

# SNET: a Statistical Normalisation Method for Twitter

---

by

Phavanh Sosamphan

A thesis submitted for the degree of  
Master of Computing

Department of Computing and Information Technology  
Unitec Institute of Technology  
Auckland, New Zealand

March 2016



## Abstract

One of the major problems in the era of big data use is how to ‘clean’ the vast amount of data on the Internet, particularly data in the micro-blog website Twitter. Twitter enables people to connect with their friends, colleagues, or even new people who they have never met before. Twitter, one of the world’s biggest social media networks, has around 316 million users, and 500 million tweets posted per day (Twitter, 2016). Undoubtedly, social media networks create huge opportunities in helping businesses build relationships with customers, gain more insights into their customers, and deliver more value to them. Despite all the advantages of Twitter use, comments – called tweets - posted on social media networks may not be all that useful if they contain irrelevant and incomprehensible information, therefore making it difficult to analyse. Tweets are commonly written in ‘ill-forms’, such as abbreviations, repeated characters, and misspelled words. These ‘noisy tweets’ become text normalisation challenges in terms of selecting the proper methods to detect and convert them into the most accurate English sentences. There are several existing text cleaning techniques which are proposed to solve the issues, however they possess some limitations and still do not achieve good results overall. In this research, our aim is to propose the SNET, a statistical normalisation method for cleaning noisy tweets at character-level (which contain abbreviations, repeated letters, and misspelled words) that combines different techniques to achieve more accurate and clean data.

To clean noisy tweets, existing techniques are evaluated in order to find the best solution by combining techniques so as to solve all problems with high accuracy. This research proposes that abbreviations are converted to their standard form by using abbreviations dictionary lookup, while repeated characters are normalised by the Natural Language Toolkit (NLTK) platform and a dictionary based approach. Besides the NLTK, the edit distance algorithm is also utilised as a means of solving misspelling problems, while “Enchant” dictionary can be used to access the spell checking library. Furthermore, existing models, such as a spell corrector, can be deployed for conversion purposes, while text cleanser is advanced as superior for comparing the SNET with a baseline model. With experiments on a Twitter sample dataset, our results show that the SNET satisfies 88% accuracy in the Bilingual Evaluation Understudy (BLEU) score and 7% in the word error rate (WER) score, both of

which are better than the baseline model. Devising such a method to clean tweets can make a great contribution in terms of its adoption in brand sentiment analysis or opinion mining, political analysis, and other applications seeking to make sound predictions.

## **Acknowledgements**

In the first place, I would like to take this opportunity to express my sincerest gratitude to my principal supervisor, Dr. Sira Yongchareon, for his professional support, dedication and guidance as well as treasured comments during the research. His knowledge and patience brought out the best in me, in this way leading me beyond mainstream study and onto the pathway of an academic researcher. He has been an endless source of incentive throughout this thesis. I could not have asked for a better supervisor.

Furthermore, I would like to express thanks to my associate supervisor, Dr. Veronica Liesaputra, for her contributions through valuable comments as well as motivational reinforcement from the initial stage through to the completion of this thesis. Last but not least, I do thank my second associate supervisor, Dr. Mahsa Mohaghegh, for her immense support and guidance, especially for the time she put into offering many critical pieces of advice on how to solve difficult issues and improve the quality of this research. Again, thank you so much to my three supervisors for being there for me. I also thank all the Unitec staff I met during my study as well as all of my lecturers and classmates.

I would like to express my sincere appreciation to the New Zealand Aid Programme (NZ Aid) for enabling me to have the great opportunity of experiencing and studying in New Zealand and, in particular, at Unitec Institute of Technology. Without this support, this research may not have been completed. Additionally, I appreciated having useful discussions with friends who have been supportive in terms of both academic and personal life while studying in New Zealand – Pavel Marunin, Kittiya Boriboune, and Ketmany Phimmathat – you guys are such wonderful people.

Leaving the best for last, I wish to express my gratitude to my beloved parents, Chanthavone and Thongiane Sosamphan, who are always there with their motivating words, strength and unconditional love whenever I suffer any misfortune and have to manage harsh circumstances. I would like to thank Thongvone Sosamphan – my beloved sister. She is my great inspiration and so she makes me feel I can push myself beyond any perceived limitations. I dedicate this thesis to all the people who I have mentioned above.

# Table of Contents

Abstract .....	ii
Declaration .....	iv
Acknowledgements .....	v
List of Figures .....	ix
List of Tables .....	x
List of Abbreviations .....	xi
Chapter 1: Introduction .....	1
1.1. Background of the study .....	2
1.2. Objective of the Research .....	4
1.2.1. Problem Statement .....	4
1.2.2. Hypothesis .....	6
1.3. Research Contributions .....	6
1.4. Thesis Structure .....	8
Chapter 2: Literature Review .....	9
2.1. Informal Texts .....	10
2.2. Noisy Texts .....	11
2.3. Text Normalisation .....	13
2.4. Problem Classification .....	14
2.4.1. Problems at a character-level .....	14
2.4.1.1. Misspellings .....	14
2.4.1.2. Abbreviations .....	18
2.4.1.3. Repeated characters .....	20
2.4.1.4. OOV words .....	20
2.4.2. Problems at a Sentence-level .....	25
2.4.2.1. Ungrammatical sentences .....	25
2.4.2.2. Missing words and punctuation issues .....	26
2.5. Summary of Existing Normalisation Models at Character-level and Sentence-level .....	27
2.6. Evaluation Approach .....	32
Chapter 3: Normalisation Techniques .....	36
3.1. Normalisation Setup .....	37

3.2. Data Collection .....	38
3.3. Preparing of Collected Tweets.....	39
3.4. The Use of English Dictionaries for Text Normalisation .....	43
3.5. Types of Noisy Tweets .....	43
3.6. Tokenising Text Algorithm .....	46
3.7. Individual Techniques for Solving Each Noisy Problem .....	47
3.7.1. Techniques for Removing Repeated Characters .....	49
3.7.2. Techniques for Detecting Abbreviations .....	50
3.7.3. Techniques for Correcting Misspelled Words .....	50
3.7.4. Normalisation Process.....	52
3.8. Statistical Normalisation Method for Twitter (SNET) .....	56
3.9. Chapter Summary .....	57
Chapter 4: Experimental Results and Discussion .....	58
4.1. Experimental Setup.....	58
4.2. Evaluation Metrics .....	60
4.2.1. BLEU Metric.....	60
4.2.2. WER Metric .....	61
4.2.3. Paired t-tests .....	62
4.2.4. Run-Time Efficiency.....	63
4.3. Evaluation Results .....	63
4.3.1. Results from Individual Techniques .....	64
4.3.1.1. Removing repeated characters.....	64
4.3.1.2. Detecting abbreviations .....	66
4.3.1.3. Correcting misspelled words .....	67
4.3.2. Results from Combination of Techniques.....	69
4.3.2.1. The combination of RRC → DAB → SC techniques .....	70
4.3.2.2. The combination of RRC → SC → DAB techniques .....	72
4.3.2.3. The combination of DAB → RRC → SC techniques .....	74
4.3.2.4. The combination of DAB → SC → RRC techniques .....	76
4.3.2.5. The combination of SC → RRC → DAB techniques .....	78
4.3.2.6. The combination of SC → DAB → RRC techniques .....	79
4.3.3. Baseline Model (Text Cleanser (TC)).....	81

4.4. Discussion.....	83
4.5. Limitations of Research .....	87
4.6. Chapter Summary .....	87
Chapter 5: Conclusion and Recommendations .....	89
5.1. Summary of the Research Objective .....	89
5.2. Significance of Final Findings.....	91
5.3. Summary and Future work .....	92
References.....	95
Appendix - Twitter Data Collection .....	104

## List of Figures

Figure 3.1: The general framework of normalisation steps. ....	38
Figure 3.2: The summary of Twitter data collection. ....	39
Figure 3.3: A sample tweet. ....	39
Figure 3.4: A sample tweet after removing HTML characters. ....	40
Figure 3.5: A sample tweet after decoding into UTF-8 format. ....	40
Figure 3.6: A sample tweet after removing an emoticon. ....	41
Figure 3.7: A sample tweet after splitting concatenated words. ....	41
Figure 3.8: A sample tweet after removing a URL. ....	42
Figure 3.9: An example of tokenising a given sentence into word-level. ....	46
Figure 3.10: A motivation tweet used as a sample input in testing each cleaning technique... ..	47
Figure 3.11: The flowchart of normalising repeated letters. ....	53
Figure 3.12: The flowchart of normalising abbreviations. ....	54
Figure 3.13: The flowchart of normalising misspelled words. ....	55
Figure 4.1: The normalised outputs from RRC1, RRC2, and RRC3. ....	65
Figure 4.2: The normalised outputs from DAB1 and DAB2. ....	66
Figure 4.3: The normalised outputs from SC1, SC2, and SC3. ....	68
Figure 4.4: The normalised outputs from TC. ....	82

## List of Tables

Table 2.1: Summary of normalisation techniques addressing the misspelling of words.....	27
Table 2.2: Summary of normalisation techniques addressing abbreviations.....	28
Table 2.3: Summary of normalisation techniques addressing repeated characters. ....	29
Table 2.4: Summary of normalisation techniques addressing OOV words.....	30
Table 2.5: Summary of normalisation techniques addressing noisy text at sentence-level.....	31
Table 3.1: Occurrence frequency of various informal characteristics in English tweets. ....	44
Table 3.2: The division of noisy tweets found in the annotated dataset. ....	45
Table 3.3: Existing cleaning techniques experimented in solving each problem. ✓ means solve and ✗ means not solve. ....	48
Table 4.1: The evaluation results of three techniques from elimination of repeated characters. ....	65
Table 4.2: The evaluation results of two techniques from detection of abbreviations. ....	67
Table 4.3: The evaluation results of three techniques from correction of misspelled words. ..	68
Table 4.4: The evaluation results of the combination of RRC, DAB, and SC techniques. ....	70
Table 4.5: The evaluation results of the combination of RRC, SC, and DAB techniques. ....	72
Table 4.6: The evaluation results of the combination of DAB, RRC, and SC techniques. ....	74
Table 4.7: The evaluation results of the combination of DAB, SC, and RRC techniques. ....	76
Table 4.8: The evaluation results of the combination of SC, RRC, and DAB techniques. ....	78
Table 4.9: The evaluation results of the combination of SC, DAB, and RRC techniques. ....	80
Table 4.10: The evaluation results of Text Cleanser developed by Gouws et al. (2011) as the baseline model. ....	82
Table 4.11: The group of the best combinations from six groups. ....	83
Table 4.12: The comparison between baseline model and a proposed normalisation model...	84
Table 4.13: The comparison of normalised output between the TC and The SNET.....	85
Table 4.14: The combination of the SNET with the TC.....	86

## List of Abbreviations

ACL	Association for Computational Linguistics
ACM	Association for Computing Machinery
API	Application Programming Interface
ASR	Automatic Speech Recognition
BLEU	Bilingual Evaluation Understudy
CER	Classification Error Rate
CMU	Carnegie Mellon University
CRF	Conditional Random Field
CSV	Comma-separated Values
DAB1	Detecting Abbreviations (technique) 1
DAB2	Detecting Abbreviations (technique) 2
DCRF	Dynamic Conditional Random Files
FSMs	Finite-state Machines
HMM	Hidden Markov Model
HTML	Hyper Text Markup Language
IJDAR	International Journal on Document Analysis and Recognition
IV	In-Vocabulary
JSON	JavaScript Object Notation
LM	Language Model
ML	Machine Learning
MT	Machine Translation
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
OOV	Out-of-Vocabulary
POS	Part-of-Speech
RRC1	Removing Repeated Character (technique) 1
RRC2	Removing Repeated Character (technique) 2
RRC3	Removing Repeated Character (technique) 3
SC1	Spelling Correction (technique) 1

SC2	Spelling Correction (technique) 2
SC3	Spelling Correction (technique) 3
SER	Sentence Error Rate
SMS	Short Message Service
SMT	Statistical Machine Translation
SNET	Statistical Normalisation Method for Twitter
TC	Text Cleanser
TREC	Text Retrieval Conference
TTS	Text-to-Speech
URLs	Uniform Resource Locators
WER	Word Error Rate

# Chapter 1: Introduction

Social media networks have been playing an increasingly important role to the extent that they are becoming integrated into our everyday life. People around the world are connected through the use of social network sites thus establishing their social presence via online communication. Vast amounts of data are posted and shared. Data generated by social media sites are considered to be valuable sources of information. Hence, researchers in text analytics fields have used online data in their studies. These people have successfully shown that important information, such as flu trends and presidential voting polls, can be gathered from such data (Tumasjan et al., 2010, Conover et al., 2011). However, more than 80% of online data, especially from Twitter, is unstructured and written in ill-formed English such that data users may not understand it very well (Akerkar, 2013; Bridgwater, 2010; Pritchard, 2012). For the text analytics algorithms to be able to successfully extract information from the raw data, the text has to be “clean”, which means that it is written in perfect English. Thus it is our research goal to find the best methodology of normalisation to clean Twitter messages.

This chapter explains the motivation behind this research, the scope of the study and the hypotheses that we evaluated. The contributions of the research to Natural Language Processing (NLP) community and society are discussed before providing an outline of the structure of the thesis.

## **1.1. Background of the study**

“How to clean Twitter data?” is the question that is to be answered in this research. Twitter is a micro-blog where people can openly communicate with each other on a vast array of topics, and this capacity has led Twitter to be the one of the world’s biggest social media sites (Alexa, 2016; Milanovic, 2015; Thelwall, Buckley, & Paltoglou, 2011). A large amount of users regularly “tweet” (the name for the text written in Twitter) their opinions and ideas on Twitter (Cotelo, Cruz, Troyano, & Ortega, 2015). According to the information of Twitter usage in 2015, provided on Twitter website, there are 288 million monthly active users, 500 million Tweets are sent per day, 80% of Twitter active users are on mobile devices, 77% of accounts are outside the U.S., and 33 languages are supported on Twitter (Twitter, 2015b). According to the latest update from Twitter’s official website, the number of monthly active users has increased to 320 million (Twitter, 2016). Moreover, the number of tweets sent is also growing exponentially every year.

Despite the many advantages of social media networks, comments posted may not be that useful insofar as they may be difficult to analyse if they contain irrelevant and unintelligible information. Compared to the size of other online texts (such as comments on Facebook), tweets are much smaller as each tweet must contain less than 140 characters (Saloot, Idris, & Aw, 2014). Because of the restriction on the maximum number of characters that can be sent in a tweet, tweets are commonly written in shorthand. Furthermore, the vast majority of them are generally written using devices like smartphones, so they are composed hastily with no corrections before posting them on the Twitter website (Cotelo et al., 2015). Additionally, tweets are written by users who come from different backgrounds and have their own writing style. Twitter users have built up new types of expressions, words including lexical variations, SMS-style abbreviations, letter repetitions, emoticons, and so forth. The impact of poor data

has been recognised as an issue which needs to be addressed by data analysts and data users alike.

While data cleaning has been a longstanding issue, it has become critical again with the increasing interest in text normalisation in big data. In this study, text normalisation is considered to be similar to data cleaning since it deals with erroneous and inaccurate text or data in order to clean and enhance the quality of that data (Rahm & Do, 2000). Indeed various studies have emphasised at length the significance of efficient and effective methods for handling dirty data (Han, Cook, & Baldwin, 2013; Saloot, Idris, & Mahmud, 2014; Xue, Yin, Davison, & Davison, 2011). Even though this problem has received huge attention over the years in the orthodox database literature, the resultant state-of-the-art methods and approaches still miss the mark concerning an effective solution for cleaning web data (Han et al., 2013). Selection of proper data cleaning methods and techniques out of various alternatives has ended up being a difficult task as it depends on several variables, such as the bias of a given data source (Chawade, Alexandersson, & Levander, 2014) and the lack of domain knowledge (Fayyad, 1996; Luebbers, Grimmer, & Jarke, 2003).

Similarly, organising the order of the different data cleaning methods involved during the data cleaning process is a challenging task (Peng, 2008). The difficulty here is how to enhance the accuracy, effectiveness and efficiency of the data cleaning algorithms all at the same time. Generally there are some trade-offs between accuracy and efficiency. The most accurate technique usually requires a large amount of time to clean the data. Another challenge is how to enhance the degree of automation during the process of data cleaning. For example, we would like to apply advanced techniques as much as possible to increase the speed of the cleaning process while dealing with a large amount of noise texts at a given time without changing the primary purpose. As a concrete example, automated data cleaning is important. It not only reduces the time-consuming human resource input, but also reduces the cost of data cleaning both of which are significant.

We consider the above-mentioned background, and challenges in normalising noisy data to constitute the motivation for this research. We want to design and construct normalisation models that can be applied to text normalisation across different types of noisy tweets. This is

a challenging task because tweets are written in freestyles and exhibit different types of ill-formed text and informal characteristics. Motivated by studying the existing text normalisation methods, we have set up the objective of the research which is explained in detail in the following sections.

## **1.2. Objective of the Research**

The objective of this research is to propose a means of cleaning noisy tweets as much as possible by designing and constructing normalisation models. We aim for a general framework that can be applied to text normalisation across different types of noisy tweets with high accuracy. Thus, studying existing normalisation techniques and finding the best combination of the techniques to convert noisy tweets into proper English text are our strategic research priorities. To accomplish the objective, we will refer to individual instances of abbreviations, repeated characters, and unconventional spellings collectively as “noisy tweets”. We formulate our problem statement and two hypotheses in the following sections.

### **1.2.1. Problem Statement**

After studying existing work on text normalisation, it can be seen that each method is usually designed to address a specific problem occurring in relation to noisy texts. For example, Out-of-Vocabulary (OOV) words and abbreviations are problems have been the focus of most attempts for which to find solutions (Han & Baldwin, 2011; Palmer & Ostendorf, 2005; Pennell & Liu, 2011; Pennell & Lui, 2011; Yang & Eisenstein, 2013). Even though there are many methods that could tackle these problems, some noisy texts are still not identified and normalised due to misspelling and no context features being accessible for extraction. In order to convert noisy tweets into correct English sentences, problems such as misspellings, abbreviations, repeated letters, and the combination of them should be considered. Therefore, in this thesis we propose

***A normalisation method to detect noisy tweets and convert them into accurate English sentences***

The term “normalisation method” in this study refers to the use of statistics. Normalisation could have a variety of meanings (Dodge, 2006). However, we define normalisation as statistical models that are used to clean and modify ill-formed tweets, and transform them so they are more clean and consistent. Theoretically, statistics has been based on the idea that a statistical model is a parametric model of probability measures (McCullagh, 2002). In other words, a statistical model is usually indicated by mathematical equations involving either random or non-random variables (Daintith, 2004). The use of a statistical model will assist us to generate statistical hypothesis tests.

When the statistical keyword is applied using any method, it is known as a statistical method. The statistical method is intended to be applicable to all the instances of a class rather than to any particular instance, that means it belongs to the class rather than object of the class (Grunwald, Lindsay, & Zorn, 1998; Olsen & Willis, 1996). In addition, a statistical method can be invoked without the requirement for creating an instance of a class (Martin, 2009). The statistical method will receive all information from its argument without having to be an object and running on an instance. For instance, if there is the “Abbreviation” class, the method will use a statistical function to check a given abbreviation with the abbreviations “look-up” or dictionary.

In order to produce a correct English sentence that does not contain misspellings, repeated letters, abbreviations or OOV words, we will propose to use a normalisation method that utilises different techniques to correct a tweet at the character level. For example, we will study techniques proposed by Contractor, Faruque, and Subramaniam (2010), Han and Baldwin (2011), Pennell and Lui (2011), Brody and Diakopoulos (2011), Gouws, Hovy, and Metzler (2011), Xue et al. (2011), Liu, Weng, and Jiang (2012), Norvig (2012), Hassan and Menezes (2013), Saloot, Idris, and Mahmud (2014) and so forth. We will use two criteria - accuracy and run-time efficiency - for the evaluation of our proposed normalisation method, the SNET, against the existing methods. Accuracy can be evaluated based on the BLEU (Bilingual Evaluation Understudy) score and the WER (Word Error Rate) values. Run-time efficiency is measured by the time spent in cleaning noisy tweets.

### 1.2.2. Hypothesis

In this section, we set up and explain our hypotheses regarding the problem statement discussed in the previous section. Our two hypotheses are discussed in detail as follows:

**Hypothesis 1 (H1):** The combination of the best techniques for each problem will create the best normalisation method.

**Hypothesis 2 (H2):** The SNET will produce the most accurate English sentences from noisy tweets with better time efficiency.

To prove Hypothesis 1, firstly, we will present solutions for cleaning each type of noisy tweets (repeated characters, abbreviations, and misspelled words) by implementing each individual technique and carrying out an experiment to assess the performance of the technique. These experiments will assist us to find the technique that best solves each problem. Then, we will find the best combination of techniques that best solve all the problems. The best combination of techniques is considered as our proposed normalisation model. We will name our proposed method as the SNET, which stands for Statistical Normalisation Method for Twitter. To prove Hypothesis 2, we will conduct experiments to compare the performance of the SNET with the existing baseline model developed by Gouws et al. (2011). According to the two hypotheses, we aim to discover a normalisation method that accurately detects and converts noisy tweets into correct English sentences with no misspelling, no repeated characters, and no abbreviations. In addition, we also expect that our model achieves better run-time efficiency.

### 1.3. Research Contributions

This thesis presents a comprehensive study of the normalisation of noisy data in Twitter and proposes the best normalisation method for cleaning noisy tweets as much as possible. The efficiency of data cleaning method will be demonstrated in their ability to clean dirty tweets at word-level in cases such as abbreviations, misspelled words and repeated letters. This method will be proposed by combining the existing methods so they are different from traditional data cleaning strategies that focus on the detection of individual or attribute problems. Furthermore, the associations between data entities will be considered to identify the different combinations

of techniques that can work best to solve noisy problems in tweets with better accuracy and performance than the existing approaches. The result of this research and proposed model will contribute to expanding knowledge and furthering research in the field.

Besides exploration of various techniques for the proposing method for normalising noisy tweets, an annotated corpus of informal tweets will be developed. Hence, the corpus will be made publicly available to assist further research. The informal text could be transformed into the most likely formal standard English sentence by using information gathered from our corpus to develop a normalisation system for the same domain like our study. Furthermore, the future study can evaluate its combination of different systems based on our evaluation results of combined models. This examination will visualise the improvement of the model's performance.

Data cleaning is a crucial part of text pre-processing; clean English tweets generated by the SNET could contribute significantly to the use of clean data in data analysis before producing the most accurate and reliable outcome. In terms of the contribution to society, clean tweets generated by our model will be truly valuable for the determination of public opinion and promotional campaigns (Cotelo et al., 2015). Furthermore, the analysis of Twitter can be implemented in some applications, such as brand sentiment analysis (Ghiassi, Skinner, & Zimbra, 2013; Mostafa, 2013), political analysis (Conover et al., 2011; Tumasjan, Sprenger, Sandner, & Welpe, 2010), and user profiling for market analysis (Ikeda, Hattori, Ono, Asoh, & Higashino, 2013). The research work therefore makes a significant contribution to both present and future studies in the area of text normalisation especially for social media text. This research work not only provides an alternative approach to text normalisation, but also contributes to expanding knowledge among data cleaning researchers and those in related areas.

## 1.4. Thesis Structure

To provide a clear pathway to follow the unfoldings of this research, the thesis is organised into five chapters as outlined below.

**Chapter 2** critically reviews a wide range of literature around the topic of text normalisation and data cleaning. Firstly, the definitions of “informal text”, “noisy text” and “text normalisation” are provided. Secondly, research exclusively related to dirty data type classifications are extracted from the literature and reviewed. Thirdly, existing text normalisation methods and approaches are studied and compared in detail. The chapter concludes by reviewing the evaluation metrics used in the existing text normalisation works. The issues identified from existing works help us to conceptualise the hypotheses for this research.

**Chapter 3** presents the methodology and framework of this study through the normalisation setup. The methods of data collection and preparation are described in detail, and are followed by the experiments involving both existing and proposed normalisation methods in order to prove our hypotheses and address the problem statement.

**Chapter 4** discusses the evaluation results of normalisation models. The results of each experiment are detailed in this chapter and it is shown that the proposed model retains the most appealing features of the existing text normalisation methods while being able to improve the accuracy of text normalisation on a Twitter dataset.

**Chapter 5** draws conclusions from the major findings of the research, also discusses the potential for future research in the field and offers recommendations for improving cleaning data.

## Chapter 2: Literature Review

Existing normalisation works are crucial for advancing this research. They not only provide a variety of data cleaning approaches, but also encourage the researcher to conduct a study of alternative ways of text normalisation which produce better accuracy and performance. Hence, this chapter presents necessary background and existing research work around text normalisation and data cleaning. The aim of this chapter is to understand the field of text normalisation including how the existing works and our work are related. Another aim of the literature review is to study both the advantages and disadvantages of relevant works in order to identify gaps or opportunities which can be addressed by our research.

With the rapid growth of social media networking content, research in text normalisation has grown considerably over the last decade, where a typical problem is around finding ways to convert non-standard “tokens” in informal texts into standard words (Pennell & Liu, 2014). Normalisation is considered to be one of the data cleaning techniques to detect and eliminate noisy texts—referred to as “tweets” in this research. Data cleaning of a large dataset from Twitter needs to be processed as accurately and efficiently as possible. Detection and elimination of noisy tweets is one of the major research areas in data mining, and several

existing methods are available for data cleaning, including text normalisation. Yet we argue that the existing methods do not achieve high accuracy, and some of them are costly and complex. In addition, some of the methods are well suited only for particular types of errors or noisy tweets.

The literature review is structured as follows. Section 2.1 discusses informal texts; Section 2.2 explains noisy texts; Section 2.3 defines the meaning of text normalisation; Section 2.4 presents problem classification; Section 2.5 summarises the existing normalisation models; and Section 2.6 discussed evaluation approach.

## **2.1. Informal Texts**

In recent years, informal texts have turned into an undeniably prominent topic in normalisation research. Research on informal domains and noisy texts are encouraged at many academic conferences, such as the Association for Computational Linguistics (ACL) and the Association for Computing Machinery (ACM) which oversees the digital library. Both ACL and ACM have lately run numerous workshops and held sessions for paper presentations for those working with text data, including SMS messages, social media texts, and micro-blogs (Ramage, Dumais, & Liebling, 2010). According to the record of the International Journal on Document Analysis and Recognition (IJ DAR) in 2007, 2009 and 2011, a variety of work has been done ranging from revealing false online companions to subject mining and the use of informal language. Furthermore, the Text Retrieval Conference (TREC) has established a website that is used specifically for tracking their workshops, generating topic discussions, summarising papers, and presenting a variety of other issues and tasks related to website data and sentiment analysis. It is not possible to provide herein a comprehensive list of research examples; however we do discuss a variety of existing work on normalising informal text.

A relatively new line of research involves analysing texts from micro-blogging websites, in particular the most well-known one, Twitter (<http://www.twitter.com>). Researchers have carried out a micro-blog and web search for comparison purpose (Teevan, Ramage, & Morris, 2011), studied the topics displayed in individual Twitter user streams (Ramage, Dumais, & Liebling, 2010), as well as summarised the trending topics on the Twitter site in general (Liu,

Weng, Wang, & Liu, 2011; O'Connor, Krieger, & Ahn, 2010; Sharifi, Hutton, & Kalita, 2010). Research has also engaged in detecting and analysing Twitter users' sentiment with respect to wide-ranging topics (Barbosa & Feng, 2010; Bollen, Mao, & Pepe, 2011; Davidov, Tsur, & Rappoport, 2010; Diakopoulos & Shamma, 2010; Pak & Paroubek, 2010; Tumasjan et al., 2010).

Additionally, corpora based on Twitter status messages have been developing over recent years, and can be linked to a dataset of SMS messages and Twitter. The widely used Twitter collection is the Edinburgh Twitter Corpus, which consists of more than 97 million Twitter status messages collected in a standardised format by Petrovic, Osborne, and Lavrenko (2010). This Twitter corpus can be used in both general and specific terms but, due to its size, its entirety is not possible to be annotated. On the other hand, the Edinburgh Twitter Corpus is a valuable resource for researching topics in this domain.

## **2.2. Noisy Texts**

Whilst there is agreement about the data quality meaning of "fit for purpose", there is no agreement about the meaning of noise. Manago and Kodratoff (1987, p. 351) stated that "noise is present when a knowledge base does not truly reflect the environment we want to learn from". They demonstrated that the reasons for noise lead experts to construct inaccurate models. By definition, noise is a lack of information, and the wrong or unreliable data. The term "unreliable data" is interesting because the data is not incorrect, but rather "unreliable" for use in analysis.

The stored electronic communication that cannot be classified appropriately by a data mining software is the definition of noisy text (Rouse, 2012). The character of noisy text in an electronic document is defined by an inconsistency between the symbols and letters in the Hyper Text Markup Language (HTML) code and the intended meaning created by the author (Knoblock, Lopresti, Roy, & Subramaniam, 2007; Lopresti, Roy, Schulz, & Subramaniam, 2011). Noisy texts can be caused by the use of acronyms, abbreviations, idiomatic expressions and specific jargon. Noisy texts are mostly prevalent in the unstructured texts found in chat conversations, blog posts, Short Message Service (SMS) texts, and discussion threads

(Subramaniam, Roy, Faruque, & Negi, 2009). Furthermore, noisy texts can be caused by typographical errors, poor spelling and punctuation, poor speech recognition programs, and poor translation from Optical Character Recognition (OCR) (Rouse, 2012).

The above definitions of noisy texts are additionally interesting due to the fact that they do not include the issue of accuracy in data. Hence, their meaning of noise includes any case, and this could pose problems for machine learning or other Natural Language Processing (NLP) scholars. Noisy texts can be found widely in online communications, especially on Twitter. Despite Twitter enabling people to be more connected than ever, tweets posted on the Twitter website may not be that useful if they contain irrelevant and incomprehensible information to the point that they are difficult to analyse. This is likely since Tweets, restricted by limitation on the number of characters, are commonly ill-formed since they are typically written using abbreviations, misspelled words and repeated characters (Aw, Zhang, Xiao, & Su, 2006; Pennell & Liu, 2011). Compared with the texts from another social media sites such as Facebook, tweets are smaller in size as each tweet must contain less than 140 characters (Saloot, Idris, & Aw, 2014). The vast majority of these tweets are typically typing by using electronic devices like smartphones, so they are composed speedily without correction before being posted on the Twitter (Cotelo et al., 2015). Furthermore, tweets are written by users who are from different backgrounds and they are free to write in their own unique style. Consequently, Twitter users have built up new types of expressions, words including lexical variations, SMS-style abbreviation, letters repetitions, emoticons, and so forth.

The impact of ill-formed texts has been recognised as one of the issues needing to be addressed by data analysts. Noisy texts present a challenge for those who wish to analyse micro-blog data like tweets and so need a proper data cleaning technique. This is the reason for proposing the SNET, a statistical normalisation method for Twitter in this study. Noisy tweets containing abbreviations, misspelled words and repeated characters may not be understood and processed by traditional NLP applications, such as sentiment analysis, part-of-speech tagging and machine translation, if they are not converted into their standard form first (Aw et al., 2006; Saloot, Idris, & Aw, 2014). Noisy tweets need to be normalised in order to be cleaned. Clean tweets can then be presented as offering a wide range of benefits in

business. For example, cleaned texts can provide powerful information which, in turns, contributes to cutting costs and maximising profits (MacEwen, 2009).

### **2.3. Text Normalisation**

Text normalisation is usually the initial step taken before feeding data to NLP applications (Pennell & Liu, 2011). Normalisation is the procedure of arranging, analysing, and cleaning text data to raise the efficiency for data utilisation and sharing. Normalisation typically consists of data structuring and refinement, repetition and error elimination, and standardisation (Sproat et al., 2001). In addition, normalising informal text data is imperative for language processing tasks, for example, sentiment and emotion detection, information retrieval, and summarisation, which are presently receiving a great deal of attention for informal domains. Existing normalisation research studies mostly used the data from online sources to test their normalisation models since they contained the largest amount of ill-formed words. Therefore, it is important to normalise social media texts before applying the standard NLP techniques. For example, text normalisation is essential for constructing Text-to-Speech (TTS) systems; non-standard words in the social media texts will be normalised based on their pronunciation (Beaufort, Roekhaut, Cougnon, & Fairon, 2010).

For constructing the SNET, we aim for a general framework that can be applied to text normalisation across different types of noisy tweets with minimal disadvantageous impact. The goal is to save time and resources. This is a challenging task because tweets are written in free styles that exhibit different ill-formed and informal characteristics. For example, the use of symbols that look similar is one type of abbreviation. “\$hop” is an example abbreviation, “\$” is used instead of “s”. This research is informed by some existing normalisation works such as a model-based finite-state framework for text normalisation (Beaufort et al., 2010), a character-level approach for normalising SMS abbreviation (Pennell & Liu, 2011), a noisy-channel approach for the normalization of informal text (Pennell & Liu, 2014), text cleanser framework (Gouws et al., 2011), and spelling corrector (Norvig, 2012). These approaches and frameworks will play a significant role in this research in terms of finding the proper cleaning techniques and proposing a hybrid model for normalising, and so transforming a noisy tweet into an accurate sentence. Our normalisation task is detecting informal text that is

complicated by the large amount of misspelled words, repeated characters, and abbreviations. Furthermore, the aim of this study is normalising noisy tweets at word-level, since we believe that tokenising noisy tweets in an individual word will allow the normalising model to handle each problem with high accuracy, and generate the best output from the system. Hence, existing works that are related to our research will be critically analysed in the following paragraphs.

## **2.4. Problem Classification**

Existing works in the area of text normalisation is discussed in this section. Based on the literature, we identified character-level and sentence-level problems in text normalisation. In this thesis, we only address the character-level normalisation problem. We calculated that normalising noisy tweets at both character and sentence levels would be difficult task and could not be done under the limitations of time and research resources. We consider that existing work in normalising noisy texts at a sentence-level will be useful in terms of expanding our knowledge as well as providing an important base from which to build our future work. The studies related to this research are described in detail in the following sections.

### **2.4.1. Problems at a character-level**

Existing text normalisation and data cleaning methods involved in this study will be discussed based on the noises at the character-level including misspellings, abbreviations, out-of-vocabulary (OOV) words, and repeated and extra letters in words (Bangalore, Murdock, & Riccardi, 2002; Brody & Diakopoulos, 2011; Choudhury et al., 2007; Han & Baldwin, 2011). Each method will be described below under the four types of noisy problems.

#### **2.4.1.1. Misspellings**

In general, Short Messages Service (SMS) and Twitter can contain massive misspellings of words, intentionally and unintentionally on the part of the writer. These words may create some difficulty in data analysis. In order to deal with misspelling words in text language,

Choudhury et al. (2007) worked on the character-level transformations for SMS normalisation by applying a Hidden Markov Model (HMM). HMM is the model that helps to assign certain observations/states to an amount of variables with the highest probability (Blunsom, 2004; Eddy, 1996; Rabiner & Juang, 1986). Choudhury et al. (2007) dealt with both phoneme and grapheme information by tagging HMM through the supervised noisy channel model. The tags got “corrupted” into words, the “noise” was removed, and the original tags were restored. Their method achieved 89% in top-1 accuracy on decoding text language words into standard forms, but transposition errors and self-loop letters, such as “firend” and “funnyyyy” respectively, still need to be improved.

Meanwhile, Whitelaw, Hutchinson, Chung, and Ellis (2009) focused solely on normalising unintentional spelling errors generated from the web by employing the noisy channel model based on orthographic distance. The noisy channel model is considered as an effective model to conceptualise several processes in NLP, especially for tasks involving corrections such as spelling and speech recognition (Kernighan, Church, & Gale, 1990; Nelken, 2012). For example, the form of spelling errors adds some noise inadvertently. Hence, the task of the noisy channel model is to remove the noise. This system could autocorrect misspelled words when there was high confidence. The total error rate of human typed data was 10.8%, while this system achieved only 3.8%. Furthermore, an n-gram model was used in this system as well. An n-gram model is that which is used to estimate the probability (P) of each word given prior context based on the number of the relative frequency of word sequences (Brown, Desouza, Mercer, Pietra, & Lai, 1992; Fink, 2014). An n-gram model uses only N-1 words of prior context. For example, a 1-gram sequence is referred to as an “Unigram” (P (phone)), a 2-gram sequence is a “Bigram” (P (phone | call)), and a 3-gram sequence is a (P (phone | your cell)). When researchers combined an n-gram model component in their system, real-word substitutions such as grammatical errors and word usages could be detected and corrected.

Work by Xue et al. (2011) demonstrated that most of the major issues in misspellings in micro-text like Twitter and SMS are phonetic spelling, emotional emphasis, and popular acronyms. These types of misspelled words could be normalised by adopting the noisy channel framework as a multi-channel model, where each model is used to capture the form of lexicon variants based on four factors: phonetic, orthographic, acronym expansion and

contextual factors. Comparing the total accuracy scores with existing algorithms, namely Aspell (92%) and Moses (94%), their model achieved 96% in normalising Twitter dataset. However, Han et al. (2013) pointed out that it is challenging to balance the use of various channel models in one approach.

In 2012, Liu et al. aimed to increase the accuracy in top- $n$  normalisation by proposing a broad-coverage normalisation system to tackle both the intentional misspellings and the intentionally-created noisy tokens. A top- $n$  evaluation is the measurement used to demonstrate the improvement of performance in normalising (Nigro, 2007). In Liu et al.'s work, human annotations (visual priming), character-level labelling (enhanced letter transformation), and the Jazzy spell checker (Idzelis, 2005) were integrated into the system. This combined system received 64.36% in top-1 accuracy and 91.75% in top-20 accuracy. After re-ranking, the researchers were able to promote the accurate normalisation to the top-1 position. Nevertheless, the visual prime was time-consuming. Meanwhile, Han, Cook, and Baldwin (2012) argued that Liu et al. (2012) had assumed lexical variant detection used in both their work and the previous works (Gouws et al., 2011; Han & Baldwin, 2011) on lexical normalisation was perfect.

Spelling corrector, developed by Norvig (2012), describes the first steps in creating a Google-style spelling corrector that will take something like "Speling", recognise its closet word and replies "Did you mean Spelling?". The algorithm of spelling corrector is defining the conditional probability of a word given based on the edit distance (Damerau-Levenshtein distance algorithm) by finding the dictionary entries with smallest edit distance (deletes + transposes +replaces + inserts) from the query term. For example, if the edit distance is 0 then the term is spelled correctly. On the other hand, if the edit stance is  $\leq 2$  the dictionary term is used as a spelling suggestion. With the proposed edit distance algorithm, spelling corrector achieved 90% of total accuracy at the processing speed of at least 10 words per second.

Mapa et al. (2012) worked on correcting misspelled words in social media. Their approach depended on the spell correction and dictionary lookup, which was used to determine tweets containing incorrectly spelled and slang words in the formal version. Python was utilised for the advancement of the system by drawing on the NLTK library. NLTK is a main stage for

structuring Python projects to work with human language data. It gives interfaces that are easy to use for more than 50 corpora and lexical resources, for example, WordNet, together with a suite of text processing libraries for tokenisation, classification, tagging, parsing, and wrappers for modern quality NLP libraries (NLTK, 2015). Mapa et al. (2012) used a regular expression method to get rid of needless entities such as Uniform Resource Locators (URLs), hash-tags, and emoticons from the tweet to be normalised. A regular expression is a sequence of characters and symbols defining a search pattern (Lawson, 2005). It is mainly used in matching pattern with strings, for example, the “find and replace” operation (Mitkov, 2005; Python, 2015b). By using spaces and overlooking the symbols of punctuation, they tokenised the tweets to break them into tokens. The tokens were then tagged with a Part-Of-Speech (POS) tag and after that were moved to a comparator function where a token was mapped and compared with the words-list appearing in the PyEnchant and Names dictionaries. PyEnchant is a spell checking library for Python. It is used to identify the given words whether or not they are Out of Vocabulary. PyEnchant is the combination of the fundamental Enchant library’s with all the functionalities of an object-oriented interface and the flexibility of Python (Kelly, 2015). Unfortunately, this work did not provide the results of deploying the method in terms of the actual percentage for the accuracy. The authors simply claimed that their mapping process based on spelling correction would check for spelling errors and addressed misspelled word with a fair amount of accuracy.

Dutta, Saha, Banerjee, and Naskar (2015) addressed the issue of text normalisation, a frequently ignored problem in NLP, especially in social media text written by two or more languages (code-mixing). The objective of this work was to correct spelling errors in English in code-mixed text that contains English words and Romanised words (those translated from another language). The authors handled misspelled English words by using the noisy channel model of spelling correction. They also used the spell checker model for tackling wordplay, phonetic variations and contracted words. Meanwhile, they also focused on solving language identification at word-level. They employed a conditional random field (CRF) based machine learning (ML) for creating a structured prediction model to identify language in code-mixed social medial text. Lafferty, McCallum, and Pereira (2001) defined a CRF as a framework that is used for structuring probabilistic models to label and parse sequence data such as natural language text. A CRF is used for encoding the relationship between constructing consistent

interpretations and observations (A. McCallum, 2002; Wallach, 2004). For example, A CRF usually found applications in name-entity recognition (Settles, 2004), Part-of-Speech Tagging (Chen, 2012), and other object recognition works. By using the techniques mentioned above, Dutta et al. (2015) spell checker achieved 69.43% accuracy on the corrected English words, while their language identification achieved 90.5% accuracy.

#### **2.4.1.2. Abbreviations**

Abbreviations are commonly found in SMS and micro-texts like Twitter so, unknott surprisingly, there are many research studies focused on this problem. Cook and Stevenson (2009) extended Choudhury et al. (2007)'s model to build an unsupervised noisy channel model which used likelihood models for detecting common types of abbreviations, and combined the language model (LM) in order to select standard English word with the top probability. The LM is considered as a statistical model that has been used for estimating the natural language's distribution as correctly as possible (Ponte & Croft, 1998). For example, the LM will calculate a probability distribution or  $P(s)$  that is an over-string  $S$  by reflecting how often a string  $S$  appears in a given sentence. By using the same data set as Choudhury et al. (2007), their unsupervised noisy channel model acquired 59.40 % in top-1 accuracy and 87.80% in top-20 accuracy. However, this unsupervised model has limited applicability to real-word normalisation tasks because it did not attempt to differentiate such tokens from other types of OOV tokens.

Kobus, Yvon, and Damnati (2008) examined the idea that they could use phonetic symbols to represent French text messages. They then mapped characters in the abbreviation to phonemes as the output of the automatic speech recognition (ASR) system by incorporating speech recognition into the machine translation (MT) metaphor. The combination of the MT-like and ASR-like systems revealed a significant improvement by decreasing the word error rate (WER) from 12.26% to 10.82%. However, this approach required an aligned parallel corpus of noisy texts for training, which was difficult to acquire (Contractor et al., 2010). Although our research does not consider normalising French texts, we have discovered an alternative solution for detecting and converting abbreviations into their formal version from this approach. Furthermore, this approach could be adapted in some way to normalise English tweets.

Contractor et al. (2010) automatically created a noisy list of word-abbreviation pairs by using heuristics, then employed a statistical machine translation (SMT) model as an unsupervised technique to train on word-abbreviation pairs and scan for the best clean sentence. On top of the use of the SMT model, string edit distance was used to identify the most closely-akin candidate orthographic forms and then the LM was used to decode the text message. As compared to an unprocessed noisy text and sentences from human generated reference text, results generated from the system reported significantly higher BLEU scores (53.90%) than unprocessed noisy text (40.96%), and achieved 10% lower WER than the unprocessed sentences.

Pennell and Liu (2011) introduced a two-phase method using the MT system in order to normalise SMS by expanding abbreviations found in informal text. A character-level MT system was used in the first phase for generating possible hypotheses for each abbreviation without context, and a word-level LM (Shannon (1948)) with a score model were used in the second phase to choose a hypothesis in context for refining the abbreviation normalisation. Their abbreviation model yielded improvements in the overall error rate (6.67%) as compared to the use of LM alone (10.64%). Although the system brought reasonable results for the detection-based abbreviations, the researchers only tested their model on abbreviations which had been assumed by them in the first instance. Hence, the system might not perform well when applied to other types of abbreviations.

By extending the work of Pennell and Liu (2011), instead of translating a non-standard word to standard word by using character-based translation (single-step MT), Li and Liu (2012) proposed a two-stage approach using the MT model for normalising abbreviations and non-standard words in Twitter messages and SMS. Abbreviations were firstly translated to phonetic sequences (their possible pronunciation) using a normalising approach similar to that of (Kobus et al., 2008). Then phonetic sequences were translated back to In-Vocabulary (IV) words by using a dictionary to eliminate words that were not in the dictionary and kept N-best candidates. In order to enhance their proposed model, Li and Liu (2012) combined character-based translation with a two-step MT model. Compared with the one-step MT, two-stage MT

and Pennell and Liu (2011) model on the same dataset, their combined system received 83.11% accuracy in the top-20 coverage.

#### **2.4.1.3. Repeated characters**

Another common phenomenon in Twitter is a sentiment expressed through lengthening words that repeat a single letter in one word. Brody and Diakopoulos (2011) introduced a simple approach to deal with repeated letters before detecting sentiment in Twitter. The general concept informing their experiment is detecting lengthened words and associating them with a canonical form. They assumed that the canonical form of a word is the most frequent form identified by the U.S. Standard English dictionary in Aspell Unix utility (Atkinson, 2004). Only 2.63% of non-standard words were not recognised by the dictionary. The rest (97%) were recognised, which indicated that the strategy of selecting the most frequent form as the canonical one is highly accurate and reliable.

Roy, Dhar, Bhattacharjee, and Das (2013) proposed an algorithm for the pre-processing of repeated letters and punctuation to transform them to their most appropriate form for sentiment analysis and other NLP tools. Saloot, Idris, and Mahmud (2014) introduced an algorithm for eliminating repeated letters from Malay tweets by basing them on patterns setup. Extra letters were eliminated when a token was detected as a non-standard word by tagging with IV words and a normalised token label. After a token with repeated letters was converted to word patterns, the regular expression module was used as a pattern finder to determine whether a token fitted into the patterns. Then repeated letters were deleted based on the match pattern. Although the three different methods mentioned above did not provide the actual results of their system's accuracy in terms of handling repeated letters, we believe that we can utilise them in our study.

#### **2.4.1.4. OOV words**

Identifying an OOV word is challenging and certainly not straightforward in text normalisation (Cotelo et al., 2015). Generally, an OOV word is characterised as a word that does not exist in the vocabulary of a dictionary or a given framework (Hassan & Menezes, 2013). In order to address this problem, Beaufort et al. (2010) received good results on

normalising French SMS by combining the noisy channel model with weighted finite-state machines (FSMs). This hybrid approach shared similarities with both the MT method and spelling correction in order to rewrite rules of OOV lexicons normalisation. Even though the system was efficient, given the 83% score in the BLEU and 9.3% in the WER, the complexity of the phonetic alphabet in French language resulted in a disappointing sentence error rate (SER) score of 65.07%. Although our research does not focus on normalising French texts, we have learned of another way to detect and convert OOV words into their accurate version. Furthermore, the approach could be adapted in some way to normalise English tweets.

Han and Baldwin (2011) worked with classifiers by using the noisy channel model to pre-identify all OOV words as being ill-formed words or not, and then converted those ill-formed OOV words based on the phoneme and grapheme similarity into standard forms. Their combined method (dictionary look-up, context support, and word similarity) yielded better results in both F-score (75.30%) and BLEU score (93.40%). Despite this, the researchers found some limitations when handling highly noisy tweets (mostly containing misspelled words and special symbols). Their approach did not perform well because of the lack of enough contexts for expanding and converting ill-formed words. Furthermore, their approach also required a large corpus annotated by humans for training and tuning (Liu et al., 2012).

Some research has supported lexical normalisation by building a normalisation dictionary. Clark and Araki (2011), for example, created a dictionary that contains 1043 phrases and allowed end-users to add more. Meanwhile, Gouws et al. (2011) constructed a novel, unsupervised exception dictionary by using automatically paired OOV words and IV words. By using some similarity functions, namely a phonetic edit distance-based function, a heuristic string similarity function, and a subsequence overlap function, OOV words were identified based on the list of IV words and created the output as a word mesh which contained the most likely clean candidates for each word. Then the model grouped them as a set of confusion lattices for decoding clean output by using an n-gram language model from SRI-LM (Stolcke, 2002). An n-gram language model was used to re-score the probability of the confusion lattices in order to find the most likely clean tokens for decoding cleaning output in terms of their most likely correct English form. This approach reduced around 20% in the WER score over existing state-of-the art approaches, such as a Naïve baseline and IBM-baseline

(Contractor et al., 2010). To evaluate a method's WER accuracy, Gouws et al. (2011) used a Twitter dataset developed by Han et al. (2012), who produced pairs of potential variant and normalisation, and developed a dictionary of lexical variations of known words, which were further positioned by string similarity. This dictionary could improve the lexical normalisation through straightforward string substitution.

Liu et al. (2011) proposed a method called letter transformation (LetterTran) that normalised OOV words into standard English words without a human supervisor. Furthermore, their proposed approach could normalise OOV words without pre-categorising the non-standard words into deletions, insertions, and substitutions. Their proposed LetterTran worked under the noisy channel model for finding the sequence of all standard English words that maximised the confidence of the sequence labelling model. By using the CRF model along with the Carnegie Mellon University (CMU) pronunciation dictionary (Lenzo, n.d.), their LetterTran approach automatically collected a set of noisy training word pairs (which is a pair of OOV words and their dictionary word or their best candidate), and then aligned them at the character-level for training and normalising OOV words to their standard forms. Furthermore, they had combined the Jazzy spell checker (Idzelis, 2005) with their LetterTran model for selecting the best candidate where the LetterTran was not confident. As a result, the combined model received a higher accuracy than the Liu et al.'s (2011) standalone model in normalising OOV words from a tweets dataset. The LetterTran model achieved 59.19 % while the combined model achieved 68.88% in top-1 accuracy and 78.27% in top-3 accuracy. However, their LetterTran model could not normalise the tokens that contain inserting and switching letters (i.e. "possibililty" for "possibility" and "vocaburaly" for "vocabulary"). Moreover, another type of OOV words, such as acronyms, was not considered in this research.

Hassan and Menezes (2013) generated lexicon by proposing a Contextual Graph Random Walks on a large unlabelled text corpus based on constructing the bipartite graph, which implied two groups of node representing context and noisy words respectively. Then a noisy word was linked to its context by edge weight. As a result, pairs of normalised and noisy words were identified as normalisation equivalences. For generating the most likely candidate for OOV words, they used the spell checker with Aspell dictionary. Then they used the Viterbi decoding process, which assisted them to select the best candidate for normalising OOV words

based on an n-gram language model. The Viterbi decoding process or the Viterbi path is a dynamic algorithm used to find the most possible sequence of hidden states (Pennell & Liu, 2011). Hassan and Menezes (2013) used the Viterbi decoder to score the best correction candidates and decoded clean output. Their system achieved a promising result with a 92.43 % score in precision, a 56.4% score in recall, and a 6% score in improvement in translation quality when it was used as a pre-processing step in the MT system.

In Han et al. (2013) work, OOV words were defined by the use of different unorthodox language in Twitter. The method focused on the lexicon normalisation of OOV words based on the dictionary-based normalisation performance. This normalisation technique was performed in an unsupervised manner by considering morphophonemic varieties of words (i.e. “f” in “leaf”, but “v” in the plural “leaves”) and the connection where the words occur. They expanded upon the work of Han et al. (2013) with more comprehensive experimentation and explanation. By comparing the accuracy between a Part-of-Speech (POS) Twitter tagger and a POS Stanford tagger (Klein & Manning, 2003) on both the original and normalised tweets, it could be seen that the normalised tweets in Twitter POS achieved a higher accuracy than POS Stanford from 70% to 94.7%.

Saloot, Idris, Shuib, Raj, and Aw (2015) continued their previous work (Saloot, Idris, & Aw, 2014) by proposing an approach for normalising OOV words based on the maximum entropy model. The maximum entropy model basically refers to the estimation of probability of unknown knowledge or events based on estimating the possibility on the given evidence, and defines them with a statistical model (Berger, Pietra, & Pietra, 1996; Jaynes, 1957). Saloot et al. (2015)’s proposed an approach that could detect OOV words found in tweets by building a dictionary-based module as an unsupervised normalisation system, which was based on candidate generation and candidate selection stages. They firstly generated a set of normalised candidates for each OOV word based on the similarities of lexicon, phoneme, and linguistics. Then the most appropriate candidates were selected by calculating three different probability scores (including lexical dependency, positional indexing and ranked distance) by using a dependency-based frequency feature, positional index, and LM model. After the model parameters obtained the optimal values in the training stage, the model could calculate the final probability value for each candidate and then produce the clean output. By using a large

dataset from Choudhury et al. (2007) and a Singaporean English Twitter corpus in their testing, their approach achieved 83.12% in the BLEU score.

Adedamola, Modupe, and Dehinbo (2015) focused on misspelled words and slang from social media, and considered them as OOV words. They normalised OOV words based on the following steps. They firstly tokenised the string (raw tweet) into tokens based on a regex pattern. Secondly, the output from the tokenising process was sent to the token classifier component. By using the regular expression as the token classifier (Python, 2015b), each token was classified into three categories including a Meta-elements category, an IV category, and an OOV category. The Meta-elements category included hash tags (#apple, #smile), user IDs and mentions (@taylorswift13, @TheGRAMMYs), Retweets (RTs), and punctuation. Each token classified in this category was considered as a normalised token. For classifying OOV words from IV words, they used an Enchant function (enchant\_dict\_check) (Lachowicz, 2010) to check whether each given token appeared in the Aspell dictionary (Atkinson, 2004) by checking the spelling of a token. All OOV words were considered as misspelled words, and then they were corrected by spell corrector as a final step. The spell corrector normalised OOV words by generating the list of candidates and then selecting the most suitable one. The Enchant library combined with edit distance was used to generate the list of suitable IV candidates for OOV words. Then the correct candidate that was most suitable for the OOV word was selected from the list by using the Levenshtein function for calculating the distance between two strings. Their system received 83 % in the BLEU score after normalising tweets containing slang words into the formal versions. However, the researchers used a small dataset of tweets to test their experiments - only 10 tweets with 135 tokens. The size of their dataset means the experimentation is not accurate enough to evaluate the performance of the system.

Based on the existing research looking at character-level problems, we have found challenges as well as opportunities in normalising non-standard words. We have firstly observed that high levels of annotated training data are required for evaluating the performance of their methods. Furthermore, the majority of the above methods are specially designed to handle a specific problem for the normalisation task. For example, the normalisation methods developed by Xue et al. (2011), Norvig (2012), and Mapa et al. (2012) can only handle the problem of misspellings. The methods developed by Contractor et al. (2010), Pennell and Liu

(2011) and Li and Liu (2012) can solely resolve abbreviations, while the methods developed by Brody and Diakopoulos (2011) and Roy et al. (2013) can only handle letters repetition. On the other hand, OOV words normalised by the methods of Han and Baldwin (2011), Liu et al. (2011) and Han et al. (2013) were not clearly defined. Their proposed methods might not handle all non-standard words. Hence, there is still some room to improve the performance of text normalisation at character-level. By combining these existing models it would be possible to accurately normalise all type of non-standard words rather than a single one. Furthermore, the combination of different normalising techniques might provide an alternative approach for text normalising at character-level and perform better than the existing methods by generating the most accurate and clean output.

## **2.4.2. Problems at a Sentence-level**

This section reviews related work that handles noisy texts at a sentence-level, including ungrammatical sentences, missing words and punctuation issues (Aw et al., 2006; Wang & Ng, 2013).

### **2.4.2.1. Ungrammatical sentences**

Incorrect grammar is a sentence-level problem that can be found in both SMS and Twitter. Aw et al. (2006) adopted the SMT model to normalise SMS at the phrase-level; sentences were split to the  $k$  most possible phrases and the grammar was corrected based on standard English rules. The method produced messages that were collated well with manually normalised messages, obtaining the 80.70% BLEU score beating the 69.58% baseline score. Without adjusting the MT model, the accuracy of SMS translation was also improved from 19.26% to 37.70% in the BLEU score. While the performance of this model was good, it still could not handle missing punctuation that affected the MT output. This method also required an aligned parallel corpus of noisy text and sentences for training, which was difficult to acquire (Contractor et al., 2010).

Raybaud, Lavecchia, Langlois, and Smaili (2009) applied confidence measures to detect ungrammatical sentences by estimating a probability of correctness. They relied on several features of confidence measures including an n-gram, backward n-gram LM, and linguistic

features. These models were used to estimate a correctness of lexicon and grammar of the hypothesis sentences. Furthermore, different combinations of an n-gram, backward n-gram LM, and linguistic measures were implemented. Their work's accuracy was evaluated over the task of classification error rate (CER). Their results achieved 30% in CER and 71% in F-measure at a sentence-level.

Kaufmann and Kalita (2010) took a two-step approach to normalising tweets. The first step is to perform pre-processing using syntactic disambiguation to remove noise from tweets as much as possible. Then the processed tweets are fed into an SMT model in order to be converted into correct grammatical English sentences. For Tweets normalisation, combining these two steps allowed the researchers' system to achieve an improvement in the BLEU score (from 67.99% to 79.85%), which is an 18% increase.

#### **2.4.2.2. Missing words and punctuation issues**

Besides the grammatical problem, Wang and Ng (2013) focused on solving missing words and correcting punctuation errors in Chinese social media texts. They proposed a novel beam-search decoder to solve the sentence level problem based on the CRFs for solving missing words and dynamic conditional random files (DCRF) for correcting punctuation. English text was normalised in this study as well. This was achieved by following the same hypothesis for normalising Chinese text but adding some specific hypothesis producers such as re-tokenisation to split informal words, inserting quotations, and time normalisation. These researchers also implemented MT for translating Chinese to English and English to Chinese. The decoder achieved 65.05% for the BLEU score in normalising non-standard English messages to standard Chinese messages and this rose to 66.54% when processing the hypothesis with contextual filtering.

Ling, Dyer, Black, and Trancoso (2013) introduced a data-driven approach to normalise non-standard tweets based on paraphrasing. A corpus of tweets was built and used in parallel with normalising deploying MT methods. Then, noisy sentences were paraphrased by combining the phrase-level and character-level models during decoding. The BLEU score of the original tweets was 19.90%. However, after combining character-level and phrase-level, the score rose to

22.91%. Although the accuracy was not significantly increased, this approach proved that when combining the appropriate methods to solve the right problem, the yield of sentence normalisation could be increased.

According to our observations, the number of research studies handling text normalisation in phrase- or sentence-level is quite small. We assume that normalising noisy text is far more difficult than normalisation at character-level. It requires advanced techniques and approaches to generate the most accurate sentences without non-standard words, grammatical errors, and fragments. Although this research does not focus on solving the sentence-level problem, some of the above approaches have provided resourceful directions for normalising noisy sentences which could be utilised to enhance the proposed normalisation method designed in this study and also in our subsequent research; or other researchers in order to build upon our proposed research.

## 2.5. Summary of Existing Normalisation Models at Character-level and Sentence-level

In order to have a clear picture of the above-mentioned existing normalisation models, this section will briefly summarise each method including its result. By providing an overview of which methods were used and their accuracy results, we are able to make an evaluation as how each method performed.

<b>Misspelled Words</b>		
<b>Authors</b>	<b>Methods/Techniques</b>	<b>Accuracy</b>
Choudhury et al. (2007)	Hidden Markov Model (HMM)	89% in top-1 accuracy
Whitelaw et al. (2009)	The noisy channel model with n-gram language model	3.8% in total error rate
Xue et al. (2011)	Multi-noisy channel framework	96% in total accuracy
Liu et al. (2012)	<ul style="list-style-type: none"> <li>• Visual priming</li> <li>• The enhanced letter transformation</li> <li>• Jazzy spell checker</li> </ul>	64.36% in top-1 accuracy and 91.75% in top-20 accuracy
Norvig (2012)	Edit distance algorithm	90% of total accuracy

Table 2.1: Summary of normalisation techniques addressing the misspelling of words.

Table 2.1 shows a summary of normalisation methods used for correcting misspelled words. Five existing systems are studied to obtain an idea of how recent methods have handled all types of misspelled words. These systems used both different datasets and accuracy metrics to evaluate their method's performance, in general terms the performance of each system is acceptable. The average accuracy result is more than 80%. However, it can be seen that the noisy channel framework, could generate the highest accuracy than the others.

<b>Abbreviations</b>		
<b>Authors</b>	<b>Methods/Techniques</b>	<b>Accuracy</b>
Kobus et al. (2008)	<ul style="list-style-type: none"> <li>• The MT-like system</li> <li>• The ASR-like system</li> </ul>	11% in the WER
Cook and Stevenson (2009)	<ul style="list-style-type: none"> <li>• The noisy channel model</li> <li>• LM</li> <li>• Maximum likelihood estimates (MLEs)</li> </ul>	59.40 % in top-1 accuracy and 87.80% in top-20 accuracy
Contractor et al. (2010)	<ul style="list-style-type: none"> <li>• SMT model</li> <li>• LM</li> </ul>	53.90% in the BLEU score
Pennell and Liu (2011)	<ul style="list-style-type: none"> <li>• The Moses MT system</li> <li>• LM</li> </ul>	6.67% of overall error rate
Li and Liu (2012)	<ul style="list-style-type: none"> <li>• Moses MT system</li> <li>• Giza++</li> <li>• HMM model</li> </ul>	83.11% in top-20 accuracy

Table 2.2: Summary of normalisation techniques addressing abbreviations.

Table 2.2 summarises normalisation methods used for detecting abbreviations. It can be seen that the language model and the machine translation model are widely used for transforming an abbreviation into its standard version. The idea of combining techniques is generally found in many recent works. Kobus et al. (2008) and Contractor et al. (2010) combined two methods and achieved unsatisfactory accuracy while Cook and Stevenson (2009) and Li and Liu (2012) combining more than two methods received better results based on an improved accuracy percentage. Hence, the number of methods used in the combination effect the performance of the system.

Table 2.3 demonstrates methods used for removing repeated characters found in words. Unfortunately, there is only a small amount of published work in this specific area. We only

found two research studies handling this problem by using different techniques. The first approach eliminated repeated characters by counting the length of words and then compared them to the dictionary lookup. Meanwhile, the second approach removed repeated letters by using the regular expression model to identify a word containing repeated letters based on the word patterns. This approach could generate a promising result.

Table 2.4 presents a variety of methods used for detecting OOV words. Systems listed in this table identified OOV words in different ways, thus they handled OOV words with different techniques. For example, some of the systems used more than one technique to address more than one type of noisy words, because they considered abbreviations and misspelling words as OOV words. Furthermore, this table provides a clear picture of research examples using combinations of statistical methods and those that are dictionary-based.

Table 2.5 summarises the methods used for addressing noisy texts at a sentence-level, including ungrammatical sentences, missing words and punctuation. The problems at sentence-level are complicated to handle and require highly advanced techniques. Although some advanced techniques, such as SMT, Moses MT and n-gram LM, are used to deal with a sentence containing grammatical errors, missing words and punctuation, the results from the above systems were still not satisfactory. The unpromising results indicate that there is still some room for improvement in order to increase the accuracy of the models mentioned in table 2.5 and generate a well-formed English sentence that does not contain any grammatical errors.

<b>Repeated Characters</b>		
<b>Authors</b>	<b>Methods/Techniques</b>	<b>Accuracy</b>
Brody and Diakopoulos (2011)	<ul style="list-style-type: none"> <li>• Detecting lengthened words</li> <li>• Transferring them into canonical form</li> </ul>	N/A
Saloot, Idris, and Mahmud (2014)	<ul style="list-style-type: none"> <li>• Regular expression model based on the setup patterns.</li> </ul>	N/A

Table 2.3: Summary of normalisation techniques addressing repeated characters.

OOV Words		
Authors	Methods/Techniques	Accuracy
Beaufort et al. (2010)	<ul style="list-style-type: none"> <li>Finite-state machines (FSMs) (LM + spell checking system)</li> </ul>	9.3% in WER
Liu et al. (2011)	<ul style="list-style-type: none"> <li>The noisy channel model</li> <li>LM</li> <li>Letter transformation + Jazzy spell checker</li> </ul>	68.88% in top-1 accuracy and 78.27% in top-3 accuracy
Han and Baldwin (2011)	<ul style="list-style-type: none"> <li>The noisy channel model as classifiers</li> <li>Dictionary lookup</li> <li>Word similarity</li> <li>Context support modelling.</li> </ul>	93% in the BLEU score and 75% in F-score
Gouws et al. (2011)	<ul style="list-style-type: none"> <li>Exception Dictionary</li> <li>Edit distance-based function</li> <li>Heuristic string similarity function</li> <li>Subsequence overlap function</li> <li>SRI-LM</li> </ul>	20% in WER
Hassan and Menezes (2013)	<ul style="list-style-type: none"> <li>Bipartite Graph</li> <li>n-gram LM</li> </ul>	70.05 % in F-score
Han et al. (2013)	<ul style="list-style-type: none"> <li>Extract (OOV and IV) pairs</li> <li>Re-rank the extracted pair</li> </ul>	94.70% of total accuracy
Saloot et al. (2015)	<ul style="list-style-type: none"> <li>lexical one-edit distance</li> <li>Phonemic generation</li> <li>Blending the previous methods</li> <li>Lexical two-edit distance generation</li> <li>Dictionary translation</li> <li>Heuristic rules</li> <li>LM</li> </ul>	83.12% in the BLEU score
Adedamola et al. (2015)	<ul style="list-style-type: none"> <li>The regular expression model</li> <li>Spell corrector by the Enchant dictionary function + edit distance method</li> </ul>	83 % in the BLEU score

Table 2.4: Summary of normalisation techniques addressing OOV words.

<b>Problems in Sentence-level</b>		
<b>Ungrammatical sentences</b>		
<b>Authors</b>	<b>Methods/Techniques</b>	<b>Accuracy</b>
Aw et al. (2006)	<ul style="list-style-type: none"> <li>• Phrase based SMT</li> </ul>	80.70% in the BLEU score
Raybaud et al. (2009)	<ul style="list-style-type: none"> <li>• An n-gram</li> <li>• Backward n-gram LM</li> <li>• Linguistic features</li> </ul>	30% in error rate and 0.71 in f-measure
Kaufmann and Kalita (2010)	<ul style="list-style-type: none"> <li>• Syntactic disambiguation</li> <li>• SMT</li> </ul>	79.85% in the BLEU score
<b>Missing words and punctuations</b>		
<b>Authors</b>	<b>Methods/Techniques</b>	<b>Accuracy</b>
Wang and Ng (2013)	<ul style="list-style-type: none"> <li>• CRF</li> <li>• DCRF</li> <li>• MT</li> </ul>	66.54% in the BLEU score
Ling et al. (2013)	<ul style="list-style-type: none"> <li>• A data-driven approach based on paraphrasing</li> <li>• Moses MT</li> </ul>	22.91 % in the BLEU score

Table 2.5: Summary of normalisation techniques addressing noisy text at sentence-level.

To summarise, it can be seen that noisy texts can contain either character or sentence-level problems or both and which can be solved using different techniques. However, there is still room for improvement. Most problems found at character-level, with the MT model, the noisy channel model, spell checking model, n-gram LM method, and edit distance as the commonly used methods. In order to accurately handle noisy texts, some researchers propose normalisation models which combining existing models. For example, the combined system developed by Liu et al. (2011) scored 78.27% of n-best accuracy; Li and Liu (2012)'s scored 83.11% of n-best accuracy and the combined system developed by Gouws et al. (2011) obtained 20% in the WER accuracy. According to the normalised results and the accuracy of these existing combined systems, we believe that we can propose combined methods which can yield better accuracy in terms of handling all types of noisy tokens such as abbreviations, misspellings, and repeated letters. Hence, this research will address all types of non-standard words by combining some independent normalisation methods. Instead of relying on a single approach, the combined methods will allow each model to implement its own technique as

much as possible. Each model will be designed to address a specific normalisation problem, thereby increasing the model accuracy. Furthermore, we have observed that some normalisation methods within the high range of accuracy mostly used combined models with complex techniques in normalising non-standard words to standard words. To provide an alternative solution for resolving problems at character-level, we propose the method with the least complexity but which can still produce better accuracy and running-time efficacy without changing main aim of normalisation .

## 2.6. Evaluation Approach

Applying appropriate evaluation criteria and measures is a vital step in evaluating the performance of a proposed method. Currently, there are many evaluation metrics that can be used for measuring the performance of a normalisation system. Most research has used F-measure, Precision, and recall for evaluating the effectiveness of their system (Derczynski et al., 2015; Han & Baldwin, 2011; Han et al., 2012, 2013; Hassan & Menezes, 2013; Liu et al., 2012; Marton & Zitouni, 2014; Whitelaw et al., 2009; Xue et al., 2011; Yang & Eisenstein, 2013; Zhang, Baldwin, Ho, Kimelfeld, & Li, 2013). Other common evaluation metrics are BLEU, WER, SER, and top-n accuracy. The use of these evaluation metrics in existing works will be discussed in detail in the following paragraphs.

**F-measure or F-score** is used to measure an experiment's accuracy. It considers both the Precision and Recall of the examination to compute the score. Both Precision and Recall scores are widely used to evaluate the effectiveness of NLP systems (Melamed, Green, & Turian, 2003; Raghavan, Bollmann, & Jung, 1989). The simple idea of Precision is to identify the percentage of selected items that are correct and Recall refers to the percentage of correct items that are selected (Jizba, 2000). In terms of normalisation evaluation, Precision is calculated by the number of true correct positive results divided by the number of all positive results, while Recall is calculated by true correct positive results divided by the total of the true positive and false negative results. Han and Baldwin (2011), for example, used Precision, Recall, and F-score for classifying output of lexical variant detection. To sum up, these measures provide accurate scores which assist researchers in understanding the overall performance of their systems.

**BLEU** metric is commonly used among researchers (Beaufort et al., 2010; Han & Baldwin, 2011; Han et al., 2013; Marton & Zitouni, 2014; Saloot et al., 2015) who employed the MT model in their normalisation system. Typically, the BLEU metric is an automatised tool designed to evaluate the accuracy of a normalisation model's performance, and it requires several standard references which contain the annotation carried out by humans, for generating a BLEU score (Papineni, Roukos, Ward, & Zhu, 2002). The BLEU score is defined on a scale of 0 to 1, but usually is represented through a percentage value. When a score is closer to 1, the accuracy of the methods' performance is high. Furthermore, some existing normalisation works compared the BLEU score achieved by their system with the baseline BLEU score in order to evaluate the performance of their normalisation system (Aw et al., 2006; Hassan & Menezes, 2013; Kaufmann & Kalita, 2010). Melamed et al. (2003) argued that even though the BLEU score is beneficial in terms of using evaluated accuracy for comparison between different MT model outputs, it may not be a sufficiently informative measure in order to improve the MT system. Another drawback of the BLEU is the lack of inability and synonym matching to detect the orders of multiple correct words (Olive, Christianson, & McCary, 2011). Thus the quality of model may not be defined by the BLEU metric.

**WER** is another evaluation metric that has been widely used to evaluate the performance of text normalisation models (Beaufort et al., 2010; Han et al., 2012, 2013; Marton & Zitouni, 2014; Sproat et al., 2001). The WER is used to score the percentage of words, which are to be deleted, inserted, or replaced in the normalisation in order to acquire the sentence of reference (Thoma, 2013). Gouws et al. (2011) used the WER metric to find the accuracy of selecting the most likely clean tokens for normalising OOV words, while Han et al. (2012) evaluated dictionary-based and normalisation based on the WER accuracy. Despite the fact that the WER is calculated efficiently and is successful in terms of reproducing the same results from the same data, the WER does not consider the syntactic and contextual roles of a word. It only measures an error rate at the surface level (He, Deng, & Acero, 2011). Furthermore, its dependency on the sentences of reference is the main downside because the WER metric considers only one normalised word to be correct based on the nearest reference.

**SER** or Sentence Error Rate is the metric used to score the percentage of a normalised sentence which does not match with an annotated sentence of reference. SER has similar

advantages and disadvantages to the WER (Tomás, Mas, & Casacuberta, 2003). An example of using the SER metric is found in the work of Beaufort et al. (2010). They evaluated the performance of their hybrid normalisation framework in normalising noisy sentences and they found that their system achieved a low SER score. Tomás et al. (2003) also used the SER metric, but for evaluating the accuracy of a machine translation's performance in translating sentences from one language into another language.

**Top- $n$  accuracy** is one of the common evaluation standards used in text normalisation research to consider the correctness of a system based on the correct standard form in the  $n$  most probable standard forms (Cook & Stevenson, 2009; Li & Liu, 2014; Liu et al., 2012; Pennell & Liu, 2011; Pennell & Liu, 2014; Pennell & Liu, 2011). A top- $n$  performance can be evaluated directly based on accuracy metrics like Precision and Recall (Cremonesi, Koren, & Turrin, 2010). This metric is used to perceive what damage to performance occurs because of the extra difficulty of decoding an abbreviation, for example, when there is another abbreviated word nearby. An abbreviation is corrected in top- $n$  if the word is correctly normalised in at least one of the top  $N$  sentences. However, when using this metric to evaluate the model at a score of 100% in top- $n$ , it will not guarantee that a single top- $n$  sentence will be 100% correct (Pennell, Ng, Hansen, & Schweitzer, 2011). For example, one abbreviation might be incorrect in sentence  $i < N$  but correct in sentence  $j < N$ , where the opposite is true of a second abbreviation.

To summarise, automated metrics is required for evaluating advanced normalisation models in order to evaluate performance and quality of normalisation. Review of the above existing evaluation metrics shows that each metric has different advantages and shortcomings in testing the performance of the model. The F-measure can provide an indication as the accuracy of the normalisation that a model will produce while the BLEU can give several insights into how good the accuracy of the normalised output from a model will be. The WER and SER can determine the accuracy of normalisation's performance that based on the top score hypothesis is indeed correct. Last but not least, the top- $n$  accuracy can demonstrate how well the normalisation system determines the accurate standard form. However, each metric can be suitable in different cases. For instance, the F-measure, BLEU, WER and top- $n$  accuracy can be more appropriate than SER in testing normalisation model in character-level. Since these

evaluation metrics will score accuracy by counting the number of words in the normalisation model's output that occur in the reference dataset. However, these metrics will not perform well when using in evaluating normalisation systems that handle noisy texts in sentence-level. SER, thus, can perform well in this case because it scores accuracy based on the number of nonmatching words or sentences in its nearest reference. Additionally, there is a correlation between word and sentence error rate (Evermann, 1999). For example, when the overall performance of normalisation system is poor, it indicates that the correlation of WER and SER is weak. In this case, if the SER is generally near 100% and the WER is changed, there is hardly any impact on the sentence error rate. On the other hand, if the performance of normalisation system is quite good and most sentences have either no or only one or two errors, then the correlations is relatively strong as an additional WER also tends to yield a new SER.

In this study, we will propose the SNET for cleaning noisy texts in character-level. Since the baseline model (Gouws et al., 2011), which will be used to compare the normalisation performance with the SNET, it uses BLEU and WER metric for evaluation. Hence, the BLEU and WER metrics will be used to evaluate the accuracy of the performance of both existing normalisation approaches and SNET as well. F-measure and Top-n accuracy will be used in our future study. Besides the well-known and appropriate evaluation metric in normalisation studies (Contractor et al., 2010; Schlippe, Zhu, Gebhardt, & Schultz, 2010), the BLEU metric has been transformed into an easy-to-use version, namely iBLEU. The iBLEU will not only assist us to handle the large amount of normalised output that will be used to evaluate the performance of a model, but also reduce the time consumed in generating the BLEU score. Additionally, the WER metric is used in this research because its baseline model (Gouws et al., 2011) can be functioning as a means of comparison with the SNET. The WER metric can also be used for testing the method's accuracy. Hence, if both the SNET and the baseline model use the same criteria to evaluate results, then the comparison will be easier. This will also reveal the clear differences between the two approaches. Besides the evaluation of the method's accuracy, we will also evaluate the efficiency of a technique based on its running time. Last but not least, a paired t-test will also be used to find the difference between techniques.

## Chapter 3: Normalisation Techniques

Texts generated in online social media sites such as Twitter are usually very different from “Standard English orthography”. The goal of this research is to propose a normalising method in order to 'normalise' an ill-formed tweet to its most likely correct English representation with the highest accuracy. For example, "u r sooooo gorgeuos 2nite" to "you are so gorgeous tonight". The methodology and general framework to normalise a sample noisy tweet conducted in this research is described in detail in this chapter.

Quantitative methods are used to describe the experimental design and analysis to answer our problem statement defined in Chapter 1. This chapter discusses the methodology that is used in this study. The steps involved in normalising noisy tweets are presented in section 3.1, followed by Twitter data collection, pre-processing, and the knowledge-based approach. The types of noisy tweets are explained in section 3.6. The existing methods that are used in proposing an approach for noisy tweets normalisation are described in section 3.7. Subsequent sections explain how the experiments will be conducted on tweets with various noise problems such as repeated characters, abbreviations, and misspelled words, with further analysis to validate the results from both existing and proposed normalisation methods.

### 3.1. Normalisation Setup

There are 5 major steps in proposing the method for normalising noisy tweets, as outlined in Figure 3.1. A general framework of normalisation steps is as follows:

1. Collection of Twitter data by using Twitter streaming API (Application Programming Interface) and R programming language.
2. Preparing data by removing Hyper Text Markup Language (HTML) characters, Uniform Resource Locators (URLs), punctuation and expression symbols, and decoding them into UTF-8 format. Through this the testing dataset and reference dataset are created.
3. Categorising the types of noisy tweets by using the identifying method discussed in section 3.5.
4. Implementing existing normalisation techniques based on the problems that can be solved, such as repeated characters, abbreviations and misspelled words. Existing techniques are tested in order to find the best technique for each problem. Then, different combinations of each technique are implemented in order to find which combination of techniques can work best in solving all noisy problems.
5. The best combination of techniques is considered the proposed normalisation method. The proposed method is named the SNET, a **S**tatistical **N**ormalisation **M**ethod for **T**witter. To prove the SNET is better in terms of the accuracy and time efficiency, we compare our system/baseline with the text cleaning framework developed by Gouws et al. (2011) by using the same dataset for evaluation.

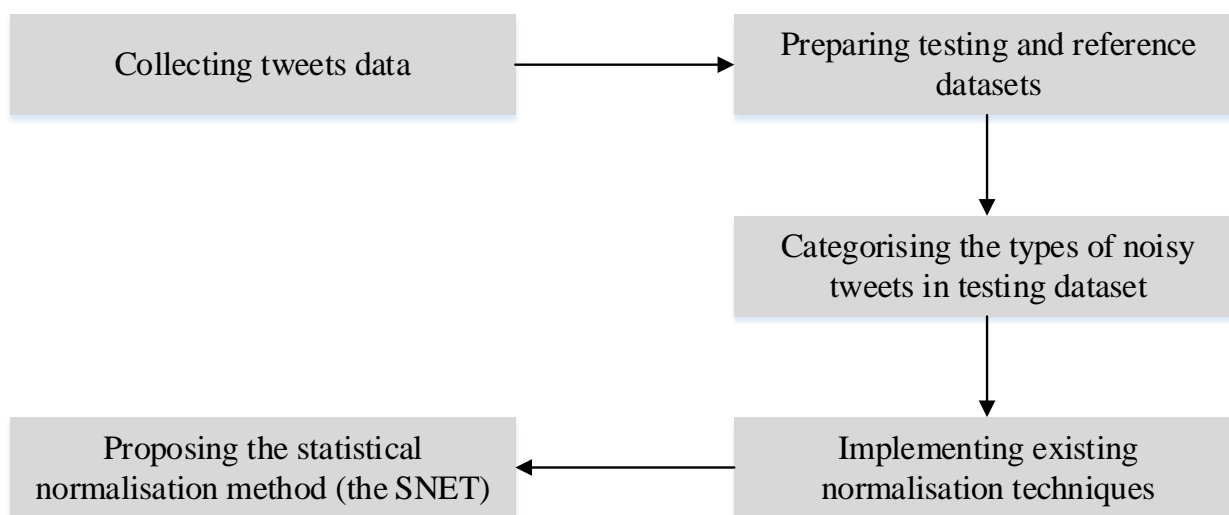


Figure 3.1: The general framework of normalisation steps.

### 3.2. Data Collection

The first application of this research is the collection of the Twitter dataset. At this stage we focus on collecting only English tweets. The diagram shown in Figure 3.2 is the summary of data collection in this research. After gathering a list of famous singers and celebrities (in order to find them on Twitter and save their handles), we use the Twitter Streaming API to collect their tweets and use them as our dataset. Twitter’s API provides a direct approach to search for users and returns results in a JavaScript Object Notation (JSON) format which makes it simple to parse using a Python script (Twitter, 2015a). To connect us to Twitter’s stream, we have to create the Twitter application to obtain API keys for access to Twitter’s API through R commands, to search for users, then download tweets and save them in a comma-separated values (CSV) file. The details of Twitter data collection will be explained in an appendix.

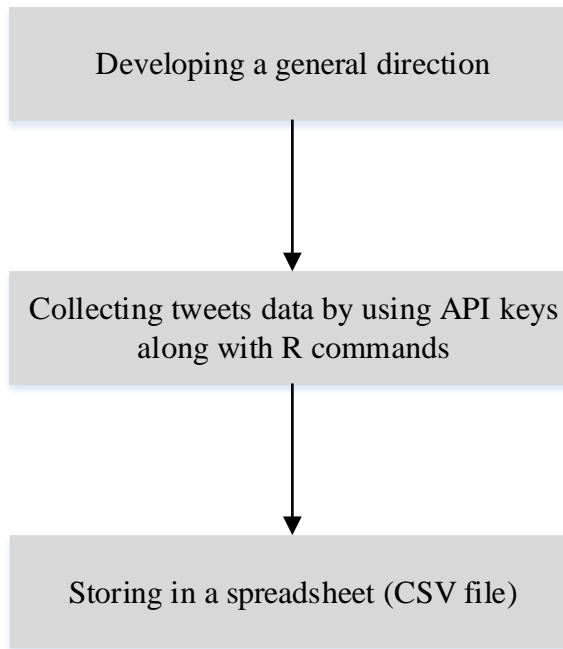


Figure 3.2: The summary of Twitter data collection.

### 3.3. Preparing of Collected Tweets

Before creating the Twitter dataset as a test dataset and a reference dataset for evaluating both existing and proposed models, basic steps of data preparation are deployed. After collecting a sample of tweets, the five steps of data preparation are implemented as follows, using an original tweet as an example.

**An original tweet:** “I luv my &lt;3 iphone & you’re awsm #apple. DisplayIsAwesome, sooo happpppppy 😊 @applenws http://www.apple.com”

Figure 3.3: A sample tweet.

The first step is removing the HTML characters. Most of the data obtained from social media sites typically contains a number of html entities such as &amp; &gt; &lt; and so forth, which is implanted in the original data. Thus, it is necessary to remove these entities. Our approach is to remove them directly using the function `html.unescape` (Python, 2016), a function which

converts these entities to standard HTML sequences. For instance, `&amp;` is converted to “&” and `&lt;` is converted to “<”.

**Snippet:**

```
>>> import html
>>> html.unescape('original_tweet')
```

**Output:**

```
“I luv my <3 iphone & you’re awsm #apple. DisplayIsAwesome, sooo
happppppy 😊 @applenws http://www.apple.com”
```

Figure 3.4: A sample tweet after removing HTML characters.

The second step is decoding data. We transform complex symbols found in data into simple and easier to understand characters. Text data could be subject to different forms of decoding like “Latin”, “UTF8”, and other encoding formats. We consider that, in order to produce data that is readable, it is necessary to keep the complete data in standard encoding format. We encode our tweets in UTF-8 format because it is widely accepted and recommended.

**Snippet:**

```
>>> tweet = original_tweet.decode("utf8").encode('ascii','ignore')
```

**Output:**

```
“I luv my <3 iphone & you’re awsm #apple. DisplayIsAwesome, sooo
happppppy 😊 @applenws http://www.apple.com”
```

Figure 3.5: A sample tweet after decoding into UTF-8 format.

The removal of punctuation and expressions is the third step of basic cleaning. All the punctuation marks should be dealt with according to certain priorities. For Twitter, @usernames and #tags are considered as in-vocabulary, thus they are not removed. Similarly, “:”, “.”, “,”,”?” are important punctuation marks that should be preserved, while others have to be removed. Besides removing unnecessary punctuation, other expressions also need to be eliminated. Tweets, which are usually written as speech transcripts, could contain human expressions such as [crying], [laughing], and other emoticons. Observation has shown that these expressions are usually irrelevant to the content of the speech. Hence, we use a simple regular expression to remove them in this case.

**Output:**

“I luv my <3 iphone & you’re awsm #apple. DisplayIsAwesome, sooo happpppppy @applenws  
http://www.apple.com”

Figure 3.6: A sample tweet after removing an emoticon.

**Snippet:**

```
>>> import re
>>> ans = ""
>>> for a in re.findall('[A-Z][^A-Z]*',"original_tweet"):
    ans+=a.strip()+'
```

**Output:**

“I luv my <3 iphone & you’re awsm #apple. Display Is Awesome, sooo happpppppy @applenws  
http://www.apple.com”

Figure 3.7: A sample tweet after splitting concatenated words.

After removing unwanted punctuation and all types of emoticons, concatenated or run-together words must be split into individual words. When humans in social forums create text

data, it is sometimes totally informal in nature. An enormous number of tweets contain multiple joined or attached words, such as `PlayingInTheClod`, `RainyDay`, and so forth. These attached words reduce the capacity of the normalisation model due to the fact that they will not be recognised by the model. Therefore, these entities need to be split into their ordinary form using straightforward rules and the regex method (Python, 2015b).

**Snippet:**

```
>>> import re
>>> text = re.sub(r"(?:\https?\:\/\/)\S+", "", original_tweet)
```

**Output:**

“I luv my <3 iphone & you’re awsm #apple. Display Is Awesome, sooo happpppppy @applenws”

Figure 3.8: A sample tweet after removing a URL.

The final step is removal of URLs. In order to have the best raw tweets for normalisation, URLs and/or hyperlinks in tweets should be removed. The five steps of preparing raw data mentioned above do not only allow us to get a useful raw dataset for evaluating our models, but they also assist us to identify what types of noisy tweets we are facing. Consequently, identifying problems easily directs us towards selecting the proper solutions and methods for normalising noisy tweets, as well as narrowing down existing normalisation process theories for utilisation in this research.

After preparing Twitter data, two types of datasets are established from 1200 tweets which have been randomly selected from world celebrities’ Twitter account by using techniques mentioned in 3.2, including testing datasets and reference datasets. The main guidelines are to normalise noisy tweets to their corresponding clean tweets in a consistent way according to the evidence in the context. To achieve this we use 1200 tweets, which have not been normalised yet, as “a testing dataset” for evaluating both existing and proposed models. Additionally, we construct a dataset of 1200 tweets which have been manually corrected by a human annotator.

We refer to these correct normalised 1200 tweets as “reference datasets” and use them as a reference list for testing each individual cleaning technique and combined techniques’ BLEU and WER accuracy.

### **3.4. The Use of English Dictionaries for Text Normalisation**

In order to normalise noisy tweets that contain misspelled words and abbreviations, using only a statistical method will not be enough. Hence, normalisation dictionaries for micro-blogs (Han et al., 2012) will be utilised in this research work as well. We have divided the dictionary-based approach for normalising noisy tweets into two types. The first dictionary is the standard English dictionary lookup which is a type-based approach to normalisation of non-standard words that are incorrectly spelt. The “en\_US” dictionary from Aspell library has been used to support spell checking techniques in order to correct misspelled words. Besides the US standard Dictionary, we have created an abbreviation dictionary as well. The abbreviations reference is created by collecting abbreviations and their full versions from reliable online sources such as [www.urbandictionary.com](http://www.urbandictionary.com), [www.noslang.com](http://www.noslang.com), [www.abbreviations.com](http://www.abbreviations.com) and [www.internetslang.com](http://www.internetslang.com).

### **3.5. Types of Noisy Tweets**

After obtaining Twitter data which does not contain URLs, HTML characters, unnecessary punctuations, non-relevant expressions, and is decoded in UTF-8 encoding format, we take a closer look at our data in order to identify the types of noisy tweets. Using the raw data we have, we categorise noisy tweets into three main classifications based on their problems: abbreviations, repeated characters, and misspelled words. Understanding each type of noisy problem assists us in handling each problem efficiently. It not only helps us to know what kind of noisy tweets we are dealing with, but also guides us in applying the correct techniques to each type of problem. Hence, we have reused the type of abbreviations defined by Pennell and Liu (2014) as a guideline and re-categorised them under these three main problems. Each type of noisy text is categorised and briefly defined in the following Table 3.1.

Main Category	Subcategory	Definition	Example
<b>Abbreviations</b>	Acronym	Using the initial letters of words	NASA (National Aeronautics and Space Administration)
	Silent character	Silent 'e' removal	mayb (maybe)
	Deletion	Deleting some letters.	tmro (tomorrow)
	Clipping	Deleting entire syllables	pro (professional)
	General	Keeping a single word, and not covered above	ppl (people)
	Initialisation	Using a first letter of each word	lol (laughing out loud)
	Substitution	Replacing characters with others	2day (today)
	As sound	Reading the token as a word	l8r (later), c (see)
	As character	Pronouncing character's names	ne (any)
	Symbolic	Using symbols that look similar	\$hop (shop)
	Combination	Using two or more of the above.	2sdy (Tuesday)
<b>Repeated Characters</b>	Repetition	Using letters repeated for emphasis	soooooooo (so)
<b>Misspelled Words</b>	Error	Generally caused by a typo	litarature (literature)
	Stylistic	Intentional changing	infruent (influent)
	Insertion	Increasing the Word length	fightting (fighting)
	Location	Removing -ing 'g'	cookin (cooking)
	Swap	Typing letters in wrong positions	friedg (fridge)

Table 3.1: Occurrence frequency of various informal characteristics in English tweets.

Table 3.1 shows the types of noisy words that are commonly seen on Twitter and other social media sites. By using the formation method used by Pennell and Lui (2014) which adopts various forms of noise in different situations, we propose three broad categories of noisy tweets with their subcategories.

The first category is “Abbreviations”. Most abbreviations can be classified by observing the characters alone using our divisions, although uncertainty arises from silent “e” removal. Some abbreviations will remove silent “e” and replace with the character that sounds like “e”. For example, “be” is replaced with “b”. Furthermore, initialisation and acronyms have been considered as abbreviations due to their prevalence as another type of shortening, formed by using only the first letter of each word as a whole word. Some abbreviations are constructed by replacing characters with others that have a similar sound but are written in a different way. The last type of abbreviation is the combination of two or more of all types of abbreviations that are mentioned above, in which is sometimes too difficult to decipher the meaning.

The second category is “Repeated characters.” This type of noisy text is easily identified by our division due to the unexpected repetition of one or two letters in a word. Repeated characters are used when people want to express or emphasise their feeling through writing. The final category is “Misspelled words.” Despite the general mistake of spelling caused by a typo, we also consider the complexities of misspelled words. Insertion is defined when the word length is increased accidentally. In this case, the letter that is inserted requires removal.

However, sometimes words are intentionally spelled incorrectly, so the stylistic problem is used to define this subcategory. According to our Twitter data, the vast majority of misspelled words are of the location problem. This problem defines the removal of silent first or last letters in words. Besides the location problem, swap is commonly found in the Twitter data as well. Swap often results from correct letters in the wrong position (typos).

<b>Total Tokens</b>	<b>Abbreviations</b>	<b>Repeated Characters</b>	<b>Misspelled Words</b>
<b>16.867</b>	489	152	375
<b>Total Noisy Tweets</b>			960

Table 3.2: The division of noisy tweets found in the annotated dataset.

Additionally, some noisy words are considered to be OOV in some existing research (Gouws et al., 2011; Schaaf, 2001) because there are some types of noisy words that cannot be identified. Despite repeated characters and misspelled words, our research also considers

unrecognised noisy words as abbreviations. Furthermore, we consider the types of noisy words mentioned above as the feature set, which is implemented for identifying the potential candidates that require normalisation. All these are the key tasks in implementing existing text normalisation models within the proposed work. In addition, the number of each type of noisy tweets found in the annotated 1200 tweets data, which contains 16.867 tokens, is shown in Table 3.2. It can be seen that the vast majority of noisy tweets are abbreviations. For this reason, each type of noisy tweet is categorised and treated separately in this study.

### 3.6. Tokenising Text Algorithm

During our normalisation, a tokenisation process is necessary. Overcoming the traditional problems using a dictionary-based approach required us to inspect variations in given words that have a potential to be noisy words in a given text (Egorov, Yuryev, & Daraselina, 2004) so that they will be factored out in the matching process while normalising misspelled words and abbreviations. Hence, a tokenisation process will be used to target text and dictionary entries. Tokenisation is splitting or breaking the input text into token sequences or a list of tokens. A token itself means an instance of a sequence of characters found in several documents that are gathered together as a useful document unit for processing (Zitnick, 2013). For example, such a sequence could be as follows: “to”, “much”, “to”, “eat”. The interface of tokenising text is defined by NLTK and called the “TokenizerI” class (Perkins, 2010). This interface can be independently used in any level of tokenisation – such as breaking the text into paragraphs, sentences, words, phonemes, or syllables (Egorov et al., 2004). For this study, we will use word tokenisation based on a regular expression-based tokeniser, which will split text according to whitespace and punctuation. An example of processing word tokenisation is shown as follows.

**Given sentence:** Hello, how are you?

**Tokens:** ['Hello', ',', 'how', 'are', 'you', '?']

Figure 3.9: An example of tokenising a given sentence into word-level.

Combined with the use of regular expression, the tokenisation process is more powerful. The regular expression-based tokeniser will determine how the input text should be split up by

specifying the format of the valid word. We will split punctuation from words in order to reduce the confusion of word identification during normalisation. The sequence of tokens shown in the above example will allow both existing and proposed methods to identify which word in a given sentence is ill-formed and which one is not. Hence, the model will normalise only ill-formed words and return the most likely correct form to the sentence as a normalised sentence.

### 3.7. Individual Techniques for Solving Each Noisy Problem

In this research, we consider three noisy problems, which cause noisy tweets, including repeated characters, abbreviations, and misspelling words. We also consider OOV words, which is one type of noisy words, in order to increase the capacity of SNET to handle uncommon noisy words. As such, several existing techniques are evaluated by our own dataset to find the best technique for solving each problem.

As shown in Table 3.3, we experiment with existing cleaning techniques by using algorithms from the existing model mentioned in chapter 2. In order to explain how each technique works in solving each problem and to demonstrate its cleaning result, the following paragraphs will present the three different issues when solving problems: repeated characters, abbreviations and misspelled words.

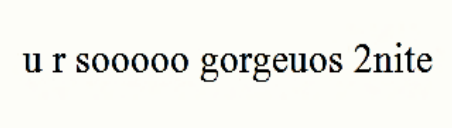
The image shows a tweet with the text "u r soooooo gorgeuos 2nite". The text is centered and displayed in a black serif font against a light yellow background. The tweet contains several examples of noisy text: "u" and "r" are abbreviations, "soooooo" consists of repeated characters, "gorgeuos" is a misspelled word, and "2nite" is an abbreviation.

Figure 3.10: A motivation tweet used as a sample input in testing each cleaning technique.

To visualise how each technique cleans each noisy problem in one tweet, we will use the example shown in Figure 3.10 as a motivation tweet and is also used as a sample input in testing each cleaning technique. A motivation tweet contains examples of three problems that we want to normalise including abbreviations (i.e. “u”, “r”, and “2nite”), repeated characters (i.e. “sooooo”), and a misspelled word (i.e. gorgeuos).

Technique	Abbreviations	Repeated Characters	Misspelled Words
Using sample regular expression function (Kuchling, 2015)	✗	✓	✗
Using the replace() function with a WordNet lookup (Perkins, 2010)	✗	✓	✓
Using regular expression module and replace() method from Perkins (2014) with Enchant library	✗	✓	✓
Using the replace() method to check given abbreviation with Python dictionary (Cooper, 2013; Python, 2015a)	✓	✗	✗
Expanding abbreviations by CSV file replacement and the word_map dictionary function	✓	✗	✗
Constructing spelling correction with PyEnchant(Perkins, 2014)	✗	✗	✓
Constructing a probability model as a train function with the edit distance (Norvig, 2012)	✗	✗	✓
Using spelling correction by TextBlob (Loria, 2015)	✗	✗	✓
a phonetic edit distance-based function with lattice-tool programme of SRILM (Gouws et al., 2011)	✓	✓	✓
Using NLTK library with a regular expression and PyEnchant library for spell corrector (Mapa et al., 2012)	✗	✗	✓

Table 3.3: Existing cleaning techniques experimented in solving each problem. ✓ means solve and ✗ means not solve.

### 3.7.1. Techniques for Removing Repeated Characters

The first technique is eliminating a repeated character using a simple regular expression function (Kuchling, 2015). We name this first technique as a RRC1 (Removing Repeated Character 1) in order to be more easily to remember. The regular expression allows us to delete repeated characters by setting up a pattern, using the (r" (\w)\1+") format, given that a word containing repeated letters should be deleted until only two letters remain. This technique converts "u r sooooo gorgeuos 2nite" to "u r soo gorgeuos 2nite". As can be seen, the repeated letters in "soo" are not completely corrected. Then we use the Python 3 spelling corrector method developed by J. McCallum (2014) based on the spell corrector (Norvig, 2012), for correcting the word after removing repeated characters. Hence, "soo" is corrected to "so".

The second technique for the removal of repeated characters (named RRC2) is developed by Perkins (2010), by using a regular expression to compile the patterns, and then run a *replace()* method to take a single word found in patterns and return a more correct version of that word, with questionable repeating characters removed. By using our motivation tweet in testing this technique, repeated letters in "sooooo" are removed until we are left with "so". However, this technique may go too far and for example, end up changing "good" into "god". To correct this issue, we use the *replace()* function with a WordNet lookup – the NLTK corpus reader and works as a lexical database for the English language (Loper & Bird, 2002; Miller, 1995; Perkins, 2014). This system will stop replacing characters if WordNet recognises the word.

It can be seen that using dictionary lookup after removing repeated characters can generate the most likely correct word; thus we decide to construct a third technique (named RRC3). It is an alternative solution of using dictionary lookup to check spell errors after removing repeated letters, by using Enchant dictionary– a spelling correction API (Lachowicz, 2010) instead of WordNet in the technique. We still use *replace()* method to check Enchant dictionary to see whether the word is valid in the dictionary or not, and then return to its correct version. Testing this technique with our motivation tweet removes repeated characters and returns the most correct word- "u r sooooo gorgeuos 2nite" to "u r so gorgeuos 2nite". Legitimate repeated letter words (such as "good") are unchanged if these words are found in an Enchant dictionary; no character replacement will take place.

### 3.7.2. Techniques for Detecting Abbreviations

The first technique is expanding abbreviations by using only the `replace()` method. We name this first technique as a DAB1 (Detecting Abbreviation 1). A Python string `replace()` method works by returning a copy of string *s* which all occurrences of *old* have been replaced by *new* (Cooper, 2013; Python, 2015a). The `replace()` method will take a single given abbreviation and return its expansion if it is found in the abbreviation lookup. Based on the result of using our motivation tweet for testing the technique, it can be seen that all abbreviations are replaced with the correct version- “u r sooooo gorgeuos 2nite” → “you are sooooo gorgeuos tonight”.

The second technique (named DAB2) is expanding abbreviations by using CSV file replacement. The CSV file is the format of the abbreviation dictionary lookup in this research work. This technique is very similar in operation to the first technique, but it can work as a base class that constructs the `word_map` dictionary from the CSV file format. The given abbreviation will be read line by line in the CSV file and, if it encounters the matching one, it will replace the abbreviation with the corresponding word. The result of using this technique is; “u r sooooo gorgeuos 2nite” → “you are sooooo gorgeuos tonight”.

### 3.7.3. Techniques for Correcting Misspelled Words

The first technique is spelling correction with Enchant (Perkins, 2014). We name this first technique as a SC1 (Spelling Correction 1). A reference to an enchant dictionary is created and then the `replace()` method is used to check whether the given word is recognised by the dictionary or not. Spelling correction is not necessary if the given word is present in the dictionary, and the word is returned. But if the word is not found, it will look up a list of suggestions generated by the less or equal to its edit distance to `max_dist`. The edit distance is the number of character changes necessary to transform the given word into the recommended word. By using motivation as a testing tweet, it can be seen that the “gorgeuos” is misspelled word because “u” and “o” are in the wrong position. This technique can identify this type of misspelled word by changing the positions of “u” and “o” into their correct places. For instance, “gorgeuos” is corrected to “gorgeous”.

The second technique (named SC2) uses a spelling corrector developed by Norvig (2012). This technique uses the concept of  $P(c|w)$  By Bayes' Theorem for using probabilities to find the correction  $c$ , out of all possible corrections, which maximises the probability of  $c$  given the original word  $w$ . The  $P(c)$  will read a text file considered to be the dictionary, which consists of more than one million words. Then the function “train” is used to train a probability model in order to count how many times each word occurs. The function “correct” selects as the set of candidate words (i.e. “seething”, “soothing”, “smoothing”, “something”) which has the direct edit distance to the original word accrued in the dictionary. The candidate set is defined for consideration when the component with the highest  $P(c)$  value is chosen which is estimated by the “NWORDS” model. Finally, the edit distance algorithm is used to enumerate the possible corrections  $c$  (candidate) of  $w$  (a given word) and then returns a set of all words  $c$  that is one edit away from  $w$ . This generates all possible terms with an edit distance  $\leq 2$  (*deletes + transposes + replaces + inserts*) from the query term and searches them in the dictionary. For a word of length  $n$ , an alphabet size  $a$ , an edit distance  $d=1$ , there will be  $n$  deletions,  $n-1$  transpositions,  $a * n$  alterations, and  $a * (n+1)$  insertions, for a total of  $2n+2an+a-1$  terms at search time. With this technique, again “gorgeuos” is corrected to “gorgeous”.

The third technique (named SC3) is spelling correction with TextBlob. TextBlob is a Python library for handling textual data. It gives a straightforward API for undertaking common NLP tasks such as part-of-speech tagging, classification, sentiment analysis, translation, and more (Loria, 2015). TextBlob provides a feature that supports spelling correction based on Peter Norvig’s spelling corrector, as implemented in the pattern library. This model uses the `correct()` method to attempt spelling correction. The given word is detected and listed as a word with its own confidence (word, confidence) tuples of spelling corrections. The model then returns the word with the highest confidence, as a most likely correct word, to the sentence. With this technique, the same result is achieved – “gorgeuos” is corrected to “gorgeous”.

After the experiment of applying each existing cleaning technique to each problem at the first stage, outputs from each technique are compared in order to find the best technique for solving each problem. For example, three techniques of spelling correction are compared based on scores generated from evaluation metrics, which are BLEU and WER. The technique that has

the highest BLEU and the lowest WER scores is considered the best model for correcting misspelled words. When all the techniques mentioned above have been applied, the results obtained from them assist us to identify which technique is the best for each problem. Furthermore, the three best cleaning techniques for solving three problems will be used to prove our first hypothesis, which is whether or not combining the best technique for each problem results in the best overall model.

#### **3.7.4. Normalisation Process**

According to the existing data cleaning techniques mentioned in the previous section, we would like to have a clear vision of how we will normalise each problem based on the existing normalisation methods. Hence, we have designed the general flowcharts of normalising each problem and they will be detailed as follows.

Figure 3.11 presents the process of normalisation of removing repeated letters contained in token through two main classes. Each token will be firstly checked for repeated characters, and it will be ignored when it does not contain repeated letters. For the token that contains repeated letters will be ignored if it is defined as username and hash-tag. Then the token with repeated letters will be normalised by removing all repeated letters and will be finally rechecked by dictionary lookup for correcting normalised token in most likely correct version.

Figure 3.12 demonstrates how we will normalise abbreviations. We design to divide the normalisation process into two classes. The first class will check given token and identify it whether abbreviation or not. Then the token identified by an abbreviation will be normalised by using abbreviation dictionary lookup and generated it into the correct form. If the abbreviation does not accrue in the abbreviation dictionary, it will be ignored and remained its original form.

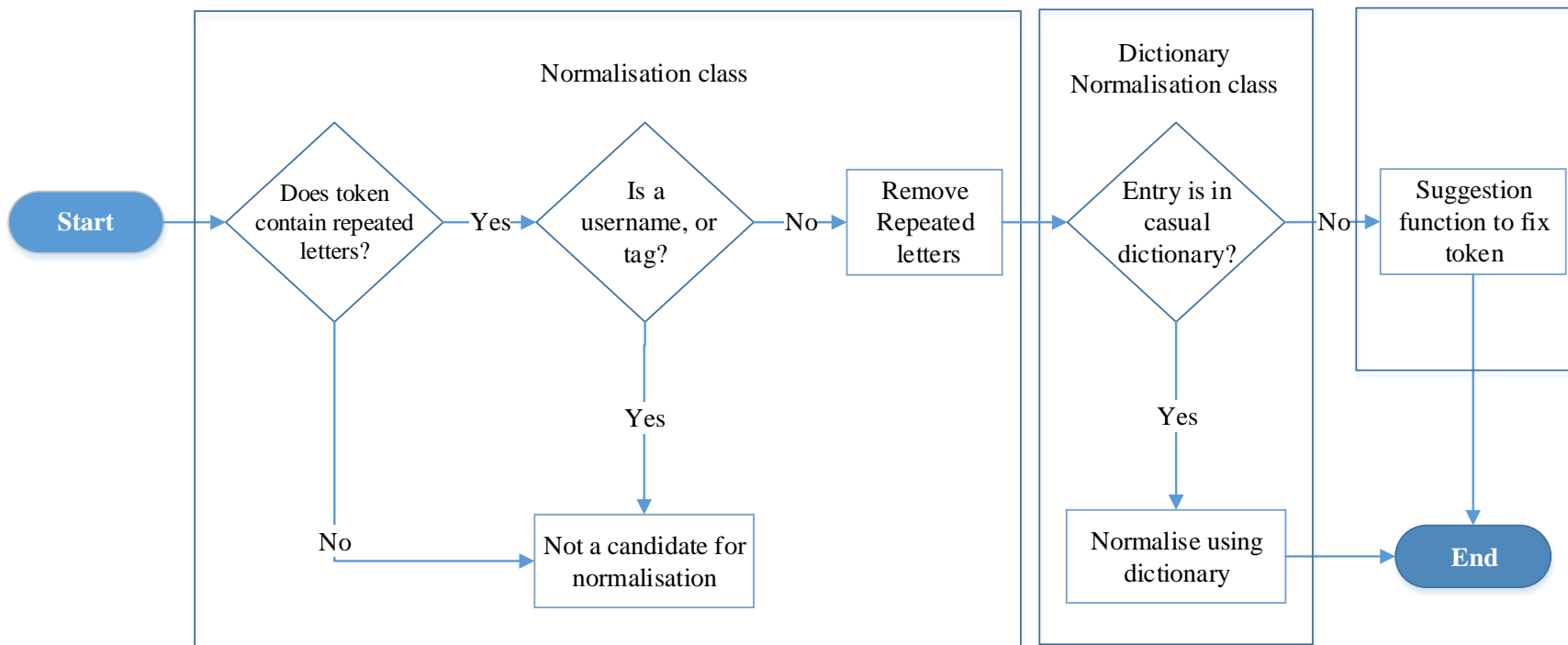


Figure 3.11: The flowchart of normalising repeated letters.

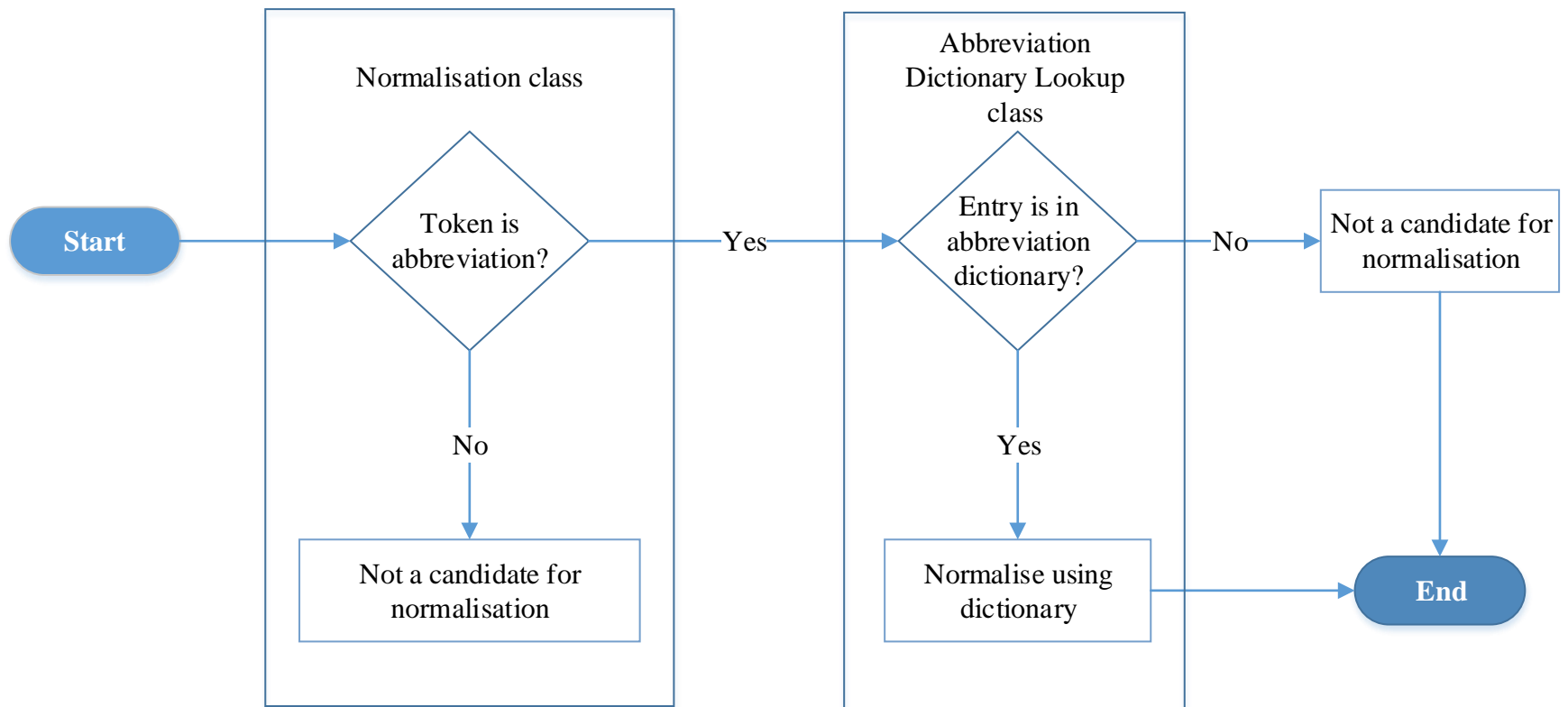


Figure 3.12: The flowchart of normalising abbreviations.

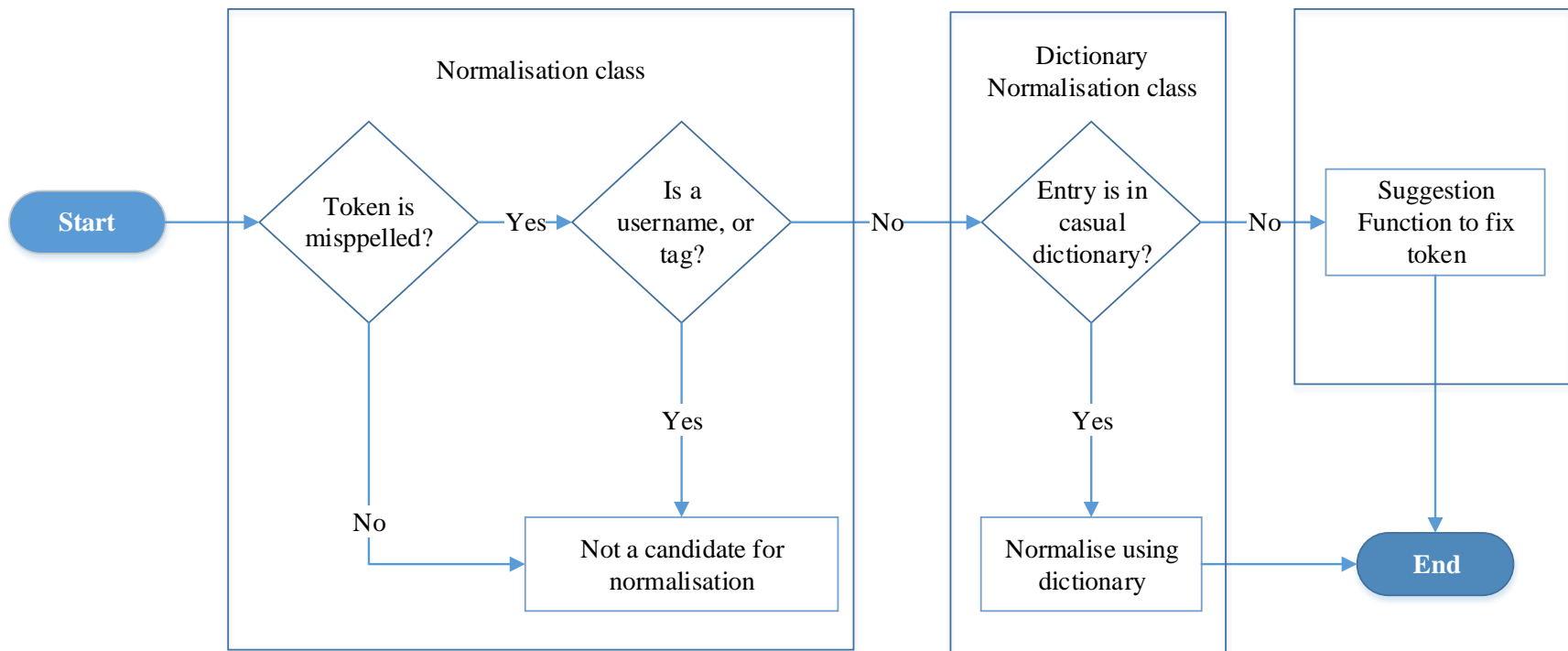


Figure 3.13: The flowchart of normalising misspelled words.

Figure 3.13 visualises how we will normalise misspelled words. Two main classes will be used to correct misspelled words. Each token will be firstly checked for misspelling, and it will be ignored when it is already correct based on the dictionary lookup. For the token identified as a misspelled word will be ignored if it is a username or hash-tag. Then the misspelt word will be normalised by using the dictionary. If the given token does not accrue in the dictionary, the suggestion function will be used to fix the token by providing the most likely correct word.

### **3.8. Statistical Normalisation Method for Twitter (SNET)**

The second experiment is a crucial part of our research in order to prove our second hypothesis, which is whether or not the proposed normalisation method can clean noisy tweets as accurately as the baseline model. The first phase for undertaking this is to combine each technique from each problem in descending order based on which combination provides the best normalising result. For example, the first group of combination techniques starts by eliminating repeated characters, then detecting abbreviations, and then finally checking the spelling. As such, there are 108 combinations between the two techniques from abbreviation detection, the three techniques from repeated characters elimination, and the three techniques from spelling correction. These combinations will give the results (based on the BLEU and WER scores) and dictate which problem should be solved in which order.

After knowing the order of solving noisy problems, we consider the use of the best techniques for each problem as a combination that will provide the best normalisation method. Thus, our 108 possible combinations of techniques will be compared to each other in order to select the best normalisation method. The best normalisation method will be considered as our proposed normalisation method and will be named SNET. Our final experiment is to compare the SNET with the baseline system, which is the Text Cleaner system developed by Gouws et al. (2011), in order to ascertain whether or not our proposed model is better as well as testing our second hypothesis. We also seek to measure the accuracy of a combination of the baseline system and our proposed normalisation method, against the baseline model alone.

Finally, the results from 108 combinations of cleaning techniques can help us to prove the first hypothesis as well, which was whether or not a combination of the best normalisation techniques from each problem will yield the best model. Also, we wish to determine if a combination in a different order will still provide the best result compared with other groups within the combination. Furthermore, the time taken to process each technique and combination is considered as well. To claim that the SNET is the best normalisation approach, the highest accuracy is not the only criteria that we seek to optimise, but also the time taken, and the consumption of processing power must be better than the baseline model.

### **3.9. Chapter Summary**

This chapter describes the normalisation techniques through five normalisation steps. These steps will be used to achieve our proposed normalisation method, the SNET, which can normalise noisy tweets into accurate English sentences. The framework of the normalisation method is presented in order to set out the step by step process of how we will perform our research to answer our problem statement and prove the hypotheses mentioned in chapter 1. This chapter not only provides us with a measurable goal, but also enables non-technical readers clearly to understand this study. Our normalisation approach is straightforward, starting with data collection and data pre-processing. After preparing datasets, noisy texts that are commonly found on Twitter are categorised. Categorising the type of noisy tweets will not only help us comprehend what noisy tweets need to be cleaned, but also helps us apply the proper cleaning techniques to each problem. Then testing and reference datasets are created for use in the evaluation of both existing normalisation techniques and the SNET. After this, the experiments used to prove the first and second hypotheses are discussed by demonstrating the experiments with existing cleaning techniques and for finding the best techniques for each problem. This is followed by an examination of how we will find the best combination of techniques and investigate the combination of the best techniques to provide the best model, and whether or not this can be claimed as the SNET. Finally, the comparison analysis is explained to test our second hypothesis which is whether or not the SNET is better than the baseline model.

# Chapter 4: Experimental Results and Discussion

This chapter aims to demonstrate how the objective of our research has been met by proving the hypothesis mentioned in Chapter 1. In this research, we hypothesise that the combined best technique addressing each problem will create the best normalisation method, and the SNET will produce the most accurate English sentences from normalising noisy tweets with better time efficiency. To test the hypothesis, we have conducted two types of experiments on a Twitter dataset in order to generate evaluation results. This chapter outlines our evaluation setup, criteria and results, and discusses our experiments and findings in relation to the problem statement set out in Chapter 1. It will also identify the limitations of the research.

## 4.1. Experimental Setup

Two types of experiments were conducted to prove our hypothesis. The first experiment contained two stages. Re-implementing existing normalisation methods represented the first stage in finding the best cleaning technique for solving each problem (misspelled words, abbreviations, and repeated characters). The second stage involved combining techniques used to address each problem in a different order. This experiment was intended to help us find the

best combination of techniques that could solve all problems. The second experiment used the same tweets dataset to prove that the SNET (the best combination of normalisation techniques found in the first stage) is better than the baseline model, in terms of accuracy and time efficiency, at normalising a noisy tweet into a clean and readable sentence. To evaluate our hypothesis, 177 experiments were performed in this research. These experiments consisted of evaluations of the two techniques for detecting abbreviations, three techniques for removing repeated characters, and three techniques for correcting misspelled words. These techniques are detailed in Chapter 3, section 3.6. Additionally, 108 experiments were carried out to examine the performance of every possible combination of techniques in various orders, and one experiment against the baseline model.

The experiments were run on a dataset of 1200 tweets containing messages from popular celebrities in the entertainment area and the replies from their fans. Our tweets dataset contains a number of different combinations of abbreviations, repeated characters, and misspelled words. Overall, there are 489 numbers of abbreviations, 152 numbers of words with repeated characters, and 375 numbers of words with misspelled word. The experiments for proposing the SNET were performed on Windows 8; meanwhile, the baseline model was implemented on Linux Ubuntu v.14.04 operating systems. All techniques described in Chapter 3 were implemented in Python and NLTK (NLTK, 2015; Perkins, 2010, 2014; Python, 2015a) framework. In order to acquire a set of word candidates and select the most appropriate candidate to replace the noisy word, the baseline model (Text Cleanser developed by Gouws et al. (2011)) utilised the SRI-LM toolkit (SRI-International, 2011; Stolcke, 2002) to extract lexical n-gram features and lattice-tool. As SRI-LM can only be installed on the Linux operating system, all experiments related to the baseline model were performed only in Linux.

For our evaluation setup, we then formed the datasets for each of our tests by manually normalising those 1200 tweets and creating four reference datasets. In the first reference dataset, we corrected all the abbreviations from the original tweets. For instance, if the original tweet was “That viedo is fuuunnnny LOL”, in the first reference dataset (Ref\_AB) the tweet became “That viedo is fuuunnnny laugh out loud.” In the second dataset (Ref\_RC), we corrected only the repeated characters. Thus, the tweet became “That viedo is funny LOL”. In the third dataset (Ref\_MSW), we corrected only the misspelled words, i.e. “That video is

fuuunnnny LOL”. In the last dataset (Ref\_All), we corrected all of those cases, i.e. “That video is funny laugh out loud.” To sum up, the first three references datasets were used for evaluating each technique that was used to address each problem and the fourth reference dataset was used to evaluate the combined models with themselves, the baseline model, and a proposed model with the baseline model.

## **4.2. Evaluation Metrics**

We now use the method of quantitative analysis for evaluating the accuracy and time efficiency of each cleaning technique. The BLEU and WER metrics are widely used as automatic evaluation metrics for finding a normalisation method’s accuracy (Aw et al., 2006; Beaufort et al., 2010; Han & Baldwin, 2011; Han et al., 2012; Hassan & Menezes, 2013; Marton & Zitouni, 2014; Sproat et al., 2001). In this study, we are going to evaluate the accuracy of each normalisation technique when they clean noisy cases in character-level from tweets, including misspelled words, repeated characters, and abbreviations. As discussed in Chapter 2, grammatical correctness and intelligibility are not considered in this research. The technique’s efficiency refers to the time required to normalise the tweets data. Finally, the paired t-test is used to determine whether there is a significant performance differential in terms of accuracy and efficiency between the normalisation techniques.

### **4.2.1. BLEU Metric**

The BLEU is a metric used to evaluate the quality of a sentence which has been transformed from one language form to another (Dreyer & Marcu, 2012). The BLEU metric measures how many words that overlap in a given normalised sentence when compared to a reference dataset, and then gives the higher scores to successive words. A normalised sentence is scored on a scale of 0 to 1; however it is frequently shown as a percentage value. The central idea behind the BLEU is that the closer the BLEU score is to 1, the more the normalised sentence correlates to a human annotate (Papineni et al., 2002). Additionally, the BLEU score can be increased by adding extra reference texts (Callison-Burch, Osborne, & Koehn, 2006). The following equation describes how the BLEU score is calculated:

$$P = \frac{m}{w_t}$$

where:

- $P$  is the precision used to compare the candidate against reference texts.
- $m$  is the number of words from the candidate that are found in the reference.
- $w_t$  represents the total number of words in the candidates.

Using the BLEU metric for evaluation brings several advantages. It can be set up speedily, it is language independent, expends less memory and time to operate, and correlates highly with human evaluation (Denoual & Lepage, 2005). To score the accuracy of each normalisation technique in this study, we use the iBLEU version 2.6 developed by Madnani (2011) which is state-of-the-art web technology that can provide a visual and interactive way to generate the BLEU scores. It can run locally in the user's browser and includes all external dependencies.

#### **4.2.2. WER Metric**

The WER is another metric that we use to evaluate the accuracy of each normalisation technique. For each sentence produced by a technique (or defined as 'hypothesis' by Thoma, 2013), the metric calculates the number of word substitutions, deletions and insertions that need to be done in the hypothesis to match the human-authored reference sentence (Klakow & Peters, 2002)). For example, if "I love dogs" is the reference and "I dogs" is the hypothesis, it is considered as a deletion because "love" was deleted. If "I love black dogs" is the hypothesis, it is considered as an insertion because "black" was inserted. If "I love cats" is the hypothesis, it is considered as a substitution because "cats" was substituted for "dogs". Therefore, achieving a lower WER indicates higher data cleaning accuracy. We use the evaluator.py file from Gouws et al. (2011), for testing each normalisation method's WER accuracy.

The WER is a valuable metric for comparing different normalisation models as well as testing improvements within the model itself (Klakow & Peters, 2002; Thoma, 2013; Zechner &

Waibel, 2000). The WER is calculated based on the Levenshtein edit distance. Word errors are measured based on “substitutions” when a word is replaced, “deletions” when a word is missed out, and “insertions” when a word is added. Additionally, the WER works at the word-level rather than the phoneme-level. First, it uses dynamic string alignment to align the recognised word sequence in the hypothesis with the reference word sequence. Next, it calculates the WER score of the hypothesis using the following equation:

$$WER = \frac{S + D + I}{S + D + C} = \frac{S + D + I}{N}$$

where:

- *S* means the number of substitutions.
- *D* means the number of deletions.
- *I* means the number of insertions.
- *C* means the number of the corrects.
- *N* means the number of words in the reference ( $N=S+D+C$ ).

### 4.2.3. Paired t-tests

A paired t-test is a statistical method that is used to compare the means of two sets of data and examine whether their difference is statistically significant (Shier, 2004). In this study, we use a paired t-test to find the difference between two techniques by comparing their BLEU and WER scores. For example, six t-tests are conducted to demonstrate how the three cleaning techniques for correcting misspelled words are different from each other. The equation of a paired t-test that is used in this study is shown as follows:

$$t = \frac{\bar{d}}{\sqrt{s^2/n}}$$

where:

- $t$  means the paired sample t-test where  $n-1$  degree of freedom.
- $\bar{d}$  or  $d$  bar means the difference mean between two samples.
- $s^2$  means the variance of the sample.
- $n$  means the size of the sample.

Subsequent to the calculation of the parameter, the calculated value (p-value) is compared with the chosen significance level (we use 0.005 as the standard error of difference or 95% confidence level). If the calculated p-value is greater than 0.005, then the difference between the two values is considered to be not statistically significant, i.e. the null hypothesis is accepted. On the other hand, if the calculated p-value is less than 0.005, the null hypothesis is rejected — there is a significant difference between the two paired samples.

#### **4.2.4. Run-Time Efficiency**

The efficiency of a technique is our last evaluation criteria in this study. Efficiency is evaluated by the time that is required by a normalisation technique to perform a data cleaning procedure within the same operating system (Rossi, Lipsey, & Freeman, 2003; Wholey, 1979). Each technique is timed from when the input is inserted until when the output is generated. The central ideal behind run-time efficiency is that a short amount of run-time process implies a high level of efficiency.

### **4.3. Evaluation Results**

To identify the best technique to solve a particular noisy problem, we compare the performance of each technique in cleaning the dataset. The results are shown in Section 4.3.1. Next, we evaluate the performance of every possible combination of techniques and order of executions on the last reference dataset. The results from 108 combinations of techniques are presented in section 4.3.2. The best combination model will be considered as our proposed

normalisation model, and then the results from that will be compared with baseline model in section 4.3.3. This is in order to prove our second hypothesis, which is that our proposed model can normalise a noisy tweet into an accurate English sentence and the model can perform more efficiently than the baseline one.

### **4.3.1. Results from Individual Techniques**

To test our first hypothesis, a comparison of the techniques to solve the same noisy problem on the same tweet dataset is required to identify which technique is the best. The findings from the first experiment will help us to examine whether or not combining the best techniques for solving each problem into a single model will produce the best model for solving all types of noisy text. The results of the first experiment are presented according to the type of noisy problems it is trying to solve and they are explained in detail as follows.

#### **4.3.1.1. Removing repeated characters**

The evaluation results of the three techniques (RRC1, RRC2, and RRC3), discussed in Section 3.6.1, for eliminating repeated characters are shown in Table 4.1. Ref\_RC is used as the reference dataset. It can be seen that RRC1 and RRC3 achieved better results than the RRC2 scores of only 79.76% in the BLEU and 11.01% in the WER. RRC1 is considered to be the best technique for eliminating the 152 repeated words found in the 1200 tweets with the highest BLEU score (83.65%), the lowest WER (8.89%), and shortest processing time (25 seconds).

As a result, the accuracy of RRC2 and RRC3 are marginally lower than RRC1. On the other hand, RRC2 and RRC3 are slightly different in terms of correcting a word after removing repeated letters. However, there is a vast difference between RRC2 and RRC3 in time spent in the cleaning process. For RRC3, although its BLEU score and WER value are better than RRC2, it spends a long period cleaning the noisy tweet. We suspect that the time has been spent mostly on checking Enchant dictionary to see whether or not the word is valid. RRC3 spends 22 minutes on processing, while RRC2 uses only 1 minute. As such, RRC3 is better than RRC2 regarding accuracy. But in terms of time efficiency, RRC2 is far better. The paired t-test showed that the difference in the BLEU score of RRC1 compared with RRC2 and RRC1

compared with RRC3 are statistically significant at 95% CI level. However, there is no significant difference in the WER score of RRC1 to RRC2 or RRC3.

Techniques	BLEU (%)	WER (%)	Time
Technique A: use regular expression function based on the format of pattern with Python 3 spelling corrector (RRC1)	83.65	8.89	25Sec
Technique B: use the replace() function with a WordNet lookup (RRC2)	79.76	11.01	1min
Technique C: use the replace() function with an Enchant dictionary (RRC3)	80.13	10.79	22mins

Table 4.1: The evaluation results of three techniques from elimination of repeated characters.

In order to allow the readers to have a clear idea of how each technique normalises a noisy tweet, the actual example from our normalised tweets dataset will be explained using the following special notations. We have used **bold** on words that are supposed to be normalised and underlines on words that have been incorrectly normalised. Thus, words that are both **bolded** and underlined are words that are supposed to be normalised but were transformed into incorrect forms. These special notations will be used throughout the chapter.

**Input:** @cassieventura Britney concert is soooooo funny

**RRC1:** @cassieventura Britney concert is **so funny**

**RRC2:** @casieventura Britney concert is **so fun**

**RRC3:** @casieventura Britney concert is **so funny**

Figure 4.1: The normalised outputs from RRC1, RRC2, and RRC3.

As can be seen in the Figure 4.1, RRC1 has better performance than RRC2 and RRC3 in eliminating the repeated characters in “sooooo” to “so” and “funny” to funny. Although

RRC3 has removed the noisy words containing repeated letters, it has also removed repeated characters in the Twitter username, “@cassieventura” to “@casieventura”, and this reduces the accuracy of the sentence. Meanwhile, RRC2 not only removes repeated characters found in Twitter usernames, but also changes “funny” to “fun” due to the limitation of WordNet lookup.

#### 4.3.1.2. Detecting abbreviations

Two techniques (DAB1 and DAB2) for normalising abbreviations, discussed in Section 3.6.2, are evaluated to show how each technique performs. Ref\_AB is used as the reference dataset in the BLEU and WER score calculations. As can be seen in Table 4.1, both techniques achieve high percentages of accuracy; both are more than 90% in the BLEU score. At the same time, the values of WER are less than 4% and only 30 seconds is spent detecting 498 abbreviations found on 1200 tweets (16.867 tokens). According to the normalised outputs from both techniques, the percentage of abbreviations that are detected and converted into their formal form is approximately 90%; the rest is the number of detected abbreviations that are correct at word-level but not at sentence-level. For example, both techniques resolve “ur” in “I love that **ur** in” to “I love that **your** in”, which is incorrect. Based on the Ref-AB, “ur” is “you are” in a reference sentence. Although our abbreviation dictionary has defined “ur” with two separate meanings, neither techniques can select the right meaning of the given sentence.

**Input:** @serenawilliams When **u** listen to it **u Wld** see a lot of similarity

**AB1:** @serenawilliams When **you** listen to it **you Wld** see a lot of similarity

**AB2:** @serenawilliams When **you** listen to it **you would** see a lot of similarity

Figure 4.2: The normalised outputs from DAB1 and DAB2.

Overall, the results from both techniques yield high accuracy, especially from AB2. It can be seen in Figure 4.2, that DAB2 performs slightly better than DAB1 regarding the accuracy results in both BLEU and WER scores. As is evident from the examples below, DAB1 cannot detect an abbreviation that contains the upper case letter (i.e. “Wld”) due to our normalisation

(abbreviation) dictionary merely having the reference abbreviations that have the lower case letters (i.e. “luv”). On the other hand, DAB2 can detect abbreviations with upper-letters by transforming them into lower-letters before resolving.

Techniques	BLEU (%)	WER (%)	Time
Technique A: use <i>replace()</i> method and manually created python dictionary (DAB1)	93.77	3.87	30 Sec
Technique B:use <i>replace()</i> method with the word_map dictionary from CSV file format (DAB2)	<b>95.26</b>	<b>2.65</b>	<b>30 Sec</b>

Table 4.2: The evaluation results of two techniques from detection of abbreviations.

To determine whether the differences between the two techniques for detecting abbreviations are significant, the use of a paired t-test will help us to find the answer. We conclude that regarding both the BLEU and WER scores, the difference between the two techniques is not statistically significant at the 95% confidence interval (CI) level.

#### 4.3.1.3. Correcting misspelled words

Three techniques (SC1, SC2, and SC3), described in Section 3.6.3, for resolving misspelled words are compared. Ref\_MSW is set as the reference dataset. As can be seen in Table 4.3, SC1 achieves the better results in the BLEU, WER and time for correcting 375 misspelled words found on 1200 tweets. SC1 achieves 79.88% for the BLEU, 12.40% for the WER and spends 2 minutes for processing. Although SC3’s BLEU score is a bit higher than SC2’s, the time spent on the cleaning process is longer at 21 minutes. As shown by the Figure 4.3, which presents the examples of output produced by the three techniques, SC1 handles the noisy input tweet the best.

By using Aspell for an Enchant dictionary along with the edit distance method, SC1 can correct “colege” to “college”, “comunications” to “communications”, and “skils” to “skills”. SC1 works by first looking for a list of alternative suggestions for each word that are not found in the dictionary. The list is sorted based on the edit distance of the suggested word to the original word. All words with an edit distance greater than the specified maximum distance are removed from the list. The technique will then choose the best match words on the list.

**Input:** @puranjaycom beyond **colege** amped up **comunications** **skils** improved networking human **skils**

**SC1:** @puranjaycom beyond **college** amped up **communications** **skills** improved networking human **skills**

**SC2:** @puranjaycom beyond **college** amped up comunication skill improved networking human skill

**SC3:** @puranjaycom beyond **college** camped up **communications** skill improved networking human skill

Figure 4.3: The normalised outputs from SC1, SC2, and SC3.

Techniques	BLEU (%)	WER (%)	Time
Technique A: spelling correction with <i>Enchant</i> with the edit distance method (SC1)	79.88	12.40	2mins
Technique B: spelling corrector developed by Norvig (but we use different dictionary) (SC2)	68.47	17.97	4mins
Technique C: use TextBlob library with <i>correct()</i> method (SC3)	69.39	16.68	21mins

Table 4.3: The evaluation results of three techniques from correction of misspelled words.

The reason SC3 changes “amped” to “camped” is because the word “amped” is not found in the TextBlob dictionary, so it chooses the most likely word to replace the original word for spelling correction. Similarly, because the word “skills” is not contained in both SC2 and SC3 dictionaries, “skills” has been converted from plural to singular form. In SC2, the set of the candidate words to replace the original word is ordered based on the highest probability value of the candidate set estimated by NWORDS model, which counts the number of times the word has been seen in the dictionary.

By using a paired t-test to determine the mean difference between the three techniques for correcting misspelled words, at the 95% CI level we found that SC1 is significantly different from SC2 and WSM3 regarding the BLEU and WER scores. SC1 is better than the other two techniques because it not only uses the functions that can check misspelled words, but also uses two dictionaries for checking, and one of them can be edited or added to with new vocabularies at any time. Additionally, there is no significant difference between SC2 and SC3 in both BLEU and WER scores, but the run-time is significantly different. While SC2 spends only 4 minutes in correcting misspelled words, SC3 spends 21 minutes. Given a large amount of input, SC3 will take a considerably long time to process the whole input.

#### **4.3.2. Results from Combination of Techniques**

From the previous section, we know that the best techniques for resolving abbreviation, misspelling and repeated characters are DAB2, SC1 and RC1 respectively. To test whether our first hypothesis is correct, the next step is to set up an experiment to identify the best combination of DAB, SC and RRC cleaning techniques and the best order to execute those techniques. Thus, we know which type of problems should be addressed first and which one should be addressed last. 108 combinations have been tested and evaluated with the BLEU, WER, and time criteria. Ref\_All is the reference dataset used for calculating the BLEU and WER score for each combination of techniques. The results of this experiment are organised according to the execution order of each technique. As such, there are six groups of combinations presented and described in detail in the following sections.

#### 4.3.2.1. The combination of RRC → DAB → SC techniques

The first combination group began with the elimination of repeated characters, followed by the rectification of abbreviations and, finally, the correction of misspelled words. The experiments are evaluated based on the BLEU score, WER score and time efficiency.

Combinations	BLEU (%)	WER (%)	Time
RRC1 → DAB1 → SC1	87.39	8.28	2mins55sec
RRC1 → DAB1 → SC2	76.45	13.85	4mins55sec
RRC1 → DAB1 → SC3	78.92	11.09	21mins55sec
<b>RRC1 → DAB2 → SC1</b>	<b>88.51</b>	<b>7.14</b>	<b>2mins55sec</b>
RRC1 → DAB2 → SC2	77.55	12.75	4mins55sec
RRC1 → DAB2 → SC3	78.92	11.09	21mins55sec
RRC2 → DAB1 → SC1	83.40	10.61	3mins30sec
RRC2 → DAB1 → SC2	73.42	15.61	5mins30sec
RRC2 → DAB1 → SC3	73.41	14.77	22mins30sec
RRC2 → DAB2 → SC1	84.80	9.33	3mins30sec
RRC2 → DAB2 → SC2	74.78	14.36	5mins30sec
RRC2 → DAB2 → SC3	74.84	13.48	22mins30sec
RRC3 → DAB1 → SC1	84.25	9.99	24mins30sec
RRC3 → DAB1 → SC2	74.45	15.06	26mins30sec
RRC3 → DAB1 → SC3	74.36	14.22	43mins30sec
RRC3 → DAB2 → SC1	85.49	8.83	24mins30sec
RRC3 → DAB2 → SC2	75.59	13.92	26mins30sec
RRC3 → DAB2 → SC3	75.58	13.02	43mins30sec

Table 4.4: The evaluation results of the combination of RRC, DAB, and SC techniques.

Table 4.4 presents 18 combinations of three techniques for each problem in a different order. Overall, this group performs well with promising results in the BLEU, WER and time. It can be seen that the outstanding combination is RRC1 → DAB2 → SC1. This combination achieves the highest BLEU score (88.51%), the lowest WER value (7.14%), and spends only 2

minutes and 55 seconds cleaning 1200 tweets. There are several reasons why RRC1 → DAB2 → SC1 handles the three problems better than the rest in this group. By comparing its output with other combinations, we note that all repeated characters are removed and corrected into their formal form by RRC1, most of the abbreviations are detected and converted by DAB2, and misspelled words are mostly corrected by SC1. Because individually RRC1, DAB2 and MWS1 spend a short amount of time normalising noisy tweets, the combination of these techniques also requires only a small amount of time to clean the noisy tweets.

On the other hand, some techniques (i.e. RRC3 → DAB1 → SC3 and RRC3 → DAB2 → SC3) spend a long period of time normalising noisy tweets, due to face that they combine the techniques (RRC3 and SC3) that individually consume a lot of time cleaning repeated letters and misspelled words. Thus, there is no doubt why they spend a longer time than the others while their yields are not satisfactory as expected. Furthermore, RRC2 → DAB1 → SC2 and RRC2 → DAB1 → SC3 are the combinations that achieve the lowest BLEU score (~73%) and the highest WER values (>14%). RRC3 → DAB2 → SC3 spends nearly 44 minutes normalising noisy tweets, and its BLEU score is a bit higher than RRC3 → DAB1 → SC3 which spends the same amount of time in normalising.

Additionally, the most outstanding difference that can be defined is between RRC1 → DAB2 → SC1 and RRC2 → DAB1 → SC3. Their calculated p-values from the BLEU and WER are both less than 0.005 at the 95% CI level, 0.0002 and 0.0004 respectively. This mean difference between them is considered to be extremely statistically significant. Furthermore, the difference between RRC1 → DAB2 → SC1 and RRC1 → DAB1 → SC1 is deemed to be statistically significant as well. Their calculated p-values from the BLEU and WER are both less than 0.005 at the 95% CI level, 0.0029 and 0.0028 respectively. Therefore, RRC1 → DAB2 → SC1 is significantly different to any technique with the BLEU score  $\leq 87\%$  or the WER score  $\geq 8\%$ .

#### 4.3.2.2. The combination of RRC → SC → DAB techniques

The second combination group began with the elimination of repeated characters, followed by the correction of misspelled words and, finally, the rectification of abbreviations. The experiments are evaluated based on the BLEU score, WER score and time efficiency.

Combinations	BLEU (%)	WER (%)	Time
RRC1 → SC1 → DAB1	82.29	10.65	2mins55sec
<b>RRC1 → SC1 → DAB2</b>	<b>84.41</b>	<b>9.60</b>	<b>2mins55sec</b>
RRC1 → SC2 → DAB1	76.35	13.78	4mins55sec
RRC1 → SC2 → DAB2	77.45	12.40	4mins55sec
RRC1 → SC3 → DAB1	75.72	13.29	4mins55sec
RRC1 → SC3 → DAB2	78.82	12.87	21mins55sec
RRC2 → SC1 → DAB1	79.30	11.89	3mins30sec
RRC2 → SC1 → DAB2	74.70	14.11	3mins30sec
RRC2 → SC2 → DAB1	73.32	14.90	5mins30sec
RRC2 → SC2 → DAB2	74.68	14.44	5mins30sec
RRC2 → SC3 → DAB1	71.31	15.99	5mins30sec
RRC2 → SC3 → DAB2	72.74	15.26	22mins30sec
RRC3 → SC1 → DAB1	75.15	13.98	24mins30sec
RRC3 → SC1 → DAB2	77.39	12.75	24mins30sec
RRC3 → SC2 → DAB1	74.35	14.78	26mins30sec
RRC3 → SC2 → DAB2	75.49	13.51	26mins30sec
RRC3 → SC3 → DAB1	72.26	15.97	43mins30sec
RRC3 → SC3 → DAB2	75.48	13.62	43mins30sec

Table 4.5: The evaluation results of the combination of RRC, SC, and DAB techniques.

As can be seen in Table 4.5, 18 combinations of three techniques for each problem in a different order have been presented. According to the evaluation results, this group still performs well with acceptable results in the BLEU score, WER score and time spent. The outstanding combination in this group is RRC1 → SC1 → DAB2, which achieves the highest

BLEU score (84.41%), the lowest WER value (9.60%), and spends only 2 minutes and 55 seconds cleaning 1200 tweets. The RRC1 → SC1 → DAB2 technique provides a double check of spelling from RRC1 and SC1 before expanding the abbreviations. According to the high accuracy of RRC1 in removing repeated characters and SC1 in correcting misspelled words, some noisy words are cleaned as much as possible before cleaning the abbreviations. However, RRC1 → SC1 → DAB2 still achieves low accuracy when compared with RRC1 → DAB2 → SC1. In RRC1 → SC1 → DAB2, SC1 considers some of the abbreviations as misspelled words, as such, those words are normalised into incorrect form. For example, SC1 corrects “deze” to “daze” instead of “these”, “whatchu” to “watch” instead of “what are you”. This action lowers the accuracy of the techniques for normalising noisy tweets. As a result, RRC1 → SC1 → DAB2 is not working well when compared with RRC1 → DAB2 → SC1.

Another notable combination in this group is RRC2 → SC3 → DAB1, which achieves the lowest BLEU score (71.31%) and the highest WER value (>15%). Even though RRC2 → SC3 → DAB1 spends less time normalising noisy tweets than RRC2 → SC3 → DAB2 and RRC3 → SC3 → DAB1, its BLEU score is still lower than theirs and the WER value is still higher. On the other hand, some techniques (i.e. RRC3 → SC3 → DAB1 and RRC3 → SC3 → DAB2) are very slow in processing normalisation. As mentioned in the previous, first combination group, both techniques use RRC3 and SC3; these require the longest amount of time to treat repeated letters and misspelled words. Although a lot of time has been spent in running RRC3 → SC3 → DAB1 and RRC3 → SC3 → DAB2, the accuracy of their performance in normalising noisy tweets is still not high enough.

From determining the difference between 18 combinations in this group, the most significant difference is observed when we compare RRC1 → SC1 → DAB2 to RRC2 → SC3 → DAB1, the calculated p-values from the BLEU and WER are both less than 0.005 at the 95% CI level, 0.0002 and 0.0005 respectively. This mean difference between them is considered to be very statistically significant. Furthermore, the difference between RRC1 → SC1 → DAB2 and RRC1 → SC1 → DAB1 is considered to be significant due to their p-values from the BLEU (0.0015) and the WER (0.0030) scores being less than 0.005 at the 95% CI level. Consequently, RRC1 → SC1 → DAB2 is significantly different to any technique with the BLEU score  $\leq 82\%$  or the WER score  $\geq 10\%$ .

#### 4.3.2.3. The combination of DAB → RRC → SC techniques

The third combination group began with the rectification of abbreviations, and then followed with the elimination of repeated characters and, finally, the correction of misspelled words. The experiments are evaluated based on the BLEU score, WER score and time efficiency.

Combinations	BLEU (%)	WER (%)	Time
DAB1 → RRC1 → SC1	87.27	8.29	2mins55sec
DAB1 → RRC1 → SC2	76.46	13.84	4mins55sec
DAB1 → RRC1 → SC3	77.79	12.17	21mins55sec
DAB1 → RRC2 → SC1	84.60	9.57	3mins30sec
DAB1 → RRC2 → SC2	74.61	14.94	5mins30sec
DAB1 → RRC2 → SC3	74.54	14.04	23mins30sec
DAB1 → RRC3 → SC1	84.83	9.43	24mins30sec
DAB1 → RRC3 → SC2	74.83	14.78	26mins30sec
DAB1 → RRC3 → SC3	74.71	13.91	40mins30sec
<b>DAB2 → RRC1 → SC1</b>	<b>88.55</b>	<b>7.10</b>	<b>2mins55sec</b>
DAB2 → RRC1 → SC2	77.62	12.69	4mins55sec
DAB2 → RRC1 → SC3	79.01	11.02	21mins55sec
DAB2 → RRC2 → SC1	85.90	8.35	3mins30sec
DAB2 → RRC2 → SC2	75.76	13.78	5mins30sec
DAB2 → RRC2 → SC3	75.78	12.88	22mins30sec
DAB2 → RRC3 → SC1	86.12	8.23	24mins30sec
DAB2 → RRC3 → SC2	75.96	13.66	26mins30sec
DAB2 → RRC3 → SC3	75.95	12.75	43mins30sec

Table 4.6: The evaluation results of the combination of DAB, RRC, and SC techniques.

This combination group provides promising results regarding satisfying the higher BLEU score and the lower WER value, but the run-time is not different from the previous groups. In Table 4.6, the outstanding combination is DAB2 → RRC1 → SC1, which achieves the highest BLEU score (88.55%) and the lowest WER (7.10%) and spends only 2 minutes and 55

seconds normalising abbreviations, repeated characters and misspelled words. According to the tweets normalised by this combination, more than 80% have been corrected to the same format as in reference dataset (Ref\_All). When comparing this group with the previous ones, we observe that handling abbreviations as a first step can ensure that all given abbreviations have been checked and replaced by their formal format. For example, “deze” to “these” and “whatchu” to “what are you”. Then, the elimination of repeated characters has made sure that words containing repeated characters will be addressed thoroughly. Consequently, when these types of noisy tweets are cleaned, they have been returned to the normalised tweets dataset without any spelling correction from SC1. These first two steps have not only increased the capacity of the spell corrector to deal with only incorrect words, but also can normalise the final output with the highest accuracy. Hence, DAB2 → RRC1 → SC1 is better than RRC1 → SC1 → DAB2 and RRC1 → DAB2 → SC1.

While DAB2 → RRC1 → SC1 is the best combination in the group, DAB1 → RRC2 → SC3, DAB1 → RRC2 → SC3, DAB1 → RRC3 → SC2 and DAB1 → RRC3 → SC3 are the combinations that achieve the lowest BLEU score (~74%) and the highest WER values (>14%). On the other hand, the combination that spends the longest time processing normalisation in this group is DAB2 → RRC3 → SC3 (43minutes and 30seconds). However, it performs better than DAB1 → RRC2 → SC3, which spends a shorter time in normalisation (23minutes and 30 seconds), based on both BLEU score and WER value. But the most notable difference among 18 combinations is between DAB2 → RRC1 → SC1 and DAB1 → RRC2 → SC3, the calculated p-values from the BLEU and WER are both less than 0.005 with 95% CI level, 0.0016 and 0.0011 respectively. Hence, this mean difference is considered to be very statistically significant.

Meanwhile, the p-values difference in the BLEU and WER scores of DAB2 → RRC1 → SC1 and DAB1 → RRC1 → SC1 are 0.0025 and 0.0027, both of which are less than 0.005 at the 95% CI level. This difference is considered to be statistically significant as well. Another difference that is seen as statistically significant is when a comparison is made between DAB2 → RRC1 → SC1 with DAB2 → RRC3 → SC1. The calculated p-values from the BLEU and WER are both less than 0.005, 0.0013 and 0.0028 respectively. Therefore, RRC1 → SC1 →

DAB2 is significantly different to any technique with the BLEU score  $\leq 87\%$  or the WER score  $\geq 8\%$ .

#### 4.3.2.4. The combination of DAB $\rightarrow$ SC $\rightarrow$ RRC techniques

The fourth combination group began with the rectification of abbreviations, followed by the correction of misspelling words and, finally, the elimination of repeated characters. The experiments are evaluated based on the BLEU score, WER score and time efficiency.

Combinations	BLEU (%)	WER (%)	Time
DAB1 $\rightarrow$ SC1 $\rightarrow$ RRC1	82.12	10.85	2mins55sec
DAB1 $\rightarrow$ SC1 $\rightarrow$ RRC2	79.55	11.49	3mins30sec
DAB1 $\rightarrow$ SC1 $\rightarrow$ RRC3	79.78	11.09	24mins30sec
DAB1 $\rightarrow$ SC2 $\rightarrow$ RRC1	71.41	15.68	4mins55sec
DAB1 $\rightarrow$ SC2 $\rightarrow$ RRC2	79.56	11.53	5mins30sec
DAB1 $\rightarrow$ SC2 $\rightarrow$ RRC3	69.78	16.39	26mins30sec
DAB1 $\rightarrow$ SC3 $\rightarrow$ RRC1	72.75	15.31	21mins55sec
DAB1 $\rightarrow$ SC3 $\rightarrow$ RRC2	69.49	16.79	23mins30sec
DAB1 $\rightarrow$ SC3 $\rightarrow$ RRC3	69.66	16.48	40mins30sec
<b>DAB2 <math>\rightarrow</math> SC1 <math>\rightarrow</math> RRC1</b>	<b>83.50</b>	<b>9.69</b>	<b>2mins55sec</b>
DAB2 $\rightarrow$ SC1 $\rightarrow$ RRC2	80.85	11.29	3mins30sec
DAB2 $\rightarrow$ SC1 $\rightarrow$ RRC3	81.07	10.99	24mins30sec
DAB2 $\rightarrow$ SC2 $\rightarrow$ RRC1	72.57	15.68	4mins55sec
DAB2 $\rightarrow$ SC2 $\rightarrow$ RRC2	70.71	16.31	5mins30sec
DAB2 $\rightarrow$ SC2 $\rightarrow$ RRC3	70.91	16.12	26mins30sec
DAB2 $\rightarrow$ SC3 $\rightarrow$ RRC1	74.01	15.89	21mins55sec
DAB2 $\rightarrow$ SC3 $\rightarrow$ RRC2	70.73	16.24	22mins30sec
DAB2 $\rightarrow$ SC3 $\rightarrow$ RRC3	70.90	16.10	43mins30sec

Table 4.7: The evaluation results of the combination of DAB, SC, and RRC techniques.

According to Table 4.7, DAB2 → SC1 → RRC1 performs better than other combinations in this group, with an 83.50% in the BLEU score and 9.69% in the WER value. However, the performance of DAB2 → SC1 → RRC1 is not good enough in comparison to DAB2 → RRC1 → SC1. After expanding abbreviations into their formal form, some words containing repeated characters initially have been corrected by SC1 before sending to RRC1. For example, “sooo” will be corrected to “soon” instead of removing repeated letters to get “so”. Hence, most of the words that contain repeated characters have been transformed into another word which causes a change in the final output of normalised tweets as well as having an effect on the accuracy.

Furthermore, DAB1 → SC2 → RRC3, DAB1 → SC3 → RRC2 and DAB1 → SC3 → RRC3 are the combinations that achieve the lowest BLEU score (~69%) and the highest WER values (>16%). In addition, the DAB2 → SC3 → RRC3 technique spends nearly 44 minutes normalising noisy tweets. These drawbacks of these combinations are caused by the use of cleaning techniques that do not work properly when combined with this type of order. Furthermore, some techniques (SC3 and RRC3) require a lot of time to clean and normalise input text, and thus the run-time processing is automatically increased when used with other cleaning techniques.

A noticeable difference is apparent within this group, and that is the significant difference between DAB2 → SC1 → RRC1 and DAB1 → SC3 → RRC2. Their p-values generated from the BLEU and WER are 0.0002 and 0.0004. This difference between them is considered to be extremely statistically significant because their p-values are less than 0.005 at the 95% CI level. Meanwhile, the difference between DAB2 → SC1 → RRC1 and DAB1 → SC1 → RRC1 is considered to be very statistically significant as well. The calculated p-values from the BLEU and WER are both less than 0.005, 0.0023 and 0.0027 respectively. Therefore, the DAB2 → SC1 → RRC1 technique is significantly different to any technique with the BLEU score  $\leq 82\%$  or the WER score  $\geq 10\%$ .

#### 4.3.2.5. The combination of SC → RRC → DAB techniques

The fifth combination group began with the correction of misspelled words, followed by the elimination of repeated characters and rectification of abbreviations was the final technique. The experiments are evaluated based on the BLEU score, WER score and time efficiency.

Combinations	BLEU (%)	WER (%)	Time
SC1 → RRC1 → DAB1	86.04	9.21	2mins55sec
<b>SC1 → RRC1 → DAB2</b>	<b>86.94</b>	<b>8.30</b>	<b>2mins55sec</b>
SC1 → RRC2 → DAB1	81.92	11.56	3mins40sec
SC1 → RRC2 → DAB2	82.89	10.67	3mins40sec
SC1 → RRC3 → DAB1	82.93	10.88	24mins30sec
SC1 → RRC3 → DAB2	83.96	9.90	24mins40sec
SC2 → RRC1 → DAB1	75.67	14.17	4mins55sec
SC2 → RRC1 → DAB2	76.98	12.92	4mins55sec
SC2 → RRC2 → DAB1	73.05	15.70	5mins30sec
SC2 → RRC2 → DAB2	74.37	14.49	5mins30sec
SC2 → RRC3 → DAB1	73.89	15.21	26mins30sec
SC2 → RRC3 → DAB2	75.25	13.95	26mins30sec
SC3 → RRC1 → DAB1	73.77	14.56	21mins55sec
SC3 → RRC1 → DAB2	75.48	13.45	21mins55sec
SC3 → RRC2 → DAB1	70.12	16.83	22mins30sec
SC3 → RRC2 → DAB2	71.71	15.76	22mins30sec
SC3 → RRC3 → DAB1	70.82	16.40	43mins30sec
SC3 → RRC3 → DAB2	72.49	15.29	43mins30sec

Table 4.8: The evaluation results of the combination of SC, RRC, and DAB techniques.

This combination group also provides the satisfactory results regarding the high BLEU scores and the low WER values. In Table 4.8, the outstanding combination is SC1 → RRC1 → DAB2, which achieves the highest BLEU score (86.94%) and the lowest WER value (8.30%) and spends 2 minutes and 55 seconds processing normalisation. However, we are surprised by

the results of this combination when we compare it with DAB2 → SC1 → RRC1. We find that the ill-formed words identified as abbreviations are not treated by SC1 at the first stage; only misspelled words are corrected. Thus, abbreviations and repeated letters are normalised by the proper techniques. The normalised results contradict with the fourth combination group, especially, the normalised result from the DAB2 → SC1 → RRC1 technique. It indicates that the original format of some words containing repeated letters has been changed by SC1. While SC1 → RRC1 → DAB2 is the best combination in the group, SC3 → RRC2 → DAB1 and SC3 → RRC3 → DAB1 are the techniques that achieve the lowest BLEU score (~70%) and the highest WER values (>16%). On the other hand, the combinations of SC3 → RRC3 → DAB1 and SC3 → RRC3 → DAB2, which spent the longest run-time in normalising than any other combination, perform better than SC3 → RRC2 → DAB1 on both BLEU score and WER value.

There is a significant difference between SC1 → RRC1 → DAB2 and SC3 → RRC2 → DAB1: the calculated p-values from the BLEU and WER scores are both less than 0.005 at the 95% CI level, 0.0003 and 0.0005 respectively. This mean difference between them is considered to be extremely statistically significant. Furthermore, there is a significant difference between SC1 → RRC1 → DAB2 and SC1 → RRC1 → DAB1. The calculated p-values from the BLEU and WER are both less than 0.005, 0.0036 and 0.0030 respectively. Moreover, the paired t-test results between SC1 → RRC1 → DAB2 and SC1 → RRC3 → DAB2 indicate that their difference is considered to be extremely statistically significant. Their p-values from both BLEU and WER scores are less than 0.005 at the 95% CI level, 0.0009 and 0.0020 respectively. Therefore, the DAB2 → SC1 → RRC1 technique is significantly different to any technique with the BLEU score  $\leq 83\%$  or the WER score  $\geq 9\%$ .

#### **4.3.2.6. The combination of SC → DAB → RRC techniques**

The final combination group began with the correction of misspelled words, followed by the rectification of abbreviations and the final technique was the elimination of repeated characters. The experiments are evaluated based on the BLEU score, WER score and time efficiency.

Combinations	BLEU (%)	WER (%)	Time
SC1 → DAB1 → RRC1	87.27	7.91	2mins55sec
<b>SC1 → DAB2 → RRC1</b>	<b>87.90</b>	<b>7.40</b>	<b>2mins55sec</b>
SC1 → DAB1 → RRC2	83.95	9.54	3mins40sec
SC1 → DAB2 → RRC2	84.87	9.25	3mins40sec
SC1 → DAB1 → RRC3	83.90	9.61	24mins30sec
SC1 → DAB2 → RRC3	84.90	9.11	24mins40sec
SC2 → DAB1 → RRC1	76.60	13.48	4mins55sec
SC2 → DAB2 → RRC1	77.91	12.45	4mins55sec
SC2 → DAB1 → RRC2	75.07	13.93	5mins30sec
SC2 → DAB2 → RRC2	75.39	13.79	5mins30sec
SC2 → DAB1 → RRC3	74.90	14.20	26mins30sec
SC2 → DAB2 → RRC3	76.52	13.32	26mins30sec
SC3 → DAB1 → RRC1	74.80	14.30	21mins55sec
SC3 → DAB2 → RRC1	76.50	13.42	21mins55sec
SC3 → DAB1 → RRC2	71.15	15.87	22mins30sec
SC3 → DAB2 → RRC2	72.91	15.26	22mins30sec
SC3 → DAB1 → RRC3	71.85	15.61	43mins30sec
SC3 → DAB2 → RRC3	73.51	14.79	43mins30sec

Table 4.9: The evaluation results of the combination of SC, DAB, and RRC techniques.

As can be seen in Table 4.9, this group performs slightly better than the previous group regarding the accuracy. More than half of the combined techniques have the highest BLEU scores (over 75%) while the WER values are lower than 13%. The outstanding combination of this group is SC1 → DAB2 → RRC1. This combination achieves 87.90% in the BLEU score and 7.40% in the WER. Based on the normalised result generated from SC1 → DAB2 → RRC1, we find that both abbreviations and repeated characters are not identified as misspelled words and corrected by SC1 at the first stage. Thus, noisy words that have been ignored by SC1 are detected and normalised by DAB2 and then RRC1 as a latter technique.

SC3 → DAB1 → RRC2 and SC3 → DAB1 → RRC3 are the combinations that achieve the lowest BLEU score (~71%) and the highest WER values (>15%). In addition, both SC3 → DAB1 → RRC3 and SC3 → DAB2 → RRC3 have spent nearly 44 minutes normalising noisy tweets. But SC3 → DAB2 → RRC3 seems to perform marginally than SC3 → DAB1 → RRC3 in terms of the BLEU and WER, due to DAB2 being able to resolve abbreviations better than DAB1. Another good example of using the right detecting abbreviations technique along with other proper cleaning techniques can be seen in SC1 → DAB1 → RRC1 and SC1 → DAB2 → RRC1. Based on their highest BLEU scores and lowest WER values, they can be claimed as the top combinations of this group. However, there is a small difference between them. SC1 → DAB2 → RRC1 is slightly better than SC1 → DAB1 → RRC1 because DAB2 assists the combined technique in generating the most correct sentences by resolving abbreviations with a high accuracy.

There is a noticeable difference apparent within this group, and the outstanding difference is between SC1 → DAB2 → RRC1 and SC3 → DAB1 → RRC2. Their p-values generated from the BLEU and WER are 0.0002 and 0.0004. This difference between them is considered to be extremely statistically significant because their p-values are less than 0.005 at the 95% CI level. Meanwhile, the difference between SC1 → DAB2 → RRC1 and SC1 → DAB2 → RRC3 is considered to be very statistically significant as well. The calculated p-values from the BLEU and WER are both less than 0.005, 0.0011 and 0.0019 respectively. Therefore, the DAB2 → SC1 → RRC1 technique is significantly different to any technique with the BLEU score  $\leq 84\%$  or the WER score  $\geq 9\%$ .

### **4.3.3. Baseline Model (Text Cleanser (TC))**

We have selected Text Cleanser (TC) developed by Gouws et al. (2011) as the baseline model for the following reasons. Firstly, Gouws et al. (2011) claimed that the system can handle all types of noisy words, which they consider as OOV words. Secondly, the system is an open source which allows researchers and developers to utilise it in further studies. For these reasons, we think that using this system as the baseline model is the best idea for conducting a comparison with our proposed normalisation technique.

Techniques	BLEU (%)	WER (%)	Time
Text Cleanser (TC)	63.10	38.22	3mins

Table 4.10: The evaluation results of Text Cleanser developed by Gouws et al. (2011) as the baseline model.

Table 4.10 shows the results of the baseline model on our testing dataset. Ref\_All is also used for calculating the BLEU and WER scores of the baseline model. TC achieves 63.10% of the BLEU and 38.22% of the WER on our 1200 tweets dataset. The system spends 3 minutes on generating and decoding noisy tweets. We hypothesise that the absence of the various abbreviation word-lists as well as the capacity in allowing one-to-many normalisation, such as “imma” to “I am going to” and detecting run-on words such as “cu” to “see you”, cause the baseline model to obtain the low BLEU and the high WER score. Furthermore, the model also changes a given word that is already correct into another word; an example can be seen as below.

**Input:** Do you hear me @katyperry im ur big fan lol

**Output:** do you here me @katyperry im your big fan lola

Figure 4.4: The normalised outputs from TC.

As shown by Figure 4.4, TC changes “hear” which is already correct in the sentence to “here”. Its IV-but-invalid tokens detection needs to be improved so that a valid word will not be transformed into another valid word. Furthermore, run-on-words such as “im” cannot be detected and normalised to “I am”. In terms of detecting abbreviations, TC does not convert “lol” to “laughing out loud”, but changes it into “lola” instead. This action indicates that TC cannot normalise abbreviations that have one-to-many case. For example, “a&f” → “always and forever”.

## 4.4. Discussion

This section summarises our experiment results and findings according to the problem statement mentioned in Chapter 1. Based on the evaluation results obtained from the experiments of 8 individual techniques and 108 combinations of normalising techniques, we can conclude the outcome as follows.

The first group of results relate to finding the best technique for each noisy text problem. We discover that the best technique for detecting abbreviations is DAB2 (see Table 4.2), while the best technique for correcting misspelled words is SC1 (see Table 4.3), and RRC1 is the best technique for removing repeated characters (see Table 4.4). To test our first hypothesis that the best combination of best techniques for each problem is the best model, 108 combinations of techniques and execution orders have been tested. Since most of the best results show the positive correlation of two factors (accuracy and running time), it is indicated that the use of less complicated techniques and the proper order of solving each problem provide the best results in this research. As can be seen in Section 4.3.2 and summarised in Table 4.11, the combinations of RRC1, DAB2 and SC1 in different order provide the best results. Thus, we can claim that our first hypothesis is true.

<b>The Best Combinations</b>	<b>BLEU</b>	<b>WER</b>
RRC1 → DAB2 → SC1	88.51%	7.14%
RRC1 → SC1 → DAB2	84.41%	9.60%
<b>DAB2 → RRC1 → SC1</b>	<b>88.55%</b>	<b>7.10%</b>
DAB2 → SC1 → RRC1	83.50%	9.69%
SC1 → RRC1 → DAB2	86.94%	8.30%
SC1 → DAB2 → RRC1	87.90%	7.40%

Table 4.11: The group of the best combinations from six groups.

From Table 4.11, it can be seen that the best ways to clean a noisy text are by expanding abbreviations with CSV file replacement as a first step (DAB2). This is then followed by removing repeated characters with regular expression function along with spell checking (RRC1). The correcting of misspelled words with Enchant dictionary and replace() method is

the final step (SC1). Additionally, there is a significant difference between all of those results in Table 4.11, except RRC1 → DAB2 → SC1 and DAB2 → RRC1 → SC1. Their p-values from the BLEU and WER scores are greater than 0.005 with 95% CI. Thus, the difference between them is considered to be not quite statistically significant. The difference that is extremely significant is between RRC1 → DAB2 → SC1 and DAB2 → SC1 → RRC1. Their calculated p-values from the BLEU and WER scores are less than 0.005, 0.0006 and 0.0012 respectively. Therefore, DAB2 → RRC1 → SC1 is considered to be the best normalisation method from our 108 experiments of normalisation methods. It is considered as our proposed statistical normalisation method for Twitter, the SNET.

To test our final hypothesis, which is the SNET that can detect and convert a noisy tweet into an accurate English sentence; we compare the performance of our method with the performance of Text Cleanser (TC) system developed by Gouws et al. (2011) on the same dataset. We chose TC because it is designed to deal with the same types of noisy problems which we are trying to normalise. Hence, the result of this system can be comparable with our proposed normalising model.

<b>Normalisation models</b>	<b>BLEU (%)</b>	<b>WER (%)</b>	<b>Time</b>
A baseline model (TC)	63	38.22	3mins
The SNET	88.55	7.10	2mins55sec

Table 4.12: The comparison between baseline model and a proposed normalisation model.

As can be seen in Table 4.12, the SNET performs better than TC in terms of achieving higher accuracy. Our model achieves 88.55% of the BLEU score and 7.10% in the WER score, while Gouws et al.’s (2011) model achieves 63% in the BLEU score and 38.22% in the WER score. In terms of time efficiency, although both models spend a short amount of time on normalisation in the different operating systems, our proposed model is somewhat faster than the baseline model. In order to find out whether or not the difference between those two numbers is significant, scores from the BLEU and WER are calculated to generate p-values. Their calculated p-values are less than 0.005 at the 95% CI level, 0.0127 and 0.0001 respectively. By conventional evaluation criteria, this difference is considered to be statistically significant.

Original tweets	Tweets normalised by the TC	Tweets normalised by the SNET
#Grammys: See Last Year's Worst Dressed HERE	#grammys: see last <u>years</u> <u>wrest</u> dresser here	#Grammy: See Last Year's Worst Dressed HERE
@NICKIMINAJ like when You're standing there facing the ocean <b>&amp;</b> the wave comes <b>riight</b> up just to touch Your feet <b>&amp;</b> say hello	@nickiminaj like when <u>your</u> standing there fading the ocean <b>&amp;</b> the wave <u>comer</u> <b>right</b> up just to touch your <u>feat</u> <b>&amp;</b> say hello	@NICKIMINAJ like when You're standing there facing the ocean <b>and</b> the wave comes <b>right</b> up just to touch Your feet <b>and</b> say hello
@KimKardashian what <b>im</b> going <b>2</b> tell them	@kimkardashian what <b>im</b> going <b>2</b> tell them	@KimKardashian what <b>I am</b> going <b>to</b> tell them
YO Impress all <b>ur</b> homies <b>&amp;</b> stay straight <b>stuntin</b> by learning <b>deez</b> lyrics to keep <b>ur</b> party trill: #ThisIsHowWeDo	yo impress all <b>your</b> <u>hamas</u> <b>&amp;</b> stay straight <b>stanton</b> by learning <b>diaz</b> lyric to keep <b>your</b> party <u>troll</u> #thisishowwedo	YO Impress all <b>your</b> <u>homes</u> <b>and</b> stay straight <b>stunting</b> by learning <b>these</b> lyrics to keep <b>your</b> party trill: #ThisIsHowWeDo
@taylorswift13 he <b>ses</b> me I turn around <b>&amp;</b> instead of leaving I decide to go talk to him <b>&amp;</b> there I have some other things I find out <b>conections</b>	@taylorswift13 he <b>ses</b> me i turn around <b>&amp;</b> instead of leaving i decide to go talk to him <b>&amp;</b> there i have some other things i find out <b>conditions</b>	@taylorswift13 he <b>sees</b> me I turn around <b>and</b> instead of leaving I decide to go talk to him <b>and</b> there I have some other things I find out <b>connections</b>
@ParisHilton I' m off to <b>chefin'</b> princess	@parishilton i' m off to <b>chieftain</b> princess	@Paris Hilton <b>I' million</b> off to <b>chaffing</b> princess

Table 4.13: The comparison of normalised output between the TC and the SNET.

Table 4.13 contains the sample of tweets normalised by the TC and the SNET. It can be seen that the normalised tweets generated by a baseline model are less clean than tweets produced by our proposed model. The baseline model could not resolve the problems of some abbreviations and misspelled words, and incorrectly replaced an already correct word with another word. For example, “worst” to “wrest” “deez” to “diaz”, and “conections” to “conditions”. A run-on word such as “im” being replaced with “i am” – is another problem that could not be detected and handled by the baseline model. Furthermore, both the proposed model and the baseline model could not handle “homies”. Actually, “homies” has a meaning - a fellow townspeople or a close friend (homie, n.d.) - but some standard dictionaries have categorised it as informal vocabulary. Thus, this word is not in the dictionaries used either by our proposed model or the baseline model. Both models corrected “homie” to the most likely word by choosing the element with the highest probability value.

Another error that we found in normalised tweets is when there is a whitespace in a given word (i.e. “I\_m” is in the last example shown in Table 4.13). The SNET recognises it as a noisy word due to it not appearing in the dictionary lookup and reference text. The “I\_m” is transformed into “I million” instead of remaining “I m”. By using the tokenising text algorithm, the model treats the white space as a token separator which is not considered a token. Hence, the whitespace in a word will separate one word into two words. Then the “I” is recognised by dictionary lookup while the “m” is not. The “m” is recognised as an abbreviated term which means “million” according to our abbreviation dictionary. Therefore, this minor issue is another factor that reduces the accuracy of our proposed model’s performance in normalising noisy tweets.

Combinations	BLEU (%)	WER (%)	Time
TC → DAB2 → RRC1 → SC1	64.85	36.79	5mins55sec
<b>DAB2 → TC → RRC1 → SC1</b>	<b>67.62</b>	<b>34.97</b>	<b>5mins55sec</b>
DAB2 → RRC1 → TC → SC1	67.60	34.98	5mins55sec
DAB2 → RRC1 → SC1 → TC	66.44	37.27	5mins55sec

Table 4.14: The combination of the SNET with the TC.

Additionally, we combined our proposed model with the baseline model to see whether or not our model could increase the accuracy of the baseline model. As shown in Table 4.14, there are slight improvements, especially by way of the third combination which improves from 63% to 67.60% in the BLEU score and from 38.22% to 34.98% in the WER score. However, the difference is not significant at the 95% CI level. We find that the limitations of the baseline model mentioned in the previous paragraphs bring an unsatisfactory result in these extra experiments, in terms of the low BLEU scores and the high WER scores.

To sum up, although our proposed normalisation model cannot define some informal vocabularies such as slang, it can perform better than the standalone model in terms of accuracy. It not only handles most noisy problems, but also reduces the time taken to clean data while generating the most accurate English sentence. Our proposed model can be considered as state-of-the-art because it contains the most updated techniques to handle noisy tweets and other informal languages used in social media messages. As such, this research represents a great contribution to the field of cleaning noisy tweets before feeding them into NLP applications such as sentiment analysis.

## **4.5. Limitations of Research**

In this section, we describe some threats to the validity of our experiments. Firstly, our dataset is small and focuses on one domain. Because our dataset contains tweets that are related to famous people in the entertainment industry, our abbreviation lookup list mostly contains abbreviations that are commonly used by entertainers and their fans in the social media. Secondly, our reference dataset, containing the correctly normalised tweets, is not manually cleaned by native English speakers. As such, our clean reference might not be as completely clean as we expected. Due to these biases, it is possible that our results cannot be generalised to all noisy tweets from the general population and/or topics.

## **4.6. Chapter Summary**

This section has outlined our experiment methodology and results used to test our hypotheses. Our evaluations showed that the best normalisation performance to clean abbreviations,

misspellings and repeated characters in noisy tweets can be achieved by combining the best technique addressing each problem. The best normalisation orders are resolving abbreviations, removing repeated characters and rectifying the misspelled words. Furthermore, the results also showed that our proposed normalisation method, the SNET, which combines both statistical methods and a dictionary-based approach, performs significantly better than the baseline model. As such the SNET supports all of our hypotheses and answers our problem statement described in Chapter 1.

# Chapter 5: Conclusion and Recommendations

The principal objective of this study is to propose the best normalisation method detecting and converting noisy tweets into accurate English sentences. To accomplish this objective it is necessary to answer the problem statement and evaluate hypotheses discussed in Section 1.2.1 and 1.2.2 respectively. Previous chapters outline the normalisation setups and experiments that we have used in this research to validate results for proving hypotheses as well as answer the problem statement. This chapter summarises the major findings of the research and offer responses to the problem statement. Finally, the conclusion drawn from the study, the significance and limitation of the research, and suggestions for future studies are discussed.

## 5.1. Summary of the Research Objective

Twitter is considered to be one of the world's biggest online social networking services enabling a large amount of users to post and read short "140-character messages" called "tweets" (Alexa, 2016; Twitter, 2016). Twitter not only gives people a better connection to each other but it also contains a vast amount of various data sources, which can be utilised for earning the high profit (Cotelo et al., 2015; Ikeda et al., 2013; Mostafa, 2013; Tumasjan et al., 2010). Despite the many advantages that can Twitter provides, a tweet posted on Twitter

website may not be that useful if it contains irrelevant and unintelligible information. Most text analytics algorithms cannot handle all type of noisy or ill-formed tweets (Brody & Diakopoulos, 2011; Mapa et al., 2012; Pennell & Liu, 2011; Whitelaw et al., 2009). Mainly, due to the restriction on the maximum number of characters, many tweets are ill-formed. Noisy tweets might contain the user's personal short-hands, abbreviations, repeated characters, and misspelled words. Solving all these problems is both tough and challenging (Gouws et al., 2011; Han & Baldwin, 2011)

There are several existing text cleaning techniques proposed by researchers to solve these issues; however they have limitations and still do not achieve good results overall (Han & Baldwin, 2011; Liu et al., 2011; Pennell & Liu, 2011). Furthermore, it is a challenge to select the proper data cleaning techniques out of the various alternatives and to order the chosen techniques in the best way possible during the data cleaning procedure. The best data cleaning process must accurately clean the noisy data in the shortest amount of time possible.

Instead of normalising one type of ill-formed words, we have considered all types of ill-formed words found on the sample tweets and cleaned them under three main categories; misspelled words, abbreviations and repeated letters. The main reason why we have categorised these ill-formed words is because we would like to ensure that all subcategories of these three main problems are normalised into their correct version by the best-suited techniques. Thus, the objective of this research is to find the best normalisation approach in order to efficiently and accurately clean tweets containing misspellings, abbreviations and repeated characters.

To accomplish this objective, the problem statement and hypotheses of this research was outlined in Chapter 1. Related works mentioned in Chapter 2 do not only provide the background knowledge of text normalisation, but also point out where data cleaning techniques need to be improved and built on for both present and future studies. Furthermore, Chapter 3 discussed the algorithms and methods used to tackle misspellings, abbreviations, repeated characters and combinations of them. The experiment setup and evaluation results from the experiments of eight individual techniques and 108 combinations of techniques were analysed in Chapter 4 to prove the research hypotheses.

## 5.2. Significance of Final Findings

The objective of the research has been met because both hypotheses mentioned in Section 1.2.2 are validated.

**H1:** The combination of the best technique for each problem is the best model.

The best cleaning technique for a given problem is one that produces the highest BLEU and the lowest WER score in the shortest time possible. As mentioned in Chapter 3, the first hypothesis was evaluated in two steps. The first step was to find the best technique for each problem. According to the evaluation results shown in Section 4.3.1, we can conclude:

- By detecting 498 abbreviations, the best technique of detecting abbreviations is technique number two (DAB2) with 95.26% in the BLEU score, 2.65% in the WER score and 30 seconds for run-time.
- By correcting 375 misspelled words, the best technique of correcting misspelled words is techniques number one (SC1) with 79.88% in the BLEU score, 12.40% in the WER score and 20 minutes for run-time.
- By cleaning 152 words that contain repeated characters, technique number one (RRC1) is the best, with 83.65% in the BLEU score, 8.89% in the WER score and 20 seconds for run-time.

After finding the best techniques for each problem, the second step of proving the first hypothesis is to find the best combination of techniques and the best ordering of those techniques. Based on the evaluation results displayed in Section 4.3.2, the combination of DAB2, RRC1, and SC1 is the best combined technique, and the best cleaning process is detecting abbreviations first, followed by resolving repeated characters and correcting misspelled words as a final step.

**H2:** The proposed method will produce the most accurate English sentences from noisy tweets with better time efficiency.

Based on the evaluation results in Section 4.4, the second hypothesis has been proved. DAB2 → RRC1 → SC1 is the proposed normalisation method and named the SNET. By comparing it with the baseline model, the SNET achieves a significant result in terms of having a higher accuracy in cleaning all types of noisy words found on tweets. In terms of run-time efficiency, they are not significantly different to each other. However, the proposed approach is the best model to normalise a noisy tweet into an accurate English sentence.

### **5.3. Summary and Future work**

It has been established that data cleaning is a crucial part of text pre-processing. Therefore, a noisy tweet needs to be normalised to a cleaned sentence to provide good quality data. The aims of this research have been met as outlined. The major tasks undertaken for this research included the intensive study of text normalization techniques, the development of a rule-based normalisation method, the evaluation of performance measurement, the comparison of all the techniques studied and the proposed technique.

Three main issues in noisy tweets have been considered in text normalisation. Existing techniques have been evaluated with the same dataset in order to identify and select the best technique to deal with abbreviations, repeated characters, and misspelled words. According to our final finding, the proposed model needs to adopt the following techniques step-by-step to normalise a noisy tweet into an accurate English sentence.

- An abbreviation is converted to its correct form through the use of the abbreviation dictionary lookup.
- Repeated characters are removed through the use of regular expression, and then each word is rechecked through using Aspell dictionary.
- Misspelled words are corrected through Enchant library and used Aspell dictionary.

Based on our experiment with 108 combinations of cleaning techniques, the above order and combination provides the best performance. It not only provides the highest score of the BLEU score and the lowest WER, but also generates sentences with minimum dirty data;

therefore the cleaned texts can be effectively used in sentiment analysis and other NLP applications.

Nevertheless, there is still room for improvement for future work. Our dataset itself has some limitation regarding its overall size, which is not large, and is applied on one domain only. Further exploration is needed to see if applying the SNET on different domains will yield a similar result. Besides that, this research merely focuses on dealing with noisy tweets at the character level. We believe that we would achieve higher accuracy and performance if sentence-level cleaning techniques are included in our study, therefore improving our method, such as correcting ambiguous abbreviations.

The two techniques for removing repeated characters (RRC2 and RRC3), which have low accuracy, can be enhanced by incorporating name entity recognition. Since there are many names containing repeated characters which appear on the tweets dataset, the function of name recognition may assist in recognising a named entity and maintain it without removing any repeated letters. Another solution is to set up a simple rule to ignore a word that starts with “@” and “#”. Then the Twitter username and hashtag do not change.

Furthermore, the limitation of dictionary based approach is another issue needed to be improved. Some vocabularies are considered to be the informal word in dictionary, but technically they have their meaning which is denied by standard dictionaries. When this type of vocabulary has transformed to another word based on the use of standard dictionary lookup, the meaning of sentence might be changed as well. Hence, the use of normalisation dictionary contained special vocabularies might solve this problem.

Our normalisation method, the SNET, can be improved by linking abbreviations lookup with web pages that contain Internet language such as abbreviations and are regularly updated. By coding a function that can update an abbreviation list from the Internet the normalisation model should be able to detect more abbreviations. All these challenges are possible areas for future work.

In conclusion, normalising noisy tweets into the most accurate version is a crucial step to achieving better quality data analysis in many real-world applications. The SNET has been proven to achieve higher accuracy and performance than the baseline model developed by Gouws et al. (2011). However, there is room for further improvements which are mentioned on previous paragraphs. We believe that our research makes contributions and provides benefits to our research fields and communities and advances the current stage of the related research overall.



- Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., & Basu, A. (2007). Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJ DAR)*, 10(3-4), 157-174.
- Clark, E., & Araki, K. (2011). Text normalization in social media: Progress, problems and applications for a pre-processing system of casual English. *Procedia - Social and Behavioral Sciences*, 27(0), 2-11. doi: <http://dx.doi.org/10.1016/j.sbspro.2011.10.577>
- Conover, M., Ratkiewicz, J., Francisco, M. R., Gonçalves, B., Menczer, F., & Flammini, A. (2011). Political Polarization on Twitter. *The International Conference on Weblogs and Social Media*, 133, 89-96.
- Contractor, D., Faruque, T. A., & Subramaniam, L. V. (2010). Unsupervised cleansing of noisy text. *Proceedings of the 23rd International Conference on Computational Linguistics: Posters* (pp. 189-196). Stroudsburg, USA: Association for Computational Linguistics.
- Daintith, J. (2004). Statistical methods. Retrieved from <http://www.encyclopedia.com/doc/1O11-statisticalmethods.html>
- Davidov, D., Tsur, O., & Rappoport, A. (2010). Enhanced sentiment learning using twitter hashtags and smileys. *Proceedings of the 23rd international conference on computational linguistics: posters* (pp. 241-249). Stroudsburg, USA: Association for Computational Linguistics.
- Denoual, E., & Lepage, Y. (2005). BLEU in characters: towards automatic MT evaluation in languages without word delimiters. *Proceedings of the Second International Joint Conference on Natural Language Processing* (pp. 79-84). Association for Computational Linguistics.
- Derczynski, L., Maynard, D., Rizzo, G., van Erp, M., Gorrell, G., Troncy, R., Petrak, J. & Bontcheva, K. (2015). Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2), 32-49. doi: <http://dx.doi.org/10.1016/j.ipm.2014.10.006>
- Diakopoulos, N. A., & Shamma, D. A. (2010). Characterizing debate performance via aggregated twitter sentiment. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1195-1198). New York, USA: Association for Computing Machinery. doi: 10.1145/1753326.1753504
- Dodge, Y. (2006). *The Oxford dictionary of statistical terms*: Oxford University Press on Demand.
- Dolinar, S. (2015). Collecting Twitter data: Getting started. Retrieved from <http://stats.seandolinar.com/collecting-twitter-data-getting-started/>
- Dreyer, M., & Marcu, D. (2012). Hyter: Meaning-equivalent semantics for translation evaluation. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 162-171). Stroudsburg, USA: Association for Computational Linguistics.
- Dutta, S., Saha, T., Banerjee, S., & Naskar, S. K. (2015). Text normalization in code-mixed social media text. *Proceedings for the Recent Trends in Information Systems (ReTIS), IEEE 2nd International Conference on* (pp. 378-382). IEEE.
- Eddy, S. R. (1996). Hidden markov models. *Current opinion in structural biology*, 6(3), 361-365.
- Egorov, S., Yuryev, A., & Daraselia, N. (2004). A simple and practical dictionary-based approach for identification of proteins in Medline abstracts. *Journal of the American Medical Informatics Association*, 11(3), 174-178.
- Evermann, G. (1999). Minimum word error rate decoding. *Cambridge University, UK*, 45-67.

- Fayyad, U. M. (1996). Data Mining and Knowledge Discovery: Making Sense Out of Data. *IEEE Expert: Intelligent Systems and Their Applications*, 11(5), 20-25. doi: 10.1109/64.539013
- Fink, G. A. (2014). n-Gram Models. In G.Fink (Ed.), *Markov Models for Pattern Recognition* (pp. 107-127). London, UK: Springer.
- Ghiassi, M., Skinner, J., & Zimbra, D. (2013). Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network. *Expert Systems with applications*, 40(16), 6266-6282.
- Gouws, S., Hovy, D., & Metzler, D. (2011). Unsupervised mining of lexical variants from noisy text. *Proceedings of the First workshop on Unsupervised Learning in NLP* (pp. 82-90). Stroudsburg, USA: Association for Computational Linguistics.
- Grunwald, D., Lindsay, D., & Zorn, B. (1998). Static methods in hybrid branch prediction. *Proceedings for the Parallel Architectures and Compilation Techniques, 1998. Proceedings. 1998 International Conference on* (pp. 222-229). IEEE.
- Han, B., & Baldwin, T. (2011). Lexical normalisation of short text messages: Makn sens a# twitter. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (pp. 368-378). Stroudsburg, USA: Association for Computational Linguistics.
- Han, B., Cook, P., & Baldwin, T. (2012). Automatically constructing a normalisation dictionary for microblogs. *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning* (pp. 421-432). Stroudsburg, USA: Association for Computational Linguistics.
- Han, B., Cook, P., & Baldwin, T. (2013). Lexical normalization for social media text. *ACM Trans. Intell. Syst. Technol.*, 4(1), 1-27. doi: 10.1145/2414425.2414430
- Hassan, H., & Menezes, A. (2013, August). Social Text Normalization using Contextual Graph Random Walks. *Association for Computational Linguistics (1)* (pp. 1577-1586).
- He, X., Deng, L., & Acero, A. (2011, May). Why word error rate is not a good metric for speech recognizer training for the speech translation task?. *Proceedings for the Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on* (pp. 5632-5635). IEEE.
- homie. (n.d.). *The American Heritage® Abbreviations Dictionary, Third Edition*. Retrieved from <http://www.dictionary.com/browse/homie>
- Idzelis, M. (2005). Jazzy: The java open source spell checker. Retrieved from <http://jazzy.sourceforge.net/>
- Ikeda, K., Hattori, G., Ono, C., Asoh, H., & Higashino, T. (2013). Twitter user profiling based on text and community mining for market analysis. *Knowledge-Based Systems*, 51(0), 35-47. doi: <http://dx.doi.org/10.1016/j.knosys.2013.06.020>
- Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical review*, 106(4), 620.
- Jizba, R. (2000). *Measuring search effectiveness*. Retrieved from [https://www.creighton.edu/fileadmin/user/HSL/docs/ref/Searching\\_-\\_Recall\\_Precision.pdf](https://www.creighton.edu/fileadmin/user/HSL/docs/ref/Searching_-_Recall_Precision.pdf)
- Kaufmann, M., & Kalita, J. (2010). Syntactic normalization of twitter messages. *International conference on natural language processing*, 1-7.
- Kelly, R. (2015). PyEnchant. *a spellchecking library for Python*. Retrieved from <http://pythonhosted.org/pyenchant/>
- Kernighan, M. D., Church, K. W., & Gale, W. A. (1990, August). A spelling correction program based on a noisy channel model. *Proceedings of the 13th conference on*

- Computational linguistics-Volume 2* (pp. 205-210). Stroudsburg, USA: Association for Computational Linguistics.
- Klakow, D., & Peters, J. (2002). Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2), 19-28. doi: [http://dx.doi.org/10.1016/S0167-6393\(01\)00041-3](http://dx.doi.org/10.1016/S0167-6393(01)00041-3)
- Klein, D. & Manning, C. D. (2003). *Fast Exact Inference with a Factored Model for Natural Language Parsing*. Retrieved from <http://www-nlp.stanford.edu/~manning/papers/lex-parser.pdf>
- Knoblock, C., Lopresti, D., Roy, S., & Subramaniam, L. V. (2007). Special issue on noisy text analytics. *International Journal on Document Analysis and Recognition*, 10(3), 127-128.
- Kobus, C., Yvon, F., & Damnati, G. (2008, August). Normalizing SMS: are two metaphors better than one?. *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1* (pp. 441-448). Stroudsburg, USA: Association for Computational Linguistics.
- Kuchling, A. M. (2015). Regular Expression HOWTO. Retrieved from <https://docs.python.org/3.5/howto/regex.html>
- Lachowicz, D. (2010). Enchant. Retrieved from <http://www.abisource.com/projects/enchant/>
- Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001)* (pp. 282-289). Association for Computing Machinery.
- Lawson, M. V. (2005). Finite Automata. In D. Hristu-Varsakelis & W. S. Levine (Eds.), *Handbook of Networked and Embedded Control Systems* (pp. 117-143). Boston, MA: Birkhäuser Boston. doi:10.1007/0-8176-4404-0\_6
- Lenzo, K. (n.d.). The CMU Pronouncing Dictionary. Retrieved from <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>
- Li, C., & Liu, Y. (2012). Normalization of Text Messages Using Character-and Phone-based Machine Translation Approaches. In *INTERSPEECH* (pp. 2330-2333).
- Li, C., & Liu, Y. (2014). Improving Text Normalization via Unsupervised Model and Discriminative Reranking. *Proceedings of the Association for Computational Linguistics 2014 Student Research Workshop* (pp. 86-93). Association for Computational Linguistics.
- Ling, W., Dyer, C., Black, A. W., & Trancoso, I. (2013). Paraphrasing 4 Microblog Normalization. In *Proceedings of the Empirical Methods in Natural Language Processing*.
- Liu, F., Weng, F., & Jiang, X. (2012). A broad-coverage normalization system for social media language. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1* (pp. 1035-1044). Jeju Island, Korea: Association for Computational Linguistics.
- Liu, F., Weng, F., Wang, B., & Liu, Y. (2011). Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2* (pp. 71-76). Stroudsburg, USA: Association for Computational Linguistics.
- Loper, E., & Bird, S. (2002). NLTK: The natural language toolkit. *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language*

- processing and computational linguistics-Volume 1* (pp. 63-70). Stroudsburg, USA: Association for Computational Linguistics.
- Lopresti, D., Roy, S., Schulz, K., & Subramaniam, L. V. (2011). Special issue on noisy text analytics. *International Journal on Document Analysis and Recognition*, 14(2), 111-112.
- Loria, S. (2015). TextBlob: Simplified Text Processing. Retrieved from <https://textblob.readthedocs.org/en/dev/>
- Luebbers, D., Grimmer, U., & Jarke, M. (2003). Systematic development of data mining-based data quality tools. *Proceedings of the 29th international conference on Very Large Data Bases-Volume 29* (pp. 548-559). VLDB Endowment.
- MacEwen, R. (2009). Benefits of Implementation - Clean Data. Retrieved from <http://www.audaxium.com/benefits-of-implementation-clean-data/>
- Madnani, N. (2011). iBLEU: Interactively debugging and scoring statistical machine translation systems. *Proceedings for the Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on* (pp. 213-214). IEEE.
- Manago, M., & Kodratoff, Y. (1987). Noise and Knowledge Acquisition. *Proceedings of the 10th international joint conference on Artificial intelligence - Volume 1* (pp. 348-354). Milan, Italy: International Joint Committee on Artificial Intelligence.
- Mapa, E., Wattaladeniya, L., Chathuranga, C., Dassanayake, S., Silva, N. D., Kohomban, U., & Maldeniya, D. (2012). Text Normalization in Social Media by using Spell Correction and Dictionary Based Approach. *Systems learning, 1*, 1-6.
- Martin, R. C. (2009). *Clean code: A handbook of agile software craftsmanship*. Massachusetts, USA: Prentice Hall.
- Marton, Y., & Zitouni, I. (2014). Transliteration normalization for Information Extraction and Machine Translation. *Journal of King Saud University - Computer and Information Sciences*, 26(4), 379-387. doi: <http://dx.doi.org/10.1016/j.jksuci.2014.06.011>
- McCallum, A. (2002, August). Efficiently inducing features of conditional random fields. *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence* (pp. 403-410). Morgan Kaufmann Publishers Inc..
- McCallum, J. (2014). Python 3 Spelling Corrector. Retrieved from <https://pypi.python.org/pypi/autocorrect/0.1.0>
- McCullagh, P. (2002). What is a statistical model?. *The Annals of Statistics*, 30(5), 1225–1310.
- Melamed, I. D., Green, R., & Turian, J. P. (2003). Precision and recall of machine translation. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003--short papers-Volume 2* (pp. 61-63). Edmonton, Canada: Association for Computational Linguistics. doi:10.3115/1073483.1073504
- Milanovic, R. (2015). The World's 21 Most Important Social Media Sites and Apps in 2015. Retrieved from <http://www.socialmediatoday.com/social-networks/2015-04-13/worlds-21-most-important-social-media-sites-and-apps-2015#sthash.KSgXJicq.dpuf>
- Miller, G. A. (1995). WordNet: a lexical database for English. *Commun. ACM*, 38(11), 39-41. doi: 10.1145/219717.219748
- Mitkov, R. (2005). *The Oxford handbook of computational linguistics*. New York, USA: Oxford University Press.
- Mostafa, M. M. (2013). More than words: Social networks' text mining for consumer brand sentiments. *Expert Systems with Applications*, 40(10), 4241-4251. doi: <http://dx.doi.org/10.1016/j.eswa.2013.01.019>

- Nelken, R. (2012). Natural Language Processing: What is a noisy channel model?. Retrieved from <https://www.quora.com/Natural-Language-Processing/What-is-a-noisy-channel-model>
- Nigro, H. O. (2007). *Data Mining with Ontologies: Implementations, Findings, and Frameworks: Implementations, Findings, and Frameworks*. New York, USA: IGI Global.
- NLTK. (2015). Natural Language Toolkit. Retrieved from <http://www.nltk.org/>
- Norvig, P. (2012). How to Write a Spelling Corrector. Retrieved from <http://norvig.com/spell-correct.html>
- O'Connor, B., Krieger, M., & Ahn, D. (2010). TweetMotif: Exploratory Search and Topic Summarization for Twitter. *Proceedings of the Fourth AAAI ICWSM* (pp. 384-385). International Conference on Weblog and Social Media.
- Olive, J., Christianson, C., & McCary, J. (2011). *Handbook of natural language processing and machine translation: DARPA global autonomous language exploitation*. New York, USA: Springer Science & Business Media.
- Olsen, R. G., & Willis, M. C. (1996). A comparison of exact and quasi-static methods for evaluating grounding systems at high frequencies. *Power Delivery, IEEE Transactions on, 11*(2), 1071-1081.
- Pak, A., & Paroubek, P. (2010, May). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. *Proceedings of the Seventh conference on International Language Resources and Evaluation - Volume 10* (pp. 1320-1326). Valletta, Malta: Language Resources and Evaluation Conference.
- Palmer, D. D., & Ostendorf, M. (2005). Improving out-of-vocabulary name resolution. *Computer Speech & Language, 19*(1), 107-128. doi: <http://dx.doi.org/10.1016/j.csl.2004.03.002>
- Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 311-318). Stroudsburg, USA: Association for Computational Linguistics.
- Peng, T. (2008). A framework for data cleaning in data warehouses. *Proceedings of the 10th International Conference on Enterprise Information Systems (ICEIS)* (pp. 473-478). Edinburgh, UK: Springer Verlag.
- Pennell, D., & Liu, Y. (2011). A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations. *Proceedings of the 5th International Joint Conference on Natural Language Processing*. (pp. 974-982). Chiang Mai, Thailand: Asian Federation of Natural Language Processing.
- Pennell, D. L., & Liu, Y. (2014). Normalization of informal text. *Computer Speech & Language, 28*(1), 256-277. doi: <http://dx.doi.org/10.1016/j.csl.2013.07.001>
- Pennell, D. L., Ng, V., Hansen, J. H., & Schweitzer, H. (2011). *Normalization of informal text for text-to-speech* (Doctoral dissertation). The University of Texas at Dallas, Texas, USA.
- Pennell, D., & Liu, Y. (2011). Toward text message normalization: Modeling abbreviation generation. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on* (pp. 5364-5367). IEEE.
- Perkins, J. (2010). *Python Text Processing with NLTK 2.0 Cookbook*. Birmingham, UK: Packt Publishing.
- Perkins, J. (2014). *Python 3 Text Processing with NLTK 3 Cookbook*. Birmingham, UK: Packt Publishing.

- Petrovic, S., Osborne, M., & Lavrenko, V. (2010). The edinburgh twitter corpus. *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media* (pp. 25-26). Stroudsburg, USA: Association for Computational Linguistics.
- Ponte, J. M., & Croft, W. B. (1998, August). A language modeling approach to information retrieval. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 275-281). New York, USA: Association for Computing Machinery.
- Pritchard, S. (2012). How to manage unstructured data for business benefit. Retrieved from <http://www.computerweekly.com/feature/How-to-manage-unstructured-data-for-business-benefit>
- Python. (2015a). Common string operations. Retrieved from <https://docs.python.org/2/library/string.html>
- Python. (2015b). Regular expression operations. Retrieved from <https://docs.python.org/2/library/re.html>
- Python. (2016). html — HyperText Markup Language support. Retrieved from <https://docs.python.org/3.4/library/html.html?highlight=html.unescape#html.unescape>
- Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1), 4-16. doi: 10.1109/MASSP.1986.1165342
- Raghavan, V., Bollmann, P., & Jung, G. S. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *Association for Computing Machinery Trans. Inf. Syst.*, 7(3), 205-229. doi: 10.1145/65943.65945
- Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Engineering* 23(4), 3-13.
- Ramage, D., Dumais, S. T., & Liebling, D. J. (2010). Characterizing Microblogs with Topic Models. *International Conference on Weblogs and Social Media 10*, 130-137.
- Raybaud, S., Lavecchia, C., Langlois, D., & Smaili, K. (2009). Word-and sentence-level confidence measures for machine translation. *Proceedings of the 13th Annual Conference of the EAMT 09* (pp. 104–111). Barcelona, Spain: European Association for Machine Translation.
- Rossi, P. H., Lipsey, M. W., & Freeman, H. E. (2003). *Evaluation: A systematic approach* (7<sup>th</sup> ed.). California, USA: Sage publications.
- Rouse, M. (2012). noisy text. Retrieved from <http://searchbusinessanalytics.techtarget.com/definition/noisy-text>
- Roy, S., Dhar, S., Bhattacharjee, S., & Das, A. (2013). A lexicon based algorithm for noisy text normalization as pre processing for sentiment analysis. *International Journal of Research in Engineering and Technology*, 2(2), 67-70.
- Saloot, M. A., Idris, N., & Aw, A. (2014). Noisy Text Normalization Using an Enhanced Language Model. *Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition* (pp. 111-122). Kuala Lumpur, Malaysia: The Society of Digital Information and Wireless Communications.
- Saloot, M. A., Idris, N., & Mahmud, R. (2014). An architecture for Malay Tweet normalization. *Information Processing & Management*, 50(5), 621-633. doi: <http://dx.doi.org/10.1016/j.ipm.2014.04.009>
- Saloot, M. A., Idris, N., Shuib, L., Raj, R. G., & Aw, A. (2015). Toward Tweets Normalization Using Maximum Entropy. *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (pp. 19-27). Beijing, China: Association for Computational Linguistics.

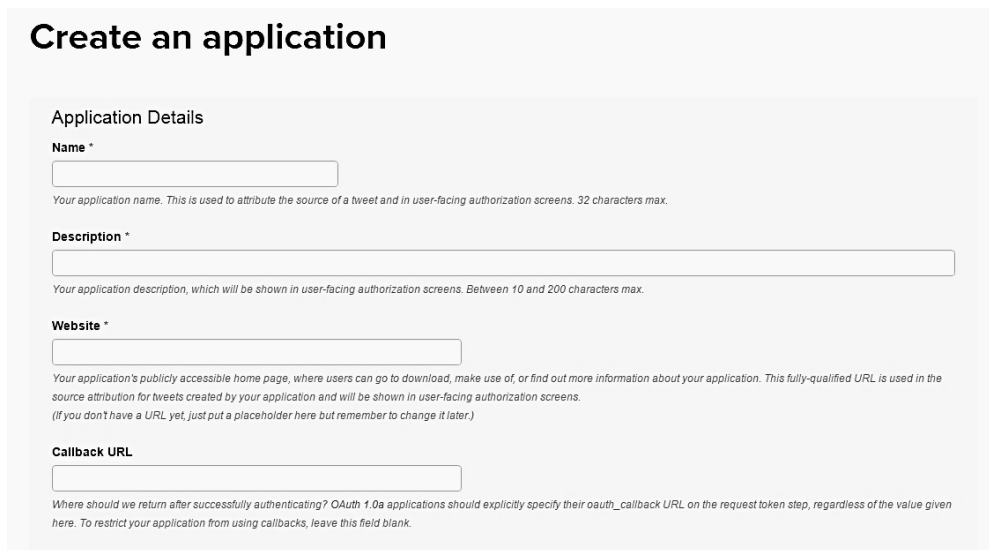
- Schaaf, T. (2001, September). Detection of OOV words using generalized word models and a semantic class language model. *Proceedings of the 7th European Conference on Speech Communication and Technology* (pp. 2581-2584). Aalborg, Denmark: INTERSPEECH.
- Schlippe, T., Zhu, C., Gebhardt, J., & Schultz, T. (2010). Text normalization based on statistical machine translation and internet user support. *Proceedings of the 11th Annual Conference of the International Speech Communication Association* (pp. 1816-1819). Makuhari, Japan: INTERSPEECH.
- Settles, B. (2004). Biomedical named entity recognition using conditional random fields and rich feature sets. *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* (pp. 104-107). Stroudsburg, USA: Association for Computational Linguistics.
- Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27, 379-423.
- Sharifi, B., Hutton, M. A., & Kalita, J. K. (2010). Experiments in microblog summarization. *Proceedings of the Social Computing (SocialCom), IEEE Second International Conference on* (pp. 49-56). Minnesota, USA: IEEE.
- Shier, R. (2004). *Paired t-tests*. Retrieved from <http://www.statstutor.ac.uk/resources/uploaded/paired-t-test.pdf>
- Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., & Richards, C. (2001). Normalization of non-standard words. *Computer Speech & Language*, 15(3), 287-333.
- SRI-International. (2011). SRILM - The SRI Language Modeling Toolkit. Retrieved from <http://www.speech.sri.com/projects/srilm/>
- Stolcke, A. (2002). SRILM-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing* (pp. 257-286). INTERSPEECH.
- Subramaniam, L. V., Roy, S., Faruque, T. A., & Negi, S. (2009). A survey of types of text noise and techniques to handle noisy text. *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data* (pp. 115-122). New York, USA: Association for Computing Machinery. doi:10.1145/1568296.1568315
- Teevan, J., Ramage, D., & Morris, M. R. (2011). # TwitterSearch: a comparison of microblog search and web search. *Proceedings of the fourth ACM international conference on Web search and data mining* (pp. 35-44). New York, USA: Association for Computing Machinery. doi:10.1145/1935826.1935842
- Thelwall, M., Buckley, K., & Paltoglou, G. (2011). Sentiment in Twitter events. *Journal of the American Society for Information Science and Technology*, 62(2), 406-418. doi: 10.1002/asi.21462
- Thoma, M. (2013). Word Error Rate Calculation. Retrieved from <http://martin-thoma.com/word-error-rate-calculation/>
- Tomás, J., Mas, J. À., & Casacuberta, F. (2003). A quantitative method for machine translation evaluation. *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?* (pp. 27-34). Stroudsburg, USA: Association for Computational Linguistics.
- Tumasjan, A., Sprenger, T. O., Sandner, P. G., & Welp, I. M. (2010). Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. *International Conference on Weblogs and Social Media*, 10, 178-185.

- Twitter. (2015a). Documentation. Retrieved from <https://dev.twitter.com/overview/documentation>
- Twitter. (2015b). Twitter usage. Retrieved from <https://about.twitter.com/company>
- Twitter. (2016). Twitter Usage/ Company Facts. Retrieved from <https://about.twitter.com/company>
- Wallach, H. M. (2004). Conditional random fields: An introduction. *Technical Reports (CIS)*, 22.
- Wang, P., & Ng, H. T. (2013). A Beam-Search Decoder for Normalization of Social Media Text with Application to Machine Translation. *Proceedings of NAACL-HLT* (pp. 471–481). Georgia, USA: Association for Computational Linguistics.
- Whitelaw, C., Hutchinson, B., Chung, G. Y., & Ellis, G. (2009). Using the web for language independent spellchecking and autocorrection. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2* (pp. 890-899). Stroudsburg, USA: Association for Computational Linguistics.
- Wholey, J. S. (1979). *Evaluation: Promise and performance*. Washington, DC., USA: Urban Institute Washington, DC.
- Xue, Z., Yin, D., Davison, B. D., & Davison, B. (2011). Normalizing microtext. *Analyzing Microtext*, 11, 74–79.
- Yang, Y., & Eisenstein, J. (2013). A Log-Linear Model for Unsupervised Text Normalization. *Proceedings of the Empirical Methods in Natural Language Processing* (pp. 61-72). Association for Computational Linguistics.
- Zechner, K., & Waibel, A. (2000, April). Minimizing word error rate in textual summaries of spoken language. *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference* (pp. 186-193). Stroudsburg, USA: Association for Computational Linguistics.
- Zhang, C., Baldwin, T., Ho, H., Kimelfeld, B., & Li, Y. (2013). Adaptive Parser-Centric Text Normalization. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (1)* (pp. 1159-1168). Sofia, Bulgaria: Association for Computational Linguistics.
- Zitnick, C. L. (2013). Handwriting beautification using token means. *ACM Transactions on Graphics (TOG)*, 32(4), 1-8. doi: 10.1145/2461912.2461985

## Appendix - Twitter Data Collection

The following steps are Twitter data are collected. In order to comply with the requirements, it is necessary to have a Twitter account to create an “application” using the following steps:

1. Sign into Twitter.
2. Go to <https://apps.twitter.com/app/new> and create a new application by filling the form shown below in Figure 3.2.



**Create an application**

**Application Details**

**Name \***

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

**Description \***

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

**Website \***

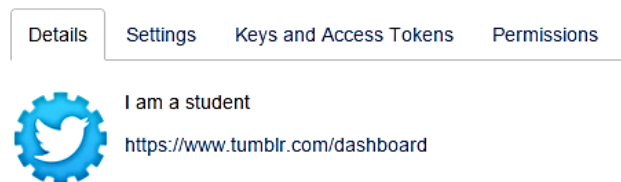
Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.  
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

**Callback URL**

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth\_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Screenshot of creating a new application page on Twitter’s development website

3. Click on “Keys and Access Tokens” on the application page.



Screenshot of the page after creating a new application

4. Obtain and copy your Consumer Key, Consumer Secret Key, Access Token, and Secret Token.

### Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	IrtvjdkwiyPnMBHGGrRaBLHmZ
Consumer Secret (API Secret)	NcfbYepi3C9bVGqcPitUsG0tLR1iLAsCCcBK1epaVv6rID1bON
Access Level	Read and write (modify app permissions)
Owner	thongiane
Owner ID	4750726099

### Application Actions

### Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	4750726099-0ltbhowc08fDnRYmlm5RdHrsJNKdXYr5iaLehB
Access Token Secret	ga5vGqJ7X3iYGksPR1X1wicbIfEasl0KmRzN7RWdFnnPy
Access Level	Read and write
Owner	thongiane
Owner ID	4750726099

### Token Actions

## Screenshot of API keys

- After receiving the API keys, we follow the tutorial of collecting Twitter data published by Dolinar (2015) to download sample tweets. He suggests using R for Tweet scraping due to its simplicity in collecting and parsing data without significant understanding of programming. For this procedure, like most R scripts, firstly the libraries should be installed and called.

```
#install.packages("streamR")
#install.packages("ROAuth")
library(ROAuth)
library(streamR)
```

## Screenshot of R commands for installing packages

- The API keys acquired from Twitter's development website are used in the first part of the code for a Twitter scrubber. Personal API keys are inserted where the **\*\*KEY\*\*** is in

the code. For this method of verification in R, it merely uses the CONSUMER KEY and CONSUMER SECRET KEY and then it gets ACCESS TOKEN from a PIN number using a web address that is opened in the browser.

```
#create your OAuth credential
credential <- OAuthFactory$new(consumerKey='**CONSUMER KEY**',
                              consumerSecret='**CONSUMER SECRET KEY**',
                              requestURL='https://api.twitter.com/oauth/request_token',
                              accessURL='https://api.twitter.com/oauth/access_token',
                              authURL='https://api.twitter.com/oauth/authorize')

#authentication process
options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))
download.file(url="http://curl.haxx.se/ca/cacert.pem", destfile="cacert.pem")
credential$handshake(cainfo="cacert.pem")
```

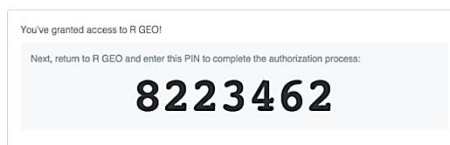
Screenshot of R commands for authentication in Twitter streaming API

7. After the code is executed correctly, R will output the following in the console:

```
> credential$handshake(cainfo="cacert.pem")
To enable the connection, please direct your web browser to:
https://api.twitter.com/oauth/authorize?oauth_token=nWhcHceqTtQetPr0
When complete, record the PIN given to you and provide it here:
```

Screenshot of URL that is used to get the PIN

8. After logging into Twitter and authorising the application, we copy the **Error! Hyperlink reference not valid.** from the console into a browser for obtaining the PIN number. When the PIN number appears, we copy it into the R console and press enter.



Screenshot of PIN number

9. Subsequently, the validation is completed; R is able to utilise those authorisations to make API calls. The `filterStream()` function in the `streamR` package is used to call the Streaming API. This function will connect us to Twitter's stream based on the amount of time that has been adjusted or the certain number of tweets that will be collected.

```
#function to actually scrape Twitter
filterStream( file.name="tweets_test.json",
              track="twitter", tweets=1000, oauth=cred, timeout=10, lang='en' )
```

Screenshot of R commands for connecting to Twitter's stream

10. The `filterStream()` will stay open as long as it is instructed to in the `timeout` parameter [in seconds], so we don't set it too long if we want our data rapidly. The Twitter data returns in a `.json` file, which is a JavaScript data file.

```
"created_at":"Fri Jan 23 13:13:31 +0000 2015", 1
  "id":"558613499606478849",
  "id_str":"558613499606478849",
  "text":"Looking for a new #job to make a fresh start in 2015? Join GLC as a load control ag 2
0Qi4Lunfby",
  "source":"\u003ca href=\"http://twitter.com\" rel=\"nofollow\"\u003eTwitter Web Client\u00
  "truncated":false,
  "in_reply_to_status_id":null,
  "in_reply_to_status_id_str":null,
  "in_reply_to_user_id":null,
  "in_reply_to_user_id_str":null,
  "in_reply_to_screen_name":null,
  "user":{"
    "id":228715466,
    "id_str":"228715466",
    "name":"Global Load Control",
    "screen_name":"GlobalLoadContr",
```

Screenshot of detail of each tweet in JSON file

11. The JSON file is an extract from a tweet that has been organising to be easier to read. (1) is a time stamp which shows the date and time the Tweet was posted. (2) is the Tweet content. Then we use the `parseTweets()` function that is additionally in `streamR` to parse the Twitter data into something useful for our research.

```
#Parses the tweets
tweet_df <- parseTweets(tweets='tweets_test.json')
```

### Screenshot of R command for parsing Twitter data

12. The parseTweets() is a simple function that reads the JSON file produced by filterStream() function, and converts the file to a full data frame shown below in Figure 3.12. Obviously, a full version of the data frame can be overwhelming compared to a single JSON file, because there is so much metadata accessible that may be useful in some areas of text normalising projects. However, we only require tweet content, so the rest of the data is discarded. We select tweets with their “text” value and save them as a CSV file.

	text	retweet_count	favorited	truncated	id_str
1	apologizing to a very nedy cat for being gone so long.	1	FALSE	FALSE	1042494957
2	leaving Lawrence Kansas. Colege I wil mis you. Thanks for leting me sit in on abi is journalism clas.	0	FALSE	FALSE	1649345788
3	Hanging with Rachel from @glorianatheband Liz and Caitlin in the dresing rom in S. Dakota. Lady A are here tonight to watch the show!	0	FALSE	FALSE	2857749494
4	...there is a shewolf in the closet open up and set us fre... There is a shewolf in the closet... Let her out so it can breath... Ao!!!	3	FALSE	FALSE	4163066509
5	Studio in Sin City. Getting my foting back & experimenting...w/TRICKY STEWART! There is an elevator from my rom to studio perfect. Focus.	568	FALSE	FALSE	5952681142
6	- monkey busines.	237	FALSE	FALSE	7429666114
7	@logandoesntwet trust! Is that a north face jacket not a chanel?! I bet the mountain people are apaled!	67	FALSE	FALSE	9403744150
8	I have ben eating Cherios like it is my job lately.	7	FALSE	FALSE	1362126526
9	Gaga is singing Telephone! I am kinda busy!	6835	FALSE	FALSE	1931939339
10	frot lops wit no milk. #GivesmeLIFE	469	FALSE	FALSE	2271764652
11	Who is ready to se #FIREWORK!!! I have got it for you I am so proud to present...	6903	FALSE	FALSE	2896792753

Screenshot of the final Twitter data collected API in CSV format.