very noisy textual data.

As we know, the existing state-of-the-art text analysis methods are based on deep learning, which are data-dependent and data-driven approaches. Therefore, the performance of the text analysis depends on the fraction of noise in the text data. We observed that the text analysis methods performed well on the text with less textual variations; however, in the case of high-variation data the performance is poor.

We applied the proposed model to different text analysis applications such as tweet classification, review analysis, spam detection, cyberbullying detection, news article classification and COVID-19 apps review classification. The proposed model is generic as it performs well in different applications scenarios.

From experiments, we found that the performance of our proposed model was superior in the case of informal/abbreviated social network text data than the existing methods. Our system was able to tackle different variants of text data to improve the performance of the text analysis models. Our system correctly identified most of the alternatives of the text, and it updated the dictionary used in this work for future occurrences of these words, which minimizes the normalization overhead.

The performance of our model depends on both the performance of the TVH and text analysis system. Our normalization method and text analysis model outperforms the existing methods in terms of precision, recall, accuracy, and F1-scores.

Author Contributions: All authors contributed equally to this work. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by National Research Foundation of Korea (NRF) Grant funded by the Korean Government (Ministry of Science and ICT) NRF- 2020K1A3A1A47110830.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bellegarda, J. Emotion analysis using latent affective folding and embedding. In Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, Los Angeles, CA, USA, 5 June 2010; pp. 1–9.
2. Boucouvalas, A.C. Real time text-to-emotion engine for expressive internet communications. In Proceedings of the International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP-2002), Staffordshire, UK, 15–20 July 2002.
3. John, D.; Boucouvalas, A.C.; Xu, Z. Representing Emotional Momentum within Expressive Internet Communication. In Proceedings of the EuroIMSA, Innsbruck, Austria, 13–15 February 2006; pp. 183–188.
4. Liu, H.; Lieberman, H.; Selker, T. A model of textual affect sensing using real-world knowledge. In Proceedings of the 8th International Conference on INTELLIGENT User Interfaces, Miami, FL, USA, 12–15 January 2003; pp. 125–132.
5. Mohammad, S. # Emotional tweets. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), Montreal, Canada, 7–8 June 2012*; Omnipress, Inc.: Madison, WI, USA, 2012; pp. 246–255.
6. Neviarouskaya, A.; Prendinger, H.; Ishizuka, M. Affect analysis model: Novel rule-based approach to affect sensing from text. *Nat. Lang. Eng.* **2011**, *17*, 95–135. [CrossRef]
7. Ahmad, K.; Alam, F.; Qadir, J.; Qolomany, B.; Khan, I.; Khan, T.; Suleman, M.; Said, N.; Hassan, S.Z.; Gul, A.; et al. Sentiment Analysis of Users' Reviews on COVID-19 Contact Tracing Apps with a Benchmark Dataset. *arXiv* **2021**, arXiv:2103.01196.
8. Pak, A.; Paroubek, P. Twitter as a corpus for sentiment analysis and opinion mining. In Proceedings of the LREc, Valletta, Malta, 17–23 May 2010; Volume 10, pp. 1320–1326.
9. Liu, X.; Zhang, S.; Wei, F.; Zhou, M. Recognizing named entities in tweets. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Stroudsburg, PA, USA, 19–24 June 2011; pp. 359–367.
10. Foster, J.; Cetinoglu, O.; Wagner, J.; Le Roux, J.; Hogan, S.; Nivre, J.; Hogan, D.; Van Genabith, J. # hardtoparse: POS Tagging and Parsing the Twitterverse. In Proceedings of the Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–8 August 2011; pp. 20–25.
11. Khan, J.; Lee, S. Enhancement of Sentiment Analysis by Utilizing Noisy Social Media Texts. *J. Korean Inst. Commun. Sci.* **2020**, *45*, 1027–1037.
12. Lertpiya, A.; Chalothorn, T.; Chuangsuwanich, E. Thai Spelling Correction and Word Normalization on Social Text Using a Two-Stage Pipeline With Neural Contextual Attention. *IEEE Access* **2020**, *8*, 133403–133419. [CrossRef]
13. Baldwin, T.; Li, Y. An in-depth analysis of the effect of text normalization in social media. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, CO, USA, 31 May–5 June 2015; pp. 420–429.
14. Baldwin, T.; Cook, P.; Lui, M.; MacKinlay, A.; Wang, L. How noisy social media text, how diffrnt social media sources? In Proceedings of the Sixth International Joint Conference on Natural Language Processing, Nagoya, Japan, 14–18 October 2013; pp. 356–364.
15. Jianqiang, Z.; Xiaolin, G. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access* **2017**, *5*, 2870–2879. [CrossRef]
16. Haddi, E.; Liu, X.; Shi, Y. The role of text pre-processing in sentiment analysis. *Procedia Comput. Sci.* **2013**, *17*, 26–32. [CrossRef]
17. Singh, T.; Kumari, M. Role of text pre-processing in twitter sentiment analysis. *Procedia Comput. Sci.* **2016**, *89*, 549–554. [CrossRef]
18. Saif, H.; Fernández, M.; He, Y.; Alani, H. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In Proceedings of the LREC 2014, Ninth International Conference on Language Resources and Evaluation, Reykjavik, Iceland, 26–31 May 2014; pp. 810–817.
19. Saif, H.; He, Y.; Alani, H. Alleviating data sparsity for twitter sentiment analysis. In Proceedings of the CEUR Workshop Proceedings (CEUR-WS. org), Buffalo, NY, USA, 26–30 July 2012.
20. Bao, Y.; Quan, C.; Wang, L.; Ren, F. The role of pre-processing in twitter sentiment analysis. In Proceedings of the 10th International Conference on Intelligent Computing, ICIC 2014, Taiyuan, China, 3–6 August 2014; Springer: Berlin/Heidelberg, Germany 2014; pp. 615–624.
21. Jianqiang, Z. Pre-processing boosting Twitter sentiment analysis? In Proceedings of the 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, China, 19–21 December 2015; pp. 748–753.

22. Verma, S.; Bhattacharyya, P. Incorporating semantic knowledge for sentiment analysis. In Proceedings of the ICON 2008, 6th International Conference on Natural Language Processing, Pune, India, 20–22 December 2008.
23. Hu, M.; Liu, B. Mining and summarizing customer reviews. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 22–25 August 2004; pp. 168–177.
24. Dinakar, S.; Andhale, P.; Rege, M. Sentiment analysis of social network content. In Proceedings of the 2015 IEEE International Conference on Information Reuse and Integration, San Francisco, CA, USA, 13–15 August 2015; pp. 189–192.
25. Kiritchenko, S.; Zhu, X.; Mohammad, S.M. Sentiment analysis of short informal texts. *J. Artif. Intell. Res.* **2014**, *50*, 723–762. [CrossRef]
26. Sharma, S.; Srinivas, P.; Balabantaray, R.C. Text normalization of code mix and sentiment analysis. In Proceedings of the 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kerala, India, 10–13 August 2015; IEEE: Piscataway, NJ, USA, 2015.
27. Xu, L.; Lee, H.C. System and Method for Text Cleaning by Classifying Sentences Using Numerically Represented Features. US Patent 8,380,492, 19 February 2013.
28. Arpita; Kumar, P.; Garg, K.; Data Cleaning of Raw Tweets for Sentiment Analysis. In Proceedings of the 2020 Indo-Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN), Rajpura, India, 7–15 February 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 273–276.
29. Chu, X.; Ilyas, I.F.; Krishnan, S.; Wang, J. Data cleaning: Overview and emerging challenges. In Proceedings of the 2016 International Conference on Management of Data, San Francisco, CA, USA, 26 June–1 July 2016; pp. 2201–2206.
30. Stone, P.J.; Dunphy, D.C.; Smith, M.S. *The General Inquirer: A Computer Approach to Content Analysis*; M.I.T. Press: Cambridge, MA, USA, 1966.
31. Jain, T.I.; Nemade, D. Recognizing contextual polarity in phrase-level sentiment analysis. *Int. J. Comput. Appl.* **2010**, *7*, 12–21. [CrossRef]
32. Mohammad, S.; Turney, P. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, Los Angeles, CA, USA, 5 June 2010; pp. 26–34.
33. Mohammad, S.M.; Yang, T. Tracking sentiment in mail: How genders differ on emotional axes. *arXiv* **2013**, arXiv:1309.6347.
34. Shamsudin, N.F.; Basiron, H.; Sa'aya, Z. Lexical based sentiment analysis-Verb, adverb & negation. *J. Telecommun. Electron. Comput. Eng.* **2016**, *8*, 161–166.
35. Roark, B.; Charniak, E. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. *arXiv* **2000**, arXiv:cs/0008026.
36. Amati, G.; Ambrosi, E.; Bianchi, M.; Gaibisso, C.; Gambosi, G. Automatic construction of an opinion-term vocabulary for ad hoc retrieval. In Proceedings of the European Conference on Information Retrieval(ECIR 2008), Glasgow, UK, 30 March–3 April 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 89–100.
37. Kaity, M.; Balakrishnan, V. An integrated semi-automated framework for domain-based polarity words extraction from an unannotated non-English corpus. *J. Supercomput.* **2020**, *76*, 9772–9799. [CrossRef]
38. Li, G.; Li, B.; Huang, L.; Hou, S. Automatic Construction of a Depression-Domain Lexicon Based on Microblogs: Text Mining Study. *JMIR Med. Inform.* **2020**, *8*, e17650. [CrossRef] [PubMed]
39. Tan, S.S. Automatic Lexicon Construction for Domain-Specific Sentiment Analysis: A Frame-Based Approach. Ph.D. Thesis, Nanyang Technological University, Singapore, 2020.
40. Viegas, F.; Alvim, M.S.; Canuto, S.; Rosa, T.; Gonçalves, M.A.; Rocha, L. Exploiting semantic relationships for unsupervised expansion of sentiment lexicons. *Inf. Syst.* **2020**, *94*, 101606. [CrossRef]
41. Esposito, M.; Damiano, E.; Minutolo, A.; De Pietro, G.; Fujita, H. Hybrid query expansion using lexical resources and word embeddings for sentence retrieval in question answering. *Inf. Sci.* **2020**, *514*, 88–105. [CrossRef]
42. Masadeh, R.; Sa'ad Al-Azzam, B.H. A Hybrid Approach of Lexicon-based and Corpus-based Techniques for Arabic Book Aspect and Review Polarity Detection. *Int. J.* **2020**, *9*. [CrossRef]
43. Wang, S.; Lv, G.; Mazumder, S.; Liu, B. Detecting Domain Polarity-Changes of Words in a Sentiment Lexicon. *arXiv* **2020**, arXiv:2004.14357.
44. Filip, G.; Krzysztof, J.; Agnieszka, W.; Mikołaj, W. Text normalization as a special case of machine translation. In Proceedings of the International Multiconference on Computer Science and Information Technology, Wisła, Poland, 6–10 October 2006; pp. 51–56.
45. Mosquera, A.; Lloret, E.; Moreda, P. Towards facilitating the accessibility of web 2.0 texts through text normalisation. In Proceedings of the LREC Workshop: Natural Language Processing for Improving Textual Accessibility (NLP4ITA), Istanbul, Turkey, 27 May 2012; pp. 9–14.
46. Almeida, T.A.; Silva, T.P.; Santos, I.; Hidalgo, J.M.G. Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering. *Knowl.-Based Syst.* **2016**, *108*, 25–32. [CrossRef]
47. Silverman, K.; Naik, D.; Bellegarda, J.; Lenzo, K. Systems and Methods for Text Normalization for Text to Speech Synthesis. US Patent 8,355,919, 15 January 2013.
48. Liang, B.W.W.H.L.; Kourie, D.G. *Classification for Selected Spell Checkers and Correctors*; School of Computing, University of South Africa: Pretoria, South Africa, 2008.
49. Xie, F.; Jiang, X.M. *Error Analysis and the EFL Classroom Teaching*; Online Submission, 2007; Volume 4, pp. 10–14.

50. Hovermale, D. SCALE: Spelling correction adapted for learners of English. In Proceedings of the Pre-CALICO Workshop on "Automatic Analysis of Learner Language: Bridging Foreign Language Teaching Needs and NLP Possibilities, Citeseer, University Park, PA, USA, 18 March 2008.
51. Lee, J.H.; Kim, M.; Kwon, H.C. Deep Learning-Based Context-Sensitive Spelling Typing Error Correction. *IEEE Access* **2020**, *8*, 152565–152578. [CrossRef]
52. Kukich, K. Techniques for automatically correcting words in text. *ACM Comput. Surv. (CSUR)* **1992**, *24*, 377–439. [CrossRef]
53. Clark, E.; Roberts, T.; Araki, K. Towards a pre-processing system for casual english annotated with linguistic and cultural information. In Proceedings of the Fifth IASTED International Conference, Maui, HI, USA, 23–25 August 2010; Volume 711, pp. 44–84.
54. Han, B.; Baldwin, T. Lexical normalisation of short text messages: Makn sens a# twitter. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 368–378.
55. Chrupała, G. Normalizing tweets with edit scripts and recurrent neural embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, MD, USA, 22–27 June 2014; pp. 680–686.
56. Liu, F.; Weng, F.; Jiang, X. A broad-coverage normalization system for social media language. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Jeju, Korea, 8–14 July 2012; pp. 1035–1044.
57. Liu, F.; Weng, F. Broad-Coverage Normalization System for Social Media Language. US Patent 9,164,983, 20 October 2015.
58. Sproat, R.; Black, A.W.; Chen, S.; Kumar, S.; Ostendorf, M.; Richards, C. Normalization of non-standard words. *Comput. Speech Lang.* **2001**, *15*, 287–333. [CrossRef]
59. Hernández, A. A ngram-based statistical machine translation approach for text normalization on chat-speak style communications, In Proceedings of the CAW 2.0, Madrid, Spain, 1–5 August 2009.
60. Saloot, M.A.; Idris, N.; Aw, A. Noisy text normalization using an enhanced language model. In Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition. SDIWC, Kuala Lumpur, Malaysia, 17–19 November 2014; pp. 111–122.
61. Desai, N.; Narvekar, M. Normalization of noisy text data. *Procedia Comput. Sci.* **2015**, *45*, 127–132. [CrossRef]
62. Doshi, F.; Gandhi, J.; Gosalia, D.; Bagul, S. Normalizing Text using Language Modelling based on Phonetics and String Similarity. *arXiv* **2020**, arXiv:2006.14116.
63. Choudhury, M.; Saraf, R.; Jain, V.; Mukherjee, A.; Sarkar, S.; Basu, A. Investigation and modeling of the structure of texting language. *Int. J. Doc. Anal. Recognit.* **2007**, *10*, 157–174. [CrossRef]
64. Chatterjee, N. A Trie Based Model for SMS Text Normalization. In Proceedings of the Intelligent Computing-Proceedings of the Computing Conference, London, UK, 16–17 July 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 846–859.
65. Sikdar, A.; Chatterjee, N. An improved Bayesian TRIE based model for SMS text normalization. *arXiv* **2020**, arXiv:2008.01297.
66. Balahur, A. Sentiment analysis in social media texts. In Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Atlanta, GA, USA, 14 June 2013; pp. 120–128.
67. Arora, M.; Kansal, V. Character level embedding with deep convolutional neural network for text normalization of unstructured data for Twitter sentiment analysis. *Soc. Netw. Anal. Min.* **2019**, *9*, 12. [CrossRef]
68. Ojugo, A.A.; Eboka, A.O. Memetic algorithm for short messaging service spam filter using text normalization and semantic approach. *Int. J. Inf. Commun. Technol.* **2020**, *9*, 9–18. [CrossRef]
69. Pota, M.; Ventura, M.; Fujita, H.; Esposito, M. Multilingual evaluation of pre-processing for BERT-based sentiment analysis of tweets. *Expert Syst. Appl.* **2021**, *181*, 115119. [CrossRef]
70. Duong, H.T.; Nguyen-Thi, T.A. A review: Preprocessing techniques and data augmentation for sentiment analysis. *Comput. Soc. Netw.* **2021**, *8*, 1–16. [CrossRef]
71. Bakar, M.F.R.A.; Idris, N.; Shuib, L. An Enhancement of Malay Social Media Text Normalization for Lexicon-Based Sentiment Analysis. In Proceedings of the 2019 International Conference on Asian Language Processing (IALP), Shanghai, China, 15–17 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 211–215.
72. Philips, L. The double metaphone search algorithm. *C/C++ Users J.* **2000**, *18*, 38–43.
73. Odell, M.K. The profit in records management. *Systems* **1956**, *20*, 20.
74. Pennell, D.L.; Liu, Y. Normalization of informal text. *Comput. Speech Lang.* **2014**, *28*, 256–277. [CrossRef]
75. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
76. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv* **2016**, arXiv:1609.08144.
77. Schuster, M.; Nakajima, K. Japanese and korean voice search. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 5149–5152.
78. Eight, F. First GOP Debate Twitter Sentiment, Kaggle. 2019. Available online: https://www.kaggle.com/crowdflower/first-gop-debate-twitter-sentiment (accessed on 15 December 2020).
79. Elsafoury, F. Cyberbullying Datasets, Mendeley, Mendeley Data. 2020. Available online: https://data.mendeley.com/datasets/jf4pzyvnpj (accessed on 21 January 2021).

80. Almeida, T.A.; Hidalgo, J.M.G.; Yamakami, A. Contributions to the study of SMS spam filtering: New collection and results. In Proceedings of the 11th ACM symposium on Document Engineering, Mountain View, CA, USA, 19–22 September 2011; pp. 259–262.
81. Ghosh, U. IMDB Review Dataset, Kaggle. 2018. Available online: https://www.kaggle.com/utathya/imdb-review-dataset (accessed on 3 January 2021).
82. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. *arXiv* **2015**, arXiv:1509.01626..
83. Peter, Norvig, NORVIG Spell-Errors Dataset, Peter Norvig. Available online: http://norvig.com/ngrams/spell-errors.txt (accessed on 27 April 2021).