SCHOOL OF COMPUTER SCIENCE
Faculty of Engineering and Information Technology
University of Technology Sydney

# Hybrid Words Representation for the classification of low quality text

by

**Usman Naseem**

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

**Master of Analytics (Research)**

Sydney, Australia

2020

# Certificate of Authorship/Originality

I certify that the work in this thesis has not been previously submitted for a degree nor has it been submitted as a part of the requirements for other degree except as fully acknowledged within the text.

I also certify that this thesis has been written by me. Any help that I have received in my research and in the preparation of the thesis itself has been fully acknowledged. In addition, I certify that all information sources and literature used are quoted in the thesis.

This research is supported by the Australian Government Research Training Program (RTP).

Production Note:
Signature removed prior to publication.

# Acknowledgements

Acknowledgment goes here Foremost, I would like to express my sincere gratitude to my supervisor; Professor Longbing Cao for the continuous support of my master research degree, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor during my study.

I also would like to appreciate my co-supervisor $A$\Professor Kaska Musial-Gabrys for providing me with continuous support throughout my study and research. Without her professional guidance and persistent help, this thesis would not have been possible. Each of my supervisors helped, supported, and guided me through my research at UTS exceptionally and unforgettably. I shall always remain thankful to my supervisors.

I thank my fellow lab-mates in Advanced Analytics Institute (AAi): for the stimulating discussions, guidance, support and for all the fun we have had in the last two years. I am thankful to the office staff at the School of computer science, UTS, in particular, Margot, Janet, Teraesa and Reshma. All team members I met within the School, provided a conducive research environment. I am also thankful to the staff at the Graduate Research School at UTS, who always remained welcoming of my questions and queries, resolving my problems in a timely manner.

Last but not least, I would like to thank my family for their unconditional support, both financially and emotionally throughout the whole master studying.

# Dedication

My thesis is dedicated to the people who have supported my goals, inspired me and challenged me academically to make it to this day.

Reflecting on my path that led me to this day, after spending more than eight years in the industry and having a very comfortable life with a good job, it was very hard decision for me to come back to student life with limited income. But I decided to take this challenge and thought may be I can do it. I pushed myself hard throughout this period. Today, I submit my master by research degree at the UTS and change my earlier 'maybe' into 'yes'.

For this day today, first and foremost credit goes to my parents. My father taught me to make decisions and always encouraged and supported my bold decision. In all bad times, he encouraged and supported me. I wish I could share this moment with my mother (late). Whatever I am today is only because of her prayers. Thanks to my siblings, Salman and Haadia for always remaining a great source of support and encouragement. My grandfather (late) who always prayed for my success. I still remember that I used to call my grandfather and asked him to pray for me whenever I used to feel down.

Without the support of my wife Huda, I would have never been able to complete my master degree. Aashir and Mahd, I owe you a lot. I could not give more time to you in your childhood due to coming back to home late nights during the weekdays and spending most of my weekends in the library doing my research - you were the real owners of this time. Huda, I am eternally thankful to you for taking up all the responsibilities and letting me focus on my research. Thank you. Thank you to my father-in-law and mother-in-law and as well as siblings of my wife who offered well wishes for my studies and prayed for my success

# Contents

# List of Figures

# List of Tables

# List of Publications

**Paper(s) Accepted & Published**

1. **Naseem, U**., Musial, K. (2019, September). Dice: deep intelligent contextual embedding for Twitter sentiment analysis. In 2019 International Conference on Document Analysis and Recognition (ICDAR) (pp. 953-958). IEEE.
2. **Naseem, U.**, Khan, S. K., Razzak, I., Hameed, I. A. (2019, December). Hybrid Words Representation for Airlines Sentiment Analysis. In Australasian Joint Conference on Artificial Intelligence (pp. 381-392). Springer, Cham.
3. **Naseem, U.**, Razzak, I., Hameed, I. A. Deep Context-Aware Embedding for Abusive and Hate Speech detection on Twitter. Australian Journal of Intelligent Information Processing Systems, 69.
4. **Naseem U**., Khan S.K., Farasat M., Ali F: Abusive language detection: A Comprehensive Review: Indian Journal of Science and Technology., IJST,2019, **ACCEPTED**

**Paper(s) to be Submitted & Under review**

5. A Comparative Analysis of Pre-processing Techniques on Twitter Abusive language and Hate Speech detection.
6. A Recommended combination of pre-processing techniques to improve quality of text data.
7. A Comprehensive Survey on Word Representation Methods for Text Classification.
8. Hybrid Words Representation for text classification (Extended version of accepted paper)

# Abbreviation

NLP - Natural Language Processing

HCI - Human-Computer Interaction

NLU- Natural Language Understanding

SA- Sentiment Analysis

TSA- Twitter Sentiment Analysis

OM- Opinion Mining

SMS- Short Message Service

RT- Retweet

DICE- Deep Intelligent Contextual Embedding

OOV- Out Of Vocabulary

SVM- Support Vector Machine

NB- Naive Bayes

LR- Logistic Regression

DT- Decision Tree

RF- Random Forest

DL- Deep Learning

BoW- Bag Of Word

TF- Term Frequency

TF-IDF- Term Frequency Inverse Document Frequency

CBOW- Continous Bag Of Words

GloVe- Global Vectors

CoVe- Context Vectors

LM- Language Model

BiLM- Bidirectional Language Model

ELMo- Embedding from language model

RNN- Recurrent Neural Network

LSTM- Long Short Term Memory

GRU- Gated Recurrent unit

BiLSTM- Bidirectional Long Short Term Memory

CNN- Convolutional Neural Network

UNK- Unique

POS- Part Of Speech

ReLu- Rectified Linear Unit

DCNN- Deep Convolutional Neural Network

HyRank- Hybrid Ranking

Re*- Refine Embedding

IWV- Improved Word Vectors

# ABSTRACT

**Hybrid Words Representation**

**for the classification of**

**low quality text**

by

Usman Naseem

Language enables humans to communicate with others. For instance, we talk, give our opinions and suggestions all using natural language; to be more precise, we use words while communicating with others. However, in today's world, we wish to communicate with computers, just like humans. It is not an easy task because human communicate in an unstructured and informal way, whereas computers need structured and clean data. So it is essential for computers to understand and classify text accurately for proper human-computer interactions. For classifying a text, the first question we must address is how to improve the low-quality text. The next immediate challenge is to have the best representation so that text can be classified accurately. The way text is organized reflects polysemy, semantic and syntactical coupling relationships which are embedded in its contents. The effective capturing of such content relationships is thereby crucial for a better understanding of text representations. This is especially challenging in the environments where the text messages are short, informal and noisy, and involves natural language ambiguities. The existing sentiment classification methods are mainly for document and clean textual data which can not capture relationship, different attributes and characteristics within tweet messages.

Social media analysis, especially the analysis of tweet messages on Twitter has become increasingly relevant since the significant portion of data is ubiquitous in nature. The social media-based short text is valuable for many good reasons, ex-

plored increasingly in text analysis, social media analysis and recommendation. In the same time, there is a number of challenges that need to be addressed in this space. One of the main issues is that the traditional word embeddings are unable to capture polysemy (assigns the same representation of a word irrespective of its context and meaning) and out of vocabulary words (assigns a random representation). Furthermore, traditional word embeddings fail to capture sentiment information of words which results in similar word vector representations having the opposite polarities. Thus, ignoring polysemy within the context and sentiment polarity of words in a tweet reduces the performance for tweets classification.

In order to address the above-mentioned research challenges and limitations associated with word-level representations, this thesis focuses on improving the representation of low-quality text by improving the unstructured and informal nature of tweets to utilize the information thoroughly and manages the natural language ambiguities to build a more robust sentiment classification model. As compared to previous studies, the proposed models can deal with the ubiquitous nature of the short text, polysemy, semantic and syntactical relationships within a content, thereby addressing the natural language ambiguity problems.

Chapter 4 presents the effects of pre-processing techniques using two different word representation models with the machine and deep learning classifiers. Then, we present our recommended combination (approach) of different pre-processing techniques which improves the low quality, by performing sentiment-aware tokenization, correction of spelling mistakes, word segmentation and other techniques to utilize most of the information hidden in unstructured text. The experimental result shows that the proposed combination performs well as compared to other combinations.

Chapter 5 presents the hybrid words representation. In this chapter, we proposed our Deep Intelligent Contextual Embedding for Twitter sentiment analysis. Proposed model addresses the natural language ambiguities and is devised to capture polysemy in context, semantics, syntax and sentiment knowledge of words. Bi-directional Long-Short Term Memory wth attention is employed to determine the sentiment. We evaluate the proposed model by performing quantitative and

qualitative analysis. The experimental results show that the proposed model outperforms various word embedding models in the sentiment analysis of tweets.

Above mentioned methods can be applied to any social media classification task. The performance of proposed models is compared with different models which support the effectiveness of the proposed models and bound the information loss in their generated high-quality representations.

# Chapter 1

# Introduction

Natural Language Processing (NLP) enables computers to understand, process and classify the language just like humans. In natural language processing (NLP), text classification is one of the most basic and important task that can be applied to many different tasks such as document classification, sentiment classificaion, language detection, topic classification, product and movie reviews classifcation etc (Aggarwal and Zhai; 2012a; Sebastiani; 2002). Text is enourmous source of getting useful information but extracting insights is a challenging task due to unstructured and low quality nature of text. In following sections, we breifly discussed text classification especially short text in terms of its benefits in today's world followed by its benefits and challenges in different areas. Afterwards, we briefly discussed sentiment analysis and presented characteristics, benefits and challenge of Twitter Sentiment Analysis in terms of natural language ambuiguites and low quality of text.

## 1.1 Text classification

Natural Language Processing (NLP) is linked to the area of human-computer interaction (HCI) which involves different challenges such as natural language understanding (NLU) which empowers computers to understand the meaning from input received from natural language or humans and involves natural language generation (Pang and Lee; 2008a). Predefined labels are assigned in the text classification task to the text documents based on its content (Sebastiani; 2005). So, in other words, by utilizing NLP, we can assign tags to the informal and raw text which makes complex and low-quality text into a meaningful organized and structured text. This structured, clean and organized text is easy to interpret by machines. Usually, these labels (classes) are assigned by humans. With the rapid growth of information avail-

able online, it has become difficult for people to process and extract the relevant information manually which has brought opportunities and challenges for the development of Natural language processing (NLP) techniques such as text classification. Text classification either use machine learning or deep learning algorithms to classify the sheer amount of textual information available.

Short text classification has become very popular with the rapid growth of social media platforms such as Twitter, Facebook and e-commerce industry such as Amazon for product descriptions and reviews. Classification of short text can be a challenging and hard task because traditional methods, which work well on large texts and documents, will not work well on the short text. The reason is that short text is very sparse, has fewer features and does not give enough word co-occurrence. Furthermore, it is unstructured and informal with a lot of spelling and grammar mistakes, abbreviations, slangs, emoticons and every person has its own writing style which makes it more challenging to mine and classifies the text with better results (Wang et al.; 2017; Adhi et al.; 2019; Chen et al.; 2011). Due to mentioned issues with a short text, it is essential to properly clean the text and get better representations of raw text before feeding extracted features to classifier for the classification task. In this thesis, we focus on the sentiment analysis task.

## 1.2 Sentiment Analysis

Sentiment analysis, often known as opinion mining is an attempt to understand peoples views, reviews, evaluations, attitudes, opinions, and emotions towards people, topics, objects, evaluation and many other things. The attitude or opinion of the person (writer) towards any certain products or topic can be recognized by using sentiment analysis (Sa et al.; 2018; Giachanou and Crestani; 2016; Silva et al.; 2016).

Millions of users around the globe use the internet where they read and write their opinions or sentiments. In todays world, people around the world mostly rely on the opinions and recommendations given by different people before making a decision. The ratio of people who consider online reviews as a personal recommen-

dation has increased drastically *. So understanding the content and sentiment of information published online, including the social media where users share sentiments and views is crucial from the perspective of improving the services, products and recommendations for those users (Cambria et al.; 2013; Aggarwal and Zhai; 2012a)

Sentiment analysis helps companies to improve their business models and increase their profit margins by mining public sentiments, reviews by customer and recommendations to online users. It is extensively being used for understanding online reviews and in social media for many different applications such as customer service or marketing. Sentiment analysis has developed into one of the famous areas in the natural language processing (NLP) to determine and extract subjective information(Pang and Lee; 2008b; He et al.; 2013; Medhat et al.; 2014)

Sentiment analysis is one of the hardest tasks to be addressed by the machines. Handling ubiquitous, unstructured and informal nature of short text and language ambiguities such as lexical, syntactic and semantic ambiguities as well as some patterns, features and entities are hard for the machines to recognize as human beings. For example, it is difficult for the machines to understand the sarcasm and ironies just like human beings because of their opposite meanings. Also, it is easy for human beings to understand the informal and unstructured text used by online users such as using different abbreviations, hashtags, acronyms, different punctuation, incorrect spellings, emoticons, slang, URLs, etc. but it is tough for the machines to handle this informal and unstructured text which causes noise to handle the text posted online by different users. Similarly, other linguistic constraints/language ambiguities such as capturing polysemy, complicated attributes of words usage in semantics and syntax and understanding the polarities of words, all these language restrictions are easy for human being to understand but much effort is needed to make the machines understand the sentiments just like human beings by handling this language imperfections (Batrinca and Treleaven; 2015; Kharde and Sonawane; 2016; Hussein; 2018; Thelwall et al.; 2010; Kiritchenko, Zhu and Mohammad; 2014;

*https://www.inc.com/craig-bloem/84-percent-of-people-trust-online-reviews-as-much-.html

O'Connor et al.; 2010). There are many different types and areas for applying sentiment analysis, but in the next section, we will restrict ourselves to only Twitter sentiment analysis which is relevant to this thesis.

## 1.3 Twitter Sentiment Analysis

In this section, first, we discuss Twitter, its characteristics, along with its importance. After that, we briefly discuss Twitter sentiment analysis and its challenges. Twitter is a social network application where people micro-blog (post messages or tweets) about a variety of different topics. Earlier it was restricted to 160 characters to be an SMS kind of service. 140 characters limit for every tweet and 20 characters left for the user-name. Twitter users can post any kind of information within this 140 characters but can provide a URL link for a more detailed information source. This restriction of 140 characters forces users to use abbreviations, emoticons and make spelling mistakes intentionally or unintentionally, which makes data noisy and makes it challenging to extract useful information. Twitter users use user mentions referring to other Twitter user, re-tweets and hashtags. $\alpha$ user-name in a tweets shows that a tweet is a posted in a response of another tweet by another user. Users might re-post the tweet if a tweet is interesting and compelling to a Twitter user. RT $\alpha$ user-name format is used to re-tweet (re-post) the tweet posted by another user. Twitter allows its users to tag their tweets using hashtags in the form of #tag-name. Hashtags generally are used to convey the context of the tweet message. The characteristics of Twitter and its character restrictions are unique which are not common in other traditional media (Giachanou and Crestani; 2016; MARTNEZ-CMARA et al.; 2014)

Due to its popularity, users across the world are posting millions of posts on a daily basis which serves as a rich source of information. The massive amount of content published on Twitter offers many opportunities for the researchers to understand and analyze the trends. Twitter sentiment analysis can be helpful for companies in many ways. It can provide qualitative insights to companies about how people talking about their products, brands and services. Millions of tweet

messages are posted by Twitter users daily, which makes it most popular among all social media platforms for getting information, daily news and updates about famous people around the globe. Due to the mentioned reasons, companies see Twitter a vital tool when compared to different social media platforms for making their marketing decisions, improving services, products and recommendations for other users. Twitter connects users and businesses around the globe without any boundaries, but on another hand, it can negatively impact the businesses and brands name if any harmful content is posted by the users, especially when it goes viral. This can cause a significant dent to the services and businesses. That is why monitoring social media, understanding users behaviour and resolving their concerns have become essential elements for any business in order to improve the business and maximise the profits. Twitter is a great source to understand and discover the new trends in the industry (Giachanou and Crestani; 2016; Go et al.; 2009; Agarwal et al.; 2011; Cambria et al.; 2013)

In recent literature, many different techniques have been proposed for Twitter sentiment analysis but still not fully able to handle natural language ambiguities such as words with different meanings in a context (Polysemy), sentiment of words, semantic and syntactical information. To handle the polysemy issue, different language model-based models are proposed, which generates context-dependent vectors. (Liu et al.; 2015; McCann et al.; 2017a; Peters et al.; 2018c; Melamud et al.; 2016a; Cambria et al.; 2018) and other language complexities such as sentiment specific word representation which considers sentiment of words, semantic and syntactical information of words (Tang et al.; 2016a; Rezaeinia et al.; 2017a; Khan et al.; 2014). This is vivid in the case of low-quality text such as tweets which makes the analysis more challenging from conventional text such as documents, blogs or forums. Bermingham and Smeaton (2010) studied the difficulty level of short and long text for detecting sentiment analysis. Similarly, the impact of low-quality tweet messages was studied by Brody and Diakopoulos (2011) and found that Twitter users use informal language in one of every six tweet messages. Saif et al. (2012) presented a method to reduce the noise in tweet messages by semantic smoothing.

## 1.4 Research Contributions

This research makes the following contributions which depict the relationship between our research objectives, problems and research questioned defined in Chapter 3.

- The effects of common and advance pre-processing techniques on text classification performance are analyzed. Two different word representation models with machine learning and deep learning classifiers are employed to analyze the effects on text classification performance (Chapter 4).
- A recommended combination of pre-processing techniques in a systematic pattern is proposed to improve the quality of the text. The proposed method helps to get better text representation by sentiment-aware tokenization, replacing abbreviations and slangs, correcting spelling mistakes, word segmentation and remove noise from the text (Chapter 4).
- Proposal of hybrid words representation to capture the language ambiguities such as Polysemy (words with a different meaning in a context), Sentiment knowledge of words, Semantic and syntactical information. (Chapter 5).
- Application of hybrid words representation on text classification task, which focuses on improving the text quality and addresses the language ambiguities (Chapter 4 & 5).

## 1.5 Thesis Structure

The structure of this thesis is shown in Figure 1.1. We divide the proposed methods in this thesis into two parts. (1) First part presents the analysis and recommended combination of pre-processing techniques to improve the quality of text as in Chapter 4; (2) Second part presents hybrid words representation model (DICE) which handles language complexity as in Chapter 5. The summary of each chapter is as follows:

- *Chapter 1*: This chapter is the brief introduction of the thesis, including the text classification, Sentiment Analysis, Twitter sentiment analysis, the thesis contributions and structure.

- *Chapter 2*: This chapter presents a fundamental and relevant literature review to this thesis which includes text classification pipeline, text pre-processing, different word representation and classification methods including related word to our proposed methods. Also, we present a gap analysis in this chapter to highlight the gap in the current state of the art models.

- *Chapter 3*: This chapter provides the methodology used in this thesis, including research problems, questions and objectives of this research in a detail.

- *Chapter 4*: This chapter provides a comprehensive analysis of common and advance pre-processing techniques with different feature extraction and classification models. Further, we present our first model which improves the quality of text and helps the learner to learn better features which improves classification performance. In this study, we recommend the techniques which gives better results individually and which does not. We also compared the results of our proposed method with the baseline method and presented the comparison that which classifier benefits the most using our proposed method.

- *Chapter 5*: This chapter proposes the hybrid words representation model ( DICE: Deep Intelligent contextual embedding) which addresses the language complexity such as words with different meanings (Polysemy) within a context, out of vocabulary (OOV) words, semantics, syntax and sentiment knowledge of words. We compared our proposed model with different models from literature such as classic, continuous, hybrid and sentiment specific state of the art word representation models. The empirical analysis of our proposed model for Twitter sentiment analysis demonstrates its effectiveness with respect to managing the language complexity compared to the state-of-the-art models.

- *Chapter 6*: A summary of the thesis, its contributions and recommendation for future work is given in the final chapter.

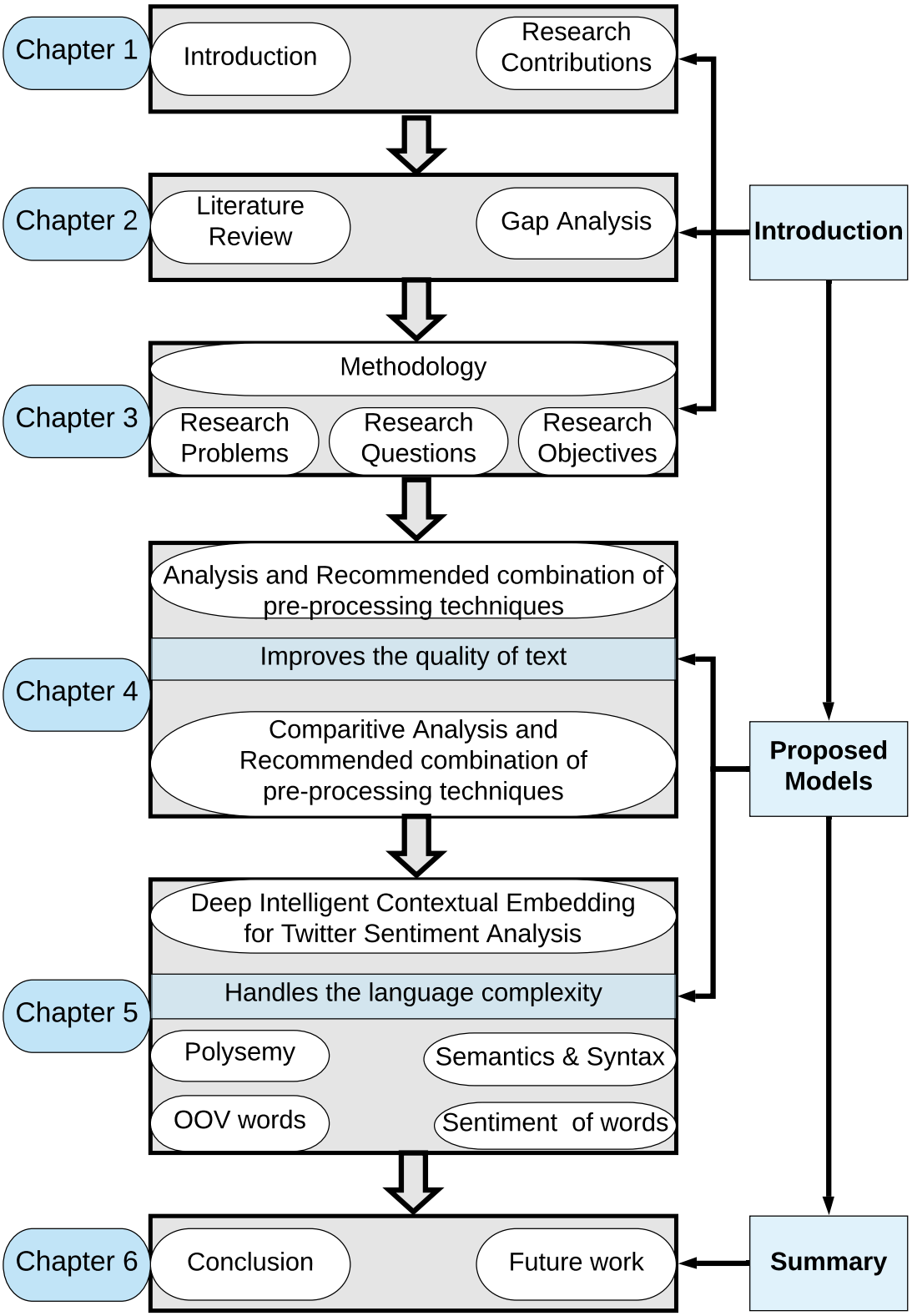


Figure 1.1 : Thesis Structure

# Chapter 2

# Literature review

In this chapter, we present some background and relevant techniques for our research. First, we present the text classification pipeline followed by the details of each section in the pipeline.

## 2.1 Text classification Pipeline

Text mining helps to examine vast volumes of unstructured and raw text and discover appropriate insights. Coupled with machine learning, it can formulate different models for text analysis to give labels or obtain information based on prior training. In any text classification task, first, we have to gather data which we are interested in analysing. The next step is to represent the raw unstructured data in a form that machine learning classification algorithms can understand. Text representation can be divided into main two parts: i) Text Pre-processing and, ii) Features Extraction and then classify the learned representations using an appropriate classifier (Kowsari et al.; 2019; Aggarwal and Zhai; 2012b). Below we discuss briefly the text classification pipeline illustrated in Figure 2.1.

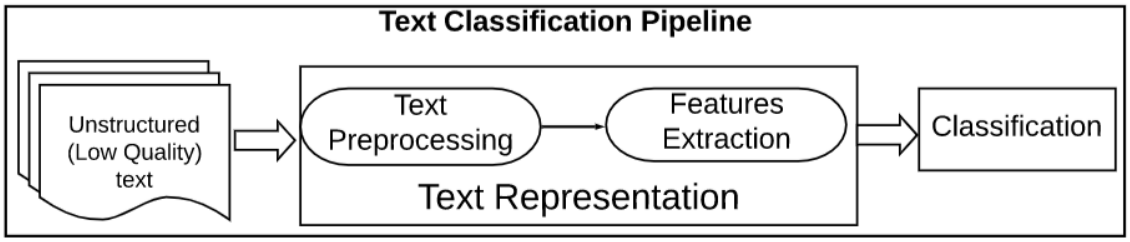


Figure 2.1 : Text Classification Pipeline

- **Unstructured (Low Quality) Text:** Unstructured (Low Quality) text is a form of written text which requires metadata and cannot easily be listed or classified. Usually, it is the information generated by users on social media postings, documents, email or messages. Raw text is scattered and sparse with less number of features and does not give sufficient word co-occurrence information. It is an important source of information for businesses, research institutes and monitoring agencies. Often companies mine it for getting the data to improve their marketing strategies and achieve an edge in the marketplace. It plays a big part in predictive analytics and in analysing sentiments of users to find-out the overall opinion of customers. It helps to discover unique insights by revealing hidden information, discovering trends and recognising relationships between irrelevant bits of the data (Gurusamy and Kannan; 2014; Haddi et al.; 2013; Uysal and Günal; 2014).

- **Text Representation**: For text classification, the text should be converted into a form which the computer can understand. First, we need to improve the quality of raw and unstructured text and then extract features from it before classification. Both of these steps are briefly discussed below.

    * **Text Preprocessing:** Pre-processing is the crucial step, especially in the classification of short text. Pre-processing techniques are valuable techniques for decreasing the data adequacy, sparsity and helps to improve the low quality of text especially in the case of short text where everyone writes in their style, and use emoticons, abbreviations, make spelling mistakes and use URLs etc. A proper combination of common and advance pre-processing techniques can help to learn good text representation (Bao et al.; 2014; Singh and Kumari; 2016; Jianqiang and Xiaolin; 2017). Pre-processing techniques analysed in our study are briefly discussed in section 2.2.

* **Features Extraction:** Features extractions of the data is the critical step for machines to classify and understand the data like humans. It is process of transforming a raw data into a numeric data which machines can understand. Usually, this feature extraction step of transforming a raw data is called a features vector. Extracting word representations which are robust is not so easy without having a considerable amount of corpus due to diversity of expressing sentiments, emotions and intentions in the English language. However, due to social media platforms, researchers now have access to get an enormous amount of data. But assigning labels to this massive amount of data collected from social media platforms is not an easy job. To make this annotation process easy, researchers initially worked on finding sign of sentiment and emotion within the content of the text like emoticons and hashtags (Suttles and Ide; 2013; Wang, Chen, Thirunarayan and Sheth; 2012; Kowsari et al.; 2019; Felbo et al.; 2017). Some of the famous classical and current feature extraction algorithms are briefly discussed in section 2.3.

– **Classification**: Selecting the best classifier is the essential part of text classification pipeline. It is hard to find out the most effective and adequate classifier for text classification task without understanding theoretically and conceptually each algorithm. In section A we present the various famous classification algorithms for text classification task. First, famous traditional machine learning algorithms of text classification such as Logistic Regression which is used in many data mining areas (Patriche et al.; 2016; Chen et al.; 2017), Naive Bayes which is computationally not expensive and works well with less amount of memory (Larson; 2010), K-nearest Neighbour which is non-parametric methods and Support Vector Machine is a famous classifier which has been widely used in many different areas earlier. Then tree-based algorithms like random forest

and decision tree are discussed followed by deep learning-based classifiers which are a collection of methods and approaches motivated by the working mechanism of the human brain. These methods utilise the extensive amounts of training input data to achieve the high quality of semantically rich text representations which can be given as input to different machine learning methods which can make more better predictions (Korde and Mahender; 2012; Kowsari et al.; 2019).

## 2.2 Text prepossessing

Text datasets contain a lot of unwanted words such as stop-words, punctuation, incorrect spellings, slangs, etc. This unwanted noise and words can have an adverse effect on the performance of the classification task. Below first, we present briefly different pre-processing techniques followed by some literature review where researchers analyzed the effects of text pre-processing techniques.

### 2.2.1 Text preprocessing methods

In this section, methods and techniques related to short text pre-processing and cleaning will be briefly discussed.

* **Tokenization**

  A process of transforming a text (sentence) into tokens or words is known as tokenization. Documents can be tokenized into sentences, whereas sentences can be converted into tokens. In tokenization, a sequence to text is divided into the words, symbols, phrases or tokens (Balazs and Velásquez; 2016). The prime objective of tokenization is to find out the words/tokens in a sentence. Usually, tokenization is applied as a first and standard preprocessing step in any NLP task.(Giachanou et al.; 2017)

* **Removal of Noise, URLs, Hashtag and User-mentions**

Unwanted strings and Unicode such as \xc6 \x8e and \xf1 which are considered as a leftover during the crawling process, which is not useful for the machines and creates noise in the data. Also, almost all of tweets messages posted by users, contains URLs to provide extra information, User-mention/tags ($\alpha$) and use hashtag symbol "#" to associate their tweet message with some particular topic and can also express their sentiments in tweets by using hashtags. These give extra information which is useful for human beings, but it does not provide any information to machines and 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; it is replaced with tags (Agarwal et al.; 2011) whereas User-mentions ($\alpha$) are removed (Bermingham and Smeaton; 2011; Khan et al.; 2014)

* **Word Segmentation** Word segmentation is the process of separating the phrases/content/keywords used in the hashtag. And 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 their tweets with some particular topic. The phrase or keyword starting with # is known as hashtags. There are different methods available in the literature for word segmentation (Reuter et al.; 2016; Celebi and Ozgur; 2016).

* **Replacing Emoticons and Emojis**

Twitter users use many different emoticons and emojis such as:), :(, etc. to express their sentiments and opinions. So it is very important to capture this useful information to classify the tweets correctly. There are few tokenizers available which can capture few expressions and emoticons and replace them with their associated

meanings (Gimpel et al.; n.d.).

* **Replacement of abbreviation and slang**

  Character limitations of Twitter enforce online users to use abbreviations, short words and slangs in their posts online. An abbreviation (short words) is a short or acronym of a word such as MIA which stands for missing in action whereas slang is an informal way of expressing thoughts or meanings which is sometimes restricted to some particular group of people, context and considered as informal. So it is crucial to handle such kind of informal nature of text by replacing them to their actual meaning to get better performance without losing information. Researchers have proposed different methods to handle this kind of issue in a text, but the most useful technique is to convert them to an actual word which is easy for a machine to understand (Kouloumpis et al.; 2011; Mullen and Malouf; 2006)

* **Replacing elongated characters** Social media users, sometimes intentionally use elongated words in which they purposely write or add more characters repeatedly more times, such as looooovvveee, greeeeat. Thus, it is important to deal with these words and change them to their base word so that classifier does not treat them different words. In our experiments, we replaced elongated words to their original base words. Detection and Replacement of elongated words have been studied by Mohammad et al. (2013) and Balahur (2013).

* **Correction of Spelling mistakes**

  Incorrect spellings and grammatical mistakes are very commonly present in the text, especially in the case of social media platforms,

especially on Twitter and Facebook. Correction of spelling and grammatical mistakes helps in reducing the same words written indifferently. Textblob is one the library which can be used for this purpose. Norvig's spell correction* method is also widely used to correct spelling mistakes.

* **Expanding Contractions**

  A contraction is a shortened form of the words which is widely being used by online users. An apostrophe is used in the place of the missing letter(s). Because we want to standardize the text for machines to process easily so, in the removal of contractions, shortened words are expanded to their original root /base words. For example, words like how is, Im, cant and dont are the contractions for words how is, I am, cannot and do not respectively. In the study conducted by Boia et al. (2013) and Snchez-Mirabal et al. (2014), contractions were replaced with their original words or by the relevant word. If contractions are not replaced, then the tokenization step will create tokens of the word "can't" into "can" "t".

* **Removing Punctuations**

  Social media users use different punctuations to express their sentiments and emotions, which may are useful for humans but not all much useful for machines for the classification of short texts. So removal of punctuation is common practice in classification tasks such as sentiment analysis. However, sometimes some punctuation symbols like "!" and "?" shows/denotes the sentiments. Its common practice to remove punctuation. (Lin and He; 2009). whereas, replacing question mark or sign of exclamation with tags has also been studied by Balahur (2013).

*http://norvig.com/spell-correct.html

* **Removing Numbers**

  Text corpus usually contains unwanted numbers which are useful for human beings to understand but not much use for machines which makes lowers the results of the classification task. The simple and standard method is to remove them (He et al.; 2011; Jianqiang; 2015). However, we could lose some useful information if we remove them before transforming slang and abbreviation into their actual words. For example, words like "2maro", "4 u", "gr8", etc. should be first converted to actual words, and then we can proceed with this pre-processing step.

* **Lower-casing all words**

  A sentence in a corpus has many different words with capitalization. This step of pre-processing helps to avoid different copies of the same words. This diversity of capitalization within the corpus can cause a problem during the classification task and lower the performance. Changing each capital letters into a lower case is the most common method to handle this issue in text data. Although, this pre-prepossessing technique projects all tokens in a corpus under the one feature space also causes a bunch of problems in the interpretation of some words like US in the raw corpus. The word US could be pronoun and a country name as well, so converting it to a lower case in all cases can be problematic. The study conducted by dos Santos and de C. Gatti (2014) has lower-cased words in corpus to get clean words.

* **Removing Stop-words**

  In-text classification task, there are many words which do not have critical significance and are present in high frequency in a text. It means the words which does not help to improve the performance because they do not have much information for the sentiment clas-

sification task, so it is recommended to remove stop words before feature selection step. Words like (a, the, is, and, am, are, etc.). A popular and straightforward method to handle with such words is to remove them. There are different stop-word libraries available such as NLTK, scikit-learn and spaCy.

* **Stemming**

  One word can turn up in many different forms, whereas the semantic meaning of those words is still the same. Stemming is the techniques to replace and remove the suffixes and affixes to get the root, base or stem word. The importance of stemming was studied by Mejova and Srinivasan (2011). There are several types of stemming algorithms which helps to consolidate different forms of words into the same feature space such as Porter Stemmer, Lancaster stemmer and Snowball stemmers etc. Feature reduction can be achieved by utilizing the stemming technique.

* **Lemmatization**

  The purpose of the lemmatization is the same as stemming, which is to cut down the words to it's base or root words. However, in lemmatization inflection of words are not just chopped off, but it uses lexical knowledge to transform words into its base forms. There are many libraries available which help to do this lemmatization technique. Few of the famous ones are NLTK (Wordnet lemmatizer), genism, Stanford CoreNLP, spaCy and TextBlob etc.

* **Part of Speech (POS) Tagging**

  The purpose of Pat of speech (POS) tagging is to assign part of speech to text. It clubs together with the words which have the same grammatical with words together.

* **Handling Negations**

For humans, it is simple to get the context if there is any negation present in the sentence, but for machines sometimes it does not help to capture and classify accurately so handling a negation can be a very challenging task in the case of word-level text analysis. Replacing negation words with the prefix 'NEG_' has been studied by Narayanan et al. (2013). Similarly, handling negations with antonym has been studied by Perkins (2010).

### 2.2.2 Effects of Pre-processing methods

Text pre-processing plays a significant role in text classification. Many researchers in the past have made efforts to understand the effectiveness of different pre-processing techniques and their contribution to text classification tasks. Below we present some studies conducted on the effects of pre-processing techniques on text classification tasks.

Bao et al. (2014) study showed the effect of pre-processing techniques on Twitter analysis task. Uni-gram and bi-grams features were fed to Liblinear classifier for the classification of positive and negative classes. They showed in their study that reservation of URL features, the transformation of negation (negated words) and normalization of repeated tokens have a positive effect on classification results whereas lemmatization and stemming have a negative effect on classification results. Singh and Kumari (2016) showed the impact of pre-processing on Twitter dataset full of abbreviations, slangs, acronyms for the sentiment classification task. In their study, they showed the importance and significance of slang and correction of spelling mistakes and used Support Vector Machine (SVM) classifier to study the role of pre-processing for sentiment classification. Haddi et al. (2013) also explored the effect of text pre-processing on movie review dataset. The experimental shows that pre-processing methods like the transformation of text such as changing abbreviations to actual words and removal of stop word, special characters and handling of negation with the prefix NOT and stemming can significantly

improve the classification performance. The SVM classifier was used in their experiments  the study conducted by Uysal and gunal. Uysal and Günal (2014) to analyze the role of pre-processing on two different languages for sentiment classification was presented. They employed SVM classifier in their studies and showed that performance is improved by selecting an appropriate combination of different techniques such as removal of stop words, the lower casing of text, tokenization and stemming. They concluded that researchers should choose all possible combinations carefully because the inappropriate combination may result in degradation of performance. Similarly, Jianqiang and Xiaolin (2017) studied the role of six different pre-processing techniques on five datasets in their study, where they used four different classifiers. Their experimental results show that replacing acronyms ( abbreviations) with actual words and negations improved the sentiment classification, whereas removing stop-words, special characters and URLs have a negative influence on the results of sentiment classification. Role of text preprocessing to reduce the sparsity issue in Twitter sentiment classification is studied by Saif et al. (2013). Experimental results show that choosing a combination of appropriate pre-processing methods can decrease the sparsity and enhance the classification results. Agarwal et al. (2011) proposed novel tweets pre-processing approaches in their studies. They replaced URL, user mentions, repeated characters and negated words with different tags and removed hashtags. Classification results were improved by their proposed pre-processing methods. In another studies presented by Saloot et al. (2015) and Takeda and Takefuji (2015) in the natural language workshop which focus on noise user-generated text[†]. Noisy nature of Twitter messages is reduced/decreased by normalizing tweets using a maximum entropy model and entity linking. Recently, Symeonidis et al. (2018) presented the comparative analysis of different techniques on two

[†]http://noisy-text.github.io/

datasets for Twitter sentiment analysis. In their study, they studied the effect of each technique on four traditional machine learning-based classifiers and one neural network-based classifier with only TFIDF (unigram) for words representation method. Their study showed that pre-processing technique such as removing numbers, lemmatization and expanding contractions to base words performs better, whereas removing punctuation does not perform well in the classification task. Their study also presented the interactions of the limited number of different techniques with others and showed that techniques which perform well when interacted with others. However, no work has been done the recommendation of pre-processing techniques to improve the quality of the text.

## 2.3 Words Representation

Next step after cleaning and pre-processing the text is to represent to a mahine readable form. Different researchers in the past have proposed different word representation models are briefly discussed, followed by some literature review where different methods have been adopted for different NLP related tasks.

### 2.3.1 Words Representation Models

Below, we present some popular word representation methods. First, we present some legacy word representation models, followed by some famous representation learning models.

#### *Legacy words representation models*

This section presents some of the popular legacy word representation methods which were commonly used in earlier days for the text classification task. Frequency of words is the basis of this kind of words representation methods. First, we give a short description of categorical word representation methods and then weighted word representation methods.

– **Categorical word representation**: is the simplest way to represent text. In this method, words are represented by a symbolic representation either "1" or "0". One-hot encoding and Bag-of-words (BoW) are the two models which come under categorical word representation methods. Both are briefly discussed below.

  * **One hot encoding:** The most straightforward method of text representation is one hot encoding. In one hot encoding, the dimension is the same amount of terms present in the vocabulary. Every term in vocabulary is represented as a binary variable such as 0 or 1, which means each word is made up of zeros and ones. Index of the corresponding word is marked with 1, whereas all others are marked as zero (0). Each unique word has a unique dimension and will be represented by a 1 in that dimension with 0s everywhere else.

  * **Bag-of-Words (BoW)**: BoW is simply an extension of one-hot encoding. It adds up the one-hot representations of words in the sentence. An example of "Hello" and "World" as one-hot encoding and "Hello World" as BoW is given in figure 2.2.

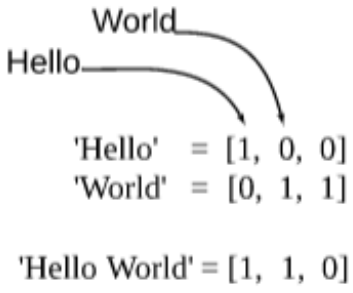


Figure 2.2 : An illustration of one-hot encoding and BoW models

– **Weighted Word representation** Here, we present the common methods for weighted word representations such as Term frequency (TF) and

Term frequency-inverse document frequency (TF-IDF). These are associated with categorical word representation methods but rather than only counting; weighted models feature numerical representations based on words frequency. Both of them are briefly discussed below.

* **Term Frequency (TF)** : TF calculates how often a word occurs in a document. A word can probably appear many times in large documents tas compared to small ones. Hence, TF is computed by dividing the length of the document. In other words, TF of a word is computed by dividing it with the total number of words in the document.

* **Term frequency Inverse document frequency (TF-IDF)**: To cut down the impact of common words such as 'the', 'and' etc. in the corpus, TF-IDF was presented by Sparck Jones (1988) for text representation. TF here stands for Term frequency (TF) which is defined in the above section, and IDF denotes inverse document frequency. Words with less frequencies are assigned higher weight than the more familiar word in Term frequency Inverse document frequency (TF-IDF). TF-IDF is represented mathematically by the below equation.

$$TF - IDF(t, d, D) = TF(t, d) \times \log(\frac{D}{df_t})$$

Where $t$ denotes the terms; $d$ denotes each document; $D$ represents the collection of documents and $df_t$ denotes sum of documents with term $t$ in it.

***Representation Learning***

The limitations of manually extracting features make it use a limited for building a suitable model in machine learning. Due to this, different methods have been proposed in the past, which discovers the representations automatically for downstream

tasks such as classification. Such methods which discover features itself are called as feature learning or representation learning.

It is very important because the performance of machine learning models heavily depends on the representations of the input (Bengio et al.; 2013). Deep learning-based model, which are good at learning important features itself, is changing traditional feature learning methods. Proper representation can be learned either by utilizing supervised learning methods or unsupervised methods.

In the area of NLP, unsupervised text representation methods like pre-trained embeddings have replaced categorical text representation methods. These word embeddings turned into very efficient representation methods to enhance the performance of many downstream tasks due to having a previous knowledge for different machine learning models. Traditional feature learning methods have been replaced by these neural network-based methods due to their good representation learning capacity.

Since categorical word representations models fail to capture syntactic and semantic meaning of the words and these models suffers the curse of high dimensionality. The shortcomings of these models led the researchers to learn the distributed word representation in low dimension space (Bolukbasi et al.; 2016).

**Continous Words Representation**

Word Embedding is natural language processing (NLP) technique in which text from the corpus is mapped as the vectors. In other words, it is a type of learned representation which allows same meaning words to have the same representation. It is the distributed representation of a text (words and documents) which is a significant breakthrough for better performance for NLP related problems. The most significant benefit of word embedding is this that it provides more efficient and expressive representation by keeping the word similarity of context and by low dimensional vectors. Nowadays, word embedding is being used in many different applications like semantic analysis, philology, psychiatry, cognitive science, social science and psychology (Elekes et al.; 2018).

An automatic feature learning technique in which every word in a vocabulary is indexed into an N dimension vector is known as distributed vectors or Word embedding. Which follows the distributional hypothesis. According to this, words which are used and appear in the same contexts tend to assure the same meanings. So these vectors tend of have the attributes of words neighbours and they capture similarity between words. During 1990, several researchers made attempts to lay down the foundation of distributional semantic learning.

Bengio et al. (2003) presented a model which learned word representations using distributed representation. Different word embedding models have been proposed, which makes unigrams useful and understandable to machine learning algorithms and usually, these models are used in the first layer in a deep neural network-based model. These word embedding are pre-trained by predicting a word based on its context without losing semantic and syntactical information. Thus, using these embedding techniques have demonstrated to be helpful in many NLP tasks because It does not lose the order of words and captures the meaning of words (syntactic and semantic information of words).

However, the popularity of word representation methods are due to two famous models, Word2Vec (Mikolov et al.; 2013) and GloVe (Manning et al.; 2014). These famous, along with other famous word representation models, are briefly discussed below.

- **Word2Vec**

  Word2vec is words representation model developed by Mikolov et al. (2013). This model uses two hidden layers which are used in a shallow neural network to create a vector of each word. The word vectors captured by CBOW and Skip-gram models of word2vec are supposed to have semantic and syntactic information of words. To have a better representation of words, it is recommended to train the corpus with the large corpus. Word2Vec have proved to be useful in many Natural language processing (NLP) related tasks (Collobert et al.; 2011). Word2Vec was developed to build training of embedding

more significant, and since then, it has been used as a standard for developing pre-trained word representation. Based on the context, Word2Vec predicts by using one of the two neural network models such as Continuous bag of words (CBOW) and Skip-gram. A predefined length of the window is moved together with the corpus in both models, and the training is done with words inside the window in each step (Altszyler et al.; 2016). Figure 2.2 shows the working principle of both Word2Vec algorithms, CBOW and Skip-Gram.

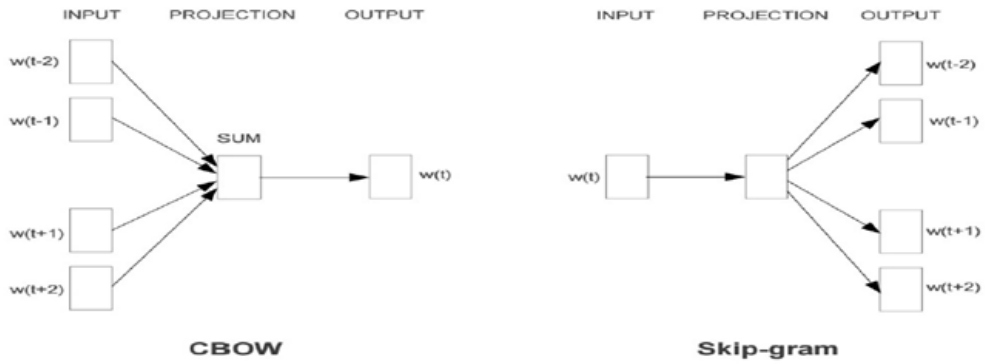


Figure 2.3 : Working principle of Word2Vec (Mikolov et al.; 2013)

- **Continuous Bag of words (CBOW)**: CBOW gives words prediction of current work based on its context. CBOW communicates with the neighbouring words in the window. Three layers are used in CBOW process. Context is considered as the first layer whereas the layer which is hidden matches with the estimation of every word from the input to the weight matrix which later on is estimated to the output which is considered as the third layer. The last phase of this method is to correlate the output and the work itself to improve the representation on the basis of backpropagation of the error gradient. In a figure 2.3, CBOW method predicts the middle word on the basis of its context in skip-gram predicts the context word on the basis of centre word (Naili et al.; 2017).

- **Skip-Gram**

Skip-Gram is the reverse of CBOW model; prediction is given on the basis of the central word after the training of context in skip-gram. Input layer correlates with the targeted word, and the output layer corresponds with the context. This model looks for the estimation of the context given the word, unlike CBOW. The last phase of this model is the correlation between output and every word of the context to adjust representation on the basis of backpropagation (Naili et al.; 2017; Elekes et al.; 2018).

Skip-gram is efficient when we have less training data, and not frequent words are well presented. Whereas CBOW is quicker and performs better with repeated words. To address the issues of learning the final vectors, two algorithms are proposed. First one is negative sampling in which we restrict the sum of output vectors which needs to be updated, so only a sample of the vectors is updated based on a noise distribution which is a probabilistic distribution used in the sample step. Moreover, the other method is Hierarchical softmax which is developed based on the Huffman tree. It is a binary tree which gives all words depending on their counts. Then normalization is done for each step from the root to the target. Negative sampling is efficient when the dimension of vectors is less and works well with repeated words. Whereas hierarchical softmax works good when we have less frequent words (Naili et al.; 2017).

- **Global Vectors (GloVe)**

  GloVe is an expansion of the word2Vec for learning word vectors efficiently where the words prediction is made on the basis of surrounding words. Glove was proposed by Manning et al. (2014). Glove is based on the appearances of a word in the corpus, which is based on two steps. Creation of the co-occurrence matrix from the corpus is the first step, followed by the factorization to get vectors. Like word2Vec, GloVe also provided pre-trained embeddings in different dimension which are trained over the vast corpus. The objective function of GloVe is given below.

$$J = \sum_{k,j=1}^{V} f(X_k j)(w_k^T w_j^{'} + b_k + b_j - \log(X_k j))$$

where;

$V$ : is size of vocabulary,

$X$ : is co-occurrence matrix,

$X_k j$ is frequency of word $k$ co-occurring with word $j$,

$X_k$ total number of occurrences of word $k$ in the corpus,

$P_k j$ is the probability of word $j$ occurring within the context of word $k$,

$w$ is a word embedding of dimension d,

$w^{'}$ is the the context word embedding of dimension $d$

- **FastText**

  Bojanowski et al. (2016a) proposed FastText. Word Embedding methods like Word2Vec and GloVe does not consider the issue of morphology/rare words/Out of vocabulary (OOV). FastText model solved above-mentioned issues. FastText breaks a word in an N-grams instead of full word for feeding into a neural network. FastText gives better results by having better word representation primarily in the case of rare words. Facebook has presented pre-trained word embeddings for 294 different languages, trained on Wikipedia using fastText embedding on 300 dimensions and utilized word2Vec skip-gram model with its default parameters (Joulin et al.; 2016).

- **Contextual word representations**

  Recently, researchers have focused on solving the issue of polysemy in the context and proposed new word embedding models which are briefly discussed below.

  - **Generic Context word representation (Context2Vec):** Context2Vec was proposed by Melamud et al. (2016a) in 2016 to generate context-dependent word representations. Their model is based on word2Vecs CBOW model but replaces its average word representation within a fixed

window with more better and powerful Bi-directional LSTM neural network. A large text corpus was used to learn neural model which embeds context from a sentence and target words in the same low dimension which later is optimized to reflect the interdependencies between target and their entire sentential context as a whole.

– **Contextualized word representations Vectors (CoVe)**

McCann et al. (2017b) presented their model Contextualized word representations (CoVe) which is based on context2Vec. They used machine translation to build CoVe instead of the approach used in Word2Vec (skip-gram or CBOW) or Glove (Matric factorization). Their basic approach was to pre-train two-layer BiLSTM for attention sequence to sequence translation, starting from GloVe word vectors and then they took the output of sequence encoder and called it a CoVe, Combine it with GloVe vectors and use in a downstream task-specific mode using transfer learning.

– **Deep Contextualized word Embedding (ElMo)**

Peters et al. (2018c) proposed ELMo, which gives deep contextual word representations. The final word vectors are learned from bi-directional language model (forward and backward LMs). ELMO uses the linear concatenation of representations learnt from bidirection language model instead of only just the final layer representations like other contextual word representations. ELMo produces varied word representations for the same word in different sentences. ELMo embeddings are based on the representation learned from Bi language model (BiLM). Log-likelihood of sentences in both forward and backward language models is involved in training process of BiLMs and final vector is computed after the concatenation of hidden representations from forward language model $\overrightarrow{h}^{LM}_{n,j}$ and backward language model $\overleftarrow{h}^{LM}_{n,j}$, where $j = 1, ...., L$ and is given below

$$BiLM = \sum_{n=1}^{k} (\log p(t_n|t_1, ....., t_{n-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_n|t_{n+1}, ...., t_n; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)$$

where $\theta_x$ and $\theta_s$ are the token representation parameters and softmax parameters, respectively, which are shared between forward and backward directions. And $\overrightarrow{\Theta}_{LSTM}$ and $\overleftarrow{\Theta}_{LSTM}$ are the forwarded back backward LSTM parameters respectively. ELMo abstracts the representations learned from an intermediate layer from BiLM and execute linear combination for each token in a downstream task. BiLM contains *2L+1* set representations as given below.

$$R_n = (X_x^{LM}, \overrightarrow{h}_{n,j}^{LM}, \overleftarrow{h}_{n,j}^{LM} | \quad j = 1, ...., L)$$
$$= (h_{n,j}^{LM} | \quad j = 0, ..., L)$$

where $h_{n,0}^{LM} = x_n^{LM}$ is the layer of token and $h_{n,j}^{LM} = [\overrightarrow{h}_{n,j}^{LM}, \overleftarrow{h}_{n,j}^{LM}]$ for each bi directional LSTM layer. ELMo is a task specific combination of these features where all layers in $M$ are flattened to single vector and is given below

$$ELMo_n^{task} = E(M_n; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{h,j}^{LM}$$

where $s^{task}$ are weights which are softmax normalized for the combination of different layers representations and $\gamma^{task}$ is a hyperparameter for optimization and scaling of ELMo representation.

Recently, many different State-of-the-art models have been presented in different studeis. Some of them are briefly given below:

- **Universal Language Model Fine-Tuning (ULMFiT)**

  Proposed by fast.ais Jeremy Howard and NUI Galway Insight Centers Sebastian Ruder, ULMFiT Howard and Ruder (2018) is essentially a method to

Table 2.1 : Comparison of Word Representation Models

| **Word Representation Models** | **Model** | **Architecture** | **Type** | **Pros** | **Cons** |
|---|---|---|---|---|---|
| **Legacy words Representations** | One Hot Encoding and BoW | - | Countbased | i) Easy to compute<br>ii) Works with the unknown word<br>iii) Basic metric to extract terms in a document | i) It does not capture the semantics & syntactic info.<br>ii) Common words effect on the results<br>iii) Can not capture sentiment of words |
| | TF and TF-IDF | - | | i) Easy to compute<br>ii) Basic metric to extract the most descriptive terms in a document<br>iii) Common words do not affect the results due to IDF | i) It does not capture the semantics & syntactic info.<br>ii )Can not capture the sentiment of words |
| **Continous words Representations** | Word2Vec | Log Bilinear | Prediction based | i)It captures the text semantics & syntactic<br>ii) Trained on huge corpus ( Pre-trained) | i) Fails to capture polysemy & words sentiments.<br>ii) It cannot capture OOV words<br>iii) Need huge corpus to learn |
| | GloVe | Log Bilinear | Count based | i) It enforce vectors to capture sub-linear relationships in the vector space<br>ii)Lower weight for highly frequent word pairs such as stop words will not dominate the training progress | i) It cannot capture the meaning of the word from the text (Fails to capture polysemy)<br>ii)Memory consumption for storage<br>iii) it cannot capture out-of-vocabulary words from the corpus<br>iv) Need huge corpus to learn ( Pre-trained) |
| | FastText | Log Bilinear | Prediction based | i) Works for rare words- n-grams which are still shared with other words)<br>ii) Solve OOV words issue. | i) It cannot capture the meaning of the word from the text (Fails to capture polysemy)<br>ii)Memory consumption for storage<br>iii)Computationally is more expensive in comparing with GloVe and Word2Vec<br>iv) Can not capture sentiment of words |
| **Contextual words Representations** | Context2Vec CoVe & ELMo | BiLSTM | Prediction based | i) It solves the polysemy issue | i) Improves performance<br>ii) Computationally is more expensive<br>iii) Need another word embedding for all LSTM and feed-forward layers<br>iv) Can not capture the sentiment of words |

enable transfer learning for any NLP task and achieve great results. All this, without having to train models from scratch. This method involves fine-tuning a pre-trained language model (LM), trained on the Wikitext 103 dataset, to a new dataset in such a manner that it does not forget what it previously learned. UMFiT was based on the state-of-the-art language model at that time which is LSTM-based model. The architecture and training method, ULMFiT (universal language model fine-tuning), builds on similar approaches of CoVE and ELMo. In CoVe and ELMo the encoder layers are frozen. ULMFiT instead describes a way to train all layers, and does so without overfitting or running into catastropic forgetting, which has been more of a problem for NLP (vs Computer vision) transfer learning in part because NLP models tend to be relatively shallow.

- **Transformer-Based Pre-tained Models**

  **Transformer** has been proven to be more efficient and faster than LSTM or CNN for language modeling and thus the following advances in this domain will rely on this architecture.

  - **GPT (OpenAI Transformer)** is the first Transformer-based pre-trained language model proposed by Radford et al. (2019), it uses the decoder of the Transformer to model the language as it is an autoregressive model where the model predicts the next word according to its previous context. GPT has shown a good performance on many downstream tasks.

  - **Bidirectional Encoder Representations from Transformers (BERT)**

    Devlin et al. (2018) proposed BERT in 2019, BERT is another contextualized word representation modelbased on a multilayer bi-directional transformer-encoder, where the transformer neural network uses parallel attention layers rather than sequential recurrence Vaswani et al. (2017). BERT is pre-trained on two unsupervisedtasks: (1) a masked language model, where 15% of the tokens are randomly masked (i.e., replaced with

the [MASK] token), and the model is trained to predict the masked tokens, (2) a next sentence prediction(NSP)task,where the model is given a pair of sentences and is trained to identify when the second one follows the first.This second task is meant to capture more long-term or pragmatic information.

BERT is trained on the Books Corpus dataset(800Mwords) and text passages of English Wikipedia. Two pre-trained model sizes for BERT are available: BERT-Base and BERT-Large. BERT can be used directly from the pre-trained model on un-annotated data, or fine-tuned on ones task-specific data. The pre-trained publicly available model and code for fine-tuning are available online [‡].

- **Other text embedding representations**

  After few months, a cross-lingual Language Model Pretraining (XLM) Lample and Conneau (2019) from Facebook enhanced BERT for Cross-lingual Language Model. A month later, GPT-2 proposed by Radford et al. (2018) from OpenAI scaled up the GPT in terms of the number of model parameters and training data and it showed a good ability to generate human-like text so OpenAI didnt publish the large model.

  Later in the next few months, Microsoft proposed Multi-Task Deep Neural Networks for Natural Language Understanding (MT-DNN) (Liu, He, Chen and Gao; 2019), MT-DNN outperformed GPT-2 on different NLP tasks, MT-DNN is a BERT-based model, it extends BERT by using multi-task training.

  The next month June 2019, Carnegie Mellon University and Google Brain introduced: XLNet: Generalized Autoregressive Pretraining for Language Understanding which proposes a new task to predict the bidirectional context instead of the masked Language task in BERT, it is a permutation language in which we make some permutations of each sentence so the two contexts will be taken into consideration. In July 2019 RoBERTa: A Robustly Optimized

[‡]https://github.com/google-research/bert

BERT Pretraining Approach (Liu, Ott, Goyal, Du, Joshi, Chen, Levy, Lewis, Zettlemoyer and Stoyanov; 2019) has been introduced, it is like a lite version of BERT but it has fewer parameters and better performance as it removes the training on the sentence classification task. In September 2019 Five pre-trained language models were introduced, Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism (Nvidia) Shoeybi et al. (2019), Lan et al. (2019) proposed ALBERT: A Lite BERT for Self-supervised Learning of Language Representations(Google), StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding (Alibaba) Wang et al. (2019), CTRL: A Conditional Transformer Language Model for Controllable Generation(SalesForce) Keskar et al. (2019),DistilBERT, a distilled version of BERT (Hugging Face) Sanh et al. (2019) have arrived.

DistilBERT reduces the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster. MegatronLM is a scaled up transform-based model, 24 times bigger than BERT. CRTL is a 1.63 billion-parameter conditional transformer language model, it is a conditional generative model. StructBERT extends BERT by incorporating language structures into pre-training. ALBERT, a lite version of BERT, establishes new state-of-the-art results while having fewer parameters compared to BERT.

Although these models were able to solve polysemy and contextual issues but were unable to more characteristics more effectively and efficiently, considering this issue researchers proposed hybrid models (presented in next subsection) to capture more information and characteristics of data without losing any information to improve the results, as shown in Table 2.1, we present a comparison of different word representation models.

### 2.3.2 Effects of Word representation methods

Below we present some relevant studies where different word representation models have been employed for various text classification tasks.

Pang et al. (2002a) performed that binary classification task on IMDb dataset and employed unigrams, bigram and POS tags as features. For classification, they used SVM, Maximum entropy and NB classifiers in their study and found out that best results were achieved with unigrams as feature and SVM as classifier. Kwok and Wang (2013) used N-gram features along with NB classifier tweets classification. These legacy based word representation methods such as N-grams, BoW, TF, and TF-IDF have been widely used in different studies for various text classification tasks (Greevy; 2004; Davidson et al.; 2017; Liu and Forss; 2014; Kouloumpis et al.; 2011). These traditional methods for text classification are simple, computationally economical. However, their limitations such as ignoring word order, unable to capture semantic information and high dimensionality etc. restrict their use for efficient text classification tasks.

Later, representation learning methods of learning text representation directly using neural network (Collobert and Weston; 2008) was adopted, which improved classification results. Word embeddings from continuous word representation models such as Word2Vec and GloVe are the most famous and widely used ones among these methods because of low dimensionality of vectors and capture semantic relationships. Word representation models have also been used for sentence-level classification task by averaging word vectors as feature representation which is utilized later on as input for sentence-level classification (Castellucci et al.; 2015).

Word embeddings which are created based on unigrams and by averaging embeddings are not able to capture the issue of syntactic dependencies like "but" and "negations" can change the complete meaning of a sentence and long dependencies within a sentence (Castellucci et al.; 2015). Socher et al. (2013) proposed a recursive neural network which can capture and model long semantic and sentiment dependencies of words and sentence at different stages. The disadvantage of this method is that it depends on parsing, which makes it challenging to use on Twitter related text (Foster et al.; 2011). A paragraph representation model solved this issue learns word vectors and does not reply on parsing (Le and Mikolov; 2014). Both of these, recursive neural and paragraph representation models have assessed on IMDb

dataset used by (Pang et al.; 2002b), and both models improved the classification results obtained by using BoW features.

Deep neural network-based methods have also been used for Text classification tasks. Tang et al. (2016a) proposed sentiment specific word representation model, which are achieved from emoticons labelled tweet messages with the help of the neural network. Severyn and Moschitti (2015) presented another neural network-based model where they used Word2Vec to learn embedding. Tweets are presented as a matrix wherein which columns compare with words, thus retaining the position they appear in a tweet. Emoticons annotated data was utilized to pre-train the weights and then trained by hand-annotated from SemEval contestant. The experimental results tell that pre-training step enables for a better initialization of the networks' weights and therefore, has a positive role in classification results. In another study conducted by Fu et al. (2017), Word2Vec was employed to get word representation which was forwarded to the recursive encoder as an input for text classification. Ren et al. (2016) also used Word2Vec to generate word representations and proposed a new model for the Twitter classification task. Lauren et al. (2018) presented a different document representation model where they used the skip-gram algorithm for generating word representations for the classification of clinical texts.

Due to the limitations and restrictions in a few corpora, pre-trained word embeddings are preferred by researchers as an input of machine learning models. Qin et al. (2016) used pre-trained Word2Vec embeddings and forwarded these word embeddings to CNN. Similarly, Kim (2014a) utilized pre-trained embeddings of Word2Vec and forwarded to CNN neural network, which increased the classification results. Camacho-Collados et al. (2016) for concept representation in their work. Jianqiang and Xiaolin (2018) have initialized word embeddings using pre-trained GloVe embeddings in their DCNN model. Similarly, Wallace (2017) applied GloVe, and Word2Vec pre-trained word embedding in deep neural network-based algorithms and enhanced the classification results. Similarly, a study conducted by Wang et al. (2016), used pre-trained GloVe embeddings as an input to LSTM with attention model for aspect based classification and Liu et al. (2018) employed pre-trained

word embeddings Word2Vec for recommending idioms in essay writing. Recently, Ilic et al. (2018) have used contextual word embeddings (ELMo) for word representation for the detection of sarcasm and irony and shown that using ELMo word representaitons have improved the classification results. The research community has made limited efforts for solving the above mention limitations of continuous word representation models by proposing different models. For example, for handing OOV words which are not seen in the training and they are assigned UNK token and same vector for all words and degrades results if the number of OOV is large. Different methods to handle OOV words have been proposed in different studies (Dhingra et al.; 2017; Herbelot and Baroni; 2017; Pinter et al.; 2017) But still these models does not capture the polysemy issues. This issue of words with different meanings (polysemy) is addressed in different models presented by the (Neelakantan et al.; 2015; Iacobacci et al.; 2015; Pilehvar and Collier; 2016). In recent days, researchers has presented more robust models to handle OOV words and polysemy issues (Liu et al.; 2015; Melamud et al.; 2016b; McCann et al.; 2017a; Peters et al.; 2018a).

To handle domain-specific problems different studies have been conducted where researchers used existing knowledge encoded in semantic lexicons to these word embedding to improve the downsides of using the pre-trained embedding which is trained on news data which is usually different from the data we use in our tasks. Some of the models presented are proposed in the following studies which inject external knowledge in existing word embedding and improves the results. Faruqui et al. (2014); Speer et al. (2016); Mrksic et al. (2017); Seungil et al. (2017); Niebler et al. (2017). Word embeddings are beneficial in different areas beyond NLP like link prediction, information retrieval and recommendation systems. Ledell et al. (2017) proposed a model which is suitable for many of above-menitoned applications and acted as a baseline. None of the above mentioned is robust enough and fails to integrate sentiment of words in the representations and does not work well in domain-specific tasks such as sentiment analysis etc.

Studies show that adding sentiment information into conventional word repre-

sentation models improves performance. To integrate the sentiment information into word embeddings, researchers have proposed different hybrid word representations by changing existing skip-gram model (Tang, Wei, Yang, Zhou, Liu and Qin; 2014). Tang et al. (2016b) proposed several hybrid ranking models (HyRank) and developed sentiment embeddings based on C&W, which considers context and sentiment polarity of tweets. Similarly, several other models are presented, which considers context and sentiment polarity of words for sentiment analysis. (Tang et al.; 2015; Liang-Chih, Wang, Lai and Zhang; 2018; Rezaeinia et al.; 2017b). Liang-Chih, Wang, Lai and Zhang (2018) proposed sentiment embeddings by refining pre-trained embeddings Re(*) using the intensity score of external knowledge resource. Rezaeinia et al. (2017c) proposed improved word vectors (IWV) by combining word embeddings, part of speech (POS) and combination of lexicons for sentiment analysis. Recently, Cambria et al. (2018) proposed context embeddings for sentiment analysis by conceptual primitives from text and linked with commonsense concepts and named entities.

## 2.4 Gap Analysis

Text representation embeds textual data into a vector space, which significantly affects the performance of downstream learning tasks. Better representation of text is more likely to facilitate better performance if it can efficiently capture intrinsic data attributes. Below we briefly highlight the limitations of categorical and continuous word representation models. Since our study is focused on handling the words with different meanings (polysemy), handling OOV words and capturing the sentiment of words within the context so in this report we will restrict the detailed analysis of below-mentioned limitations within our scope of the study.

- Legacy word representation methods (Explained in 2.3.1 ) like categorical and weighted word representations are the most naive and most straightforward representation of textual data. These legacy word representation models have been used widely in early days for different classification tasks like document classification, Natural language processing (NLP), computer vision and infor-

mation retrieval. The categorical word representation models are simple and not difficult to implement but their limitations such as they do not consider capture semantics and syntactic information because they do not consider the order of words and do not consider any relationship between words. Further, the size of the input vector is proportional to vocabulary size, which makes them computationally expensive, which results in poor performance.

- Representation learning methods (Explained in 2.3.1) has helped the research community to build powerful models. However, its drawback is that the features need to be selected manually so to solve this shortcoming there was a need to present some methods which can discover and learn these representations automatically for any downstream task. This automatic extraction of features without human intervention is known as representation learning which has improved results drastically over the past few years in many areas such as image detection, speech recognition, NLP etc. (LeCun et al.; 2015). Continuous word representation models like Word2Vec , GloVe and Fasttext (Mikolov et al.; 2013; Manning et al.; 2014; Joulin et al.; 2016; Bojanowski et al.; 2016b) etc. have drastically improved the classification results and overcome shortcomings of categorical representations. It is found that having these continuous word representation of words is more affected as compared to traditional linguistic features because of their ability to capture more semantic and syntactic information of the textual data without losing much information. Despite their success, there are still some limitations which they are not capable of addressing such as they are unable to handle polysemy issues because they assign the same vector to word and ignores its context. Also, models like Word2Vec and GloVe assigns a random vector to a word which they did not encounter during training which means they are unable to handle out of vocabulary (OOV) words which were solved by Fasttext which breaks words into n-grams. Moreover, from our prior knowledge, we know that some words such as word *Good & Bad* have some sentiments associated with them which these models fail to capture and lowers the performance in case of sentiment

analysis etc. All of these limitations degrades the performance of text classification. Moreover, all of the current state-of-the-art methods do not perform well in the case of the low-quality text.

Table 2.2 : Gap Analysis

| Models/Language Characteristics | Semantics | Syntactical | Polysemy | Out of Vocabulay | Words Sentiment |
|---|---|---|---|---|---|
| 1-Hot encoding | [×] | [×] | [×] | [×] | [×] |
| BoW | [×] | [×] | [×] | [×] | [×] |
| TF | [×] | [×] | [×] | [×] | [×] |
| TF-IDF | [×] | [×] | [×] | [×] | [×] |
| Word2Vec | [✓] | [✓] | [×] | [×] | [×] |
| GloVe | [✓] | [✓] | [×] | [×] | [×] |
| FastText | [✓] | [✓] | [×] | [✓] | [×] |
| Context2Vec | [✓] | [✓] | [✓] | [✓] | [×] |
| CoVe | [✓] | [✓] | [✓] | [×] | [×] |
| ELMo | [✓] | [✓] | [✓] | [✓] | [×] |
| HyRank | [✓] | [✓] | [×] | [×] | [✓] |
| Re(*) | [✓] | [✓] | [×] | [×] | [✓] |
| IMV | [✓] | [✓] | [×] | [×] | [✓] |

- **Discussion**:

  Learning word representation which captures polysemy, the sentiment of words handles OOV words without losing syntactic and semantic information of words together has not been fully explored yet in the current research. Existing methods did not quantify: (1) the effectiveness of improving the quality of text systematically; (2) loss of information in selecting an inappropriate combination of pre-processing techniques; (3) effectiveness of embedding different text characteristics. Without these, it is hard to learn an appropriate word representation with satisfying quality.

  Hence we can conclude that: i) No work has proposed a systematic approach

to combine preprocessing techniques to improve the quality of text in a way that we do not lose any useful information, ii) No work has proposed a unique hybrid combination to handle language ambiguities such as polysemy; OOV words, Words sentiments and semantic and syntactical information of words.

## 2.5 Summary

In this section, a summary of the literature review discussed in this chapter.

In section 2.1, text classification pipeline is briefly discussed where we introduced the basic definitions each part in text classification pipelines such as unstructured text, text representation and classification. Since, the scope of this thesis is to have a better representation of text by improving the quality of text and extracting better features, the rest of the literature presented is focused according to the scope of our thesis. In section 2.2, first, we presented some of the common and advanced pre-processing techniques in detail how their application effect the text and some of the famous methods presented in the literature for these techniques. We looked for papers which presented the effect of pre-processing on text classification and summarised them in this section. This section, highlights the importance of the role of pre-processing in text classification and presents pre-processing techniques which should be considered and which should not be considered for the text classification task. Section 2.3 presents the working mechanism of the famous word representation models. We divided our literature review on these word representation models into different categories such as legacy word representation models where we discussed categorical and weighted word representation models, in representation learning we discussed continuous and contextual word representation models and we also discussed famous hybrid and sentiment specific word representation models followed by their related work where these models have been applied on different text classification tasks. Lastly, we critically analysed all of these models and presented and discussed the gap analysis in current literature presented in section 2.4. The classification algorithms and evaluations metrics are discussed in Appendix A.

# Chapter 3

# Methodology

In this chapter, we discuss the research methodology adopted. First, we defined the research problems, followed by research questions and objectives.

## 3.1 Research Problems, Objectives, and Questions

This research focuses on learning deep hybrid word representations for low-quality text such as tweets. Our research focuses on proposing methods to improve the low-quality text, which helps to learn better features and handle the language ambiguities for the text classification task. In recent years, handling language complexities has become a growing and famous research area in many different NLP tasks. We formally defined our research problems, Objectives and questions below.

### 3.1.1 Research Problems

Given a tweet $T_x$ which has unique characteristics such as i) unstructured and informal nature depicts low quality of text; ii) have same words with different meanings in the context (polysemy); iii) words with different sentiments at input to the classifier with a sequence of tokens $(t_1, t_2, ...t_k)$, 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 the set of fixed labels $y_1, y_2, ...y_k$ at output . Whereas at the input of learner a set of $n$ hand-labeled tweets $(T_1, y_1), ...., (T_n, y_n)$ given and a learned classifier $f(T)$ predicts the label (class) $y$ of a tweet.

Recently, different studies have been conducted which tries to handle language complexities (Cambria et al.; 2018; Tang et al.; 2016a; Rezaeinia et al.; 2017a) but all of these methods deals with the structured and clean text. In addition

to that, not any method can handle all language complexities summarized in Figure 3.1. It can be seen in the figure that all red Highlighted words in the tweets are unstructured and informal, which depicts the low quality of text in tweet messages and needs to improve for better classification results. All green Highlighted words show the problem of polysemy within the tweets where the same word can have different meaning depending on its context, which traditional word embeddings are unable to capture. Similarly, polarity of green highlighted words "**bad**" and "**good**" can not be captured by traditional methods as well. All these language/textual complexities degrades the performance which needs to be fixed.

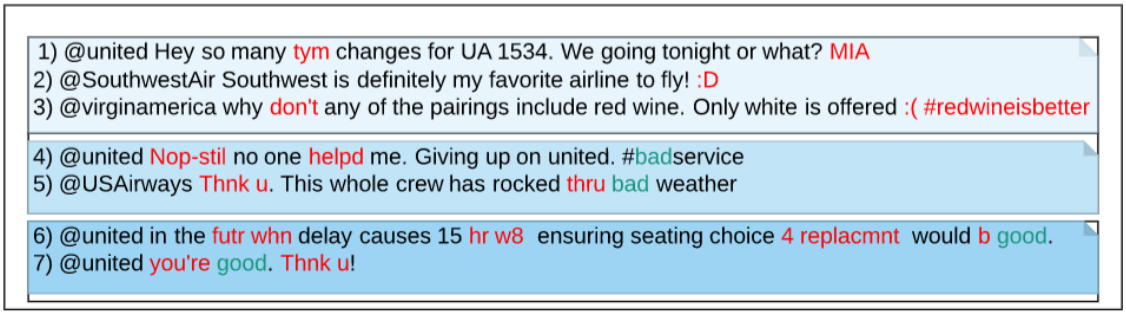


Figure 3.1 : The examples of research problems.

### 3.1.2 Research Questions

In this thesis, we try to address the following research questions:

**RQ:1** What is the effect of different common and advanced preprocessing techniques on text classification task?

**RQ:2** What is the recommended systematic approach to combine the preprocessing techniques to improve the low quality of the text?

**RQ:3** How we can enable computers to handle the following language ambuiguities: polysemy, context, syntax, words sentiments, unstructured text and out of vocabulary words" and classify text like humans?

**RQ:4** What is the best representation of a Tweet $T_x$ defined in section 3.1.1so that it can be classified accurately?

### 3.1.3 Research Obejctives

Based on the aforementioned research problems defined in the previous section, we want to have a representation $f(T)$ of a tweet $T_x$, which can capture all mentioned characteristics. Formally, we focus on the following research objectives:

- **To improve the quality of text**: In this objective, our prime focus is to improve the low-quality text to get a proper representation of the text. First, we explored the impact of some common and advanced pre-processing techniques and explored which techniques perform well and which degrades the classification results. Then we present a systematic combination of different pre-processing techniques which replaces emoticons with its associated meanings, replaces abbreviations and slangs, correct spelling mistakes, word segmentation, expand contractions and remove noise, to utilize the hidden features in raw and unstructured text. This objective addresses the limitation of low-quality text which is overlooked in previous studies and addresses the first two questions [**RQ1 and RQ2**] of our research defined in section 3.1.

- **To handle natural language ambiguities**: Ambiguity can be defined as the fact or a statement having more than one meaning and therefore can cause a confusion*. Natural language is ambiguous in nature which humans can understand, but the computer fails to understand the linguistic ambiguities. In this objective, language ambiguities we focus specifically are the words with different meanings in the context (Polysemy), Sentiment knowledge of words, semantic and syntactical information and addresses the third and fourth question [**RQ3 and RQ4**] of our research defined in section 3.1.

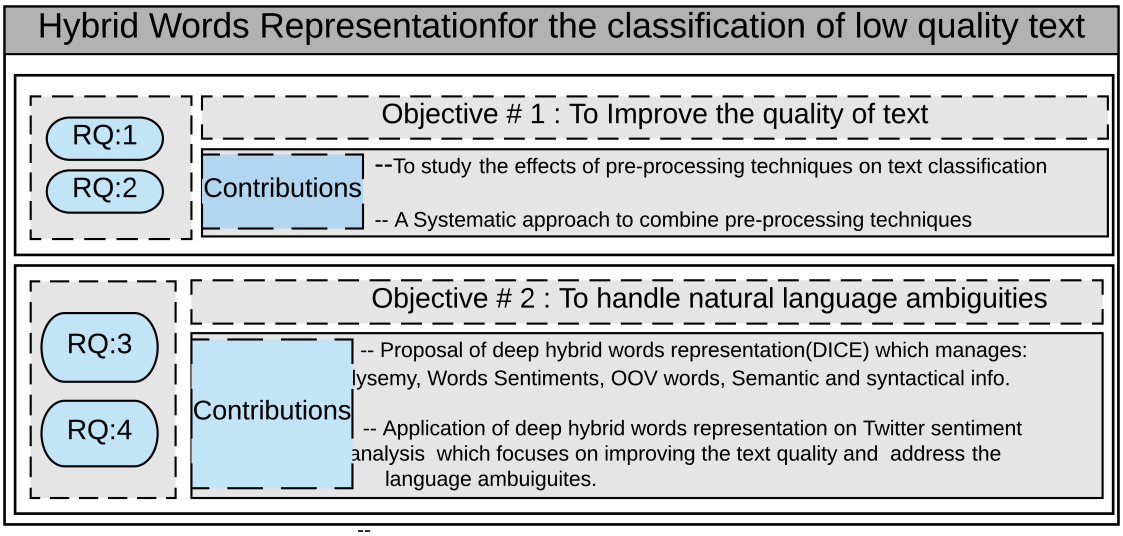


Figure 3.2 : The relationship of Research questions, Objectives and Contributions

Figure 3.2 shows the relationship of Research questions, Objectives and Contributions of this thesis.

*https://dictionary.cambridge.org/dictionary/english/ambiguity

# Chapter 4

# A Recommeneded combination of pre-processing techniques to improve quality of text

## 4.1 Introduction

Social media platforms play an essential part in understanding the content published online. Analysis of data coming from social media platforms, especially Twitter, has become one of the primary focus of the researchers. Millions of Twitter users* are sharing their opinions and views on different subjects such as political debate, stock market, hate speech, products, etc. which is crucial from the point of improving services, marketing strategies, users behaviour and trends. Since the Twitter messages are restricted to only 140 characters which enforce Twitter users to use different acronyms, emoticons, slangs, and URLs etc. to give extra information. The use of these informal and unstructured ways of writing degrades the quality text due to its unstructured and informal nature. Different pre-processing techniques have been applied for the text classification task, but there is no recommended combination of pre-processing, which would improve the quality of the text.

Pre-processing of tweets is the process of cleaning and preparing the low-quality text that is going to be classified. On social media, Twitter in our case, more than 550 millon† users post their tweet messages, and every Twitter user write in their style, use abbreviations, different punctuations, incorrect spellings, use emoticons and emojis to express their emotions and opinions, use slangs and acronyms as well as URLs, hashtags and user mentions. All of these language imperfections cause much noise in the data and degrades the classification performance.

*https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

†https://www.omnicoreagency.com/twitter-statistics/

According to a study conducted by Fayyad et al. (2003), noise in the datasets can be at the level of 40% of the data which can confuse classifiers performance and result in the bad classification results. Similarly, in another study by Haddi et al. it is shown that online text is usually unstructured and contains noise which enhances the dimensionality issues and degrades the classification performance (Haddi et al.; 2013). So, applying an appropriate combination of pre-processing techniques in a systematic way that does not lose information and helps to learn better representation for classification has not been much explored in previous studies.

Applying different pre-processing techniques randomly may lose the information and degrade the classification performance. In order to capture more information, we present step by step recommended combination of pre-processing, which helps to improve the quality of text and result in better classification performance. In this chapter first, we bring together the effects of some common and advanced pre-processing techniques one by one, and then we present our proposed recommended combination of applying different pre-processing techniques.

**Challenges:** The language used on social media and blogs is ubiquitous as very often; it is unstructured and very informal. Users write in their own words, use abbreviations, different punctuations, incorrect spelling, emoticons, slang and URLs, etc. All of those language imperfections cause much noise in the data, and one of the significant challenges is to handle this unstructured and informal text by applying appropriate cleaning and combination of pre-processing techniques. At the same time, another crucial challenge is to consider the order of pre-processing techniques to avoid losing useful information.

**Objective:** Study and understand the effects of pre-processing techniques on classification task and propose a method to improve the low quality of text which helps to improve classification performance.

Our proposed method addresses below-mentioned issues.

- Remove noise and normalize the low-quality text,
- Replace Abbreviations and acronyms with actual words,

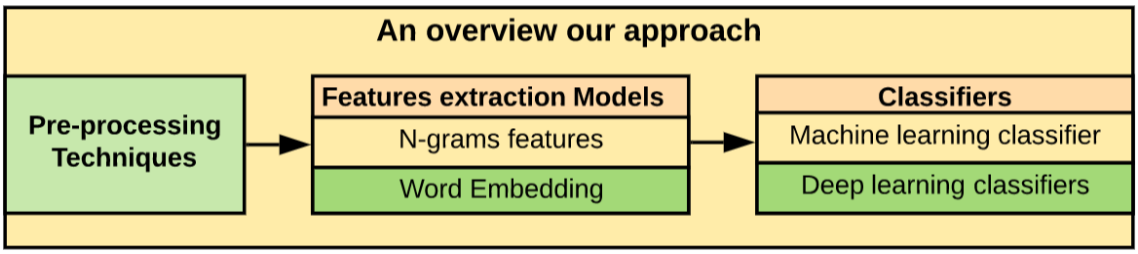


Figure 4.1 : An overview of our approach

- Replace Emoticons and Emojis with their associated meanings,
- Spell correction and Word segmentation of words used in hashtags etc.

**Contributions :**

Specially the main contributions of our work are as follows:

- Evaluation of different common and advanced pre-processing techniques and their effect on classification results.
- Presented a recommended combination of pre-processing which improves the quality of the text.

## 4.2 Methodology

In this section, we present an analysis of one pre-processing technique at a time to analyze the effects on classification results followed by the recommended combination of pre-processing techniques. The overview of our methodology is given in Figure 4.1.

### 4.2.1 Analysis of one pre-processing techniques at a time

In this section first, we present the analysis of 12 different pre-processing techniques one by one. We present the methods and techniques we used to analyze the effect of these techniques (discussed in section 2.2.1). Below we present each of these 12 pre-processing techniques:

- URLS, User-mentions and Hashtag symbols: As we are aware the in most of the tweet messages contains URLs, User-mentions and Hashtags symbols which users use in their messages to give extra information. This extra information is useful for humans, but it is considered as noise and not much use for computers. In our anlaysis, we have removed all URLs, user-mentions and hashtag symbols. An example is given below:

  **Before:** most *#hated* but the hoes favorite *#2MW #SevenOne* http://ow.ly/VIbf0

  **After:** most hated but the hoes favorite 2MW SevenOne

- Abbreviations and slangs: As mentioned earlier, that character limitation forces social media users to use different abbreviations and slangs in their messages. This problem becomes more trivial when every use writes in his style and use different abbreviations. Acronyms and phrases which sometimes are restricted to some specific context or group of people. To interpret these language imperfections, it is vital to replace them with their associated meanings which can make computer to understand them easily. In our experiments we used ekphrasis library to replace abbreviations and acronyms to replace with their associated meanings. An example is given below:

  **Before:** hey so many time for UA 1534 changes. We going tonight or what? MIA. :(

  **After:** hey so many time for UA 1534 changes. we going tonight or what? Missing In Action :(

- Expanding Contractions: Expanding contractions can be beneficial pre-processing technique especially before performing tokenization step because it will make two different tokens of contraction such as *can't* into "*can*" and "*t*". Expanding contractions can help us not to lose information because we need the word "*not*", which is essential to keep during classification task. In our analysis, we employed pycontractions 2.0.0 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.

- Removing Numbers: Keeping numbers does not give extra information in the text classification task, and it is common practice to remove numbers from the corpus. However, removing numbers randomly may lose information. For instance, if we remove "8" from "gr8" then we lose useful information which can degrade the results. So removing numbers should always be performed after replacing abbreviations and slangs with their associated 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

- Replace Emoticons: Using emoticons to express their opinions and sentiments on social media has been used excessively in recent past. Humans can understand the emotions and sentiments behind these emoticons but not for computers. So in order to get maximum information in our experiments, we used the ekphrasis library to replace these emoticons with their associated meanings. An example is given below:

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

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

- Lemmatization: In order to group different extended forms of words into one word which can make the analysis more straightforward, we used lemmatization to replace words with their root words. In our analysis, we used WordNet lemmatizer library to perform this step. An example is given below:

**Before:** poorly serviced. Give us a chance atleast once.

**After:** poorli serviced. Give us a chance atleast onc.

- Removing punctuations: One of the classic and common pre-processing in most of the text classification task is to remove punctuations. Using these punctuations are useful for humans to understand the opinion and sentiments, but for computers, it does not add much to the performance. So in our study, we removed all punctuations. An example is given below:

  **Before:** Thank you #unitedairlines for the free gift for our son at #childrens-mercy hospital in KC!

  **After:** Thank you #unitedairlines for the free gift for our son at #childrens-mercy hospital in KC

- Words Segmentation: As previously mentioned, character limitation enforces users to write in an unstructured and informal way. Social media users in their hashtag messages use different phrases/words to express their emotions which are readable and understand by humans but to make it understood by computers. In our study, we separated the remaining content/phrases after removing hashtag symbols. An example is given below:

  **Before:** goodvibes United Airlines Flies Children With Serious Illnesses To Santa's North Pole

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

- Lower-casing of words: Lower-casing of all words is also one the common pre-processing techniques. It helps to decrease the dimensionality issues and also helps to match the same words 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

- Removing Stop words: Many words in the language are used which do not help to understand the meaning of the sentence are widely used in the text.

For instance, words like ( the, am, are etc.). Removing stop words is common practice in text classification tasks. In our analysis we used NLTK stop-word library 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:** thought Comcast bad, saw bad side United Airlines

- Elongated Characters: In order to avoid for the learner to not treat elongated words from their base words we replaced characters which are repeated three times to one character and borrowed this idea from (Kiritchenko, Zhu and Mohammad; 2014). An example is given below:

  **Before:**Gooood for you, @united. United Airlines brings back free snacks

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

- Incorrect Spellings: Incorrect spelling are commonly found in social media posts and messages. Sometimes users intentionally use wrong spellings which is understandable for humans. We also study the effect of correcting spelling mistakes in this analysis. We use Norvig's spell corrector[‡] 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

Table 4.1 shows all 12 pre-processing techniques and numbers associated with them for analysis in the next section 4.3.3. The technique number 0 represents the unprocessed text which we used as the baseline for the comparison of results in our study.

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

Table 4.1 : Pre-processing Techniques and their associated Numbers

| Technique Number. | Pre-Processing Technique |
|---|---|
| 0 | Baseline |
| 1 | URLs, User-mentions & Hashtag symbol |
| 2 | Replace Abbreviations and Slangs |
| 3 | Expanding Contractions |
| 4 | Removing Numbers |
| 5 | Replace Emoticons |
| 6 | Lemmatization |
| 7 | Removing Punctuations |
| 8 | Words Segmentation |
| 9 | Lower-casing of words |
| 10 | Removing Stopwords |
| 11 | Elongated Charachters |
| 12 | Incorrect Spellings |

### 4.2.2 Recommended combination of pre-processing techniques

Following the analysis of one technique at a time now, we present recommended systematic combination of pre-processing techniques. The motivation behind this is to improve the quality of text and find a combination of techniques which performs best when compared with others.

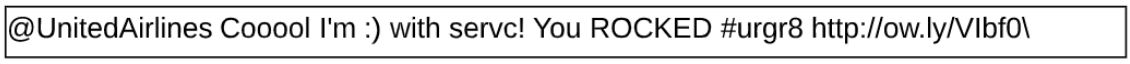


Figure 4.2 : Toy Example

Researchers usually apply four to five common data pre-processing techniques before features extraction step. However, for the poor quality of the text, especially in the case of Twitter, more pre-processing techniques need to be applied to nor-

malize and improve the quality of the text. Selecting an appropriate order with the right combination of pre-processing techniques is essential to improve the quality of the text. Language imperfections such as spelling mistakes, abbreviations and emoticons etc. do not provide useful information to computers. Hence replacing and normalizing them to their actual and base words is useful for computers to understand. There are plenty of pre-processing techniques but choosing the right combination, which improves the quality of the unstructured text is not an easy task. Also, not all techniques perform/interact well when they are combined with others; even they perform well when used alone.

Further, not following a specific order of pre-processing techniques can result in losing information which ultimately degrades the classification performance. Recommending a combination which improves the quality of text has not been explored in previous studies. Possible number of combinations for comprehensive evaluations are 12! which itself is a different research to findout which combinations interact well eachother when combined. To limit this huge possible research space, our proposed method is the result of testing different combinations which interact well with other techniques and is designed considering the motivation of improving the quality of the text. To find out the interactions and combinations of different pre-processing techniques to improve the quality of the text, we considered the tweet (toy example) given in Figure 4.2. In addition to that, to get better quality of raw text, for few techniques such as technique 1, 4, 10 and 6 has to be in the same order we proposed which also limits (reduces) the possible number of combinations to 8! . We explored different combinations but combinations with significant results are discussed here.

As mentioned earlier, selecting an appropriate order of different data pre-processing techniques is essential. For instance; if we remove numbers from the word "gr8" randomly before replacing it to actual word "great" from a toy example, then we lose a piece of useful information. Similarly expanding contractions "I'm", especially in case of negative contractions such as "couldnt" or "havent", after tokenization because tokenizer will break "couldn't" in to couldn" and "t" and "haven't" into "havn" and "t". These tokens do not capture the useful information which degrades

the performance. Further, for a word "urgr8", first removing hashtag symbol, then replacing abbreviations with actual words and in the last performing word segmentation changes it into a "you are great" and in case we change order and remove numbers or performed word-segmentation then we again we useful lose information. Similar is the case with other techniques which either does not interact well with others techniques and/or result in losing useful information which degrades the results. Keeping the mentioned motivation in mind, we experimented with different combinations[§] (given in Table 4.2) to analyze the interaction and combination of different techniques.

Table 4.2 : Different combinations of pre-processing techniques

| Combination # | Names Associated for future reference | Techniques numbers take from Table 4.1 |
|---|---|---|
| 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 |

Based on experimental results (Section 4.3.3), the best performing combination is the 8th combination which is graphically shown in Figure 4.3. In our proposed 12 steps recommended combination method, the first step we removed all the Unicode strings, URLs, user-mentions and hashtag # symbol which helps to remove the irrelevant information which is good for humans but not for computers. Then we replaced emoticons and emojis with their associated meanings, abbreviations and acronyms and correction of spelling mistakes at step no.2, 3 and 4 respectively. At step 5 and 6, we expanded contractions and replaced elongated characters, respectively. In remaining steps, we removed all punctuations, lower-cased all words, performed

[§]Other different combinations as well and only combinations with significant/best results are reported

word segmentations, removed numbers and stop-words and at the end performed lemmatization. In all steps, as mentioned above, we used the same methods and techniques we used earlier during their analysis individually. The importance and significance of the proposed combination is given in subsection 4.3.3 and subsection 4.3.4 respectively.

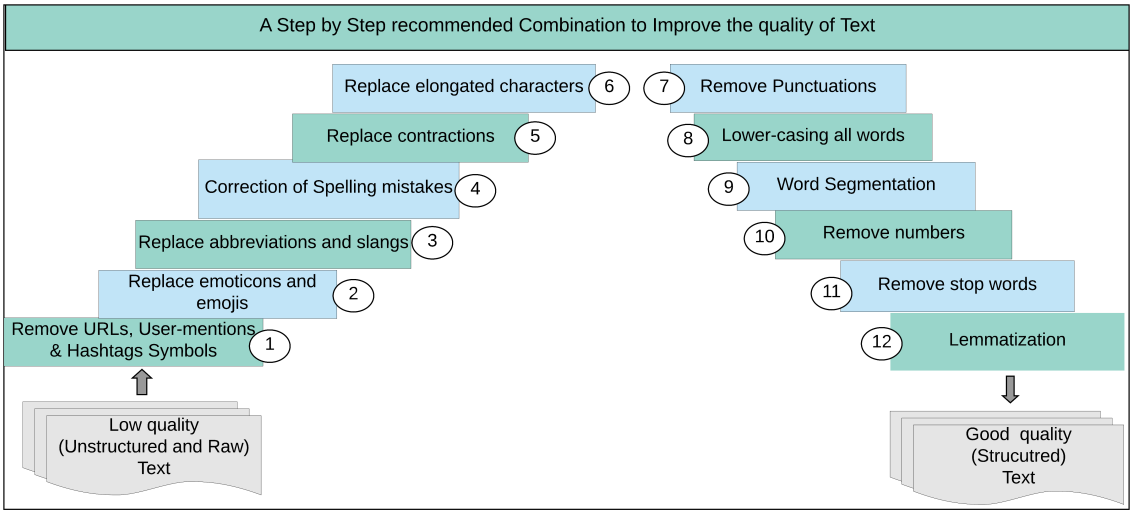


Figure 4.3 : A Recommended combination of pre-processing

## 4.3 Experimental Analysis

In this section, first, we present the experimental settings followed by the datasets used in our analysis and finally, the results are discussed.

### 4.3.1 Experiment Settings

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

#### *Words Representation*

As mentioned in chapter 2, different word representation models are available to chose from but in this study we selected one from legacy word representation methods $TF - IDF$ with N-gram ($N = 1, 2, 3$) features extraction model and one from continous word representation models ($GloVe$) explained in section 2.3.

***Classifiers***

To provide a comprehensive analysis of techniques, we have used five commonly used traditional machine learning and two deep learning-based classifiers to assess the effect of different pre-processing techniques in the 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 $TF - IDF$ word representation model.

**Deep learning classifiers**: From the deep learning-based classifiers, we used two commonly used algorithms; i) Convolutional Neural network (CNN) and Recurrent Neural network (RNN) with $GloVe$ word representation model.

In particular, we used implementation of Kim (2014b) for CNN and RNNs, we borrowed implementations of Tree-Long Short Term Memory (LSTM) by Looks et al. (2017) and bi-directional LSTM (Bi-LSTM) by Tai et al. (2015) with default parameters, as parameter optimization is not the part of this analysis.

***Evaluation Metrics***

For evaluation of our proposed methods and comparison with our models, we have used F1-Score metric which can be computed by:

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision}$$

### 4.3.2 Datasets

For the extensive evaluation, we investigate how different pre-processing techniques behave when applied to three different Twitter datasets related to Twitter hate speech and abusive language. Datasets statistics are given in Table 4.3 and briefly discussed below:

- **Golbeck et al. Dataset**

Golbeck et al. (2017) provided a large human labelled corpus for online harassment data on Twitter. First, a list of keywords was produced for the collection of tweets which contains harassing words; then coders were given guidelines to label 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.

- **Waseem et al. Dataset**

  Hovy and Waseem (2016) provided a labelled dataset of 16,914 tweets out of 136,052 collected tweets over a period of two months. Tweets were manually annotated and classified into three classes such as racism, sexism and neither. 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 so we could retrieve only 15,844 tweets with pythons Tweepy library, which are labelled into three classes.

- **Davidson et al. Dataset**

  Davidson et al. (2017) presented our third dataset, which focuses on differentiating between hateful and offensive language. Dataset was manually annotated into three classes: hateful, offensive and neither. The total number of tweets are given in this dataset is 25,112.

### 4.3.3 Results and Discussion

In this section, first, we present an analysis of different pre-processing techniques used, followed by the recommended combination of pre-processing techniques.

#### *Analysis of one pre-processing techniques at a time*

Table 4.4 to Table 4.6 presents the results[¶‖] of all pre-processing techniques for three datasets. It can be seen clearly that classification performance is not

[¶]Green & Red colors denote the highest & lowest performing techniques in each case respectively (rows)

[‖]High performing techniques in each case are marked as bold

Table 4.3 : 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 |

consistent, which depicts the effects of each technique on results. Below we discuss the effects of each pre-processing technique as compared to our baseline technique results. For the sake of simplicity, we discuss the techniques which managed to improve the performance $\geq 1$ as compared to baseline.

1. **Removal of URLs, user-mentions and Hashtag Symbols**: This technique has managed to increase the performance on two datasets. Increase in performance observed in the case of SVM (trigrams), LR (bigrams), CNN and BiLSTM classifiers on Waseem et al. dataset and SVM (unigram and bigram), LR (unigram and bigram), DT (unigram) and all neural network-based classifiers. No significant increase is observed in the case of the third dataset. The reason behind this improvement is the number of URLs, user-mentions and hashtag symbols are enormous in these two datasets as compared to Golbeck et al. dataset. The highest results achieved by this technique is 0.695 for Davidson et al. dataset.

2. **Replace Abbreviations and Slangs**: Experimental results show that performance is increased in all datasets when we replace abbreviations and slangs with their associated meanings. For Waseem et al., dataset performance increase in the case of CNN and BiLSTM. For Golbeck et al. increase observed in both RNN based classifiers whereas, in case of Davidson et al. dataset

a significant increase in performance is observed in all neural network-based classifiers and all traditional machine learning classifiers but with results varies with different features. The prime reason for this increment in performance is that the number of abbreviations and slangs in Davidson et al. dataset is more than the other two datasets. The highest results achieved by this technique is 0.669 for Davidson et al. dataset.

Table 4.4 : Comparison of preprocessing 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 |

3. **Expanding Contractions**: Expanding contractions to root words also showed the performance in all three datasets. In the case of Waseem et al. performance increased only in the case of CNN and LSTM, whereas for Golbeck et al. performance increased only for LSTM classifier. The performance increased for all classifier (with different features) in for the third dataset. The highest results achieved by this technique is 0.644 for Davidson et al. dataset.

4. **Removing Numbers**: This technique has outperformed the baseline results on one classifier in the case of Waseem et al. dataset (LSTM) and Golbeck et al. dataset. (LSTM) Whereas in the third dataset it beats baseline in SVM

(unigram and bigram), LR(trigram), DT(unigram), RF(bigram and trigram) and from neural network-based classifiers it beats baseline in both CNN and LSTM classifiers. The highest results achieved by this technique is 0.644 for Davidson et al. dataset.

5. **Replace Emoticons**: This technique has managed to improve the performance only on two datasets. In the case of Golbeck et al. performance increase in CNN based classifier whereas, for Davidson et al. dataset, performance increased in almost all classifiers. The reason for this increase is that the number of emoticons in our third dataset is more as compared to the other two datasets. The highest results achieved by this technique is 0.684 for Davidson et al. dataset.

6. **Lemmatization**: This technique has managed to show significant improvement in Davidson et al. dataset in almost all cases. Whereas, in the remaining two datasets, performance increased only in the case of CNN(Waseem et al.) and LSTM (Golbeck et al.) based classifiers. The highest results achieved by this technique is 0.671 for Davidson et al. dataset.

7. **Removing Punctuations**: Intrestignly removing punctuation did not give any significant results when used alone and able to beat the baseline results in only two datasets. LSTM (Golbeck et al.) and LST, RF (unigram) and NB (unigram) in the case of Davidson et al. dataset. The highest results achieved by this technique is 0.637 for Davidson et al. dataset.

8. **Words Segmentation**: Separating remaining content of hashtag words can beat baseline results in all datasets. In case of Waseem et al. results improved for RNN based classifiers, for Golbeck et al. only LSTM based classifier was able to beat baseline score whereas, in the third dataset, all neural network-based classifiers outperformed baselines results. Also results improved for SVM(trigram), LR(trigram) and DT(trigram) classifiers. The highest results achieved by this technique is 0.713 for Davidson et al. dataset.

9. **Lower-casing of words**: This commonly used technique showed improve-

Table 4.5 : Comparison of preprocessing 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 |

ment in all datasets. For Waseem et al. results improved in case of DT(trigram), RF(trigram) and CNN based classifiers. For Golbeck et al. dataset, only LSTM and RF(bigram) were able to beat baselines results. For Davidson et al. dataset, performance increase in case of LSTM, RF(trigram), DT(unigram and trigram), LR(bigram and trigram), NB(unigram) and SVM(bigram and trigram) classifiers. The highest results achieved by this technique is 0.648 for Davidson et al. dataset.

10. **Removing Stopwords**: This common technique has also outperformed the baseline results in all datasets. For Waseem et al. BiLSTM and NB(unigram) showed improvement. NB(bigram) and LSTM for Golbeck et al. dataset and all neural network-based classifiers and RF(unigram) for Davidson et al. dataset. The highest results achieved by this technique is 0.674 for Davidson et al. dataset.

11. **Elongated Characters**: This technique has managed to beat baseline results only in-case of two datasets ( Davidson et al. and Golbeck et al.). The reason

is that the number of elongated characters is more in these two datasets as compared to Waseem et al. dataset. The highest results achieved by this technique is 0.688 for Davidson et al. dataset.

12. **Incorrect Spellings**: Correcting spelling mistakes have been able to beat baseline results in all datasets. For Waseem et al. results are improved in case of CNN and BiLSTM based classifiers. For Golbeck et al. only RNN based classifiers showed improvement whereas, all neural network-based classifiers along with DT(trigram), LR(bigram) and SV(trigram) outperformed baselines results in Davidson et al. dataset. The highest results achieved by this technique is 0.699 for Davidson et al. dataset.

Table 4.6 : Comparison of preprocessing 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** |

***Summary***

In the previous subsection, we discussed the effects of 12 different pre-processing techniques on three Twitter hate speech and abusive language datasets. According to results given in Table 4.4, Table 4.5 and Table 4.6, we found out that results of

Table 4.7 : 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, User-mentions and Hasthag symbols |

each pre-processing techniques varies with different classification algorithm. Green highlighted are the results which outperform baseline in each case, whereas red highlighted are the lowest results. Highest results of each pre-processing techniques can be seen in bold. Each technique beats baselines results in most of the classifiers in all datasets. Results of one technique at a time is graphically presented in Figuure 1 to Figure 3.

Based on overall results, we divided results into two (Best and Worst) categories according to their performance given in Table 4.7. We found out that in the case when we use only one technique at a time, then best-performing techniques are lemmatization and lower-casing all words whereas, removing punctuations and URLs, User-mentions and Hashtag symbols also degrade the performance. So, in a nutshell, when the analysis is required in a short time, then the recommended techniques to achieve good results are lemmatization and lower-casing of all words. However, using a combination of different techniques certainly takes more time but also improves the overall results. In the next section, we present the recommended combination of pre-processing techniques which help to get better representations and improves classification results.

***Recommended combination of pre-processing***

As previously mentioned, classification results vary how we combine different pre-processing techniques. Table 4.8 to Table 4.10 shows the results of our proposed combination of different pre-processing results. We compared the results of our proposed method with results of baselines method where no pre-processing technique is applied, with the results of best performing the individual technique and other tested combinations.

Table 4.8 : Comparison of Proposed Combination on classification task (Waseem et al. Dataset)

| ClassifierTechniques | | 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 |

It is evident from the results the proposed method is beneficial to use to improve the classification results. The reason why the obtained results are better than others is that our proposed pre-processing improves the quality of text by removing noise and helps the learner to get better features. The proposed recommended combination works well because of the order of techniques to apply each pre-processing technique plays a significant role to improve the quality of the text. Changing the order of techniques results in losing information which degrades the classification results. Proposed method structures and normalizes the unstructured and informal nature of the text, which is the primary cause of performance improvement. Figure 4.4 presents the graphical comparison of our proposed method with others on all datasets.

The reason why our proposed method works well because of its step by step order which improves and normalizes poor quality text into a good quality and does

Table 4.9 : Comparison of Proposed Combination on classification task (Golbeck et al. Dataset)

| ClassifierTechniques | | 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 |

not to lose any information. A step by step working mechanism of our proposed method is applied to the toy example presented earlier is given in Table 4.11. We can see clearly from the table that by following the proposed combination in a recommended order improves the quality of the text. For instance; removing Urls, user-mentions and hashtags is useful for humans but does not give extra information for machines. Similarly, for emoticons, abbreviations, spelling mistakes and other language imperfections are easily understandable for humans but in order for computers to understand and get maximum information we replace and normalize these imperfections with their actual meanings and words and reduce dimensionality issues.

Further, the order of applying these techniques is very 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.

Table 4.10 : Comparison of Proposed Combination on classification task (David et al. Dataset)

| ClassifierTechniques | | 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 |

into their actual meanings then we lose information. Similarly, performing word segmentation before replacing slangs and acronyms also lose information. For instance, if we do not first replace abbreviations and slangs into their actual words from the phrases like "urgr8" and "emfi9" etc. then again we end up losing the necessary information. Further, expanding contractions after tokenization step loses information and breaks contractions such as "can't" into "can" and "t" and "don't" into "don" and "t". Similar is the case with other techniques. Another, important point to consider that, not all techniques interact well with other when we change their order which degrades the results. So changing the order of pre-processing techniques lose useful information and/or the pre-processing techniques does not interact well with other techniques which degrade the performance. The experimental analysis confirms that proposed systematic combination beats other combinations and addresses the challenges of improving the quality of the poor quality text. The proposed method improves the quality of the text and helps the learner to learn

Table 4.11 : 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 |

better text representation.

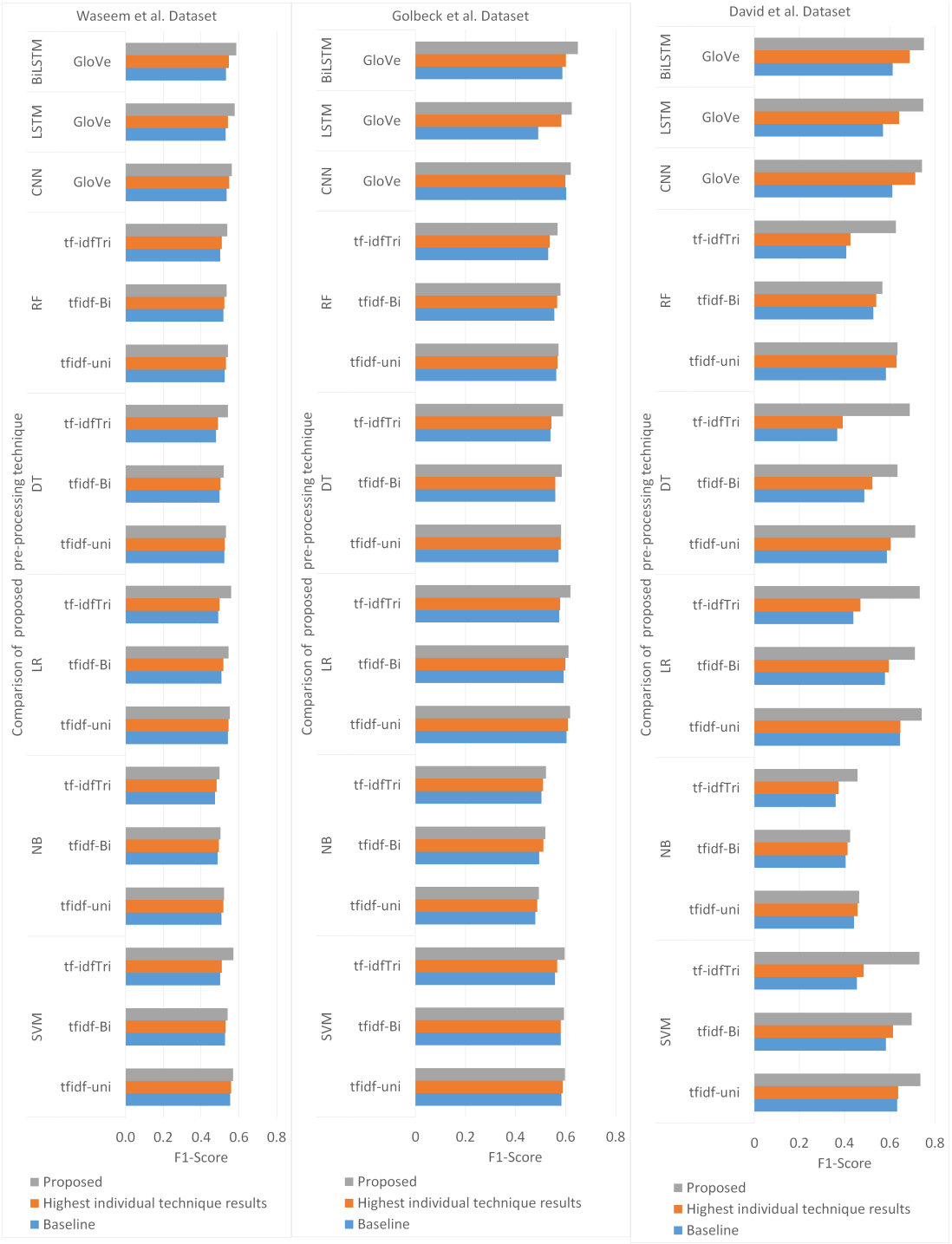


Figure 4.4 : Comparison of Proposed method on classification task

### 4.3.4 Significance of proposed pre-processing combination:

- Recommended combination with the systematic pattern of pre-processing techniques which can be applied to any low-quality text to clean, normalize and improve the quality of data to have better text representation.
- Our Sentiment tokenizer can identify and replace latest emoticons, emojis, abbreviations and acronyms.
- Social media users sometimes intentionally make spelling mistakes which are more comfortable for humans to understand but degrades the classification performance. Our recommended approach also corrects incorrect spellings.
- Removing only # symbol and performing word segmentation of remaining content to get useful information especially in case of sentiments or opinions expressed in hashtag phrases/words and removed all other information which does not help.
- Performing above mentioned steps according to proposed method help to learn better features because the learner will not to assign different vectors for the same word which has spelling mistakes, performing steps in a recommended pattern will not lose information as shown in Table 4.11. For instance, removing numbers, not capturing emoticons or removing hashtag with its content randomly may result in losing information.

## 4.4 Summary

In this chapter, we presented an analysis of different pre-processing techniques and proposed a recommended combination of pre-processing techniques. Pre-processing is the first step in any text classification task and using an appropriate combination of pre-processing techniques can help to improve the classification results.

We studied the effect of 12 different pre-processing techniques in tweets classification and conducted the number of experiments to confirm the effect of different pre-processing on three different Twitter hate-speech and abusive language

datasets. Each pre-processing technique is evaluated with five traditional and three deep learning-based classifiers with different feature extraction models. Further, we presented the worst-performing techniques and recommended techniques which give better results when used individually. Also, results vary when we change the classifier, which confirms the fact of choosing an accurate classifier in the classification task. Then after a series of experiments and different combinations and interactions of different pre-processing techniques, we presented a recommended combination of pre-processing techniques which improves the results out of different tested combinations.

Our proposed method can improve the quality of the text. The proposed method can capture emoticons and acronyms and replace them with their actual meanings. It also replaces the incorrect spelling with correct spellings, expand contractions and remove other noise from the data. By using our proposed method, the unstructured and informal text is transformed into a structured and clean text which helps to learn better features which results in better classification outcomes as compared to other combinations analyzed in this study. Because of proposed combination's good performance, we reused our recommended pre-processing combination for improving the quality of text in our other experiments as well in Chapter 5.

The following steps achieve the recommended combination of pre-processing techniques:

- At first extra information is removed such as Unicode strings, URLs, user-mentions and hashtag symbols. We do not lose useful information by removing these.

- Then in the next two steps, our social tokenizer captures emoticons and acronyms one by one and replaces them with their actual meanings. These steps are important in our proposed method because this help to get extra information which is understandable for machines by transforming emoticons and abbreviations to actual meanings.

- Another important step is correcting spelling mistakes and replacing elongated characters. These steps help the learner not to assign different vectors to the same word due to spelling mistakes.
- Expanding contractions is also one of the important steps because if we do not expand contractions to their root words then during tokenization, we might lose useful information which degrades the classification results.
- Separating remaining content of hashtags is also a crucial step because usually, users express important information in hashtags, so capturing this helps the learner to get features.
- Lastly, removing noise and other steps such as removing punctuations, removing stop-words, lower-casing and lemmatization steps also helps to normalize the text.

Further, the experimental analysis also confirms that our proposed method can effectively improve the quality of text which results in better classification results. The experiment shows that our recommended combination outperforms results obtained from other combinations analysed in this study for tweets classification.

This research opens new opportunities and explores different techniques and methods to improve the quality of the text, which has been overlooked in previous studies. In future, we plan to apply our proposed method on different domains and also explore comprehensively different combinations and interactions of pre-processing techniques which has not be studied in our current analysis.

# Chapter 5

# Deep Intelligent Contextual Embedding for Twitter Sentiment Analysis

## 5.1 Introduction

In this chapter, we present a deep hybrid words representation model which manages language ambiguities and represent this with the embeddings known as deep hybrid words representation (DICE). We applied our hybrid words representation on Twitter sentiment analysis task.

The sentiment analysis of the social media-based short text (eg., Twitter messages) is valuable for many good reasons which have been extensively explored in recent years in different communities such as text analysis, social media analysis and recommendation. Although different methods have been proposed, we are still not able to fully capture the language ambiguities such as words with different meanings in the context (Polysemy), sentiment knowledge of words, semantic and syntactical information. Capturing mentioned language ambiguities is more challenging and vivid in the case of Twitter which allows the limited number of characters to users.

Examples of polysemy and words with opposite polarity are shown in Figure 5.1 where the meaning of words *good* and *bad* and *Like* and *hate* changes depending on its context which traditional word embeddings (explained in Chapter 2) are unable to capture but assign the same representation of a word irrespective of its context and meaning. In addition to polysemy, traditional word embeddings fail to capture sentiment information of words like *good* and *bad* and *Like* and *hate* which results in similar word vector representations having the opposite polarities. Thus, ignoring polysemy within the context and sentiment polarity of words in a tweet reduces the performance of sentiment

analysis. In a nutshell, the purpose of this study is to address the unique characteristics such as i) unstructured and informal nature of text; ii) have same words with different meanings in the context (polysemy); iii) words with different sentiments.

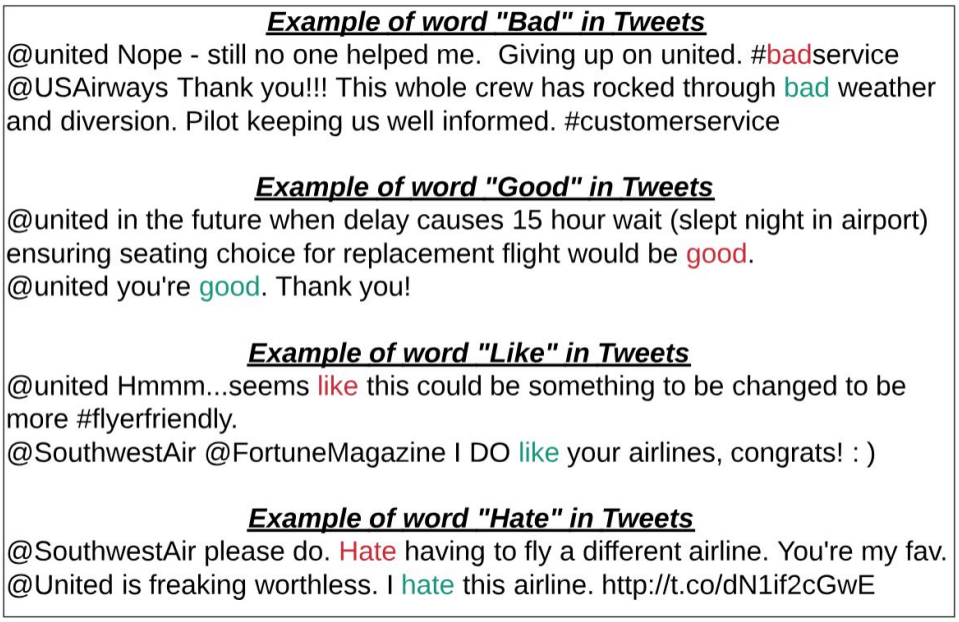


Figure 5.1 : Examples of Words with different meanings and polarities

**Challenges:**

Words representation quality is measured by how it adds syntax information and handle polysemy in a model, which improves semantic word representation. Unfortunately, traditional word representation models do not handle language ambiguities such as polysemy without losing semantic information. Also, do not analyse the sentiment of words which is a significant challenge needed to be addressed, especially for sentiment analysis tasks where the sentiment of words can help to improve the classification results. Another challenge is to select an appropriate sentiment lexicon(s) from a wide variety of sentiment lexicons available in the literature and in last to find out methods which can complement each other and extract required features for Twitter sentiment analysis.

**Objective:**

Proposed in this thesis Deep Intelligent Contextualized Embedding (DICE) addresses our second research objective explained in section 3.1.3, which is to handle natural language ambiguities. DICE is devised to handle the issues of:

- Words with multiple, different meanings (polysemy),
- Out of Vocabulary (OOV) words,
- Semantics and Syntactical information and
- Sentiment knowledge of words.

**Contributions:** Specially the main contributions of this work are as follows:

- Propose a hybrid words representation (DICE) to manage language ambiguities.
- Apply hybrid words representation (DICE) for Twitter sentiment analysis.

## 5.2 Methodology

In this section, we describe our proposed novel hybrid words representation model *Deep Intelligent Contextual Embedding (DICE)* which captures the language ambiguities defined earlier. The complete architecture of our proposed model is given in Figure 5.2. At input using our proposed pre-processing combination (chapter 4), a corpus of the processed tweet is given to our model. Then in the next two layers, we build our hybrid words representation model (DICE) by concatenating four different embeddings which, as a whole, addresses the challenges and objectives highlighted in the previous section. First, we generate contextual embedding $V_{context}$, followed by Word embeddings $V_{GloVe}$ and POS embedding $V_{POS}$ and then, at last, we generate Lexicon embedding $V_{Lexicon}$. After generating these four embeddings, we concatenate them and get hybrid words representation (DICE) $V_{DICE}$, which is

then fed to BiLSTM with attention for tweets sentiments prediction at the output layer. Below we describe each of the main components of our proposed model.

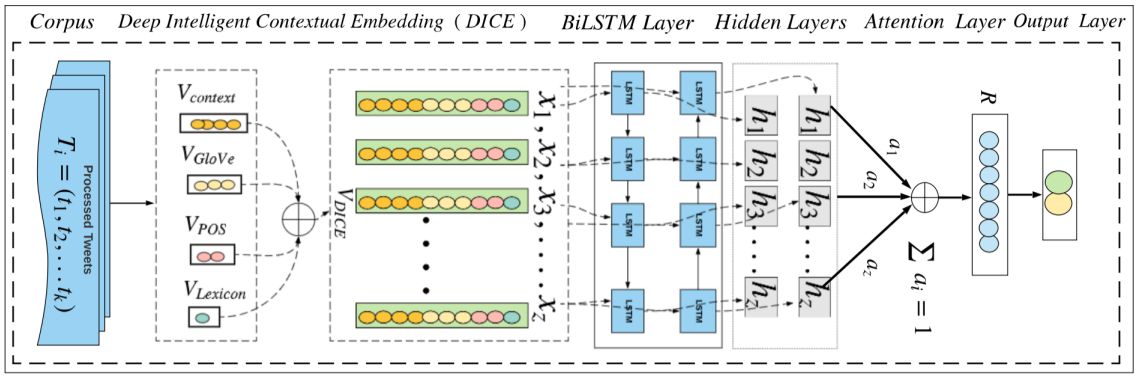


Figure 5.2 : DICE with BiLSTM & Attention
for Twitter Sentiment Analysis

### 5.2.1 Deep Intelligent Contextual Embedding (DICE)

Given a Tweet $T_i$ with a sequence of tokens $(t_1, t_2, t_3, ...t_k)$, where $i$ represents the number of a tweet and $k$ denotes the number of tokens in a tweet. Our hybrid words representation model (DICE) concatenates contextual embedding $V_{context}$ (ELMo)→ deals with polysemy and OOV words, Word embeddings $V_{GloVe}$ (GloVe)→ deals with semantic and sytactical issues, Part of speech $V_{POS}$ (POS) embedding → captures syntactical information more effectively and Lexicon embedding $V_{Lexicon}$ → deals with sentiment knowledge of words in context. Detail of each embedding is given in the following sections.

#### *Contextual Embedding* $V_{context}$

We know that language is complicated. Context can alter the meaning of the words in a sentence. For instance; the words such as *Good* and *Bad* given in Figure 5.1 are same in the different sentence but their meaning changes according to its context (Polysemy). Whilst we, humans, can comfortably

understand such language complexities, but building a model which can decipher the various nuances of the meaning of words given the surrounding text is ambiguous.

Traditional word embeddings fail to handle such language ambiguities. They only generate one vector representation per word; for that reason, these traditional word embeddings models cannot handle and capture how the meaning of each word can alter based on its context. This made it vital to build a model which captures the word meaning in changing contexts (Polysemy) and at the same time keep the contextual information. Peters et al. (2018b) proposed contextualized word embeddings, are embeddings proposed from language model (ELMo) which goes beyond traditional word representation models. Contextualized words embeddings can capture such language complexities and generates different word vectors according to its context.

Instead of word vectors based on dictionary, ELMo analyses words based on its context, which makes the working mechanism of ELMo different than traditional word embeddings. ELMo builds word vectors on the fly by passing text through a deep neural network rather than having a look-up table of words and their corresponding vectors. Further, ELMo words embeddings are character-based, which helps to learn morphological-based embeddings which help to solve the issue of out-of-vocabulary (OOV) words unseen during training. Figure 5.3 shows the working mechanism and architecture of ELMo where it uses CNN based character level words representation of raw word vectors which are fed to the first layer of biLM (layer 1) which forms the intermediate words representations. These intermediate word representations act as an input to the second layer of biLM (layer 2) and form intermediate word representations. Finally, the ELMo representations are created, which is a weighted sum of words representations (intermediate word representations) from both biLM and raw word vectors. The working mechanism of each biLM is given below.

As previously mentioned that ELMo embeddings are based on the representa-

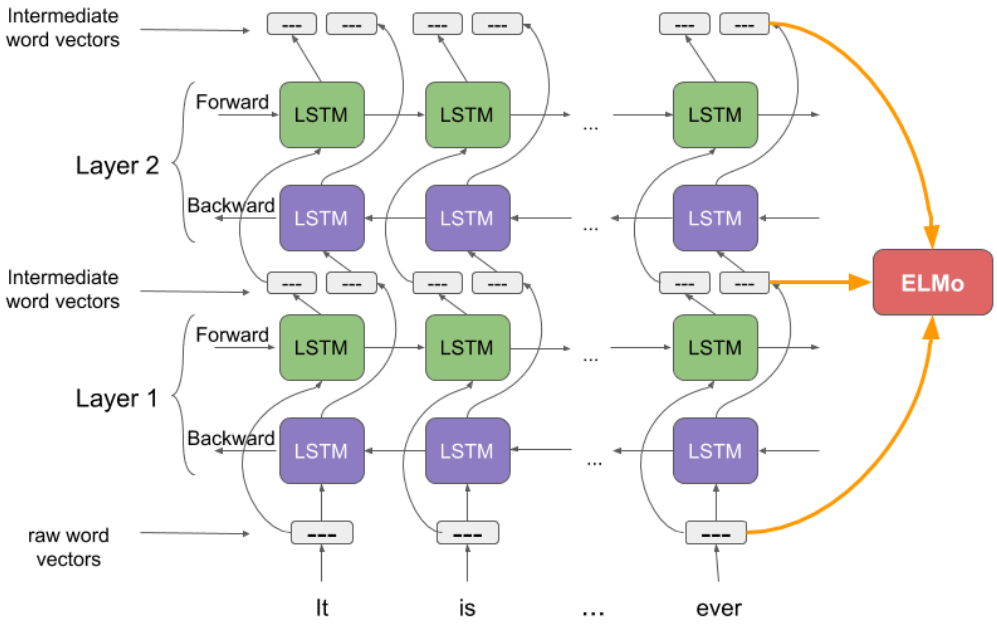


Figure 5.3 : A working mechanism and architecture of ELMo

This image is taken from Vidhya analytics blog*

tion learned from Bi-language model (biLM). A bi-LM is a concatenation of forward and backward language models where forward language model passes an input representation for each token through L layers of forward LSTM to learn representation. Being a hidden representation of RNNs these learned representations are context dependent. For a given sequence of $k$ tokens a prediction of token in a forward LM is computed by formula given below and shown in Figure 5.4:

$$P(t_1, t_2, ....., t_k) = \prod_{n=1}^{k} p(t_n|t_1, t_2, ....t_{n-1})$$

whereas, Backward LM is same as forward LM but it processes a sequence in opposite direction and is computed by formula given below:

*Source: `https:www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-\features-from-text`

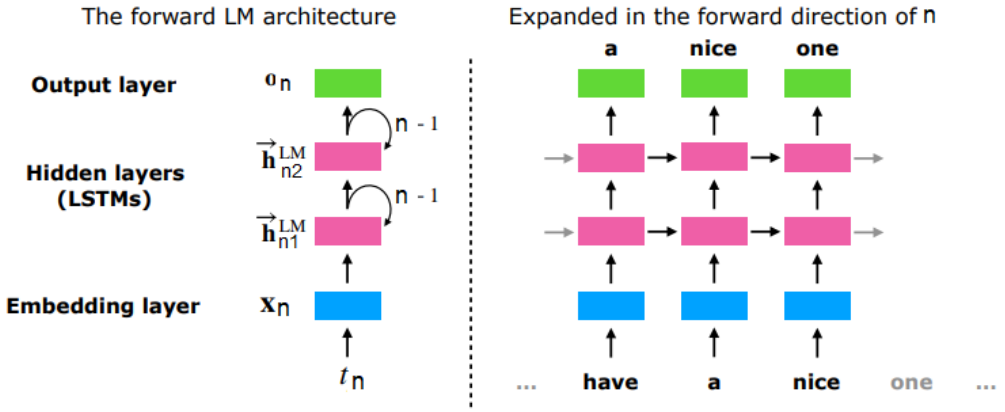


Figure 5.4 : Forward LM architecture

This image is taken from shuntaro Yada Slides[†]

$$P(t_1, t_2, ....., t_n) = \prod_{n=1}^{k} p(t_n | t_{n+1}, t_{n+2}, ....t_k)$$

Log-likelihood of sentences in both forward and backward language models is involved in training process of BiLMs and final vector is computed after the concatenation of hidden representations from forward language model $\overrightarrow{h}^{LM}_{n,j}$ and backward language model $\overleftarrow{h}^{LM}_{n,j}$, where $j = 1, ...., L$. BiLM is computed by formula given below and shown in Figure 5.5.

$$biLM = \sum_{n=1}^{k} (\log p(t_n | t_1, ....., t_{n-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s)$$

$$+ \log p(t_n | t_{n+1}, ...., t_n; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)$$

Where $\Theta_x$ and $\Theta_s$ are the token representation parameters and softmax parameters, respectively, which are shared between forward and backward directions. And $\overrightarrow{\Theta}_{LSTM}$ and $\overleftarrow{\Theta}_{LSTM}$ are then forwarded back backward LSTM parameters respectively. ELMo abstracts the representations learned from an

[†]Source: `https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized\-word-representations-peters-2018`

intermediate layer from BiLM and execute a linear combination for each token in a downstream task. BiLM contains *2L+1* set representations as given below.

$$R_n = (X_x^{LM}, \overrightarrow{h}_{n,j}^{LM}, \overleftarrow{h}_{n,j}^{LM} | \quad j = 1, ...., L)$$
$$= (h_{n,j}^{LM} | \quad j = 0, ..., L)$$

where $h_{n,0}^{LM} = x_n^{LM}$ is the layer of token and $h_{n,j}^{LM} = [\overrightarrow{h}_{n,j}^{LM}, \overleftarrow{h}_{n,j}^{LM}]$ for each bi directional LSTM layer.

ELMo is a task specific combination of these features where all layers in $M$ are flattened to single vector and is is computed by formula given below.

$$ELMo_n^{task} = E(M_n; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{h,j}^{LM}$$

Where $s^{task}$ are weights which are softmax normalized for the combination of different layers representations and $\gamma^{task}$ is a hyperparameter for optimization and scaling of ELMo representation. Our architecture is based on pre-trained ELMo embeddings with 1,024 dimensions obtained using the 1 Billion Word Benchmark, which contains about 800M tokens of news crawl data from WMT 2011 Chelba et al. (2013). ELMo gives us context Vector, $\mathbf{V}_{\boldsymbol{context}}$ of 1024 dimensions, which addresses the issues of polysemy and OOV words. Word representations learnt from ELMo can easily be integrated into current models and significantly enhances the performance in many NLP related tasks.

### ***GloVe Embedding*** $V_{GloVe}$

Pennington et al. (2014) presented Global Vectors (GloVe), which is an unsupervised learning model for obtaining word vector representations by aggregating global word-word co-occurrence statistics which counts how frequently a word appears in a context which makes it a count-based model. GloVe uses ratios of co-occurrence probabilities whereas Word2Vec (CBOW and Skip-gram) is a prediction based model which considers only local context and does not

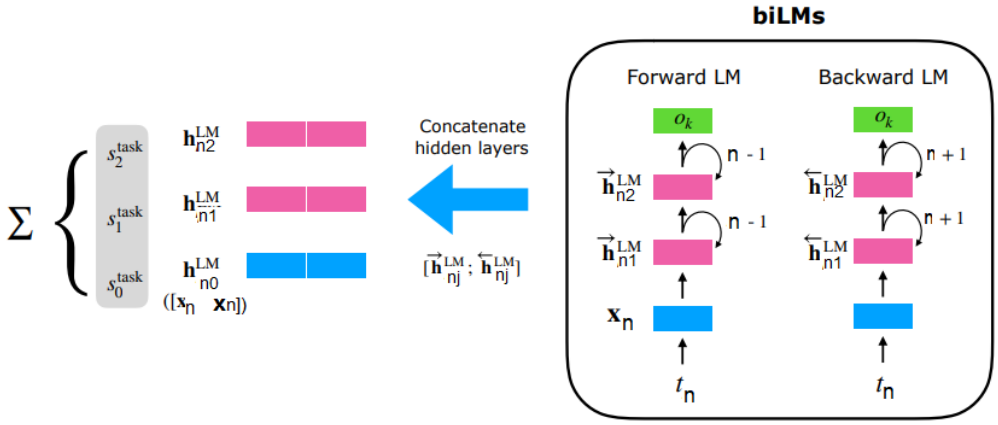


Figure 5.5 : Working mechnaism of BiLM

This image is taken from shuntaro Yada Slides[‡]

take benefit of the global context. This shortcoming of Word2Vec may not learn well substantial scale repetition and patterns as compared to GloVe.

Global Vectors (GloVe) use matrix factorization approaches to breakdown the gigantic co-occurrence intro more dense word vectors with low dimension representation. Pennington et al. (2014) studied the relationship between words *ice* and *steam* by taking the ratio of co-occurrence probabilities with different probe words. Let the probability of x words shows in the context of word y be P(x/y); ice appears more often with solid as compared to gas. Similarly steam appears more often with gas than solid. Both of these words co-occur more often with word *water* due to the same characteristics and less frequently with irrelevant word $fashion$. This means that the probability of P(solid/ice) will be quite higher as compared to the probability of P(solid/steam). Thus, the ratio of probability of P(solid/ice) to probability of P(solid/steam) will be significant as compared to the word *gas* which is associated with word *steam* but not with *ice* then the the ratio of the probability of P(gas/ice)/P(gas/steam). Further, the ratio will be near to 1 for relevant words like *ice* and *steam*. Moreover, word representations generated from GloVe are faster to train, scalable,

[‡]Source: `https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized\-word-representations-peters-2018`

give good results, even the small dataset and training can be stopped earlier if no significant improvement is observed. The objective function is given below.

$$J = \sum_{k,j=1}^{V} f(X_k j)(w_k^T w_j^{'} + b_k + b_j - \log(X_k j))$$

where;

$V$ : is size of vocabulary,

$X$ : is co-occurrence matrix,

$X_k j$ is frequency of word $k$ co-occurring with word $j$,

$X_k$ total number of occurrences of word $k$ in the corpus,

$P_k j$ is the probability of word $j$ occurring within the context of word $k$,

$w$ is a word embedding of dimension d,

$w^{'}$ is the the context word embedding of dimension $d$

As recommended by Peters et al. (2018b); it is favourable to concatenate ELMo embeddings with traditional word embeddings such as Word2Vec and GloVe. In our model, we used pre-trained GloVe embedding of 300 dimensions which are trained on 840 billion tokens from common crawl because it gives better results as compared to Word2Vec in our case. GloVe outputs a vector, $\mathbf{V}_{GloVe}$ of 300 dimensions, which has word semantics and syntactical information of tweets context.

***Part of Speech (POS) Embedding*** $V_{POS}$

The state-of-the-art word embeddings models can capture distributional semantic meanings in different NLP related tasks. These models fail to incorporate and capture more complex semantic representations because of the linguistic ambiguities of some words can have various word-senses based on their context. Grammatical language characteristics are defined to address these

language ambiguities. Part of Speech (POS) helps to capture the syntactic characteristics of individual words.

POS tagging is an important step in which each word in the context is assigned with the appropriate POS tag. Using POS has shown good results in NLP related tasks (Rezaeinia et al.; 2017a). POS gives us useful information about a word, neighbours and different syntactic types of words such as verbs, nouns, adverbs and adjectives etc. The main purpose of POS embedding is to detect Nouns /NN, Verbs /VB, Adjectives /JJ, Adverbs /RB and generate their representations. Figure 5.6 shows an example of generating POS word vectors. In our proposed model we have used Stanford parser for POS tagging which generates POS tags. Each POS tagged token is then transformed to a vector, $\mathbf{V}_{POS}$ of 50 dimensions.

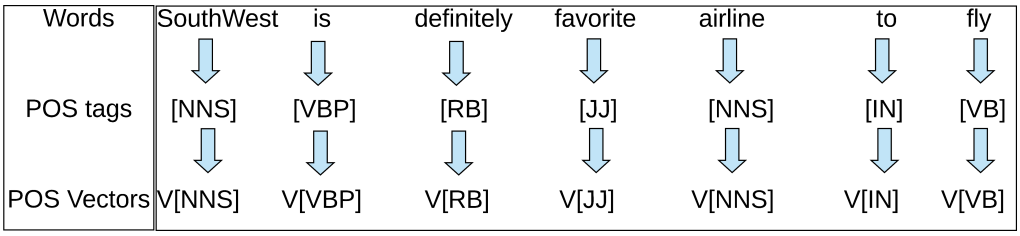


Figure 5.6 : An example of POS Embedding $\mathrm{V}_{POS}$

***Lexicon Embedding*** $V_{Lexicon}$

Words in language have sentiments associated with them which humans can understand, but Conventional word representations models fail to capture sentiment knowledge of words. In order to capture this linguistic ambiguity, we used lexicons to extract sentiments of words and generate vectors.

The lexicon embedding is based on the extraction of sentiment scores from lexicon dictionary, which is a list of words, specific terms and phrases. Using these lexicons can be useful in analyzing the text for sentiment analysis. Each lexicon contains a pair of word-sentiment where each word has its sentiment score, which is between $-1$ to 1, where value less than 0 represent negative

words and positive for values above 0. There are many sentiment and emotion lexicons resources are available, so it is crucial to select the right one or appropriate combination of lexicons. We selected a combination of six different lexicons for extracting sentiments in our lexicon embedding. If any token is not available in any of these lexicons, then we assigned a score of zero to that token. Our lexicon embedding outputs a vector, $\mathbf{V}_{Lexicon}$ of 6 dimensions. We have used the following six Lexicons in our model. Table 5.1 shows the summary and characteristics of sentiment lexicons used in our model.

1. **SenticNet 5.0** (Cambria et al.; 2018):

   SenticNet is a concept based sentiment lexicon. SenticNet has different versions available, and the latest version is senticNet 5.0, which gives sentiment and semantic information of 100,000 concepts, and it is commonly used for sentiment analysis. The parser of this lexicon gives the sentiment score and the sentic vector of each concept found in the given text. The sentiment value is a real number, whereas the sentic vector contains scores related to emotions.

2. **VADER** (Hutto and Gilbert; 2014):

   Valence Aware Dictionary and Sentiment Reasoner (VADER) sentiment lexicon were mainly created for sentiment classification of social media text which is sensitive to both sentiments, positive and negative sentiments, and its intensity. It uses the human-centric method, combines qualitative analysis and empirical validation by human raters and knowledge of the crowd. VADER works well on social media text, and it does not need any training data. It is fast and does not suffer from speed-performance tradeoff.

3. **Bing Liu Opinion Lexicon** (Hu and Liu; 2004):

   Bing Liu's sentiment lexicon is presented and maintained by Bing Liu in 2012 has been widely used. This sentiment lexicon consists of 2006

positive words and 4,683 negatives. This sentiment lexicon contains slang words, misspelled words and morphological variants aswell.

4. **SemEval Twitter English Lexicon** (Mohammad et al.; 2013):

   SemEval Twitter English Lexicon was presented in semEval-2015 task-10 (subtask E) on Twitter sentiment analysis. The sentiment scores in this lexicon are from range 0 to 1. This lexicon contains a list of about 1500 single words and two-word negations expressions and their relation with negative and positive polarities. The terms used in this lexicon are taken from Twitter (English), and it also included general English words, hashtags, misspellings and other different categories which are commonly used in Twitter.

5. **NRC Sentiment140 Lexicon** (Kiritchenko, Zhu, Cherry and Mohammad; 2014):

   NRC-Canada presented NRC sentiment140 lexicon in which the polarity value of each word is computed, and pointwise mutual information (PMI) measure among words and their sentiment tag in the tweet was used to find the bigrams. A huge corpus of 1.6 million tweets with negative and positive emotions instead of using hashtags to compute the sentiment of words.

6. **Large-Scale Twitter-Specific Sentiment Lexicon (TS-LEX)** (Tang, Wei, Qin, Zhou and Liu; 2014):

   This sentiment lexicon was built by learning representation learning approach. Sentiment information of text was integrated into a neural network with its loss function to learn sentiment specific phrase embedding. Existing phrase embedding model was tailored, and the network is trained from the huge corpus of positive and negative emoticons without annotation. Unlike other lexicons, this lexicon outputs a word representation that represents a semantic, syntactical and sentimental relationship within the words. An urban dictionary was used to expand the

list of sentiment seeds of common words, which were used to train the classifier, which gives the probability of word to be positive or negative.

Table 5.1 : Summary and characteristics of Sentiment Lexicons used

| **Lexicon Name** | **Construction based** | **Corpus Used** | **Statsitics** |
|---|---|---|---|
| SenticNet 5.0 | Dictionary | WordNet | 100K concepts in four categories |
| VADER | Dictionary | SentiWordNet | 75000 lexical features with scores |
| Bing Liu | Dictionary & Manually | WordNet | Positive words = 2006<br>Negative words = 4783 |
| SemEval Twitter English Lexicon | Corpus | Tweets | 1500 words |
| NRC Sentiment140 Lexicon | Corpus | Tweets | 1.6M Positive & Negative emoticons |
| TS-LEX | Corpus & Dictionay | WordNet Affect | Positive words with values =1,78,781<br>Negative words with Values = 1,68,845 |

After the creation four-vectors individually, we concatenated all of them to get one vector $\mathbf{V}_{DICE}$, which is clean and contains polysemy, word semantics, syntax and sentiment knowledge of words in a tweet and given by the following equation.

$$\mathbf{V}_{DICE} = \mathbf{V}_{context} \oplus \mathbf{V}_{GloVe} \oplus \mathbf{V}_{POS} \oplus \mathbf{V}_{Lexicon}$$

where element-wise symbol $\oplus$ denotes vectors concatenation. The concatenation of four embeddings is shown graphically in Figure 5.7.

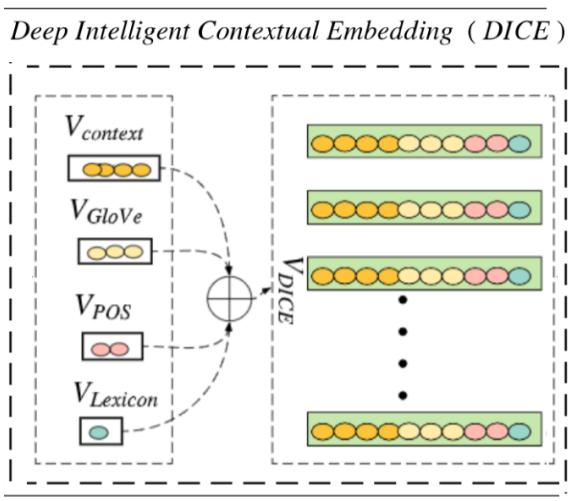


Figure 5.7 : An illustraction of DICE

### 5.2.2 BiLSTM Layer

Schuster and Paliwal (1997) presented Long Short Term Memory (LSTM) network, which is an extension of Recurrent neural network (RNN). It was to solve the problem of vanishing gradient encountered by RNN. In LSTMs, the idea was to propose an adaptive mechanism of gating which finalizes the amount to keep the earlier state and remember the obtained features of input data. Forget gate, input gate and output gate are the three gates of a typical LSTM cell. The flow of information at the current time step is determined by these three gates. LSTM processes a sequence $S = (x_1, x_2, ..., x_Z)$ where $z$ is the length of input text, word by word in a sequence. Below is the mathematical representation of a LSTM cell.

$$f_i = \sigma(W_f[x_i, h_{i-1}] + b_f)$$

$$I_i = \sigma(W_z[x_i, h_{i-1}] + b_z)$$

$$C_i = tanh(W_C[x_i, h_{i-1}] + b_C)$$

$$C_i = f_i * C_{i-1} + z_i * C_i$$

$$o_i = \sigma(W_o[x_i, h_{i-1}] + b_o)$$

$$h_i = o_i * tanh(C_i)$$

In our work, we have placed BiLSTM layer, proposed by Schuster and Paliwal (1997), on top of our DICE with attention layer for sentiment analysis to capture the information from both directions. A BiLSTM takes an input of a vector $\mathbf{V}_{DICE}$ with a sequence of $x_z$ tokens and produces hidden representation $h_i$ at a given time $i$ by concatenating the hidden representations from both forward $\overrightarrow{h_i}$ and backward $\overleftarrow{h_i}$ LSTM and is given by following equation.

$$h_i = [\overrightarrow{h_i} \parallel \overleftarrow{h_i}]$$

where $\parallel$, denotes the concatenation of outputs from both forward and backward LSTM.

### 5.2.3 Attention layer

Not all words contribute equally to understanding the meaning of the sentence. We used attention mechanism presented by Diyi et al. (2016) to enforce the contribution of essential words. Attention assigns a weight $a_i$ to each token through a softmax function, and finally, representation **R** which is a weighted sum of all tokens is calculated and given by the following equation.

$$R = \sum_{i=1}^{z} a_i h_i,$$

where,

$$a_i = \frac{exp(e_i)}{\sum_{t=1}^{z} exp(e_t)}, \quad \sum_{i=1}^{z} a_i = 1$$

$$e_i = tanh(W_h h_i + b_h)$$

where $W_h$ and $b_h$ are learned parameters, $h_i$ is the concatenation of the representations of the forward and backward LSTM.

### 5.2.4 Output Layer

We used representation **R** generated from an attention layer and fed to fully connected softmax layer to get the class probability distribution. We minimized binary Cross-entropy loss function **L** in which loss increases as the predicted probability **p** diverges from the actual label **y**, is given by following equation.

$$L = -(y \log(p) + (1 - y) \log(1 - p))$$

## 5.3 Experimental Analysis

In this section, first, we present experimental settings, datasets used in this study and then the results of the proposed model by comparing with some famous and state-of-the-art words representation models for Twitter sentiment analysis.

### 5.3.1 Experimental settings

In this section, pre-processing methods, evaluation metric and parameters used in our experiments are presented.

#### *Preprocessing*

We re-used our proposed recommended combination of pre-processing techniques to improve the quality of text explained in Chapter 4. As mentioned earlier, our pre-processing method helps the learner to learn better features which ultimately improves the classification results.

#### *Evaluation Metric*

To keep our experiments consistent and comparable with previous proposed methodology (Chaper 4), we used the F1-score evaluation metric, same as we used previously in Chapter 4.

#### *Parameters*

Parameters used in our experiments are presented in this section.

- **Regularization**: In order to avoid the overfitting problem, we have used Gaussian noise at our input layer. Also, a dropout (Srivastava et al.; 2014) at connections of the network is also used to turn off the neurons in the network randomly. Moreover, to avoid overfitting and make our method robust, we applied $L_2$ regularization technique to decrease the large weights.

- **Training**: Binary cross-entropy loss function was used to train our model using the rectified linear unit (ReLu) by back-propagation method with the batch size of 128. To tune the learning rate, we used Adam (Kingma and Ba; 2014) optimizer. 10-fold cross-validation technique was used to evaluate the classification results also, 80% of data used for training and remaining 20% for testing purpose.

Table 5.2 : Summary of all parameters

| Name | Details |
|---|---|
| Optimizer | Adam |
| Learning Rate | 0.001 |
| Back-Propagation | ReLu |
| batch Size | 128 |
| Dropout | 0.25 |
| $L_2$ Regularization | 0.0001 |
| Hidden Layer Dimension | 150 each |
| Gausian Noise | $\sigma = 0.3$ |

– **Hyper-parameters**: The Grid search optimization technique was used to find the hyper-parameters used in our experiments. Parameters used are given in Table 5.2.

### 5.3.2 Datasets

We have used three airlines-related Twitter datasets, two of them crawled and labelled by us, and one is publicly available. We chose airline-related datasets because limited work has been done on airlines sentiment analysis using deep learning methods. We call our three datasets as Dataset 1, Dataset 2 and Dataset 3 and the distribution of the tweets are given in Table 5.3.

– **Dataset 1**: This dataset is publicly available and taken from the Kaggle Datasets originally released by CrowdFlower. The total number of tweets given in the dataset is 14,640. Since we are dealing with binary classification problem so we are considering only positive and negative so after filtering the total number of tweets is 11,541. The tweets are related to six major US Airlines: American airline, United airline, US Airways, Southwest airline, Delta airline, and Virgin airline.

– **Dataset 2**: The data was collected by using Tweepy, an official python Twitter API library. The dataset contains only two months worth of tweets starting from December 2015 until January 2016. We followed guidelines by Mohammad et al. in Mohammad (2016) for annotations of tweets and instructed annotators to label tweets as positive and negative. First, a set of 200 tweets were given, four different annotators annotated the tweets collectively so that all of them can have a general understanding and agreement on the standard for annotation. We used Cohens Kappa ($\kappa$) for calculating inter-annotator agreement (IAA) between annotators. In the second phase, the same set of random 1000 tweets were given to each annotator for annotation. The disagreement was observed and resolved. In the third phase, again another same set of 500 tweets were given to all annotators, and this time we achieved good IAA score, and minor disagreement was observed. Finally, in our last phase, a large set of remaining tweets were equally divided among all annotators for annotation of tweets. The total number of tweets in dataset 2 is 16,454. The tweets are related to three airlines in dataset 2, which are Cathay Pacific, United airline and Singapore airline.

– **Dataset 3**: Dataset 3 contains tweets related to Emirates airline. We used the same method to collect and annotate our third dataset, as stated above. The total number of tweets given in our dataset 3 is 22,172.

Table 5.3 : Tweets distribution in all datasets

| **Dataset Name** | **Positive** | **Negative** | **Total** |
|---|---|---|---|
| Dataset 1 | 2,363 | 9,178 | 11,541 |
| Dataset 2 | 11,670 | 4,784 | 16,454 |
| Dataset 3 | 17,860 | 4,312 | 22,172 |

### 5.3.3 Results and Discussion

This section presents the baselines models used in comparison to our proposed model and discusses the results.

#### *Baselines*

As a baseline, we selected models from classic legacy based word representation, continuous words representation models and finally, hybrid and sentiment specific words representation models which have been used for Twitter sentiment classification task earlier.

- Weighted word representation model: We compared the performance of our model with the method proposed by Go et al. (2009) and da Silva et al. (2014) where they used weighted word representations TF-IDF with different traditional machine learning-based classifiers for Twitter sentiment analysis.

- Continous word representation model: we compared our model with
  (i) Deep convolutional neural network[§] (DCNN) where they used GloVe for word representations (Jianqiang and Xiaolin; 2018).
  (ii) CharSCNN/SCNN[¶] (dos Santos and de C. Gatti; 2014) utilized character embedding (CharSCNN) and Word2Vec (SCNN) for initializing embedding where resulting embeddings are fed to deep neural networks.

- Hybrid and sentiment specific word models: We compared our model with some recently proposed sentiment embeddings such as
  (i) Hybrid ranking (Tang et al.; 2016b) which incorporates sentiment and context of tweets for Twitter sentiment analysis (HyRank[‖]).
  (ii) Refined embeddings Re(*) (Liang-Chih, Lai and Zhang; 2018) where researchers refined the traditional word embeddings by using intensity

[§]https://nlp.stanford.edu/projects/glove/

[¶]https://code.google.com/archive/p/word2vec/

[‖]http://ir.hit.edu.cn/dytang/

score from lexicon.

(iii) Finally, we have compared our model with recently proposed improved word vectors (IWV) (Rezaeinia et al.; 2017a) where traditional pre-trained word embeddings were enhanced by adding POS and sentiment information from lexicons for sentiment analysis.

We selected those methods because they are the state-of-the-art ones and based on the conducted meta-analysis, they exhibit the highest accuracy among the techniques developed so far.

Table 5.4 : Comparison of Proposed Words Representation on a Classification task

| **Model** | **Dataset 1** | **Dataset 2** | **Dataset 3** |
|---|---|---|---|
| TF-IDF word representation with SVM | 0.790 | 0.777 | 0.802 |
| TF-IDF word representation with NB | 0.829 | 0.832 | 0.821 |
| TF-IDF word representation with DT | 0.860 | 0.851 | 0.872 |
| TF-IDF word representation with RF | 0.869 | 0.871 | 0.897 |
| TF-IDF word representation with Ensemble | 0.814 | 0.830 | 0.825 |
| DCNN (GloVe word representation) | 0.832 | 0.841 | 0.848 |
| CharSCNN (Character word representation) Pre-trained | 0.862 | 0.858 | 0.870 |
| SCNN (Word2Vec word representation) Pre-trained | 0.831 | 0.840 | 0.857 |
| CharSCNN (Character word representation) Random | 0.809 | 0.819 | 0.835 |
| SCNN (Word2Vec word representation) Random | 0.817 | 0.829 | 0.842 |
| HyRank word representation model | 0.841 | 0.842 | 0.863 |
| Re(Word2Vec word representation) model | 0.851 | 0.849 | 0.868 |
| Re(GloVe word representation) model | 0.859 | 0.857 | 0.871 |
| IWV (Improved Word Vectors) model | 0.880 | 0.872 | 0.885 |
| DICE | **0.915** | **0.905** | **0.919** |

***Proposed Model***

Results of our model are given in Table 5.4. As we can see the performance of our model is better than existing methods for sentiment analysis when testing them on three, airline-related Twitter datasets with standard pre-processing. Our proposed

model, (DICE) improved the performance ratio ($\Delta$) by 3.98%, 3.78% and 3.84% when compared to previous best results of embedding based IWV method on dataset 1, dataset 2 and dataset 3 respectively. As our model offers a consistent improvement over all other methods so we can conclude that it is a robust solution for sentiment analysis problem.

The reasons why our model achieved better results as compared to others are as follows i) we improve the quality of text by removing noise, learning sentiment aware tokenization and correcting spelling mistakes etc., which helps to learn better representation, and ii) it handles the language ambiguities by capturing deeper relationships within the text. Unlike Word2Vec, GloVe and Fasttext which can not handle words with different meanings in the context (polysemy) DICE can capture it. Along with polysemy, DICE can also handle the OOV issues and have sentiment knowledge of words which other hybrid models like IWV, Refined embedding and HyRank fails to capture. Specifically, our proposed model learns high-quality representations by adding polysemy, sentiment knowledge of words, handles OOV words issues, semantics and syntactical information of words which helps to get better classification results. We also applied our recommended pre-processing combinations on hybrid and sentiment specific models.

Table 5.5 : Comparison with recommended pre-processing on classification task

| Model\Datasets | Standard Pre-processing | | | Recommended Pre-processing | | |
|---|---|---|---|---|---|---|
| | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 1 | Dataset 2 | Dataset 3 |
| HyRank (Hybrid word representation model) | 0.841 | 0.842 | 0.863 | 0.859 | 0.865 | 0.882 |
| Re(Word2Vec word representation) model | 0.851 | 0.849 | 0.868 | 0.868 | 0.871 | 0.886 |
| Re(GloVe word representation) model | 0.859 | 0.857 | 0.871 | 0.875 | 0.878 | 0.887 |
| IWV (Improved Word Vectors) model | 0.88 | 0.872 | 0.885 | 0.895 | 0.894 | 0.902 |
| DICE | 0.915 | 0.905 | 0.915 | 0.932 | 0.926 | 0.934 |

Table 5.5 shows that performance improves when we use our proposed pre-processing (Chapter 4) as compared to standard pre-processing. It can be seen that using our proposed recommended preprocessing combination improved the accuracy ratio ($\Delta$) for Hybrid, Re(Word2Vec), Re(GloVe), IWV and DICE by 2.14%, 1.99%, 1.86%,

1.70% and 1.85% on dataset 1, 2.73%, 2.59%, 2.45%, 2.52% and 2.32% for dataset 2 and 2.20%, 2.07%, 1.83%, 1.92% and 2.07% for dataset 3 respectively. This increase in performance depicts the robustness of our proposed pre-processing combination.

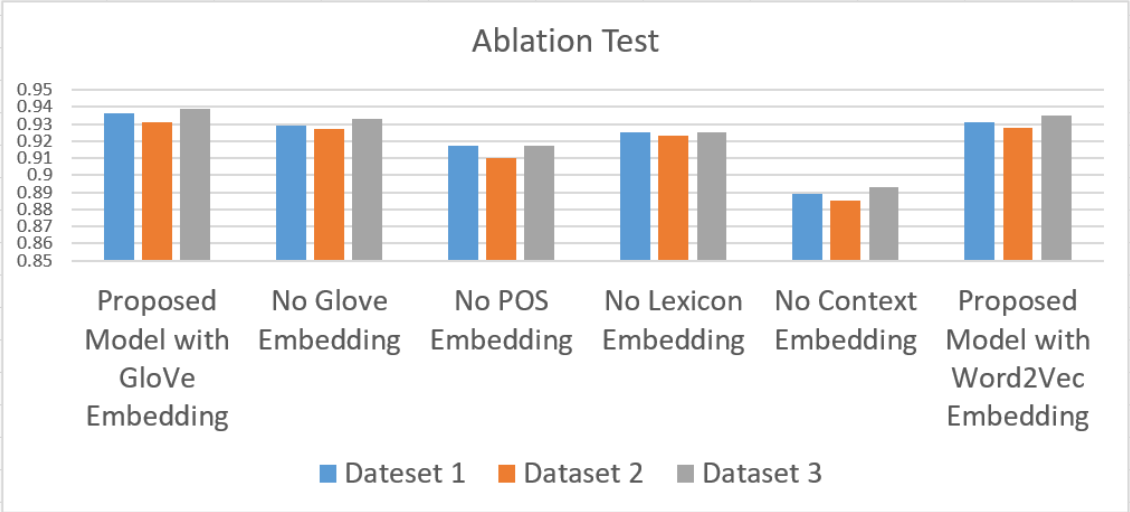


Figure 5.8 : Ablation analysis of proposed model

We extended our hybrid word representation model (DICE) and incorporated character embeddings to efficiently handle OOV words. The character-level features can exploit prefix and suffix information about words to have closer representations among words of the same category. This is useful for terms that may be OOV and can capture mitigating issues like unseen words. We have utilized the character level representations proposed by Lample et al. Lample et al. (2016) using Bi-LSTMs to produce a character-enhanced embedding for each unique word. Our parameters for the forward and backward LSTMs are 25 each and the maximum character length is 25, which results in an 50-dimensional embedding vector,$\mathbf{V}_{Char}$. Experimental anlaysis proofs that incorporating character embedding with our recommended pre-processing improves the performance by 0.65%, 0.85% and 0.63% on dataset 1, dataset 2 and dataset 3 respectively as compared to our previous deep intelligent contextual embedding (DICE) model.

We also performed the ablation analysis of our proposed model. Figure 5.8 tells that all embedding layers in our proposed model add to the overall performance. The

performance drops slightly for all datasets in both cases when we replace GloVe embedding with Word2Vec embedding or remove GloVe embedding from our model. Further, the experimental analysis also indicates that performance drops in both cases when we remove POS and lexicon embeddings from our model and noticeable drop is observed when we remove context embeddings from our model which shows the importance of contextual embedding and strength of using embedding from language model. Hence, we can conclude that one of the strengths of our model lays in the combination of different components that build the diversity, which contributes to increased accuracy of sentiment classification.

Figure 5.9 presents the visualization of common words in our datasets. The word cloud of most common words in a) Positive, b) Negative, and c) All Tweets.

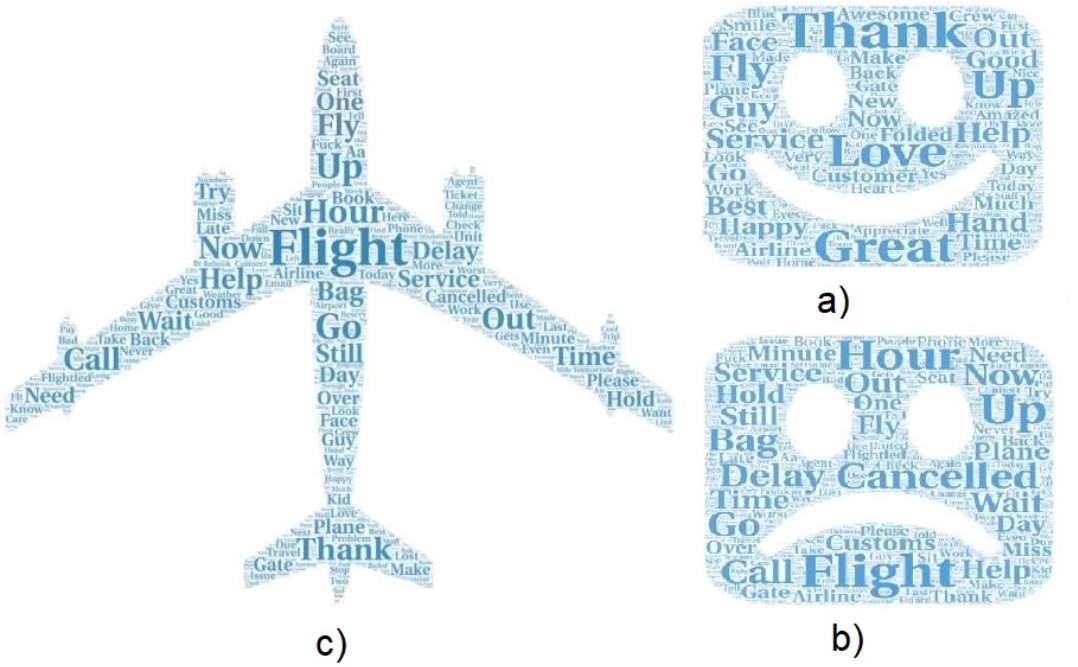


Figure 5.9 : Word Cloud of a) Positive, b) Negative and c) All Tweets

### 5.3.4 Significance of proposed model

- DICE can effectively capture complex attributes of language ambiguities.
- Hybrid word representation model (DICE) captures and compliments

data characteristics and extract useful features instead of using one-word representation model.

- Unique combination of sentiment lexicons are used which assigns its relevant sentiment to words which helps to improve classification results.
- Features extracted by the proposed hybrid word representation model is fed to BiLSTM with attention which focuses more on useful words for sentiment classification and handles data in the sequence which improves results as compared to other classifiers.

## 5.4 Summary

In this chapter, we present hybrid words representation (DICE) which handles complicated attributes of words and its usage within the noisy tweet context. DICE can handle language ambiguities like polysemy, semantics, syntax and sentiment knowledge of words by learning words representations from four different embeddings. Hybrid words representation layer is then fed to bi-directional long short term memory (BiLSTM) with attention to predict the classes.

Hybrid words representation (DICE) is achieved by the following steps;

- Words with different meanings in the context (Polysemy) and out of vocabulary (OOV) words are capture by the embeddings from language model (ELMo).
- GloVe is used to capture semantics and syntactical information.
- Sentiment knowledge of words is incorporated into words by using a combination of six different lexicons.
- Part of speech (POS) embedding helps to get better syntactical information.
- All of above embeddings are concatenated to get hybrid words representations. Words representations which handle language complexities defined earlier.

- BiLSTM is employed to capture sequential information, and attention mechanism is adopted to give importance to important words.

Further, the experimental analysis also confirms that our model can effectively manage the language complexities and can be utilised for input representation for Twitter sentiment analysis related tasks where the data is of low quality and has language ambiguities. The experiment shows that our model outperforms different baselines based on traditional word embeddings and sentiment embeddings for sentiment analysis.

This research opens new opportunities to handle the language complexities and improves the quality of words representation. In future, we can explore more different ways to learn more implicit and explicit relationships within the content to capture more complex characteristics of the data, which can improve the performance. It will be interesting to learn the representation of the words in a hierarchal way to capture more information and try different sources at the input, which can improve the quality of words representation.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusion

This thesis proposed and presented a method to improve the quality of the text, which has been overlooked in previous studies. Also, it contributes to the current research by proposing a hybrid words representation model (DICE) which can manage the natural language complexities in text for the text classification task.

A detailed introduction on the topic of this research is discussed in Chapter 1. Also, research questions, problems, objectives are outlined in Section 3.1. Further, the contributions of this research are mentioned in Section 1.4 of this thesis. Then we discussed and critically evaluated different pre-processing and words representation methods used in natural language processing and presented their pros and cons in Table 2.1. We also reviewed previous studies conducted on the effects of using different pre-processing techniques and word representation models on different NLP related tasks. Word representation models (See Section 2.3). Based on the current studies, we presented gap analysis (Section 2.4), where we highlighted the gaps in the current state-of-the-art methods and linked it to our research objectives defined in Section 3.1.3.

As we know that the language used on social media in unstructured and informal in nature which is easy for humans to understand but in order to enable computers to understand, process and classify the language just like human beings we need first to improve the quality of text which computers can understand and perform the required task efficiently. Further, natural language is complex and ambiguous in nature which humans can understand, distinguish and classify easily due to their prior knowledge but for machines; this language ambiguities can be more challenging to utilize the information and perform the task efficiently fully. The research

problems we addressed in this thesis are defined in Section 3.1.1 with more details. Research question defined in section 3.1.2 are linked with the research problems highlighted in this thesis. To answer the first research question(**RQ:1**) of this thesis, we performed a comprehensive analysis of different common and advance pre-processing techniques on text classification task in Chapter 4. We used five different traditional machine learning and three deep learning-based classifiers with different word representation models. We found out that classification performance varies with different pre-processing techniques. We also presented the best and words performing pre-processing techniques for text classification. To deal with our second research question (**RQ:2**) we proposed a method to improve the quality of the text. We combined different pre-processing techniques in a systematic pattern which improved the quality of the text and showed better performance. In this method, we improved the quality of unstructured and informal text obtaining structured and clean text by identifying and replacing emoticons, acronyms, spell corrections, replacing contraction and elongated characters etc. which can be used to improve the quality of any social media text. Experimental results show that our proposed method improves the results as compared to baseline, highest individual performing techniques and other combinations. This study addressed our first research objective mentioned in Section 3.1.3.

In order to deal our third and fourth research questions (**RQ:3**) and (**RQ:4**), we presented hybrid words, representation model, Deep Intelligent contextual Embedding (DICE) in Chapter 5, which handles the language ambiguities such as words with different meanings (polysemy) within a context, OOV words, semantics, syntactical and sentiment knowledge of words for text classification. DICE is built up with concatenating four different word representation models to handle the language complexity, which is then fed to BiLSTM with attention model to capture the important words to predict classes. We applied our proposed model to solve the problem of Twitter sentiment analysis. We conducted several experiments to show that DICE can handle the language ambiguities defined earlier better than current state-of-the-art word representation methods. In addition to this, our proposed pre-

processing method also improves the results of current state-of-the-art methods in Twitter sentiment analysis which shows the robustness of our methods. This study addressed our second research objective, as mentioned in Section 3.1.3.

Chapter 4 and Chapter 5 of this thesis, in the same time, constitute research outputs * what shows that the research community finds presented here research relevant and significant in the area of the text classification. We also started to apply our current model on different datasets other than airline related datasets and plan to submit it in reputable venues in future. Further, we are also working to improve our current model and add different models hierarchically to improve the classification perfomance. Details are listed in the List of Publications.

This thesis highlights the importance of good quality of words representation and presents different methods to improve the current word representation methods for text classification task such as sentiment and opinion classification etc. The idea of improving the representation of the words considering the natural language ambiguities/complexities can be crucial in many different NLP related tasks. The methods proposed in this thesis highlight the importance of improving the quality of the text, especially in the case of social media analysis task (e.g. Twitter).

Further, our proposed hybrid words representation is limited to only defined natural language ambiguities such as polysemy, syntax, semantc, sentiment of words and out of vocabulary words, but more work is to be done in order to improve word representations which can handle other language ambiguities such as different words with same meanings (synonyms), metaphor, homograph, irony, sarcasm etc and different ways should be explored to learn more robust word representations. Moreover, more advanced pre-processing techniques and combinations should also be explored to improve the quality of text. And lastly, further exploration should also be applied on other benchmart datasets of different domains and areas to explore more insights.

---

*Two Papers from Chapter 5 are accepted in A-rank conference(ICDAR) (Naseem and Musial (2019)) and (AI2019) (Naseem et al. (2019)). Further, extended version will be submitted to a top rank journal with datasets from other domains & Two papers from Chapter 4 are under review. Also, Survey paper from Chapter 2 is also under review.

## 6.2 Future Work

This research opens new challenges and opportunities to deeply explore methods to improve the quality of text and handle different language ambiguities/complexities, which result in high quality of words representation.

Further research efforts could be directed towards some of the issues and challenges mentioned below.

- **Improve the quality of text**:
  - We presented a method to improve the quality of tweets; however, it will be interesting to apply the proposed method on other social media platforms.
  - We experimented with datasets which are specific to Twitter Hate Speech and Abusive language. Applying the proposed method to other areas and domains will explore more insights and will present more opportunities to improve this method.
  - More advanced pre-processing techniques and their interactions with each other (combinations) can be explored with different feature extraction and classification models.
- **Handle language ambuiguities**:
  - Our proposed model solves only defined language ambiguities. However, more work needs to be done to solve other language ambiguities such as different words with same meanings (synonyms), metaphor, homograph, irony, sarcasm etc. in the English language to learn more robust word representations.
  - More ways to be explored to learn word representations. One of those could be learning words representations hierarchically to capture more implicit and explicit characteristics[†].

[†]Naseem et al. (n.d.); Review paper has already been accepted

* Proposed method applied on airline-related tweets which can be applied to another domain and other NLP related tasks.

The main significant contribution of this thesis is to handle language ambiguities like humans. We emphasized that having good text representations helps to improve the classification performance and proposed two methods; the first one improved the quality of text and the second one managed the language ambiguities for machines to understand and classify text like humans.

# Appendices

## A Appendix for Chapter 2

**Classification techniques**:

As mentioned earlier in section 2.1, choosing an appropriate classifier is one of the main steps in the text classification task. Without having a comprehensive knowledge of every algorithm, we cannot find out the most effective model for the text classification task. Out of many machine learning algorithms used in text classification, we will present some famous and commonly used classification algorithms. These are used for sentiment classification tasks such as Nave Bayes (NB), Support vector machine (SVM), logistic regression (LR), Tree-based classifiers like decision tree (DT) and random forest (RF) and neural network-based (deep learning) algorithms. Table 1 presents the pros and cons of classification algorithms.

**Machine learning based classifiers**

**Naive Bayes(NB) classifiers**: The Naive Bayes (NB) classifiers are a group of different classification algorithms which are based on Bayes theorem, presented by Thomas Bayes (Hill; 1968). All Naive Bayes algorithms have the same assumption, i.e., each pair of features being classified is independent of others. The NB classification algorithms are widely used for information retrieval (Qu et al.; 2018) and many text classification tasks (Pak and Paroubek; 2010; Melville et al.; 2009). Naive Bayes classifiers are called "Naive" because it considers that every feature is independent of other features in the input. Whereas in reality, words and phrases in the sentences are highly interrelated. The meaning and sentiment depend on the position of words in the sentence, which can change if the position is changed.

NB classifiers are derived from Bayes theorem which states that given the number of documents ($n$) to be classified into $z$ classes where $z \in \{\mathrm{x}_1, x_2, ....x_z\}$ the predicted label out is x∈X.The Naive Bayes theorem is given as follows:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

where $y$ denotes a document and $x$ refers to the classes. In simple words, the NB algorithm will take each word in the training data and will calculate the probability of that word being classified. Once the probabilities of every word are calculated, then classifier is read to classify new data by utilizing the prior calculated probabilities during the training phase. Advantages of NB classifiers are; they are scalable, more suitable when the dimension of input is high, its implementation is simple, less computationally expensive, works well when less training data is available and can often outperform other classification algorithms. Whereas the disadvantages are; NB classifiers make a solid makes a reliable hypothesis on the shape of data distribution, i.e. any two features are independent given the output class, which gives bad results (Soheily-Khah et al.; 2017; Wang, Khardon and Protopapas; 2012). Another limitation of NB classifiers is due to data scarcity. For any value of the feature, we have to approximate the likelihood value by a frequentist

**Support vector machine (SVM)**: The support vector machine (SVM) classifiers are one of the famous and common used algorithms used for text classification due to its good performance. SVM is a non-probabilistic binary linear classification algorithm which performs by plotting the training data in multi-dimensional space. Then SVM categories the classes with a hyper-plane. The algorithm will add a new dimension if the classes can not be separated linearly in multi-dimensional space to separate the classes. This process will continue until a training data can be categorized into two different classes.

The advantage of SVM classifiers is that results are obtained by using SVM are usually better. The disadvantage of SVM algorithms is that it is not easy to choose a suitable kernel, long training time in case of extensive data and more computational resources are required etc.

**Logistic Regressiong (LR) classifier**: Logistic regression (LR) is a statistical model and is one of the earliest techniques used for classification. LR predicts probabilities rather than classes (Fan et al.; 2008; Genkin et al.; 2007) or existence of an event like a win/lose or healthy/sick etc. This can be expanded to model many classes of events like deciding whether an image consists of a cat, duck, cow, etc.

Table 1 : Comparison of Classification Algorithms

| Classifiers | Pros | Cons |
|---|---|---|
| NB | i) Less computational time.<br>ii) Easy to understand & implement.<br>iii) Can easily be trained less data. | i) Relies strongly on the class features independence and does not perform well if the condition is not met.<br>ii) Issue of zero conditional probability for zero frequency features which makes total probability zero |
| SVM | i) Effective in higher dimension<br>ii) Can model non linear decision boundary<br>iii) Robust to the issue of over-fitting | i) More computational time for large datasets<br>ii) Kernal selection is difficult<br>iii) Does not perform well in case of overlapped classes |
| LR | i) Easy and Simple to implement.<br>ii) Less computationally expensive<br>iii) Does not need tuning and features to be uniformly distributed | i) Fails in case of non-linear problems<br>ii) Need large datasets<br>iii) Predict results on the basis of independent variables |
| DT | i) Interpretable and easy to understand<br>ii) Less pre-processing required<br>iii) Fast and almost zero hyper-parameters to tuned | i) High chances of over fitting<br>ii) Less prediction accuracy as compared to others<br>iii) Complex calculation in large number of classes |
| RF | i) Fast to train, flexible and gives high results ii) Less variance than single DT<br>iii) Less pre-processing required | i) Not easy and simple to interpret<br>ii) Require more computational resources<br>iii) Require more time to predict as compared to others |
| DL | i) Fast predictions once training is complete<br>ii) Works well in case of huge data<br>iii) Flexible architecture, can be utilized for classification and regression tasks | i) Require a large amount of data<br>ii) Computationally expensive and time-consuming<br>iii) DL based classifiers are like black-box (issue of model interpretability exists) |

Every object being identified in the image would be given a probability between 0 and 1 and the sum adding to one. LR predicts the results based on the set of independent values. However, if the wrong independent values are added, then the model will not predict good results. It works well in the case of categorical results but fails in the case of continuous results. Also, LR wants that every data point to be independent of all others, but if the findings are interlinked to one another, then classifier will not predict good results.

**Decision Tree (DT) classifier**: Decision tree (DT) was presented by Magerman (1995) and developed by Quinlan (1986). It is one of the earliest classification

models for text and data mining and is employed successfully in different areas for classification task (Morgan and Sonquist; 1963). The main intuition behind this idea was to create tree-based attributes for data points, but the major question is which feature could be a parent and which will be a child's level. To fix this Statistical problem modelling for features was presented by De Mantaras (De Mántaras; 1991). DT classifier design contains a root, decision and leaf nodes which denote dataset, carry out the computation and performs classification respectively. During the training phase, the classifier learns the decision need to be executed to separate labelled categories. To classify the unknown instance, the data is passed through the tree. A particular feature from the input text is matched with the fixed which was known during the training stage. The calculation at each decision node compares the chosen features with this fixed feature earlier; the decision relies on whether the feature is more prominent than or less than the fixed which creates two-way division in the tree. The text will eventually go over these decision nodes until it reaches the leaf node that describes it assigned class.

The advantages of DT classifier are; the amount of hyper-parameters which require tuning is nearly zero, easy to describe, can be understood easily by its visualizations whereas the significant disadvantages of DT classifier are; it is sensitive to minor change in the data (Giovanelli et al.; 2017) and have probability of overfitting (Quinlan; 1987), complex computations in case of large number of class labels and have difficulties with-out-of sample prediction.

**Random Forest (RF) Classifier**: Random forest which is also called an ensembles learning technique for text classification which concentrates on methods to compare the results of several trained models in line to give better classifier and performance than a single model. Ho (1998) proposed RF classifier, which is simple to understand and also gives better results in classification. RF classifier is composed of the number of DT classifiers where every tree is trained by a bootstrapped subset of the training text. An arbitrary subset of the characteristics is selected at every decision node, and the model will only examine part of these features. The primary issue with utilizing the single tree is that it has massive variation so that the arrangement of

the training data and features can impact its results.

This classifier is quick to train for textual data but slow in giving predictions when trained (Bansal et al.; 2018). Performs good with both categorical and continuous variables, can automatically handle missing values, robust to outliers and less affected by noise whereas training a vast number of trees can be computationally expensive, require more training time and utilize much memory.

**Deep learning based classifiers**

Deep learning is a group of algorithms and models motivated by the working of the human brain. It has attained state-of-the-art results in many different areas which include many NLP applications. It requires a large number of training data to achieve a semantically good representation of textual data. Deep learning models have attained excellent results compared to machine learning models on different classification tasks. Main architectures of deep learning which are commonly used in any text classification task, are briefly discussed below.

**Recurrent Neural Network (RNN)**: RNN is one of the popular neural network-based model which is widely used for different text classification tasks (Sutskever et al.; 2011; Mandic and Chambers; 2001). Previous data points of a sequence are assigned more weights in an RNN model which makes it more useful and better for any text, string or sequential data classification. RNN models deal with data from previous layers/nodes in such a good way which makes them superiors for semantic analysis of a corpus. Gated recurrent unit (GRU) and long short term memory (LSTM) are the most common types of RNNs which are used of text classification.

The one of the drawback of RNN is that they are sensitive to gradient vanishing problem and exploding gradient when gradient descents error is back propagated (Bengio et al.; 1994).

- **Long Short-Term Memory (LSTM)**: LSTM was presented by Hochreiter and Schmidhuber (1997). LSTM was presented to address the gradient descent issues of RNN by keeping the long term dependency in a more better way as compared to RNNs. It is more effective to overcome the issues of vanishing gradient (Pascanu

et al.; 2013). Even though LSTMs have an architecture like a chain which is same as RNNs but it uses different gates which handles the volume of information carefully, which is allowed from each node state. The role of each gate and node in a basic LSTM cell is explained below.

$$\mathrm{i}_t = \sigma(W_i[x_t, ht-1] + bi)$$

$$\widehat{C}_t = tanh(W_c[x_t, ht-1] + b_c),$$

$$\mathrm{f}_t = \sigma(W_f[x_t, h_{t-1}] + b_f),$$

$$\mathrm{C}_t = i_t * \widehat{C}_t + f_t C_{t-1},$$

$$\mathrm{o}_t = \sigma(W_o)[x_t, h_{t-1}] + b_o,$$

$$\mathrm{h}_t = o_t tanh(C_t),$$

where $i_t$, $\widehat{C}_t$ and $f_t$ denotes input gate, candid memory cell and forget gate activation respectively. whereas $C_t$ computes new memory cell value and $o_t$ and$h_t$ represents the final output gate. $b$ is bias vector, $W$ denotes weight matrix and $x_t$ denotes input to the memory cell at time $t$.

- **Gated Recurrent Unit (GRU)**: GRU is another type of RNNs which are presented by Chung et al. (2014) and Cho et al. (2014). GRU is the simplest form of LSTM architecture. However, it includes two gates and does not contain internal memory which makes it different from LSTM. Also, in GRU, a second non-linearity (tanh) is not applied on a network. The working of a GRU cell is given below:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$\widehat{r}_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$\mathrm{h}_t = z_t h_{t-1} + (1 + z_t)$$

$$\sigma_h(W_h x_t + U_h(r_t h_{t-1}) + b_h)$$

where $z_t$ denotes to the update gate of t, $x_t$ represents input vector, W,U and b denotes parameter matrices. $\sigma_g$ which is a activation function can be ReLU or sigmoid, $\mathrm{r}_t$ represents reset gate of t, $h_t$ is the output gate of vector t , and $\sigma_h$ denotes the hyperbolic tangent function.

**Convolutional Neural Networks (CNN)**: Another famous architecture of deep learning is CNN which is mostly used for hierarchical classification in a deep learning (Jaderberg et al.; 2014). CNN was built and used for image classification in early days, but over the period, it has shown excellent results for text classification as well (Lecun et al.; 1998). In a image classification, an image tensor is convolved with a set of kernels of size $dxd$. The convolution layers in the CNN are known as feature maps which can be stacked to have multiple filters. To overcome the computational issue due to the size of dimensionality, CNN uses pooling layer to reduce the size from one layer to the other one. Different pooling methods have been proposed by researchers to decrease the output without losing features (Scherer et al.; 2010).

Max pooling is the most common pooling technique where maximum elements in the pooling window are selected. To feed the pooled output from stacked features map to the next one, features are flattened into one column. Usually, the last layer of CNNs is fully connected. Weights and feature filters are adjusted during the backpropagation step of CNN. The number of channels is the major issue with CNN's for text classification, which is very few in case of image classification. Three channels form RGB. For text, it can be a vast number which makes dimensions very high for text classification (Johnson and Zhang; 2014).

### Evaluation Metrics

In this section, different evaluation metrics are briefly discussed. But first some related concept will be briefly discussed.

**Confusion matrix**: Confusion matrix is a unique table or a method which is used to present the efficiency of the classification algorithm. In Table 2, we present the confusion matrix. Details are given below:

**True Positives (TP)**: TP are the accurately predicted positive instances.

**True Negatives (TN)**: TN are the accurately predicted negative instances.

**False Positives (FP)**: FP are wrongly predicted positive instances.

**False Negatives (FN)**: FN are wrongly predicted negative instances.

Table 2 : Confusion Matrix

| Actual Class | | | |
|---|---|---|---|
| Predicted Class | | Positive | Negative |
| | Positive | *True Positive (TP)* | *False Negative (FN)* |
| | Negative | *False Positive (FP)* | *True Negative (TN)* |

Once we understand the confusion matrix and its parameters, then we can define and understand evaluation metrics easily, briefly explained below.
**Accuracy**: Accuracy is the simple ratio of observations predicted correctly to the total observations and is given by

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

**Precision**: Precision is the ratio of true positive (TP) observations to the overall positive predicted values (TP+FP) and is given by

$$Precision = \frac{TP}{TP + FP}$$

**Recall**: Recall is the ration of true positive (TP) observations to the overall observations (TP+FN) and is given by

$$Recall = \frac{TP}{TP + FN}$$

**F1 score** - Weighted average of Recall and Precision is knowns as F1 score which means F1-score consists of both FPs and FNs and is given by

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision}$$

# B Appendix for Chapter 4

## Graphical representation of results

1. Table 4.4: Comparison of all technique's on Waseem et al. dataset

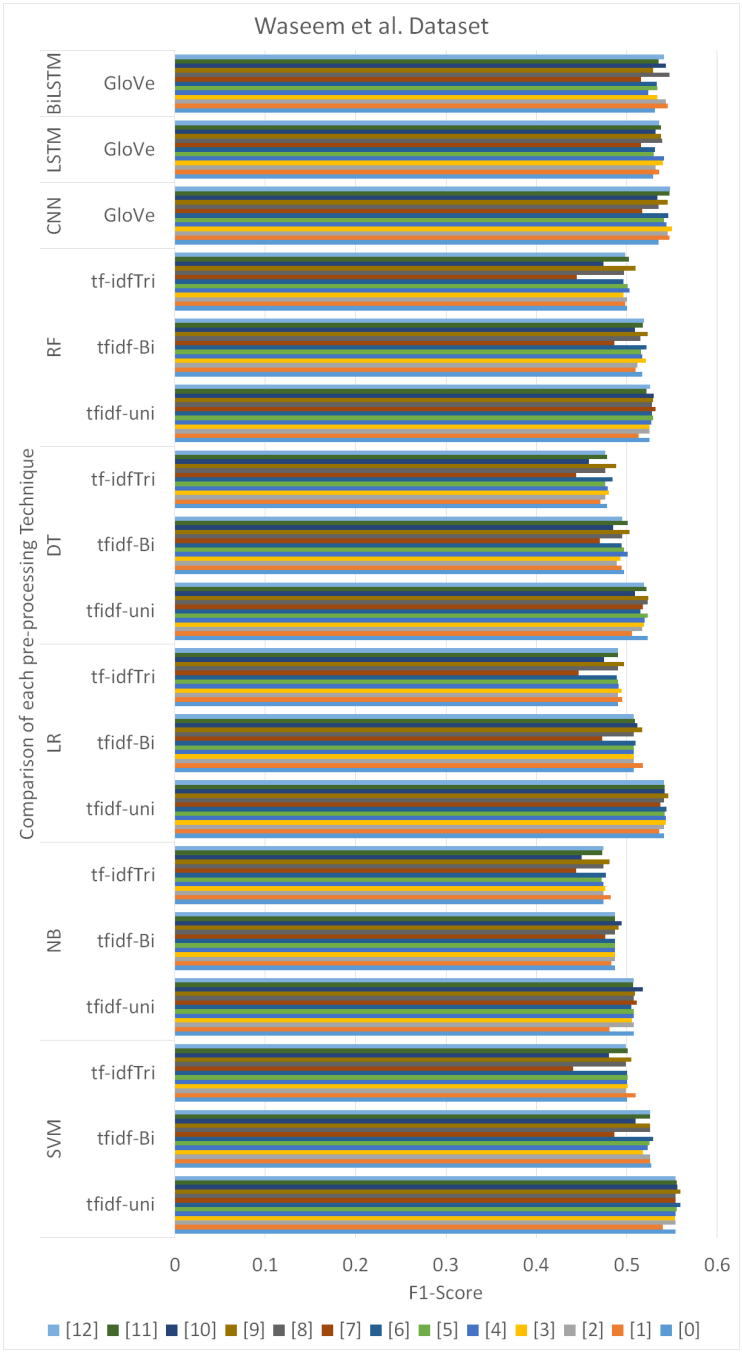


Figure 1 : Comparison of all technique's on Waseem et al. dataset

2. Table 4.5: Comparison of all technique's on Golbeck et al. dataset

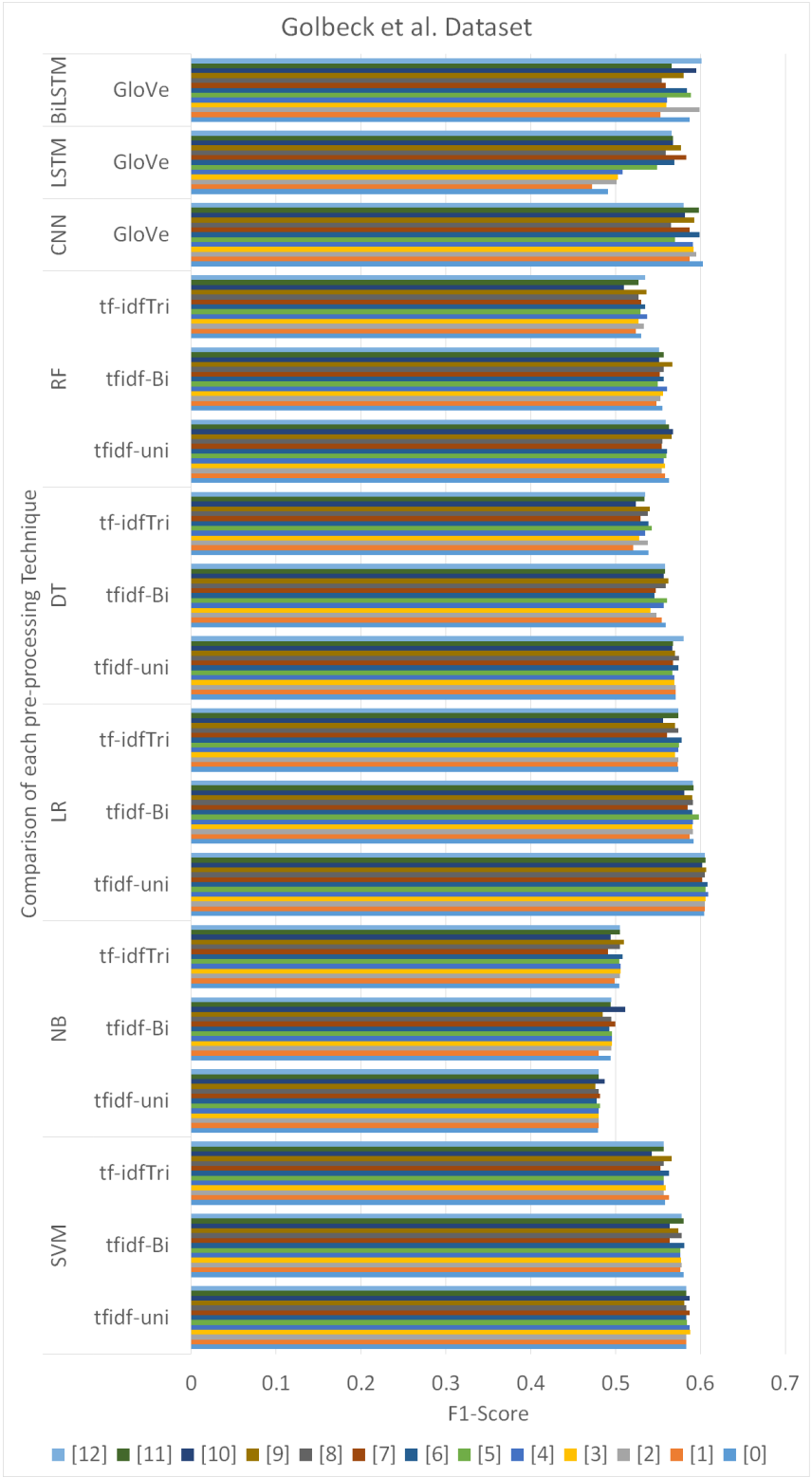


Figure 2 : Comparison of all technique's on Golbeck et al. dataset

3. Table 4.6: Comparison of all technique's on David et al. dataset

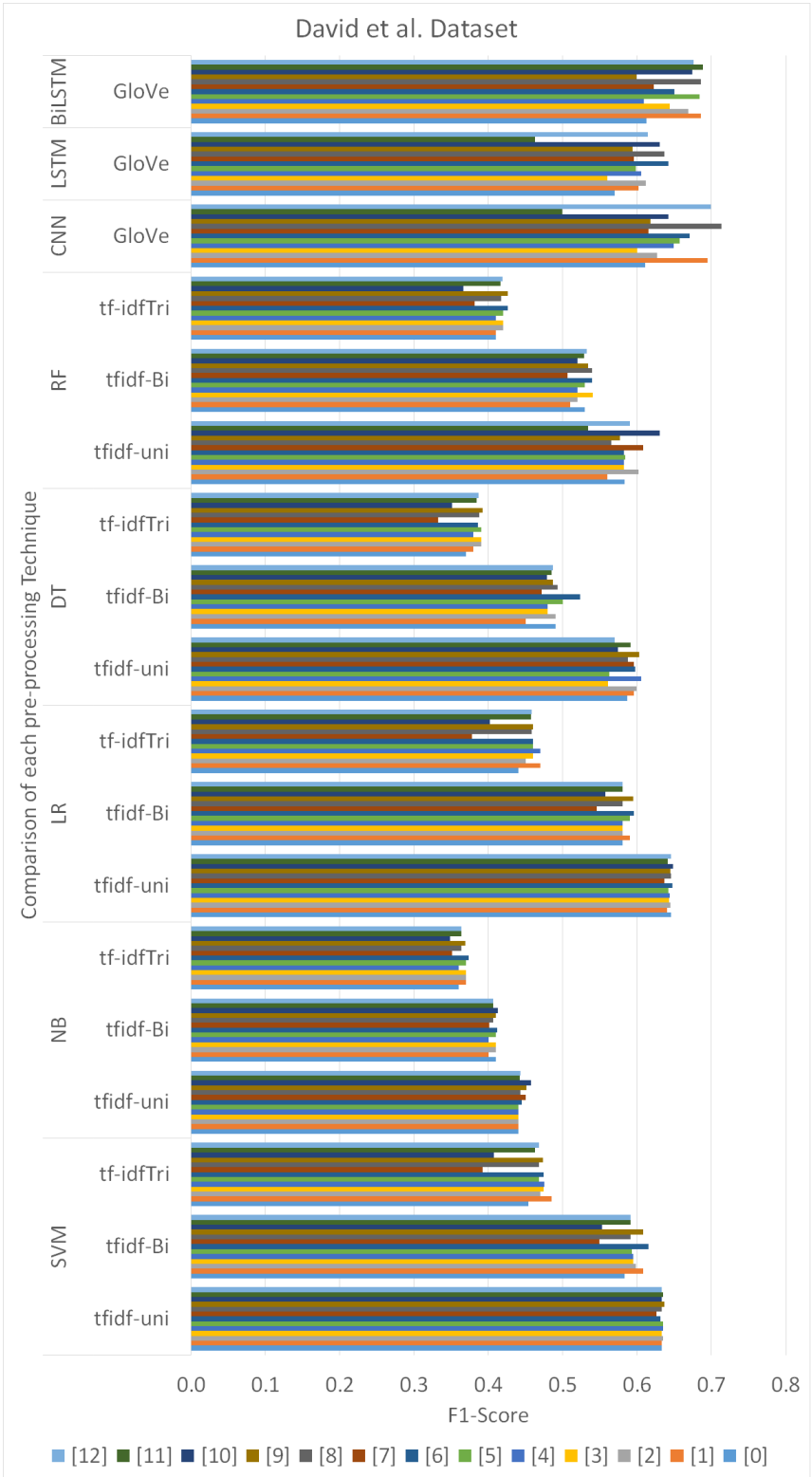


Figure 3 : Comparison of all technique's on David et al. dataset

# Bibliography

Adhi, B., Saskiah, D. and Widodo, W. (2019). A systematic literature review of short text classification on twitter, *KnE Social Sciences* **3**: 625.

Agarwal, A., Xie, B., Vovsha, I., Rambow, O. and Passonneau, R. J. (2011). Sentiment analysis of twitter data.

Aggarwal, C. C. and Zhai, C. (2012a). *A Survey of Text Classification Algorithms*, Springer US, Boston, MA, pp. 163–222.
**URL:** *https://doi.org/10.1007/978-1-4614-3223-4*$_6$

Aggarwal, C. C. and Zhai, C. (2012b). A survey of text classification algorithms, *Mining Text Data*.

Altszyler, E., Sigman, M. and Slezak, D. F. (2016). Comparative study of lsa vs word2vec embeddings in small corpora: a case study in dreams database, *ArXiv* **abs/1610.01520**.

Balahur, A. (2013). Sentiment analysis in social media texts, *WASSA@NAACL-HLT*.

Balazs, J. A. and Velásquez, J. D. (2016). Opinion mining and information fusion: A survey, *Information Fusion* **27**: 95–110.

Bansal, H., Shrivastava, G., Nhu, N. and STANCIU, L. (2018). *Social Network Analytics for Contemporary Business Organizations*.

Bao, Y., Quan, C., Wang, L. and Ren, F. (2014). The role of pre-processing in twitter sentiment analysis, *in* D.-S. Huang, K.-H. Jo and L. Wang (eds), *Intelligent Computing Methodologies*, Springer International Publishing, Cham, pp. 615–624.

Batrinca, B. and Treleaven, P. C. (2015). Social media analytics: a survey of techniques, tools and platforms, *AI & SOCIETY* **30**(1): 89–116.
**URL:** *https://doi.org/10.1007/s00146-014-0549-4*

Bengio, Y., Courville, A. and Vincent, P. (2013). Representation learning: A review and new perspectives, *IEEE transactions on pattern analysis and machine intelligence* **35**(8): 1798–1828.

Bengio, Y., Ducharme, R., Vincent, P. and Janvin, C. (2003). A neural probabilistic language model, *J. Mach. Learn. Res.* **3**: 1137–1155.
**URL:** *http://dl.acm.org/citation.cfm?id=944919.944966*

Bengio, Y., Simard, P. and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult, *Trans. Neur. Netw.* **5**(2): 157–166.
**URL:** *https://doi.org/10.1109/72.279181*

Bermingham, A. and Smeaton, A. (2011). On using twitter to monitor political sentiment and predict election results, *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011)*, Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pp. 2–10.
**URL:** *https://www.aclweb.org/anthology/W11-3702*

Bermingham, A. and Smeaton, A. F. (2010). Classifying sentiment in microblogs: is brevity an advantage?, *CIKM*.

Boia, M., Faltings, B., Musat, C. C. and Pu, P. (2013). A :) is worth a thousand words: How people attach sentiment to emoticons and words in tweets, *2013 International Conference on Social Computing* pp. 345–350.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2016a). Enriching word vectors with subword information, *arXiv preprint arXiv:1607.04606* .

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2016b). Enriching word vectors with subword information, *CoRR* **abs/1607.04606**.
**URL:** *http://arxiv.org/abs/1607.04606*

Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V. and Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings, *in* D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett (eds), *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pp. 4349–4357.
**URL:** *http://papers.nips.cc/paper/6228-man-is-to-computer-programmer-as-woman-is-to-homemaker-debiasing-word-embeddings.pdf*

Brody, S. and Diakopoulos, N. (2011). Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! using word lengthening to detect sentiment in microblogs, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Edinburgh, Scotland, UK., pp. 562–570.
**URL:** *https://www.aclweb.org/anthology/D11-1052*

Camacho-Collados, J., Pilehvar, M. T. and Navigli, R. (2016). Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities, *Artificial Intelligence* **240**: 36 – 64.
**URL:** *http://www.sciencedirect.com/science/article/pii/S0004370216300820*

Cambria, E., Poria, S., Hazarika, D. and Kwok, K. (2018). Senticnet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings, *AAAI*.

Cambria, E., Schuller, B., Xia, Y. and Havasi, C. (2013). New avenues in opinion mining and sentiment analysis, *IEEE Intelligent Systems* **28**(2): 15–21.
**URL:** *http://dx.doi.org/10.1109/MIS.2013.30*

Castellucci, G., Croce, D. and Basili, R. (2015). Acquiring a large scale polarity lexicon through unsupervised distributional methods, *in* C. Biemann, S. Handschuh, A. Freitas, F. Meziane and E. Métais (eds), *Natural Language Processing and Information Systems*, Springer International Publishing, Cham, pp. 73–86.

Celebi, A. and Ozgur, A. (2016). Segmenting hashtags using automatically created training data.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T. and Koehn, P. (2013). One billion word benchmark for measuring progress in statistical language modeling, *CoRR* **abs/1312.3005**.
**URL:** *http://arxiv.org/abs/1312.3005*

Chen, M., Jin, X. and Shen, D. (2011). Short text classification improved by learning multi-granularity topics, *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, AAAI Press, pp. 1776–1781.
**URL:** *http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-298*

Chen, W. J., Xie, X., Wang, J., Pradhan, B., Hong, H., Bui, D. T., Duan, Z. and Ma, J. (2017). A comparative study of logistic model tree, random forest, and classification and regression tree models for spatial prediction of landslide susceptibility.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, pp. 1724–1734.
**URL:** *https://www.aclweb.org/anthology/D14-1179*

Chung, J., Gülçehre, Ç., Cho, K. and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling, *CoRR* **abs/1412.3555**.
**URL:** *http://arxiv.org/abs/1412.3555*

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning, *Proceed-*

*ings of the 25th International Conference on Machine Learning*, ICML '08, ACM, New York, NY, USA, pp. 160–167.
**URL:** *http://doi.acm.org/10.1145/1390156.1390177*

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. P. (2011). Natural language processing (almost) from scratch, *CoRR* **abs/1103.0398**.
**URL:** *http://arxiv.org/abs/1103.0398*

da Silva, N. F., Hruschka, E. R. and Hruschka, E. R. (2014). Tweet sentiment analysis with classifier ensembles, *Decis. Support Syst.* **66**(C): 170–179.
**URL:** *http://dx.doi.org/10.1016/j.dss.2014.07.003*

Davidson, T., Warmsley, D., Macy, M. W. and Weber, I. (2017). Automated hate speech detection and the problem of offensive language, *CoRR* **abs/1703.04009**.
**URL:** *http://arxiv.org/abs/1703.04009*

De Mántaras, R. L. (1991). A distance-based attribute selection measure for decision tree induction, *Machine Learning* **6**(1): 81–92.
**URL:** *https://doi.org/10.1023/A:1022694001379*

Devlin, J., Chang, M., Lee, K. and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding, *CoRR* **abs/1810.04805**.
**URL:** *http://arxiv.org/abs/1810.04805*

Dhingra, B., Liu, H., Salakhutdinov, R. and Cohen, W. W. (2017). A comparative study of word embeddings for reading comprehension, *CoRR* **abs/1703.00993**.
**URL:** *http://arxiv.org/abs/1703.00993*

Diyi, Dyer, C., He, X., Smola, A. J. and Hovy, E. H. (2016). Hierarchical attention networks for document classification., *The Annual Conference of the*

*North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–9.

dos Santos, C. N. and de C. Gatti, M. A. (2014). Deep convolutional neural networks for sentiment analysis of short texts, *COLING*.

Elekes, ., Englhardt, A., Schler, M. and Bhm, K. (2018). Toward meaningful notions of similarity in nlp embedding models, *International Journal on Digital Libraries* .

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. and Lin, C.-J. (2008). Liblinear: A library for large linear classification, *J. Mach. Learn. Res.* **9**: 1871–1874.
**URL:** *http://dl.acm.org/citation.cfm?id=1390681.1442794*

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E. H. and Smith, N. A. (2014). Retrofitting word vectors to semantic lexicons, *CoRR* **abs/1411.4166**.
**URL:** *http://arxiv.org/abs/1411.4166*

Fayyad, U. M., Piatetsky-Shapiro, G. and Uthurusamy, R. (2003). Summary from the kdd-03 panel: Data mining: The next 10 years, *SIGKDD Explor. Newsl.* **5**(2): 191–196.
**URL:** *http://doi.acm.org/10.1145/980972.981004*

Felbo, B., Mislove, A., Søgaard, A., Rahwan, I. and Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Copenhagen, Denmark, pp. 1615–1625.
**URL:** *https://www.aclweb.org/anthology/D17-1169*

Foster, J., Çetinoğlu, Ö., Wagner, J., Le Roux, J., Nivre, J., Hogan, D. and van Genabith, J. (2011). From news to comment: Resources and benchmarks for parsing the language of web 2.0, *Proceedings of 5th International Joint*

*Conference on Natural Language Processing*, Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pp. 893–901.
**URL:** *https://www.aclweb.org/anthology/I11-1100*

Fu, X., Liu, W., Xu, Y. and Cui, L. (2017). Combine hownet lexicon to train phrase recursive autoencoder for sentence-level sentiment analysis, *Neurocomputing* **241**: 18–27.

Genkin, A., Lewis, D. D. and Madigan, D. (2007). Large-scale bayesian logistic regression for text categorization, *Technometrics* **49**(3): 291–304.
**URL:** *https://doi.org/10.1198/004017007000000245*

Giachanou, A. and Crestani, F. (2016). Like it or not: A survey of twitter sentiment analysis methods, *ACM Computing Surveys* **49**: 1–41.

Giachanou, A., Gonzalo, J., Mele, I. and Crestani, F. (2017). Sentiment propagation for predicting reputation polarity.

Gimpel, K., Schneider, N., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J. and Smith, N. A. (n.d.). Part-of-speech tagging for twitter: Annotation, features, and experiments.

Giovanelli, C., Liu, X. U., Sierla, S. A., Vyatkin, V. and Ichise, R. (2017). Towards an aggregator that exploits big data to bid on frequency containment reserve market, *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society* pp. 7514–7519.

Go, A., Bhayani, R. and Huang, L. (2009). Twitter sentiment classification using distant supervision, *Processing* pp. 1–6.
**URL:** *http://www.stanford.edu/ alecmgo/papers/TwitterDistantSupervision09.pdf*

Golbeck, J., Ashktorab, Z., Banjo, R. O., Berlinger, A., Bhagwan, S., Buntain, C., Cheakalos, P., Geller, A. A., Gergory, Q., Gnanasekaran, R. K., Gunasekaran, R. R., Hoffman, K. M., Hottle, J., Jienjitlert, V., Khare, S., Lau, R., Martindale, M. J., Naik, S., Nixon, H. L., Ramachandran, P.,

Rogers, K. M., Rogers, L., Sarin, M. S., Shahane, G., Thanki, J., Vengataraman, P., Wan, Z. and Wu, D. M. (2017). A large labeled corpus for online harassment research, *WebSci*.

Greevy, E. (2004). Automatic text categorisation of racist webpages.

Gurusamy, V. and Kannan, S. (2014). Preprocessing techniques for text mining.

Haddi, E., Liu, X. and Shi, Y. (2013). The role of text pre-processing in sentiment analysis, *ITQM*.

He, W., Zha, S. and Li, L. (2013). Social media competitive analysis and text mining: A case study in the pizza industry, *Int J. Information Management* **33**: 464–472.

He, Y., Lin, C. and Alani, H. (2011). Automatically extracting polarity-bearing topics for cross-domain sentiment classification, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Portland, Oregon, USA, pp. 123–131.
**URL:** *https://www.aclweb.org/anthology/P11-1013*

Herbelot, A. and Baroni, M. (2017). High-risk learning: acquiring new word vectors from tiny data, *CoRR* **abs/1707.06556**.
**URL:** *http://arxiv.org/abs/1707.06556*

Hill, B. M. (1968). Posterior distribution of percentiles: Bayes' theorem for sampling from a population, *Journal of the American Statistical Association* **63**(322): 677–691.
**URL:** *http://www.jstor.org/stable/2284038*

Ho, T. K. (1998). The random subspace method for constructing decision forests, *IEEE Trans. Pattern Anal. Mach. Intell.* **20**(8): 832–844.
**URL:** *http://dx.doi.org/10.1109/34.709601*

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory, *Neural Comput.* **9**(8): 1735–1780.
**URL:** *http://dx.doi.org/10.1162/neco.1997.9.8.1735*

Hovy, D. and Waseem, Z. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on twitter, *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, June 12-17, 2016*, pp. 88–93.
**URL:** *https://www.aclweb.org/anthology/N16-2013/*

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Melbourne, Australia, pp. 328–339.
**URL:** *https://www.aclweb.org/anthology/P18-1031*

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, ACM, New York, NY, USA, pp. 168–177.
**URL:** *http://doi.acm.org/10.1145/1014052.1014073*

Hussein, D. M. E.-D. M. (2018). A survey on sentiment analysis challenges, *Journal of King Saud University - Engineering Sciences* **30**(4): 330 – 338.
**URL:** *http://www.sciencedirect.com/science/article/pii/S1018363916300071*

Hutto, C. J. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text, *ICWSM*.

Iacobacci, I., Pilehvar, M. T. and Navigli, R. (2015). Sensembed: Learning sense embeddings for word and relational similarity, *ACL*.

Ilic, S., Marrese-Taylor, E., Balazs, J. A. and Matsuo, Y. (2018). Deep contextualized word representations for detecting sarcasm and irony, *CoRR* **abs/1809.09795**.
**URL:** *http://arxiv.org/abs/1809.09795*

Jaderberg, M., Simonyan, K., Vedaldi, A. and Zisserman, A. (2014). Reading text in the wild with convolutional neural networks, *CoRR* **abs/1412.1842**.
**URL:** *http://arxiv.org/abs/1412.1842*

Jianqiang, Z. (2015). Pre-processing boosting twitter sentiment analysis?, pp. 748–753.

Jianqiang, Z. and Xiaolin, G. (2017). Comparison research on text pre-processing methods on twitter sentiment analysis, *IEEE Access* **5**: 2870–2879.

Jianqiang, Z. and Xiaolin, G. (2018). Deep convolution neural networks for twitter sentiment analysis, *IEEE Access* **PP**: 1–1.

Johnson, R. and Zhang, T. (2014). Effective use of word order for text categorization with convolutional neural networks, *CoRR* **abs/1412.1058**.
**URL:** *http://arxiv.org/abs/1412.1058*

Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2016). Bag of tricks for efficient text classification, *CoRR* **abs/1607.01759**.
**URL:** *http://arxiv.org/abs/1607.01759*

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C. and Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation.

Khan, F. H., Bashir, S. and Qamar, U. (2014). Tom: Twitter opinion mining framework using hybrid classification scheme, *Decis. Support Syst.* **57**: 245–257.
**URL:** *http://dx.doi.org/10.1016/j.dss.2013.09.004*

Kharde, V. A. and Sonawane, S. (2016). Sentiment analysis of twitter data : A survey of techniques, *CoRR* **abs/1601.06971**.
**URL:** *http://arxiv.org/abs/1601.06971*

Kim, Y. (2014a). Convolutional neural networks for sentence classification, *arXiv preprint arXiv:1408.5882* .

Kim, Y. (2014b). Convolutional neural networks for sentence classification, *EMNLP*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization, *CoRR* **abs/1412.6980**.

Kiritchenko, S., Zhu, X.-D., Cherry, C. and Mohammad, S. (2014). Nrc-canada-2014: Detecting aspects and sentiment in customer reviews, *SemEval@COLING*.

Kiritchenko, S., Zhu, X. and Mohammad, S. M. (2014). Sentiment analysis of short informal texts, *J. Artif. Int. Res.* **50**(1): 723–762.
**URL:** *http://dl.acm.org/citation.cfm?id=2693068.2693087*

Korde, V. and Mahender, C. N. (2012). Text classification and classifiers: A survey.

Kouloumpis, E., Wilson, T. and Moore, J. D. (2011). Twitter sentiment analysis: The good the bad and the omg!, *ICWSM*.

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E. and Brown, D. E. (2019). Text classification algorithms: A survey, *CoRR* **abs/1904.08067**.
**URL:** *http://arxiv.org/abs/1904.08067*

Kwok, I. and Wang, Y. (2013). Locate the hate: Detecting tweets against blacks, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, AAAI Press, pp. 1621–1622.
**URL:** *http://dl.acm.org/citation.cfm?id=2891460.2891697*

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C. (2016). Neural architectures for named entity recognition, *CoRR* **abs/1603.01360**.
**URL:** *http://arxiv.org/abs/1603.01360*

Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining, *CoRR* **abs/1901.07291**.
**URL:** *http://arxiv.org/abs/1901.07291*

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations.

Larson, R. R. (2010). Introduction to information retrieval, *J. Am. Soc. Inf. Sci. Technol.* **61**(4): 852–853.
**URL:** *http://dx.doi.org/10.1002/asi.v61:4*

Lauren, P., Qu, G., Zhang, F. and Lendasse, A. (2018). Discriminant document embeddings with an extreme learning machine for classifying clinical narratives, *Neurocomputing* **277**: 129–138.
**URL:** *https://app.dimensions.ai/details/publication/pub.1091311082*

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents, *CoRR* **abs/1405.4053**.
**URL:** *http://arxiv.org/abs/1405.4053*

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning, *Nature* **521**: 436–44.

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition, *Proceedings of the IEEE* **86**: 2278 – 2324.

Ledell, Fisch, A., Chopra, S., Adams, K., Bordes, A. and Weston, J. (2017). StarSpace: Embed All The Things!, *arXiv e-prints* p. arXiv:1709.03856.

Liang-Chih, Lai, K. R. and Zhang, X. (2018). Refining word embeddings using intensity scores for sentiment analysis, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **26**: 671–681.

Liang-Chih, Wang, J., Lai, K. R. and Zhang, X. (2018). Refining word embeddings using intensity scores for sentiment analysis, *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* **26**(3): 671–681.
**URL:** *https://doi.org/10.1109/TASLP.2017.2788182*

Lin, C. and He, Y. (2009). Joint sentiment/topic model for sentiment analysis, *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, ACM, New York, NY, USA, pp. 375–384.
**URL:** *http://doi.acm.org/10.1145/1645953.1646003*

Liu, P., Qiu, X. and Huang, X. (2015). Learning context-sensitive word embeddings with neural tensor skip-gram model, *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, AAAI Press, pp. 1284–1290.
**URL:** *http://dl.acm.org/citation.cfm?id=2832415.2832428*

Liu, S. and Forss, T. (2014). Combining n-gram based similarity analysis with sentiment analysis in web content classification, *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1*, IC3K 2014, SCITEPRESS - Science and Technology Publications, Lda, Portugal, pp. 530–537.
**URL:** *https://doi.org/10.5220/0005170305300537*

Liu, X., He, P., Chen, W. and Gao, J. (2019). Multi-task deep neural networks for natural language understanding, *CoRR* **abs/1901.11504**.
**URL:** *http://arxiv.org/abs/1901.11504*

Liu, Y., Liu, B., Shan, L. and Wang, X. (2018). Modelling context with neural networks for recommending idioms in essay writing, *Neurocomputing*

**275**: 2287 – 2293.
**URL:** *http://www.sciencedirect.com/science/article/pii/S0925231217317198*

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach, *CoRR* **abs/1907.11692**.
**URL:** *http://arxiv.org/abs/1907.11692*

Looks, M., Herreshoff, M., Hutchins, D. and Norvig, P. (2017). Deep learning with dynamic computation graphs, *ArXiv* **abs/1702.02181**.

Magerman, D. M. (1995). Statistical decision-tree models for parsing, *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 276–283.
**URL:** *https://doi.org/10.3115/981658.981695*

Mandic, D. P. and Chambers, J. (2001). *Recurrent Neural Networks for Prediction: Learning Algorithms,Architectures and Stability*, John Wiley & Sons, Inc., New York, NY, USA.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S. and McClosky, D. (2014). The stanford corenlp natural language processing toolkit., *ACL (System Demonstrations)*, pp. 55–60.

MARTNEZ-CMARA, E., MARTN-VALDIVIA, M. T., UREA-LPEZ, L. A. and MONTEJO-REZ, A. R. (2014). Sentiment analysis in twitter, *Natural Language Engineering* **20**(1): 128.

McCann, B., Bradbury, J., Xiong, C. and Socher, R. (2017a). Learned in translation: Contextualized word vectors, *NIPS*.

McCann, B., Bradbury, J., Xiong, C. and Socher, R. (2017b). Learned in translation: Contextualized word vectors, *CoRR* **abs/1708.00107**.
**URL:** *http://arxiv.org/abs/1708.00107*

Medhat, W., Hassan, A. and Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey, *Ain Shams Engineering Journal* **5**(4): 1093 – 1113.
**URL:** *http://www.sciencedirect.com/science/article/pii/S2090447914000550*

Mejova, Y. and Srinivasan, P. (2011). Exploring feature definition and selection for sentiment classifiers.

Melamud, O., Goldberger, J. and Dagan, I. (2016a). context2vec: Learning generic context embedding with bidirectional lstm, *CoNLL*.

Melamud, O., Goldberger, J. and Dagan, I. (2016b). context2vec: Learning generic context embedding with bidirectional LSTM, *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, Association for Computational Linguistics, Berlin, Germany, pp. 51–61.
**URL:** *https://www.aclweb.org/anthology/K16-1006*

Melville, P., Gryc, W. and Lawrence, R. D. (2009). Sentiment analysis of blogs by combining lexical knowledge with text classification, *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, ACM, New York, NY, USA, pp. 1275–1284.
**URL:** *http://doi.acm.org/10.1145/1557019.1557156*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J. (2013). Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems*, pp. 3111–9.

Mohammad, S. (2016). A practical guide to sentiment annotation: Challenges and solutions, *WASSA@NAACL-HLT*.

Mohammad, S., Kiritchenko, S. and Zhu, X. (2013). Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets, *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.
**URL:** *http://aclweb.org/anthology/S13-2053*

Morgan, J. N. and Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal, *Journal of the American Statistical Association* **58**(302): 415–434.
**URL:** *http://www.jstor.org/stable/2283276*

Mrksic, N., Vulic, I., Séaghdha, D. Ó., Leviant, I., Reichart, R., Gasic, M., Korhonen, A. and Young, S. J. (2017). Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints, *CoRR* **abs/1706.00374**.
**URL:** *http://arxiv.org/abs/1706.00374*

Mullen, T. and Malouf, R. (2006). A preliminary investigation into sentiment analysis of informal political discourse, Vol. SS-06-03, pp. 159–162. cited By 68.
**URL:** *https://www.scopus.com/inward/record.uri?eid=2-s2.0-33747172751partnerID=40md5=6b12793b70eae006102989ed6d398fcb*

Naili, M., Chaibi, A. H. and Ghezala, H. H. B. (2017). Comparative study of word embedding methods in topic segmentation, *Procedia Computer Science* **112**: 340 – 349. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France.
**URL:** *http://www.sciencedirect.com/science/article/pii/S1877050917313480*

Narayanan, V., Arora, I. and Bhatia, A. (2013). Fast and accurate sentiment classification using an enhanced naive bayes model, *CoRR* **abs/1305.6143**.
**URL:** *http://arxiv.org/abs/1305.6143*

Naseem, U., Khan, S., Razzak, I. and Hameed, I. (2019). *Hybrid Words Representation for Airlines Sentiment Analysis*, pp. 381–392.

Naseem, U. and Musial, K. (2019). Dice: Deep intelligent contextual embedding for twitter sentiment analysis, *2019 15th International Conference on Document Analysis and Recognition (ICDAR)* pp. 1–5.

Naseem, U., Razzak, I. and Hameed, I. A. (n.d.). Deep context-aware embedding for abusive and hate speech detection on twitter, *Australian Journal of Intelligent Information Processing Systems* p. 69.

Neelakantan, A., Shankar, J., Passos, A. and McCallum, A. (2015). Efficient non-parametric estimation of multiple embeddings per word in vector space, *CoRR* **abs/1504.06654**.
**URL:** *http://arxiv.org/abs/1504.06654*

Niebler, T., Becker, M., Pölitz, C. and Hotho, A. (2017). Learning semantic relatedness from human feedback using metric learning, *CoRR* **abs/1705.07425**.
**URL:** *http://arxiv.org/abs/1705.07425*

O'Connor, B., Balasubramanyan, R., Routledge, B. R. and Smith, N. A. (2010). From tweets to polls: Linking text sentiment to public opinion time series, *in* W. W. Cohen and S. Gosling (eds), *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*, The AAAI Press.
**URL:** *http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1536*

Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining, *LREC*.

Pang, B. and Lee, L. (2008a). Opinion mining and sentiment analysis, *Foundations and Trends in Information Retrieval* **2**(12): 1–135.
**URL:** *http://dx.doi.org/10.1561/1500000011*

Pang, B. and Lee, L. (2008b). Opinion mining and sentiment analysis, *Found. Trends Inf. Retr.* **2**(1-2): 1–135.
**URL:** *http://dx.doi.org/10.1561/1500000011*

Pang, B., Lee, L. and Vaithyanathan, S. (2002a). Thumbs up?: Sentiment classification using machine learning techniques, *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 79–86.
**URL:** *https://doi.org/10.3115/1118693.1118704*

Pang, B., Lee, L. and Vaithyanathan, S. (2002b). Thumbs up?: Sentiment classification using machine learning techniques, *The Conference on Empirical Methods on Natural Language Processing*, Association for Computational Linguistics, pp. 79–86.

Pascanu, R., Mikolov, T. and Bengio, Y. (2013). On the difficulty of training recurrent neural networks, *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, JMLR.org, pp. III–1310–III–1318.
**URL:** *http://dl.acm.org/citation.cfm?id=3042817.3043083*

Patriche, C., Gabriel, P., Grozavu, A. and Roca, B. (2016). A comparative analysis of binary logistic regression and analytical hierarchy process for landslide susceptibility assessment in the dobrov river basin, romania, *Pedosphere* **26**: 335–350.

Pennington, J., Socher, R. and Manning, C. D. (2014). Glove: Global vectors for word representation, *In EMNLP*.

Perkins, J. (2010). *Python Text Processing with NLTK 2.0 Cookbook*, Packt Publishing.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018a). Deep contextualized word representations, *CoRR* **abs/1802.05365**.
**URL:** *http://arxiv.org/abs/1802.05365*

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018b). Deep contextualized word representations, *CoRR* **abs/1802.05365**.
**URL:** *http://arxiv.org/abs/1802.05365*

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018c). Deep contextualized word representations, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Association for Computational Linguistics, pp. 2227–2237.
**URL:** *http://aclweb.org/anthology/N18-1202*

Pilehvar, M. T. and Collier, N. (2016). De-conflated semantic representations, *CoRR* **abs/1608.01961**.
**URL:** *http://arxiv.org/abs/1608.01961*

Pinter, Y., Guthrie, R. and Eisenstein, J. (2017). Mimicking word embeddings using subword rnns, *CoRR* **abs/1707.06961**.
**URL:** *http://arxiv.org/abs/1707.06961*

Qin, P., Xu, W. and Guo, J. (2016). An empirical convolutional neural network approach for semantic relation classification, *Neurocomput.* **190**(C): 1–9.
**URL:** *https://doi.org/10.1016/j.neucom.2015.12.091*

Qu, Z., Song, X., Zheng, S., Wang, X., Song, X. and Li, Z. (2018). Improved bayes method based on tf-idf feature and grade factor feature for chinese information classification, *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)* pp. 677–680.

Quinlan, J. (1987). Simplifying decision trees, *International Journal of Man-Machine Studies* **27**(3): 221 – 234.
**URL:** *http://www.sciencedirect.com/science/article/pii/S0020737387800536*

Quinlan, J. R. (1986). Induction of decision trees, *Mach. Learn.* **1**(1): 81–106.
**URL:** *http://dx.doi.org/10.1023/A:1022643204877*

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2018). Language models are unsupervised multitask learners.
**URL:** *https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf*

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Ren, Y., Zhang, Y., Zhang, M. and Ji, D. (2016). Context-sensitive twitter sentiment classification using neural network, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, AAAI Press, pp. 215–221.
**URL:** *http://dl.acm.org/citation.cfm?id=3015812.3015844*

Reuter, J., Pereira-Martins, J. and Kalita, J. (2016). Segmenting twitter hashtags, *International Journal on Natural Language Computing* **5**: 23–36.

Rezaeinia, S. M., Ghodsi, A. and Rahmani, R. (2017a). Improving the accuracy of pre-trained word embeddings for sentiment analysis, *CoRR* **abs/1711.08609**.
**URL:** *http://arxiv.org/abs/1711.08609*

Rezaeinia, S. M., Ghodsi, A. and Rahmani, R. (2017b). Improving the accuracy of pre-trained word embeddings for sentiment analysis, *CoRR* **abs/1711.08609**.
**URL:** *http://arxiv.org/abs/1711.08609*

Rezaeinia, S. M., Ghodsi, A. and Rahmani, R. (2017c). Improving the accuracy of pre-trained word embeddings for sentiment analysis, *CoRR* **abs/1711.08609**.
**URL:** *http://arxiv.org/abs/1711.08609*

Sa, P. K., Bakshi, S., Hatzilygeroudis, I. K. and Sahoo, M. N. (2018). *Recent Findings in Intelligent Computing Techniques: Proceedings of the 5th*

*ICACNI 2017, Volume 3*, 1st edn, Springer Publishing Company, Incorporated.

Saif, H., Andres, M. F., He, Y. and Alani, H. (2013). Evaluation datasets for twitter sentiment analysis: A survey and a new dataset, the sts-gold, *ESSEM@AI*IA*.

Saif, H., He, Y. and Alani, H. (2012). Alleviating data sparsity for twitter sentiment analysis., *in* M. Rowe, M. Stankovic and A.-S. Dadzie (eds), *MSM*, Vol. 838 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 2–9.
**URL:** *http://dblp.uni-trier.de/db/conf/msm/msm2012.htmlSaifHA12*

Saloot, M. A., Idris, N., Shuib, N. L. M., Raj, R. G. and Aw, A. (2015). Toward tweets normalization using maximum entropy, *NUT@IJCNLP*.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Scherer, D., Müller, A. and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition, *in* K. Diamantaras, W. Duch and L. S. Iliadis (eds), *Artificial Neural Networks – ICANN 2010*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 92–101.

Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks, *Trans. Sig. Proc.* **45**(11): 2673–2681.
**URL:** *http://dx.doi.org/10.1109/78.650093*

Sebastiani, F. (2002). Machine learning in automated text categorization, *ACM Comput. Surv.* **34**(1): 1–47.
**URL:** *http://doi.acm.org/10.1145/505282.505283*

Sebastiani, F. (2005). Text categorization, *Encyclopedia of Database Technologies and Applications*.

Seungil, Ding, D., Canini, K., Pfeifer, J. and Gupta, M. (2017).

Deep Lattice Networks and Partial Monotonic Functions, *arXiv e-prints* p. arXiv:1709.06680.

Severyn, A. and Moschitti, A. (2015). Twitter sentiment analysis with deep convolutional neural networks, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, ACM, New York, NY, USA, pp. 959–962.
**URL:** *http://doi.acm.org/10.1145/2766462.2767830*

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J. and Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using model parallelism.

Silva, N. F. F. D., Coletta, L. F. S. and Hruschka, E. R. (2016). A survey and comparative study of tweet sentiment analysis via semi-supervised learning, *ACM Comput. Surv.* **49**(1): 15:1–15:26.
**URL:** *http://doi.acm.org/10.1145/2932708*

Singh, T. and Kumari, M. (2016). Role of text pre-processing in twitter sentiment analysis.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A. and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank, *EMNLP* **1631**: 1631–1642.

Soheily-Khah, S., Marteau, P.-F. and Béchet, N. (2017). Intrusion detection in network systems through hybrid supervised and unsupervised mining process- a detailed case study on the iscx benchmark dataset -.

Sparck Jones, K. (1988). Document retrieval systems, Taylor Graham Publishing, London, UK, UK, chapter A Statistical Interpretation of Term Specificity and Its Application in Retrieval, pp. 132–142.
**URL:** *http://dl.acm.org/citation.cfm?id=106765.106782*

Speer, R., Chin, J. and Havasi, C. (2016). Conceptnet 5.5: An open multilingual graph of general knowledge, *CoRR* **abs/1612.03975**.
**URL:** *http://arxiv.org/abs/1612.03975*

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research* **15**: 1929–1958.
**URL:** *http://jmlr.org/papers/v15/srivastava14a.html*

Sutskever, I., Martens, J. and Hinton, G. (2011). Generating text with recurrent neural networks, *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, Omnipress, USA, pp. 1017–1024.
**URL:** *http://dl.acm.org/citation.cfm?id=3104482.3104610*

Suttles, J. and Ide, N. (2013). Distant supervision for emotion classification with discrete binary values, *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 2*, CICLing'13, Springer-Verlag, Berlin, Heidelberg, pp. 121–136.
**URL:** *http://dx.doi.org/10.1007/978-3-642-37256-8*$_1$1

Symeonidis, S., Effrosynidis, D. and Arampatzis, A. (2018). A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis, *Expert Systems with Applications* **110**: 298 – 310.
**URL:** *http://www.sciencedirect.com/science/article/pii/S0957417418303683*

Snchez-Mirabal, P., Ruano Torres, Y., Hernndez Alvarado, S., Gutirrez, Y., Montoyo, A. and Muoz, R. (2014). Umcc$_d$*lsi* : $Sentimentanalysisintwitterusingpoliritylexiconsandtweetsimilarity, pp.$ 727−−731.

Tai, K. S., Socher, R. and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks, *ACL*.

Takeda, H. and Takefuji, Y. (2015). Enhancing named entity recognition in twitter messages using entity linking, *NUT@IJCNLP*.

Tang, D., Qin, B., Wei, F., Dong, L., Liu, T. and Zhou, M. (2015). A joint segmentation and classification framework for sentence level sentiment classification, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **23**(11): 1750–61.

Tang, D., Wei, F., Qin, B., Yang, N., Liu, T. and Zhou, M. (2016a). Sentiment embeddings with applications to sentiment analysis, *IEEE Transactions on Knowledge and Data Engineering* **28**(2): 496–509.

Tang, D., Wei, F., Qin, B., Yang, N., Liu, T. and Zhou, M. (2016b). Sentiment embeddings with applications to sentiment analysis, *IEEE Trans. on Knowl. and Data Eng.* **28**(2): 496–509.
**URL:** *http://dx.doi.org/10.1109/TKDE.2015.2489653*

Tang, D., Wei, F., Qin, B., Zhou, M. and Liu, T. (2014). Building large-scale twitter-specific sentiment lexicon : A representation learning approach, *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, Dublin City University and Association for Computational Linguistics, pp. 172–182.
**URL:** *http://aclweb.org/anthology/C14-1018*

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T. and Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Baltimore, Maryland, pp. 1555–1565.
**URL:** *https://www.aclweb.org/anthology/P14-1146*

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D. and Kappas, A. (2010). Sentiment strength detection in short informal text, *J. Am. Soc. Inf. Sci. Technol.* **61**(12): 2544–2558.
**URL:** *https://doi.org/10.1002/asi.v61:12*

Uysal, A. K. and Günal, S. (2014). The impact of preprocessing on text classification, *Inf. Process. Manage.* **50**: 104–112.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017). Attention is all you need, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, Curran Associates Inc., USA, pp. 6000–6010.
**URL:** *http://dl.acm.org/citation.cfm?id=3295222.3295349*

Wallace, B. (2017). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Asian Federation of Natural Language Processing, Taipei, Taiwan, pp. 253–263.

Wang, J., Wang, Z., Zhang, D. and Yan, J. (2017). Combining knowledge with deep convolutional neural networks for short text classification, *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, AAAI Press, pp. 2915–2921.
**URL:** *http://dl.acm.org/citation.cfm?id=3172077.3172295*

Wang, W., Bi, B., Yan, M., Wu, C., Bao, Z., Xia, J., Peng, L. and Si, L. (2019). Structbert: Incorporating language structures into pre-training for deep language understanding.

Wang, W., Chen, L., Thirunarayan, K. and Sheth, A. P. (2012). Harnessing twitter "big data" for automatic emotion identification, *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing* pp. 587–592.

Wang, Y., Huang, M., Zhu, X. and Zhao, L. (2016). Attention-based LSTM for aspect-level sentiment classification, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Austin, Texas, pp. 606–615.
**URL:** *https://www.aclweb.org/anthology/D16-1058*

Wang, Y., Khardon, R. and Protopapas, P. (2012). NONPARAMETRIC BAYESIAN ESTIMATION OF PERIODIC LIGHT CURVES, *The Astrophysical Journal* **756**(1): 67.