TEXT NORMALIZATION BASED ON ERROR TYPE USING PRE-TRAINED LANGUAGE MODEL

by

Youlim Ko

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Eric T. Matson, Co-Chair

Department of Computer and Information Technology

Dr. Baijian Yang, Co-Chair

Department of Computer and Information Technology

Dr. Julia M. Rayz

Department of Computer and Information Technology

Approved by:

Dr. John A. Springer

This thesis is dedicated to my family for all their love and support.

ACKNOWLEDGMENTS

I wish to gratefully acknowledge my advisors, Dr. Eric T. Matson and Dr. Baijian Yang, for all their guidance and support, as well as the confidence they have provided me throughout this process from the beginning to the end. I would also like to extend my gratitude towards my committee member, Dr. Julia M. Rayz, for her insightful comments and guidance in the field of Natural Language Processing. This thesis could not have been completed without the kind support from all of you, I am truly grateful.

Furthermore, I would like to thank Dr. Rob van der Goot, for his assistance in the MoNoise model as well as his helpful feedbacks. Finally, I would like to express my love and gratitude towards my family for all their support and encouragement throughout my studies.

TABLE OF CONTENTS

LIST OF TABLES 7

LIST OF FIGURES 8

LIST OF ABBREVIATIONS 9

ABSTRACT 10

CHAPTER 1. INTRODUCTION 11

1.1 Background 11

1.2 Problem Statement 14

1.3 Research Question 17

1.4 Significance 17

1.5 Assumptions 18

1.6 Limitations 19

1.7 Delimitations 19

1.8 Summary 19

CHAPTER 2. REVIEW OF LITERATURE 20

2.1 Normalization 20

2.1.1 Traditional Approaches 20

2.1.1.1 Spelling Correction Approaches 20

2.1.1.2 Machine Translation Approaches 21

2.1.1.3 Sequential Labeling Approaches 22

2.1.1.4 Integration of Approaches 23

2.1.2 Normalization Benchmark Dataset 27

2.2 Contextual Pre-trained Language Models 29

2.3 Recent and State-of-the-art Approaches 29

2.3.1 Neural Network Approaches 29

2.3.2 State-of-the-art Approaches 31

2.3.3 Contextual Pre-trained Language Model Approaches 32

2.4 Summary 34

CHAPTER 3. RESEARCH METHODOLOGY 37

3.1 System Design . . . 37
3.2 Preprocessing . . . 38
3.3 Experiments . . . 39
3.3.1 Dataset . . . 41
3.3.2 Evaluation Metrics . . . 43
CHAPTER 4. RESULTS AND ANALYSIS . . . 45
4.1 Experiment 1. Investigation of BERT in candidate generation . . . 45
4.1.1 Candidate generation performance on all error types . . . 45
4.1.2 Candidate generation performance on different error types . . . 46
4.1.3 Discussion . . . 48
4.2 Experiment 2. Investigation of ranking methods based on error type . . . 49
4.2.1 Candidate ranking performance with 1-best accuracy . . . 49
4.2.2 Candidate ranking performance with k-best accuracy . . . 55
4.2.3 Discussion . . . 61
CHAPTER 5. CONCLUSION . . . 65
5.1 Conclusion and Analysis . . . 65
5.2 Future Work . . . 66
REFERENCES . . . 68

LIST OF TABLES

1.1 Example of canonical and non-canonical text 12
1.2 Example of normalization 12
1.3 Example of 1:1 and 1:N replacements 14
1.4 Example of normalization error types 14
2.1 Summarization of the Multi-lingual Normalization Benchmark 28
2.2 Summary of related work in normalization 34
3.1 Experimented ranking methods 40
3.2 Experimented Error Types 41
3.3 Summarization of the NormTax Dataset 42
3.4 Comparison of the NormTax and *NormTax_single* Dataset 42
4.1 BERT candidate generation results on the NormTax_single dataset 45
4.2 K-best Ranking Results on Error 1 56
4.3 K-best Ranking Results on Error 7 56
4.4 K-best Ranking Results on Error 11 57
4.5 K-best Ranking Results on Error 3 57
4.6 K-best Ranking Results on Error 8 58
4.7 K-best Ranking Results on Error 13 58
4.8 K-best Ranking Results on Error 10 59
4.9 K-best Ranking Results on Error 12 59
4.10 K-best Ranking Results on Error 9 60
4.11 K-best Ranking Results on Error 14 60
5.1 Best Performing Ranking Method based on Error Type 66

LIST OF FIGURES

3.1 Example of Preprocessed Data 38

3.2 Error Type Distribution of the *NormTax* Dataset 43

4.1 Candidate Generation Performance of BERT based on Error Types 46

4.2 Candidate Generation Performance of $\text{BERT}_{\text{base}}$ 47

4.3 Candidate Generation Performance of $\text{BERT}_{\text{large}}$ 47

4.4 Ranking Performance of $\text{BERT}_{\text{base}}$ on Error 1 50

4.5 Ranking Performance of $\text{BERT}_{\text{large}}$ on Error 1 50

4.6 Ranking Performance of $\text{BERT}_{\text{base}}$ on Error 3 51

4.7 Ranking Performance of $\text{BERT}_{\text{large}}$ on Error 3 51

4.8 Ranking Performance of $\text{BERT}_{\text{base}}$ on Error 12 52

4.9 Ranking Performance of $\text{BERT}_{\text{large}}$ on Error 12 52

4.10 Ranking Performance of $\text{BERT}_{\text{base}}$ on Error 8 53

4.11 Ranking Performance of $\text{BERT}_{\text{large}}$ on Error 8 53

4.12 Ranking Performance of $\text{BERT}_{\text{base}}$ on Error 9 54

4.13 Ranking Performance of $\text{BERT}_{\text{large}}$ on Error 9 54

4.14 Ranking Performance of $\text{BERT}_{\text{base}}$ on Error 14 55

4.15 Ranking Performance of $\text{BERT}_{\text{large}}$ on Error 14 55

4.16 Summarization of Ranking Performance based on Error Type 61

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
UGC	User Generated Content
BERT	Bidirectional Encoder Representations from Transformers
SMS	Short Messaging Service
IV	In-Vocabulary
OOV	Out-Of-Vocabulary
MT	Machine Translation
SMT	Statistical Machine Translation
CSMT	Character-based Statistical Machine Translation
CRF	Conditional Random Field
DCRF	Dynamic Conditional Random Field
POS	Part-Of-Speech
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ERR	Error Reduction Rate
MLM	Masked Language Model
NSP	Next Sentence Prediction
MFR	Most-Frequent Method

ABSTRACT

With the emergence of Social media and its growing popularity, there has been substantial growth in User Generated Content (UGC), which holds great potential in extracting meaningful information. Due to the dynamic nature of social media contents, many Natural Language Processing (NLP) systems have suffered from performance degradation due to the original intention in development for application to standard data. To resolve this significant drop in performance, normalization of non-standard data was introduced as a pre-processing step for processing non-standard texts before being applied to these downstream tasks. This thesis focuses on investigating the incorporation of the pre-trained language model BERT in normalization and the varying performance of normalization methods based on different types of errors. In this study, the BERT model is used for the candidate generation of normalization and simple ranking methods are further applied for the candidate ranking on the normalization candidates generated through BERT. The candidate generation performance of BERT and the ranking performance of different methods are investigated based on the different types of errors.

CHAPTER 1. INTRODUCTION

This thesis concerns the normalization of non-standard data, which can be described as a preprocessing step performed on non-standard data before NLP downstream applications, with a focus on investigating the incorporation of the pre-trained language model BERT in normalization and the varying performance of normalization methods based on different error types.

Due to the increased amount of social media content and its characteristic of high volume and availability, this type of non-standard data holds great potential in the extraction of meaningful information. However, many Natural Language Processing (NLP) systems have shown a significant decrease in performance when applied to this type of non-standard data. Therefore, in applying normalization to process non-standard data to its standard form, the performance degradation can be improved significantly aiding in producing accurate results in downstream NLP tasks. The following sections will describe the background, problem, research question, and significance of the work, followed by the review of literature and research methodology.

1.1 Background

With the emergence of social media and its growing popularity, there has been substantial growth in User Generated Content (UGC), which holds great potential in extracting and analyzing meaningful information. UGC data includes many variations of out-of-vocabulary (OOV) words such as spelling mistakes, abbreviations, internet jargon, etc. (Kumar, Makhija, & Gupta, 2020). However, many NLP systems were developed considering standard text data, with no consideration for non-standard text data such as social media data. As a result, performance degradation in non-standard text data has been a growing problem in NLP systems. Numerous previous work has demonstrated the hinder of performance in various NLP tasks when introduced with noise in the data, these tasks including parsing, machine translation, POS tagging, sentiment analysis, and semantic textual similarity (Foster et al., 2011; Han, Cook, & Baldwin, 2013; Hassan & Menezes, 2013; Kumar et al., 2020; F. Liu, Weng, Wang, & Liu, 2011b; van der Goot, Plank, & Nissim, 2017; van der Goot et al., 2017; C. Zhang, Baldwin, Ho, Kimelfeld, &

Li, 2013). To alleviate this performance degradation in non-standard text data, adaption was introduced as a solution by including normalization as a preprocessing step. Thus, normalization can be conceived as a preprocessing step added to process non-standard text data (Ljubešić, Zupan, Fišer, & Erjavec, 2016).

Normalization has been defined subjectively differently in different researches, but it can be generalized to a broad definition of transforming non-canonical texts into their canonical forms (F. Liu, Weng, Wang, & Liu, 2011a). Canonical text is the standard form of text that includes formal and highly-edited texts such as texts from books and essays. Non-canonical text is the non-standard form of text that is informal and noisy, including many variations of OOV words. Non-canonical texts are commonly contained in UGC, such as Tweets and Short Messaging Service (SMS) texts. Table 1.1 shows an example of non-canonical texts and their canonical forms.

Table 1.1. *Example of canonical and non-canonical text*

Non-canonical text	what r u doin tmr
Canonical text	what are you doing tomorrow

Table 1.2. *Example of normalization*

Normalization Type	**Normalization Result**
Original Sequence	I didnt kno, its so gooooooood 2 finaly see ur face
Correct Normalization	I **didn't know**, **it's** so **good to finally** see **your** face
Normalization with Spelling Correction (a)	I **didn't know**, its so **good** 2 **finally** see **urban** face
Normalization with Simple Lookup Table (b)	I **didn't know**, its so **god two finally** see **you are** face

Normalization can be performed with different methods and can yield significantly different results based on these different methods. Table 1.2 shows an example of the original sequence, the corresponding correct normalization result, along with the normalization result performed with a traditional spelling correction and a simple lookup table of common

abbreviations.. The normalization task is quite similar to a spelling correction task, however, the former often consists of intentional errors while the latter consists of unintentional errors. Due to this factor, it can be seen in the example result in Table 1.2 (a), that solely utilizing spelling correction methods are incapable of fully performing normalization since it is unable to handle common internet abbreviations such as *ur*.

Some may then argue that a simple lookup table of common internet abbreviations and their full form pairs, as well as common errors and their correction pairs, could be sufficient for normalization. However, as it can be seen in the example result of Table 1.2 (b), abbreviations oftentimes have more than one full form (e.g. *ur* $\rightarrow$ *your*, *you are*) thus requiring either the context that it appeared in or domain-specific information and also there are too many non-standard variations for a single token (e.g. *earthqua*, *eathquake*, *earthquakee* $\rightarrow$ *earthquake*) (Han, 2014) for a simple lookup table to handle. Therefore, simply applying a lookup table without taking context into account and applying spelling correction methods, will result in poor outcomes as shown.

The normalization task can be defined in different ways, but the task can generally be divided into three steps: 1) OOV detection, 2) Candidate generation, and 3) Candidate selection (Gouws, Hovy, & Metzler, 2011; Gouws, Metzler, Cai, & Hovy., 2011). In the first step, OOV detection, nonstandard words that require normalization are detected. Once the words for normalization are identified, candidates that are likely to be the normalized form of the detected words are generated in the candidate generation step. In the last step, candidate selection, the most likely candidate is selected among the candidates to be the normalized word.

Normalization is also commonly referred to as text normalization, UGC normalization, or lexical normalization, but lexical normalization specifically refers to the normalization task with replacements only on the word-level, in other words, one-to-one replacements. One way of categorizing different types of normalization is by type of replacement: one-to-one replacement, one-to-many replacement, and many-to-one replacement. Table 1.3 shows an example of one-to-one and one-to-many replacements in normalization. Another way of categorizing is more specifically by the different types of errors, e.g. spelling errors and word lengthening errors. Examples of different types of errors are shown in Table 1.4. Some tokens consist of more than one type of error and require multiple actions for normalization. For example, the error token

no1s consists of three error types (i.e. substitution, phrasal abbreviation, and missing-apostrophes error) and thus requires substituting 1 → one, splitting, and adding an apostrophe, resulting in the final normalization result of *no one's*.

Table 1.3. *Example of 1:1 and 1:N replacements*

Category	**Source text**	**Target text**
1:1 Replacement	tmr	tomorrow
1:N Replacement	omw	on my way

Table 1.4. *Example of normalization error types*

Error Type	**Example**
Spelling error	tommorow → tomorrow
Missing-Apostrophes error	dont → don't
Missing-Whitespace Characters error	heyyou → hey you
Word Lengthening error	noooooooo → no
Phrasal Abbreviation error	omg → oh my god
Word Shortening error	tmr → tomorrow
Substitution error	2 → to

1.2 Problem Statement

To address the issue of performance degradation, active research in normalization has been conducted and various approaches have been applied for normalization. Traditional spelling correction methods utilizing noisy channel models (Shannon, 1948) were initially applied (Brill & Moore, 2000; Choudhury et al., 2007; Cook & Stevenson, 2009; Toutanova & Moore, 2002). Since spelling errors and typographical errors are typical in informal texts, this approach

was effective in resolving these particular types of errors. However, this approach was incapable of normalization outside of the word boundary which would require a broader context. In disregarding the context of the error word, ambiguity also became an issue with words that have multiple mappings. For example, when it comes to words such as ”4” that can either be the number ”4” or a phonetic substitution for ”for”, the spelling correction approach was unable to handle this ambiguity since it doesn't take context into account.

As a solution to these limitations, machine translation was investigated in performing normalization. In applying machine translation, many-to-many mappings between non-standard words and standard words were able to be handled. To incorporate context in normalization, phrase-based statistical machine translation (SMT) was first applied (Aw, Zhang, & Su, 2006). This approach was capable of taking context into account resolving the ambiguity issue but was incapable of normalizing non-standard words that were not in the training corpora, even if a similar pattern of normalization existed in the corpora. Also, in performing transformation at the phrase-level, it was impossible for lexical creativity that is commonly observed in informal texts (e.g. SMS and social media texts) to be captured through this approach (Kobus, Yvon, & Damnati, 2008).

In light of this assertion, character-based statistical machine translation (CSMT) was proposed for normalization, which was capable of capturing the character-level creativity in word variants (Li & Liu, 2012b; Ljubešić, Erjavec, & Fišer, 2014; Pennell & Liu, 2011a). In applying this approach, new abbreviations that were not in the training corpora could be handled, which was a critical limitation of phrase-based SMT since new abbreviations and words appear daily in this domain. However, since this approach performs normalization at the token-level, the context could not be taken into account. Therefore, some approaches incorporated both phrase-based and character-level SMT to take advantage of both approaches (Ling, Dyer, Black, & Trancoso, 2013). However, SMT approaches are still highly prone to suffer from an insufficient amount of annotated training data.

Sequential labeling was another popular traditional approach applied (F. Liu et al., 2011a; Pennell & Liu, 2011b; Yang & Eisenstein, 2013). In these approaches, character-level sequential labeling is applied to standard dictionary words to generate their variants, and this is formed into a look-up table that is later on used to find the best possible normalization candidates

for non-standard words. These approaches are capable of learning the patterns in letter transformation, but since every character of the standard word and their combinations are used as features, this leads to a lot of noise in the look-up process for normalization candidates (Xu, Xia, & Lee, 2015).

Many studies also combined these different approaches to benefit from the different approaches' advantages and overcome their limitations (Beaufort, Roekhaut, Cougnon, & Fairon, 2010; Gao, Li, Micol, Quirk, & Sun, 2010; Gouws, Hovy, & Metzler, 2011; Kobus et al., 2008; Li & Liu, 2012a; F. Liu, Weng, & Jiang, 2012; C. Zhang et al., 2013). These studies often incorporated external resources such as string similarity, phonetic similarity, dictionary, and target domain lexicons for accuracy improvement as well.

Most of these approaches required a large-scale manually annotated training dataset, which was often unavailable and expensive to generate. This issue was resolved to some extent, With the Text Normalization Shared Task of the ACL2015 W-NUT workshop (Baldwin et al., 2015). With the growth in the annotated dataset for the training and testing of normalization, many works started applying deep learning and neural networks following the trend of its growing application in NLP (Chrupala, 2014; Leeman-Munk, Lester, & Cox, 2015; Min & Mott, 2015; Sridhar, 2015). These methods were able to enable learning complicated text transformations and working with large diverse streams of user-generated data (Lourentzou, Manghnani, & Zhai, 2019). Although the context was incorporated in these approaches, however, most works incorporated limited contextual information at a specific context window.

In the attempt to incorporate the full context of the error word for normalization, the sequence-to-sequence (Seq2Seq) model was applied (Lourentzou et al., 2019), and was able to produce the highest accuracy among the neural models, but was not able to surpass the performance of the state-of-the-art models (Jin, 2015; van der Goot & van Noord, 2017) that utilized random forest classifiers. Although these current state-of-the-art models do not leverage any deep learning or neural networks, the limitations of these models have shown that strong contextual information is still a critical component in normalization. Therefore, an approach that takes strong context into account could potentially provide value for normalization.

Due to the recent achievement of the contextual pre-trained language model, BERT (Devlin, Chang, Lee, & Toutanova, 2019), with state-of-the-art results in various NLP tasks and

its strong context, the BERT model could potentially be a strong candidate for the incorporation of context. However, very few works have been done in incorporating the pre-trained language model BERT to the normalization of non-standard data, with only a single work incorporating BERT directly for the normalization task.

Moreover, although many studies have been conducted on different normalization methods, very few studies have investigated the performance of normalization based on different error types. Due to the variety of different error types that exists in normalization and their varying characteristics, an investigation into the performance of different normalization methods is necessary.

1.3 Research Question

The main research question investigated in this research are as follows:

- How can the pre-trained language model BERT be applied to normalization?
- How does the performance of different normalization methods vary based on the different types of errors?
  - The baseline of this study is the pre-trained language model, BERT (Devlin et al., 2019) and the normalization model, MoNoise (van der Goot, 2019).

1.4 Significance

Normalization is capable of improving performance significantly in NLP tasks that include utilizing non-canonical texts. Many approaches have been taken in normalization, but there is still room for improvement. The state-of-the-art models are generally computationally expensive and require high resource environments. This study aims to provide a simple alternative to these methods for users performing normalization in low resource environments.

Since pre-trained language models have yet been investigated thoroughly in this field, this investigation could help enhance the normalization performance in incorporating the strengths of contextual pre-trained language models. This will also be a chance to investigate the strengths of contextual pre-trained language models in normalization. Also, although many studies have been conducted on normalization, not much work has been focused on the normalization of different types of errors. The normalization of 14 different error types will be investigated in this study based on the taxonomy of van der Goot, van Noord, and van Noord (2018), and will be able to uncover how the performance of different ranking methods vary based on the different error types as well as the performance difference in candidate generation of the BERT model on different types of errors. To the best of my knowledge, there has not been any research that has applied the pre-trained language model BERT specifically for the task of normalization without any modifications or fine-tuning.

The study aims to shed some light on the potential that the BERT model holds in normalization, and through a simple BERT-infused normalization method that can be improved through different ranking methods, provide suggestion and guidance for users performing normalization to preprocess informal text data based on different types of errors it contains.

1.5 Assumptions

This research has the following assumptions:

- The baseline studies investigated in this study are state-of-the-art methods.
- The datasets used in this study are correctly annotated.
- Out-of-vocabulary (OOV) words such as named entities are unconsidered for normalization.
- The OOV detection step of normalization has been performed and further normalization steps are only performed on the detected errors.

1.6 Limitations

This research has the following limitations:

- The study was conducted with publicly available data, and are not able to represent all non-canonical text data.
- The study was conducted on English Twitter data and are not able to represent Twitter data in all languages.

1.7 Delimitations

This research has the following delimitations:

- The study focuses only on the normalization of the English language.
- The study focuses only on the normalization of OOV words excluding emojis and punctuations.
- The study uses the *NormTax* dataset (van der Goot et al., 2018) to test the accuracy of different types of normalization errors.
- The Baseline study is *BERT* (Devlin et al., 2019) and *MoNoise* (van der Goot, 2019).

1.8 Summary

This chapter described the problem of performance degradation in many NLP systems when applied to non-standard text data such as social media, and the importance of resolving this performance issue due to the rise in social media and increase in user-generated contents, which includes abundant potential in extracting meaningful information. This chapter also describes the main research question of this study and the assumptions, limitations, and delimitations. The following chapter will review the literature and previous work conducted in normalization and contextual pre-trained language models.

CHAPTER 2. REVIEW OF LITERATURE

This chapter provides an extensive review of the current and previous work on normalization. The chapter reviews traditional approaches such as spelling correction-based approaches, machine translation-based approaches, sequential labeling-based approaches, and hybrid approaches in normalization. Contextual pre-trained language models, as well as the recent and state-of-the-art normalization approaches are also reviewed.

2.1 Normalization

This section reviews studies in normalization as well as the existing limitations of the current state-of-the-art normalization systems/approaches.

2.1.1 Traditional Approaches

Research into normalization grew with the emergence of SMS texts and social media. Traditional approaches have focused on three main approaches: Spelling correction, Machine Translation, and sequential labeling-based approaches (Gouws, Metzler, et al., 2011).

2.1.1.1 Spelling Correction Approaches

Many approaches utilize spelling correction frameworks such as noisy channel models (Shannon, 1948) for normalization. This spelling correction-based approach commonly incorporates morphophonemic similarity to perform normalization and focuses on single-word replacements (Han, 2014). The strong suit of this approach is that it is capable of unsupervised learning, which makes it highly attractive in normalization due to the difficulty in obtaining properly labeled training data from the target domain. Nonetheless, the weakness of this approach tends to be in flexibility, compared to different approaches.

Although these spelling correction models have shown state-of-the-art results in Grammatical Error Correction (GEC), they have shown significantly lower performance in non-standard text data. Therefore, considering the unique characteristics of SMS and social media

data, research has been focused on capturing new forms of words such as abbreviations and word shortenings, that are largely consisted of these types of data. Multiple error generation models were proposed by Cook and Stevenson (2009), each model focused on different type forms of SMS non-standard words such as words formed from phonetic spelling or clipping. Similar error generation models were integrated for tweets by Xue, Yin, and Davison (2011).

Meanwhile, Han and Baldwin (2011) proposed a normalization method that utilizes a detection classifier, candidate generation based on morphophonemic similarity, and candidate selection based on word similarity and context. Xu et al. (2015) extends the traditional noisy channel model in proposing a syllable-based tweet normalization method. The study was conducted under the assumption that syllables are an essential part of the formation of non-canonical words in social media. Based on this assumption, syllables were utilized to represent one-to-one word replacements in both word levels and syllable levels.

2.1.1.2 Machine Translation Approaches

Another popular approach in normalization is employing monolingual machine translations. The machine translation approach was proposed to incorporate context in normalization, by translating noisy, non-standard text to standard text. This approach benefits in being more flexible, which is enabled by an extra alignment factor, allowing many-to-many word replacements in normalization (Brown, Della Pietra, Della Pietra, & Mercer, 1993). The common issue with this approach, however, is that a large amount of suitable training data is required. Also, it tends to be slower in computation, due to the processing steps included, e.g. word alignment (Han, 2014).

Aw et al. (2006) adapts a phrase-based statistical machine translation (SMT) model for the normalization of SMS texts, in viewing normalization as a translation problem. The downside of this approach is that a phrase-based model requires a large annotated dataset since the training is conducted at the word-level. Pennell and Liu (2011a) apply a character-based statistical machine translation (CSMT) model for the normalization of SMS abbreviations. Training at the character-level, the model is able to recognize common abbreviation patterns, enabling it to be more effective against new abbreviations compared to phrase-level or word-level models. Character-level models are able to resolve the issues of data sparsity that the word-level and

phrase-level models face. Ling et al. (2013) integrated these two approaches in utilizing both character-based and phrase-based machine translation systems. The character-based system first performs normalization at the word-level, and then the optimized normalization sequence is selected by the phrase-based system at the sentence-level.

Ljubešić et al. (2014) extended the work of Pennell and Liu (2011a) in training a CSMT model on a lexicon of OOV tokens from Slovene tweets. The work was further extended in (Ljubešić et al., 2016), investigating segment-level normalization. Segment refers to text longer than a single token. In investigating segment-level normalization methods, contextual information could be incorporated, overcoming the limitations of the previous token-level normalization.

2.1.1.3 Sequential Labeling Approaches

Text normalization can be framed as a sequential labeling task for an automatic speech recognition (ASR) problem. The strength of the sequential labeling-based approach is its ability to capture the mutual influence of neighboring word-level normalizations in a sequence. In other words, rather than performing individual normalization of each non-canonical word, candidates are generated for each word and the best candidate is selected based upon the overall likelihood of the normalization sequence. The best normalization sequence is selected often through the Viterbi algorithm based on a language model trained from the target domain, and a sequence modeling framework, Conditional Random Fields (CRF) (Lafferty, McCallum, & Pereira, 2001) are often used in these approaches. The sequential labeling-based approach can be viewed as a sequential generalization of the aforementioned spelling correction-based methods, but with enhanced flexibility.

Cucerzan and Brill (2004) proposed a query log misspelling correction method, utilizing a weighted edit distance on edits in individual words and applying the standard Viterbi algorithm based on a bigram language model on the full sequence, in this case, query. Contractor, Faruquie, and Subramaniam (2010) presented a similar method incorporating the longest common character subsequence and consonant edit distance for normalization candidate generation, and the Viterbi algorithm in candidate selection for the most reasonable normalized sequence. Choudhury et al. (2007) utilized a hidden Markov model (HMM) for word-level SMS text normalization, motivated by previous work in automatic speech recognition (ASR) (Jelinek, 1997) that utilized HMM to

capture pronunciation and spelling variations. This is achieved by regarding users' intentional abbreviations and unintentional typos, respectively as HMM state transitions and emissions. Through this approach, Choudury et al. were able to capture both cognitive errors and typo errors.

Zhu, Tang, Li, Ng, and Zhao (2007) presented a unified CRF model for the normalization of different levels of informal text, e.g. capitalization or punctuation errors. However, due to the shortage of sufficient training data, normalization of non-standard words was excluded in the study. Beaufort et al. (2010) utilized the finite state machine for the normalization of French SMS text, employing an SMS trigram model to select the best-normalized word sequence. This method, however, still requires a large labeled dataset for training.

F. Liu et al. (2011a) utilized a character-based CRF model, trained using the alignments of OOV tokens and normalization candidates based on the longest common subsequence, to obtain the likelihood of a token to be converted into a noisy token. Sequence tagging is performed through the character-based CRF model, to generate normalization candidates for a letter sequence. Pennell and Liu (2011b) also utilized a similar CRF model, with a focus on the normalization of deletion-based non-standard words.

2.1.1.4 Integration of Approaches

Due to the different strengths and weaknesses of these different approaches, many studies have integrated different approaches to obtain higher performance in normalization. Kobus et al. (2008) integrated a machine translation system with an ASR system for SMS normalization and showed performance enhancement compared to both separate systems. Machine translation system was used for the initial normalization, and the speech recognition system was then used to generate normalization candidates for each of the OOV tokens left from the initial normalization, which are then combined in phoneme sequences and reassembled into word sequences. A trigram model is then utilized to rescore these word sequences and produces the final normalization output.

Gao et al. (2010) follow a similar two-step process of candidate generation and candidate re-ranking, in the spelling correction of search queries. The study generalizes the noisy channel model to a ranker-based system for the spelling correction of search queries. It incorporates similarity features and phrase-based machine translation probabilities into feature vectors for each

query and generated candidate pair, and also utilizes web-scale n-gram language models to measure the likelihood of a candidate being the correction of the query.

Gouws, Hovy, and Metzler (2011) integrated a sequential labeling method with an augmented method utilizing an automatically generated exception dictionary, for the normalization of tweets. This study constructs a lookup table of OOV tokens and their standard forms from the mined lexical variant pairs. The 50 highest scored pairs with the product of the semantic similarity score and lexical similarity score, are used for the exception dictionary. This exception dictionary is then utilized to augment their baseline sequential labeling method. This augmented method is used for the initial normalization, and the baseline sequential labeling method is used for the remaining non-standard words in normalization candidate generation and normalization sequence decoding.

Li and Liu (2012a) investigate enhancing normalization by integrating character-based machine translation and other methods. The study has shown that integrating spelling correction, character-level and character segment-level machine translation models, and a character segment-level sequential labeling method, produced the highest accuracy. The study utilizes character segments rather than individual characters, based on the effectiveness in alignment and shorter decoding length.

F. Liu et al. (2012) enhanced the performance of their previous character-based CRF model (F. Liu et al., 2011a), in integrating three subnormalizers: enhanced letter transformation using character-level sequence labeling, visual priming, and spell checker using string/phonetic similarity. The incorporation of visual priming was a unique application, which prioritizes the highest frequency candidate with the longest common subsequences with the non-standard token. In integrating these sub-normalizers, the system has been able to improve recall with the following integration strategies. Two strategies of integration were employed: word-level and message-level. The word-level integration strategy is essentially to prioritize the candidates from sub-normalizers with higher precision. Among the three sub-normalizers, visual priming has shown lower precision compared to letter transformation and spell checker, which showed high precision in candidate generation (F. Liu et al., 2011b). Thus, depending on the confidence score of the best candidate, top-3 candidates from either letter transformation or spell checker are placed higher in the candidate list, and candidates from visual priming are finally utilized to fill

the total number of candidates (n) in the list. In this strategy, message-level context information is unincluded. Meanwhile, the message-level integration strategy reranks all the candidates from the word-level strategy output, utilizing local context information through message-level Viterbi decoding.

C. Zhang et al. (2013) propose a normalization framework that can be customized to specific domains, presenting a solution for the domain adaptation issue which had yet been properly investigated. This was enabled in including a small set of domain-specific replacement generators to the general model. In general replacement generators, different methods such as spelling correction, edit distance, and Internet slang lexicon were integrated for the normalization candidate generation. The proposed approach is capable of 1:N replacements (e.g., *"idk"* to "*i don't know*"), and has the advantage of flexible candidate generation and global optimization. Joint decoding of the normalization sequence in incorporating context was able to improve traditional spelling correction methods that mainly focus on the normalization of individual non-standard tokens.

Yang and Eisenstein (2013) present unLOL, a normalization system that utilizes sequential labeling methods, that captures string and context similarities between non-standard and standard words in employing a unified unsupervised log-linear model. The challenge with this approach is efficiency and computational tractability. To resolve the issue of slow computation in Viterbi decoding, this system employs the Sequential Monte Carlo (SMC) training algorithm to sample a subset of plausible candidates before applying Viterbi decoding for ranking.

Wang and Ng (2013) argue the importance of further normalization operations such as punctuation correction and missing word recovery, where most studies have focused on word substitutions. The presented method uses a CRF model for missing word recovery and a dynamic conditional random field (DCRF) model for punctuation correction. The study also proposes integrating different normalization operations through a machine translation decoder. In the normalization process, hypotheses are generated which is a partially normalized tweet, compared to a candidate word in previous studies. Hypothesis generation is conducted through various methods such as phonetic approximations. Each hypothesis is assessed with language model scores, and beam search is then further utilized to prune the hypotheses. After these processes, only the hypotheses with the highest score are kept.

Han et al. (2013) propose a lexical variant detection classifier for OOV detection, a morphophonemic similarity-based method for candidate generation, and a word similarity and context-based method for candidate selection. The proposed normalization is performed through two steps in large: Confusion set generation (i.e., candidate generation for the given type of non-canonical word) followed by candidate selection. Han (2014) further analyzed the existing methods in text normalization and based on the analysis, developed a type-based method compared to their previous token-based normalization utilizing combined lexicons based on morphophonemic information.

Many systems that participated in the Text Normalization Shared Task of the 2015 Workshop on Noisy User-generated Text "ACL2015 W-NUT" (Baldwin et al., 2015) followed the approach of integrating different methods as well. The shared task included both constrained and unconstrained systems, where constrained systems were only allowed to utilize the training data provided by the shared task, and unconstrained systems were allowed to use any additional publicly available resources. In constrained systems, Akhtar, Sikdar, and Ekbal (2015) propose a hybrid normalization approach combining a machine learning method with rule-based methods. The proposed method utilizes a supervised CRF model trained with the training dataset provided by the shared task and a derived set of domain-independent features for the detection of words that need to be normalized, and heuristic rules to perform normalization on the identified words. The incorporated features include local context, part-of-speech (POS) information, morphology features, etc.

In the unconstrained systems category, Beckley (2015) proposes a simple normalization method utilizing a combination of lexicon, rule, and ranker, with the advantage of fast performance. The first component of the proposed method is a semi-supervised dictionary of non-standard words and the respective list of normalization candidates and performs most of the normalization. For the rule-based component, two rules were selected among the different rules through experimentation and applied in their final system, which is the "ing" and "coool" rule. These hand-crafted rules are for the purpose of capturing morphology errors that frequently appear in user-generated texts. For the last component of ranking, the proposed method utilizes a sentence-level re-ranker (i.e. bigram Viterbi algorithm). Even with this very simple approach,

Beckley (2015) was able to place third among the five teams that participated in the unconstrained normalization task of the shared task.

Berend and Tasnádi (2015) propose an error type-sensitive normalization method utilizing ”efficiently indexed n-gram statistics”. This study utilized a CRF-based sequence labeling module to identify the type of correction needed for the lexical variant, and performs normalization based on external lexicons and the indexed n-gram statistics from English tweets. The external lexicons were utilized to determine the features of individual words and includes: SCOWL dictionary from Aspell spell checker project, normalization dictionaries from (Han, Cook, & Baldwin, 2012; F. Liu et al., 2012), and a web-derived slang normalization dictionary. In utilizing n-grams, the proposed normalization aims to utilize context in the normalization of OOV words. The proposed system showed the second-highest performance among the unconstrained systems in the shared task.

Supranovich and Patsepnia (2015) propose a CRF-based approach in performing both normalization detection and normalization of words that are nonexistent in the lexicon, utilizing an SVM model-based query misspelling correction module that was developed by the IHS R&D team, i.e. did-you-mean (DYM) module. The proposed system showed the highest performance among the unconstrained systems that participated in the shared normalization task (Baldwin et al., 2015).

2.1.2 Normalization Benchmark Dataset

Han and Baldwin (2011) introduced the first benchmark Twitter dataset *LexNorm* consisting of a total of 549 Tweets and 558 normalization pairs of non-standard token and standard words for normalization. This was followed by Yang and Eisenstein (2013), who presented *LexNorm1.2* with improved annotations on the original *LexNorm* corpus. Li and Liu (2015) then presented another Twitter corpus *LiLiu* that incorporates *LexNorm* along with other Twitter datasets. This corpus contains capitals compared to the *LexNorm* and *LexNorm1.2* which does not contain any capitals.

Furthermore, Baldwin et al. (2015) presented a shared task of Twitter text lexical normalization in the 2015 Workshop on Noisy User-generated Text. The dataset from this shared

task, *LexNorm2015*, has since been widely used in normalization studies. The presented corpus consists of 5,200 Tweets with 6,666 normalization pairs including all one-to-one, one-to-many, and many-to-one replacements, and was able to resolve the lack of annotated datasets for training and evaluation in normalization. It also not only covers OOV non-standard words but also covers tokens that are non-standard words but with the same spelling as a standard word.

Most recently, The state-of-the-art work from van der Goot (2019) presented a Multi-lingual Normalization Benchmark dataset for the evaluation of multi-lingual normalization performance. The Multi-lingual Normalization Benchmark is a bundle of a total of nine datasets in seven languages and includes all the previous benchmark datasets in English (Baldwin et al., 2015; Li & Liu, 2014; Yang & Eisenstein, 2013) as well as different language normalization datasets such as Spanish and Croatian. The comparison of the datasets included in the benchmark is organized in Table 2.1.

Table 2.1. *Summarization of the Multi-lingual Normalization Benchmark*

(Word #: Word Count, Lang: Language, Norm%: Percentage of normalized words, 1:N: Existence of 1-to-n replacements, Cap: Existence of capitalization, EN: English, NL: Dutch, ES: Spanish, TR: Turkish, SL: Slovenian, HR: Croatian, SR: Serbian)

Corpus	**Lang**	**Word #**	**Norm%**	**1:N**	**Cap**
LexNorm1.2 (Yang & Eisenstein, 2013)	EN	10,576	11.6	x	x
LiLiu (Li & Liu, 2014)	EN	40,560	10.5	x	o
LexNorm2015 (Baldwin et al., 2015)	EN	73,806	9.1	o	x
GhentNorm (De Clercq, Schulz, Desmet, & Hoste, 2014)	NL	12,901	4.8	o	o
TweetNorm (Alegria et al., 2013)	ES	13,542	6.3	o	o
IWT (Eryiğit & Torunoğ-Selamet., 2017)	TR	38,918	8.5	o	o
Janes-Norm (Erjavec et al., 2017)	SL	75,276	15.0	x	o
ReLDI-hr (Ljubešić, Erjavec, Miličević, & Samardžić, 2017a)	HR	89,052	9.0	x	o
ReLDI-sr (Ljubešić, Erjavec, Miličević, & Samardžić, 2017b)	SR	91,738	8.0	x	o

2.2 Contextual Pre-trained Language Models

Recent work in the field of natural language processing (NLP) has been able to achieve state-of-the-art results employing the contextual pre-trained language model, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) in various NLP tasks such as language inference and question answering. BERT is a pre-trained language model of deep bidirectional representations from unlabeled texts, enabling the incorporation of both the left and right contexts. The framework of BERT consists of two steps: pre-training and fine-tuning. BERT is pre-trained on two unsupervised tasks of the Masked Language Model (MLM) and Next Sentence Prediction (NSP). The model architecture of BERT is a multi-layer bidirectional Transformer encoder (Vaswani et al., 2017), and many downstream tasks from the fine-tuning are enabled through the self-attention mechanism of the Transformer. Other than its strength in capturing contextual information, another strong suit of the BERT pre-trained model is that it is capable of being fine-tuned to different downstream tasks with an addition of just a single output layer. Due to the success of BERT, many variants have been introduced (Clark, Luong, Le, & Manning, 2020; W. Liu et al., 2020; Y. Liu et al., 2019; Sanh, Debut, Chaumond, & Wolf, 2020; Yao, Mao, & Luo, 2019).

2.3 Recent and State-of-the-art Approaches

2.3.1 Neural Network Approaches

Recent Approaches in normalization have been applying deep learning models such as multi-layer feed-forward neural networks and recurrent neural networks (RNN), that have achieved promising results in numerous NLP tasks such as sentiment analysis (Socher et al., 2013) and speech recognition (Hinton et al., 2012). Chrupala (2014) presents a normalization model that utilizes edit sequences learned from labeled data and features from unlabeled data through recurrent neural embedding. The study has found that incorporating these features from text embeddings was able to significantly lower word error rates in normalization. Also, unlike

previous studies that have applied generative Bayesian methods, this work has chosen to edit non-canonical data into their canonical forms directly, with edits generated through a linear chain CRF model.

Some systems that participated in the Text Normalization Shared Task of the 2015 Workshop on Noisy User-generated Text "ACL2015 W-NUT" (Baldwin et al., 2015) took the approach of neural network architectures in the constrained system category and demonstrated comparable results to the best performing system (Jin, 2015).

Wagner and Foster (2015) propose utilizing a generalized perceptron for sequence labeling rather than utilizing CRF as traditional methods have done in normalization. The study compares the application of the proposed generalized perceptron method to a CRF method following the work of Chrupala (2014). Due to memory constraints, the CRF model was restricted with a smaller training dataset. The sequence labelers were utilized for edit operations that produce normalization candidates, the extracted edit operations including the following three actions: no edit, character deletion, and string insertion before the character. In the generation of word edit operations, various extracted features were included such as RNN language model hidden layer activations and edit operations (which were the same as (Chrupala, 2014)), with two new additional features: character class and editing eligibility based on the rules of the shared task. For the final selection of the best normalization candidate among the generated candidates, the system utilizes a noisy channel model and character n-gram models that are trained on the normalized tweets of the training data.

Leeman-Munk et al. (2015) applied a deep learning approach to normalization, utilizing a combination of two augmented feed-forward neural networks in the normalizer and flagger. The proposed model consists of three major components: the normalizer, flagger, and conformer. The normalizer is responsible for encoding the input and outputting the normalized result and utilizes a feed-forward neural network. The flagger is responsible for determining whether the token should be normalized or not, and has a similar structure as the normalizer in utilizing a feed-forward neural network, but has a softmax layer in the last layer that performs predictions on whether the word needs to be normalized.The last component, conformer, is responsible for correcting the failed attempts of the normalizer. Built through a dictionary derived from the training data, the conformer uses the Levenshtein distance (Levenshtein, 1966) to find the closest word in the

dictionary. Even with the lack of context incorporation, due to the constrained system not being able to include any external resources, the proposed system showed the third-highest performance in constrained systems showing comparable performance to the model that placed second.

Min and Mott (2015) also applied a deep learning approach in utilizing a contextual long-short term memory (LSTM) recurrent neural network-based model with an induced dictionary from the training data for normalization. The presented system employs LSTM (Graves, 2012; Hochreiter & Schmidhuber, 1997) for text normalization, and augments character sequences and POS tags tagged with the part-of-speech tagger (Owoputi et al., 2013) provided by the shared task. The proposed system consists of three large phases: 1) domain-specific entity filtering 2) dictionary-based normalization 3) LSTM-based normalization. The proposed system showed the second-highest performance in constrained systems following the random forest classifier approach of Jin (2015). However, both of these methods did not utilize any external lexical resources for the normalization task.

Lourentzou et al. (2019) adapt sequence-to-sequence (Seq2Seq) models for normalization. The recent study presents a hybrid word-based Seq2Seq model trained specifically for the normalization of social media data, as a solution to the limitations of previous work in only being able to include specific context windows and specific error types. The proposed model is capable of including a fuller context in incorporating long-term contextual dependencies and handling multiple errors at the same time. The results outperformed the other normalization methods employing neural architectures of ANN (Artificial Neural Network) (Leeman-Munk et al., 2015) and LSTM (Min & Mott, 2015) and showed comparable performance with the state-of-the-art works (Jin, 2015; van der Goot & van Noord, 2017).

2.3.2 State-of-the-art Approaches

Jin (2015) presents one of the state-of-the-art Twitter lexical normalization systems that perform candidate generation based on prior knowledge (largely from the training data) and a novel string similarity measurement proposed in the study, and candidate selection utilizing a binary Random Forest (RF) classifier (Breiman, 2001). The proposed novel string similarity measure represents each string with a similarity feature set where the feature includes n-grams

and k-skip-n-grams in the corresponding string. The similarity between the two strings is then evaluated with the Jaccard Index (Levandowsky & D., 1971) of the two strings' similarity feature sets.

van der Goot and van Noord (2017) further extended this work in presenting MoNoise, the current state-of-the-art normalization model, that takes a modular approach to normalization. The system employs a modular approach for candidate generation, where each module performs different types of normalization actions. The modules include a spelling correction, word embeddings, and a static lookup list module. Candidate ranking of the generated candidates is conducted through a random forest classifier (Breiman, 2001), and n-gram features along with the features from the generation modules are incorporated into the classifier. Implementation of Ranger (Wright & Ziegler, 2017) was used with the default parameters.

van der Goot (2019) improved the MoNoise model in incorporating features from the original word for normalization candidate generation and presents a new multi-lingual benchmark dataset and novel evaluation metric, Error Reduction Rate (ERR). The presented dataset is a bundle of nine datasets that includes seven languages. For an easier comparison between different datasets, the study presents ERR as a new evaluation metric, which can be conceived as a normalized accuracy for the percentage of words that are normalized in the golden standard.

In terms of the performance on different error types, MoNoise was not capable of handling the type of normalization errors that require splitting of two or more words and unknown type of errors that were unknown to annotators or in disagreement, according to the work of van der Goot et al. (2018). Based on the results, the MoNoise model also showed difficulty in Typographical errors, phonetic transformation errors, as well as merge errors.

2.3.3 Contextual Pre-trained Language Model Approaches

Recent attempts have been made to utilize contextual pre-trained language models for normalization. BERT (Devlin et al., 2019), the current state-of-the-art pre-trained language model, has especially been explored in the appliance to enhance the performance of normalization.

van der Goot and Çetinoğlu (2021) proposes a lexical normalization method for code-switched data. Code-switched (CS) data refers to data that includes multiple languages in a single utterance. The proposed systems incorporate the BERT model, in applying a BERT-based tagger (i.e. MaChAmp (van der Goot, Üstün, Ramponi, & Plank, 2020)) with the multilingual BERT model for the word-level language identification. Although the proposed method incorporated the BERT model, they were applied for the tagging and language identification step rather than the normalization step itself.

J. Zhang et al. (2020) presents a Mandarin text normalization system utilizing multi-head self-attention, which was proposed in Transformer (Vaswani et al., 2017). The study further experiments with various neural model setups, including the word-to-vector (w2v) model and fine-tuning with the BERT model, and the w2v model resulted in better performance than fine-tuning with BERT. The study suggests the overfitting of the model as a reason for the lack of performance. Since the text normalization task was performed on a Text-to-Speech (TTS) news corpus in this study, the type of non-standard words contained is quite different from that of user-generated data (e.g. social media), in the text data being more formal.

To the best of my knowledge, (Muller, Sagot, & Seddah, 2019) is the only work that has been conducted on utilizing the BERT pre-trained language model specifically for lexical normalization. The study investigates the ability of BERT in performing normalization, by framing normalization as a token prediction task and enhancing the architecture and fine-tuning BERT. To focus on the capability of the BERT model itself, the study was conducted with only the 3,000 training sentences, without the use of any other UGC resources.

Some studies have taken a different approach to enhance the performance of downstream tasks on non-standard data, in directly applying BERT for downstream tasks without performing normalization. Radivchev and Nikolov (2019) utilize BERT and ensembles for offensive tweet classification. The study compared the results of ten different models including soft voting classifier (SVC), recurrent neural network (RNN), BERT, etc., and BERT proved to show the best overall performance (Radivchev & Nikolov, 2019).

Nguyen, Vu, and Nguyen (2020) propose a pre-trained language model designated for English tweets, BERTweet. As the first large-scale and publicly available model, BERTweet was trained employing the pre-training procedures of RoBERTa (Y. Liu et al., 2019) utilizing the

model configuration of the $BERT_{base}$ model. These results outperformed $RoBERTa_{base}$ and $XLM\text{-}R_{base}$ (Conneau et al., 2020) on the tweet NLP tasks: Text classification, Named-entity recognition, and Part-of-speech (POS) tagging. This results on BERTweet were results from the above three NLP tasks rather than the task of normalization itself.

Although BERT has been proved to be a powerful tool, due to the diversity in non-standard text data and errors, it will be difficult to handle different types of errors and generalize to different downstream tasks with BERT alone. Therefore, the intuition of this study is to incorporate the pre-trained language model, BERT, with other normalization methods to be able to handle different types of normalization errors effectively as well as generalize over various downstream tasks.

2.4 Summary

The summarization of the related work conducted in normalization are shown in Table 2.2. Different approaches applied from the studies in the literature review are organized.

Table 2.2.: *Summary of related work in normalization*
(SC: Spelling Correction, MT: Machine Translation, SL: Sequential Labeling, NN: Neural Network Architecture, CP: Contextual Pre-trained Language model, LM: Language Model, NG: N-gram, DN: Dictionary)

Related Work	**SC**	**MT**	**SL**	**NN**	**CP**	**LM**	**NG**	**DN**
Cook and Stevenson (2009)	o	x	x	x	x	o	o	o
Xue et al. (2011)	o	x	x	x	x	o	o	o
Han and Baldwin (2011)	o	x	x	x	x	o	o	o
Xu et al. (2015)	o	x	x	x	x	o	o	o
Aw et al. (2006)	x	o	x	x	x	o	o	o
Pennell and Liu (2011a)	x	o	x	x	x	o	o	o
Ling et al. (2013)	x	o	x	x	x	o	o	o
Ljubešić et al. (2014)	x	o	x	x	x	o	x	x
Ljubešić et al. (2016)	x	o	x	x	x	o	x	x
Cucerzan and Brill (2004)	x	x	o	x	x	o	o	x

Table 2.2. *continued*

Related Work	SC	MT	SL	NN	CP	LM	NG	DN
Contractor et al. (2010)	x	o	o	x	x	o	o	o
Choudhury et al. (2007)	x	x	o	x	x	o	o	o
Zhu et al. (2007)	x	x	o	x	x	o	o	x
Beaufort et al. (2010)	o	o	o	x	x	o	o	o
F. Liu et al. (2011a)	x	x	o	x	x	o	o	x
Pennell and Liu (2011b)	x	x	o	x	x	o	o	o
Kobus et al. (2008)	x	o	o	x	x	o	o	o
Gao et al. (2010)	o	o	x	x	x	o	o	o
Gouws, Hovy, and Metzler (2011)	x	x	o	x	x	o	o	o
Li and Liu (2012b)	o	o	o	x	x	o	o	o
F. Liu et al. (2012)	o	x	o	x	x	o	o	o
C. Zhang et al. (2013)	o	x	o	x	x	o	o	o
Yang and Eisenstein (2013)	x	x	o	x	x	o	o	o
Wang and Ng (2013)	x	o	o	x	x	o	o	o
Han et al. (2013)	x	x	o	x	x	o	o	o
Han (2014)	x	x	o	x	x	x	x	o
Akhtar et al. (2015)	x	x	o	x	x	x	x	o
Beckley (2015)	x	x	o	x	x	x	o	o
Berend and Tasnádi (2015)	x	x	o	x	x	x	o	o
Supranovich and Patsepnia (2015)	o	x	o	x	x	x	o	o
Chrupala (2014)	x	x	o	o	x	o	o	o
Wagner and Foster (2015)	o	x	o	o	x	o	o	x
Sridhar (2015)	x	x	x	o	x	o	o	x
Leeman-Munk et al. (2015)	x	x	x	o	x	x	x	o
Min and Mott (2015)	x	x	x	o	x	x	o	o
Lourentzou et al. (2019)	x	x	x	o	x	x	x	x

Table 2.2. *continued*

Related Work	SC	MT	SL	NN	CP	LM	NG	DN
Jin (2015)	x	x	x	x	x	x	o	o
van der Goot and van Noord (2017)	o	x	x	x	x	o	o	o

CHAPTER 3. RESEARCH METHODOLOGY

This chapter describes the overall research design and methodology that is used throughout this study. The system design, experiments, dataset, and evaluation method are described in this chapter.

3.1 System Design

Normalization is commonly divided into three steps: OOV detection, candidate generation, and candidate selection. OOV detection is the detection of OOV tokens requiring normalization. Candidate generation is the generation of a list of potential normalization candidates of the detected OOV token. Candidate selection is the selection of a candidate from the generated candidate list, which will be the normalization result. This is typically performed through candidate ranking to select the best possible normalization candidate.

In this study, OOV detection is assumed to have been performed on the dataset, and normalization is performed solely on detected OOVs (i.e. normalization errors). On the OOVs detected, normalization is performed through two steps: candidate generation and candidate ranking. The candidate generation step is performed using BERT, forming candidate generation as a Masked LM (MLM) task, to produce potential candidates based on context. Based on the characteristic of the BERT model, up to 30,000 candidates can be generated. The candidate ranking step is performed through five different methods: longest common prefix, longest common sequence, edit distance, and phonetic distances; SoundEx and NYSIIS.

In integrating the contextual pre-trained language model BERT in candidate generation, this study intends to incorporate more contextual information in normalization and apply different non-complex methods for candidate ranking to achieve normalization results on different types of normalization errors with faster performance in limited resource environments as well.

3.2 Preprocessing

Datasets requiring normalization may often time include multiple normalization errors in one data. The BERT model, in performing the MLM task, can only have one masked word per input and only single words for output (i.e. candidates). Due to this characteristic of BERT, preprocessing is required on the dataset to modify the data from multiple errors to a single error per data.

For each data, all the errors that are included in the data are detected. For each detected error, all the other errors will be replaced with the corresponding normalization result. A data that includes *n* number of errors will produce *n* number of data of the same original text but with different words as errors for each data. As a result, all data will include only one error. Figure 3.1 shows an example of the preprocessed data. The errors in the data are indicated in red and the errors replaced with their corresponding normalization result are indicated with an underline. The example shows that the input data contains three errors in total, thus the preprocessed data results in three different data each including different errors with the other errors replaced with their correct normalization results.

Input Data:

Error 12 Error 12 Error 6

yea I was gonna hit **u** up but guess not **lol**

Pre-processed Data:

yea i was gonna hit you up but guess not laughing out loud

yeah i was gonna hit **u** up but guess not laughing out loud

yeah i was gonna hit you up but guess not **lol**

Figure 3.1. Example of Preprocessed Data

3.3 Experiments

To investigate the varying performance of different normalization methods based on the different types of errors, the following experiments will be conducted. The experiments along with the dataset utilized in performing the experiment and evaluation metric used to evaluate the result will be described in the following:

The first experiment is to investigate the incorporation of BERT (Devlin et al., 2019) in the candidate generation step of the normalization of non-standard data. As a BERT-based normalization method, the BERT model is used in the candidate generation step to produce top-n normalization candidates for the error word, in which then different normalization methods such as edit distance are applied for the candidate ranking to find the best candidate. Candidate generation is performed with the BERT model as a masked language modeling (MLM) task, masking the non-standard word that requires normalization and generating top-n candidate predictions of the masked word.

Due to the constraints of BERT, which can only handle one masked word for each input, this experiment is performed on data that contains only single errors. To test the varying performance of BERT in candidate generation based on different types of errors, the BERT model is tested on the different error types classified in the taxonomy of van der Goot et al. (2018). The experimented error types are shown in Table 3.2. This experiment is performed on two different BERT models to compare the performance: $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$.

The second experiment is to investigate the varying performance of different ranking methods based on different types of normalization errors. The performance of six different ranking methods, including the basic ranking of the baseline study BERT, is tested on different types of errors. The experimented ranking methods are shown in Table 3.1 and the description of each ranking methods are as follows:

Table 3.1. *Experimented ranking methods*

Ranking Methods	**Description**
LCP	Longest common prefix
LCS	Longest common subsequence
Edit_dist	Edit distance (Levenshtein)
Soundex	Phonetic distance (SoundEx)
Nysiis	Phonetic distance (NYSIIS)
(Baseline) BERT	Basic ranking performed with BERT

- First, the longest common prefix ranks candidates based on the length of their common prefix with the OOV word, ranking candidates with longer common prefixes higher.
- Secondly, the longest common subsequence ranks candidates based on the length of their common subsequence with the OOV word, ranking candidates with longer common sequences higher. The subsequences do not need to be continuous.
- Thirdly, for edit distance, the Levenshtein distance is used to calculate and perform ranking based on the minimum number of operations required to transform the candidate word to the OOV word. The edit operations consist of insertion, deletion, and substitution. Candidates with smaller edit distances are ranked higher in the candidate list.
- Lastly, two phonetic distances are used as ranking methods: SoundEx and NYSIIS. These algorithms are one of the most commonly known phonetic algorithms along with Metaphone, but SoundEx and NYSII were chosen since these algorithms output a single conversion while Metaphone outputs two conversions.

Finally, the performance of these different ranking methods is compared to the performance of the baseline models, BERT and MoNoise. The first baseline BERT refers to the basic ranking performed through the BERT model in the process of candidate generation, with no further ranking performed. The second baseline MoNoise refers to the gold error detection results of the state-of-the-art MoNoise model. For a more accurate comparison, the results will be

compared to the gold error detection results of MoNoise since gold error detection assumes correct OOV detection.

The described ranking methods are tested on the error types shown in Table 3.2, excluding the error types that are determined from the results of the first experiment to be incapable of being handled through BERT. Rankings are performed on the candidates generated with BERT, and the $\text{BERT}_{\text{large}}$ model is used in this experiment for candidate generation based on the results of the first experiment.

Table 3.2. *Experimented Error Types*

Error Type	**Error Description**	**Example**
Error 1	Typographical error	can't → ca'nt
Error 2	Missing-Apostrophes error	don't → dont
Error 3	Spelling error	neighbors → neighbours
Error 4	Split error	love → l o v e
Error 5	Merge error	no more → nomore
Error 6	Phrasal Abbreviation error	on the way → otw
Error 7	Repetition error	please → pleaseeeee
Error 8	Shortening vowels error	people → ppl
Error 9	Shortening end error	favorite → fav
Error 10	Shortening other error	because → bc
Error 11	Phonetic transformation error	though → tho
Error 12	Regular transformation error	going → goin
Error 13	Slang error	thanks → thx
Error 14	Unknown error	they → nem

3.3.1 Dataset

NormTax, the error-type annotated dataset in the work of van der Goot et al. (2018), is used for the experiments in this study. van der Goot et al. (2018) augmented the training set of the *LexNorm2015* (Baldwin et al., 2015) dataset with their proposed taxonomy of error types, annotating the normalization replacements with the respective category in the taxonomy. The summarization of the *NormTax* dataset is presented in Table 3.3. The proposed taxonomy

classifies normalization errors into fourteen different types of errors, and the distribution of the taxonomy in the LexNorm2015 dataset is shown in Figure 3.2 (van der Goot et al., 2018).

Due to the limited data containing single errors, the *NormTax* dataset was modified to produce *n* number of data rows with single errors from a data row containing *n* number of errors. This data modification process is described in section 3.2, and the modified dataset will be referred to as the *NormTax_single* dataset throughout this study. From the total of 1864 data rows, the dataset was modified resulting in a total of 3916 data rows. The *NormTax_single* dataset is used to perform both experiments 1 and 2. The comparison of the *NormTax* dataset and *NormTax_single* dataset is presented in Table 3.4.

Table 3.3. *Summarization of the NormTax Dataset*

(Tweet #: Tweet Count, Word #: Word Count, Non-standard #: Non-standard Error Count, Norm%: Percentage of normalized words, 1:N: Existence of 1-to-n replacements, Cap: Existence of capitalization, EN: English)

Dataset	**NormTax**
Source	van der Goot et al. (2018)
Language	EN
Tweet #	2,950
Word #	44,385
Non-standard #	3,928
Norm%	8.85
1:N	o
Cap	x

Table 3.4. *Comparison of the NormTax and NormTax_single Dataset*

(Data #: Data Count, Multiple_err: Includes multiple errors in one data, Non-standard #: Non-standard Error Count)

Dataset	***NormTax***	***NormTax_single***
Data #	1,864	3,916
Multiple_err	o	x
Non-standard #	3,928	3,916

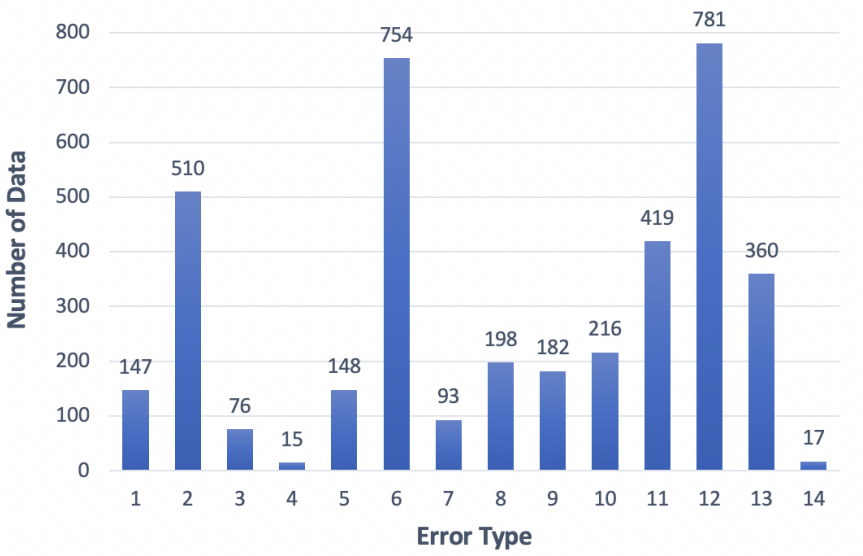

Figure 3.2. Error Type Distribution of the *NormTax* Dataset

3.3.2 Evaluation Metrics

For the evaluation of the candidate generation performance of BERT models as well as the performance of different ranking methods based on different error types, the word-level n-best accuracy score is used. In the word-level n-best accuracy score (F. Liu et al., 2012), for each OOV word, the case is considered accurate if the correct normalization result exists among the n-best candidates. In this study, the word-level n-best accuracy measure is further adapted to the experiments for better interpretation as the accuracy measure for each candidate generation and candidate ranking. For candidate generation, this evaluation metric is used to evaluate the performance of how well the candidates are generated by measuring how often the correct candidate exists in the generated candidate list prior to ranking. For candidate ranking, this evaluation metric is used to not only evaluate the accuracy of the 1-best but also the n-best results for a more accurate evaluation of the ranking performance. The accuracy measures are referred to as the following:

- **N-best accuracy:** For each OOV word, it is considered correct if the correct normalization result exists among the *n* number of candidates generated with BERT. This metric is used as the accuracy measure for the performance of candidate generation.

- **K-best accuracy:** For each OOV word, it is considered correct if the correct normalization result exists among the top-*k* ranked candidates. This metric is used as the accuracy measure for the performance of candidate ranking.

CHAPTER 4. RESULTS AND ANALYSIS

This chapter presents the results and analysis of two different experiments. The results presents the findings for adopting BERT to candidate generation and in the performance of different ranking methods based on the different types of normalization errors.

4.1 Experiment 1. Investigation of BERT in candidate generation

The candidate generation performance of the BERT model is evaluated with the n-best accuracy on the *NormTax_single* dataset. The n-best accuracy is calculated for the different number of candidates generated through the BERT model.

4.1.1 Candidate generation performance on all error types

First, the overall candidate generation performance of the BERT model was tested on the *NormTax_single* dataset, inclusive of all the different types of errors. Table 4.1 shows the overall n-best accuracy results of BERT candidate generation on the *NormTax_single* dataset. The accuracy can be seen increasing with the increase in the number of candidates. The highest accuracy was shown in 30000 candidates at 53.50%, with a close runner-up in 10000 candidates at 52.91%. The intuition from this result was that the low performance could be due to certain types of errors that could not be handled through the BERT model.

Table 4.1. *BERT candidate generation results on the NormTax_single dataset*

N-best Candidates (N)	5	10	20	30	50	70	90	100	500	1000	10000	30000
N-best Accuracy (%)	23.67	28.60	32.12	34.32	36.59	38.18	39.27	39.68	45.25	47.85	52.91	**53.50**

4.1.2 Candidate generation performance on different error types

Consequent to the intuition from the results of the previous experiment, the candidate generation performance of the BERT model was tested on the different error types in the *NormTax_single* dataset. Figure 4.1 shows the candidate generation performance of the BERT model on 30,000 candidates based on different error types. Candidate generation performance of the BERT model increased significantly when experimented on different types of errors. The highest accuracy was shown in Error 7 at 98.92% showing a 45.42% increase from the highest accuracy of the previous experiment. Other error types also show a high increase in accuracy compared to the previous experiment, except for Error 2, 4, 5, and 6 which shows 0% or close to 0% accuracy. This result reveals that the low overall performance of the BERT model on candidate generation was due to certain types of errors that show close to no performance.

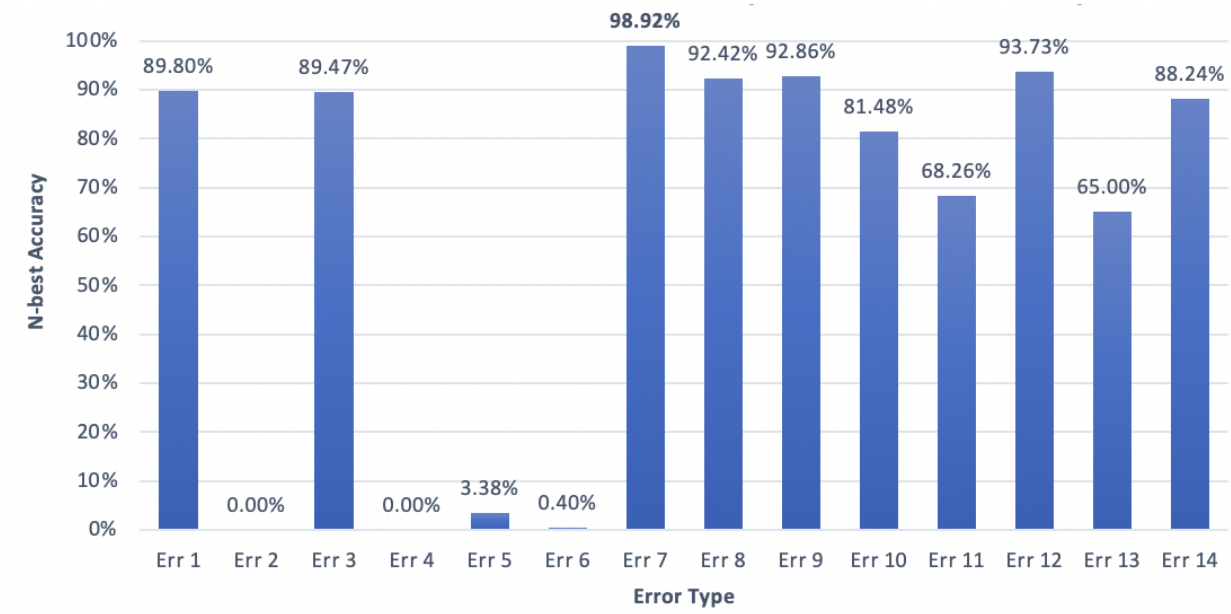

Figure 4.1. Candidate Generation Performance of BERT based on Error Types (30,000 Candidates)

For further insight, the candidate generation performance of the BERT model was tested on ten different types of errors, excluding the four non-performing error types (Error 2, 4, 5, 6), with two different BERT models: $BERT_{base}$ and $BERT_{large}$. For each error type, the candidate generation performance was tested on *n* candidates (*n = 5, 10, 50, ..., 30000*).

Figure 4.2 and Figure 4.3 show the candidate generation performance of $BERT_{base}$ and $BERT_{large}$, respectively, on different error types. Both models show an increasing trend in candidate generation performance with the increased number of candidates and similar trends based on the different types of errors, with Error 7 showing the highest accuracy in 500, 1,000, 10,000, 30,000 candidates and Error 12 in 5, 10, 50, 100 candidates. All methods show the highest performance in 30,000 candidates with similar performance in 10,000 candidates. However, most methods also show comparable performance in 500 and 1,000 candidates as well.

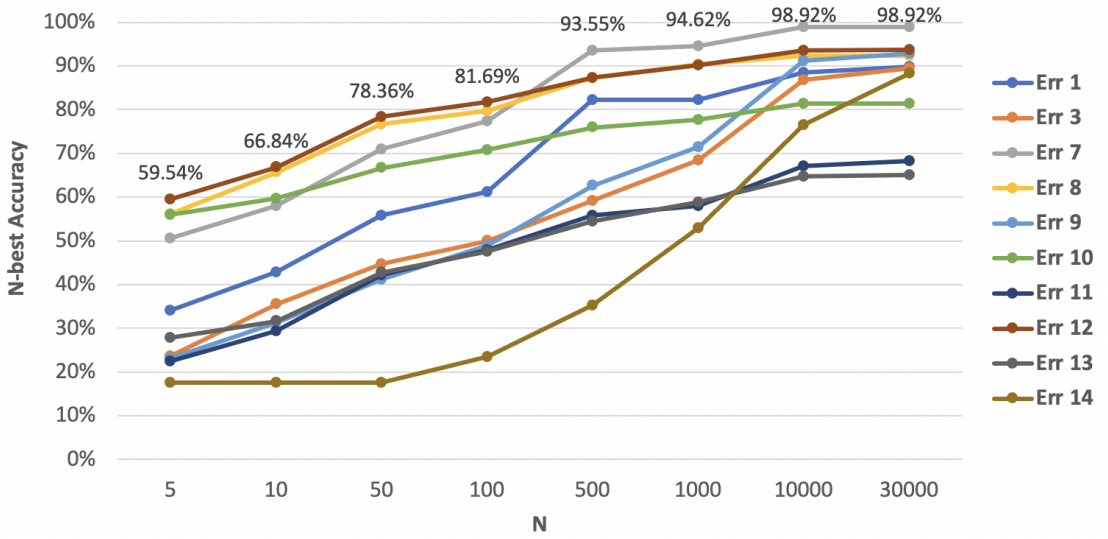

Figure 4.2. Candidate Generation Performance of $BERT_{base}$

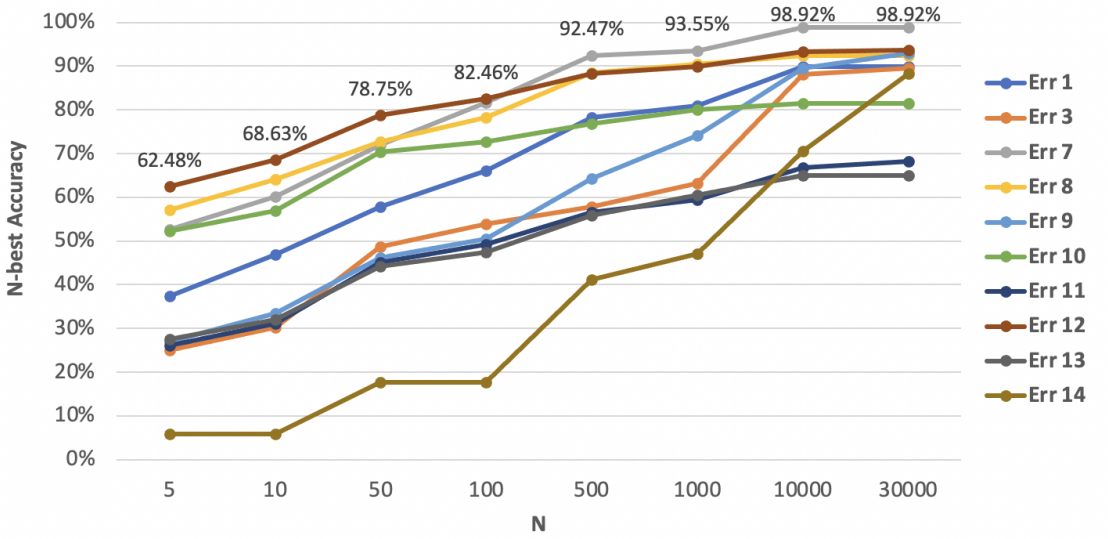

Figure 4.3. Candidate Generation Performance of $BERT_{large}$

4.1.3 Discussion

The results of the experiment on the candidate generation of different error types show that the low overall performance of the BERT model was due to the four types of errors that showed low performance (0 - 3.38%). Looking into these four types of errors (i. e. Error 2, 4, 5, and 6), the low performance of the BERT model on these specific error types can be understood.

Error type 2, missing-apostrophe's error, requires an apostrophe to be included in the correct normalization result (dont→don't), however, the candidates generated through the BERT model don't include candidates with special characters in the word (e.g. don't, can't). Meanwhile, Error type 4, split error, requires merging of the error words (l o v e→love) which require N:1 replacement of words. Error type 5, merge error, requires splitting of the error word (nomore→no more) and Error type 6, phrasal abbreviation error, requires expanding to the full form of the abbreviation (omw→on my way), both error types requiring 1:N replacement of words to perform normalization. However, the BERT model can only perform 1:1 replacement of words, since it takes a single word as input and outputs single word candidates. Therefore, the error types that include special characters, 1:N or N:1 replacements cannot be handled with the BERT model, and thus should be excluded from the further assessment of the BERT model, considered as a limitation.

Despite these limitations, the results show that the BERT model shows high potential in candidate generation for normalization in the error types that it is capable of handling, showing an average accuracy of 86.02% and the highest accuracy up to 98.92%. Moreover, it shows that in order to properly assess the potential of the BERT model in normalization, normalization performance should be tested based on different error types.

Also, the results on the $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ models show that the two models show very similar performance and trend over all the error types. Thus, one could choose the model based on the environment setting. While the two models have shown very similar performance, the $\text{BERT}_{\text{large}}$ model has shown slightly higher performance. Therefore, one with more computing resources to spare and prefer the higher accuracy could select the $\text{BERT}_{\text{large}}$ model, and for one with limited computing resources and prefer faster speed, the $\text{BERT}_{\text{base}}$ model could be perfectly suitable.

Moreover, the results on the different number of candidates suggest that although the highest candidate generation performance was shown in 30,000 candidates, same or close performance was shown in 10,000 candidates in most error types and comparable performance were shown in 500 and 1,000 candidates as well, providing insight that 1,000 candidates could be sufficient. Although the higher number of candidates are showing higher candidate generation performance, the large candidate pool could also hinder the performance of the ranking of candidates. Therefore, assessing the appropriate number of candidates based on both candidate generation and ranking will be important in the overall performance of normalization.

4.2 Experiment 2. Investigation of ranking methods based on error type

The candidate ranking performance of different ranking methods is evaluated with the k-best accuracy on the different error types in the *NormTax_single* dataset. The k-best accuracy is calculated for the ranking methods on different error types based on the different number of candidates generated through the BERT model.

For simplicity, each ranking method, longest common prefix, longest common subsequence, edit distance (Levenshtein), phonetic distance (SoundEx), phonetic distance (NYSIIS) will be referred to as LCP, LCS, Edit_dist, Soundex, Nysiis, respectively.

4.2.1 Candidate ranking performance with 1-best accuracy

First, the ranking performance of six different ranking methods, including the baseline BERT basic ranking, were tested on ten different error types. The Ranking was performed on candidates generated with two different BERT models, $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$, and tested with different *n* number of candidates (*n = 5, 10, 100, ..., 30000*). The 1-best accuracy score is calculated for each experiment.

Error 1, Error 7, and Error 11 showed a similar trend in the results, with Edit_dist showing the highest ranking performance and the baseline BERT ranking showing the lowest performance. For example, Figure 4.4 and Figure 4.5 show the ranking performance on Error 1 with *n* candidates generated with the $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ model, respectively. In both models,

Edit_dist performed much higher than the baseline BERT, and showed the highest performance from 5000 candidates with comparable performance at 1000 candidates. The second highest performing method, LCS, showed close performance in smaller number of candidates, and the baseline BERT was the lowest performing ranking method for both models. Both BERT models also showed similar performance as the number of candidates increased, but the $\text{BERT}_{\text{large}}$ model showed slightly higher performance in smaller number of candidates.

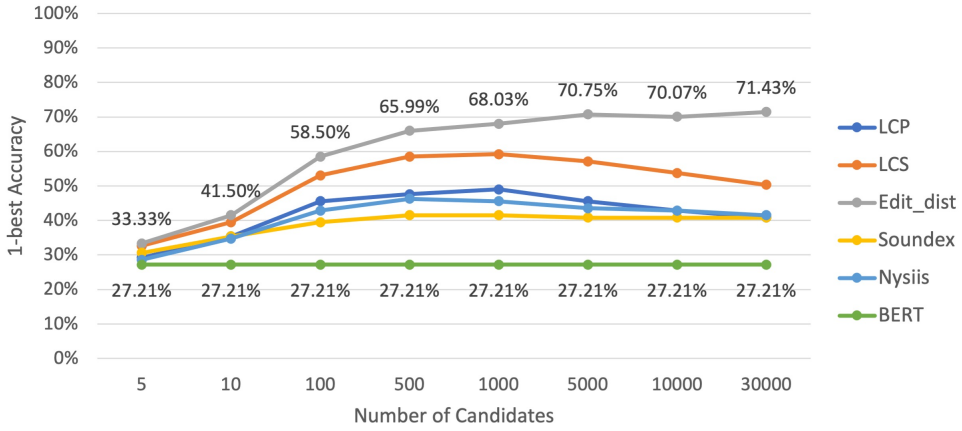

Figure 4.4. Ranking Performance of $\text{BERT}_{\text{base}}$ on Error 1

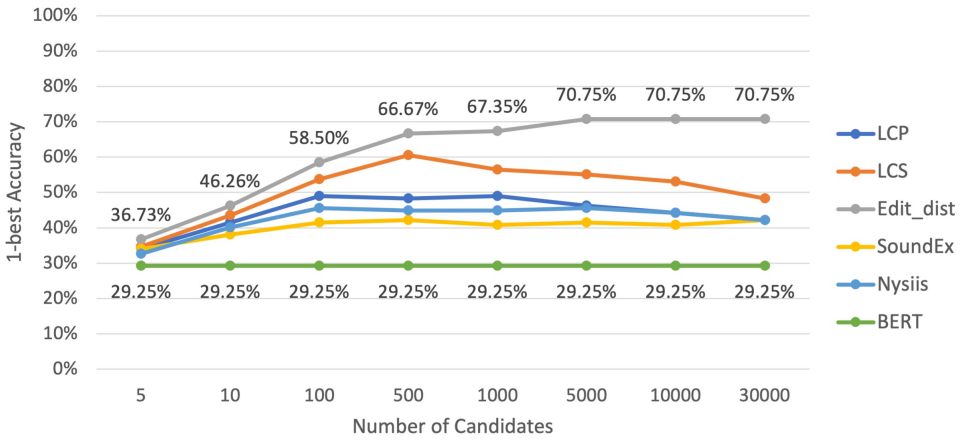

Figure 4.5. Ranking Performance of $\text{BERT}_{\text{large}}$ on Error 1

In Error 3, Edit_dist performed highly as well, however, Nysiis also showed high performance and was the highest performing method. In this specific error type, the $\text{BERT}_{\text{base}}$

showed higher performance in larger number of candidates and $BERT_{large}$ in lower, which was the opposite of the general trend of other error types.

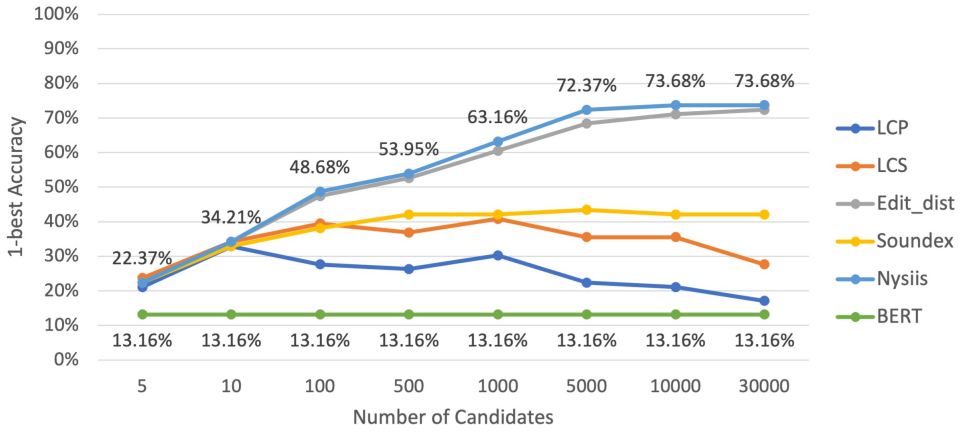

Figure 4.6. Ranking Performance of $BERT_{base}$ on Error 3

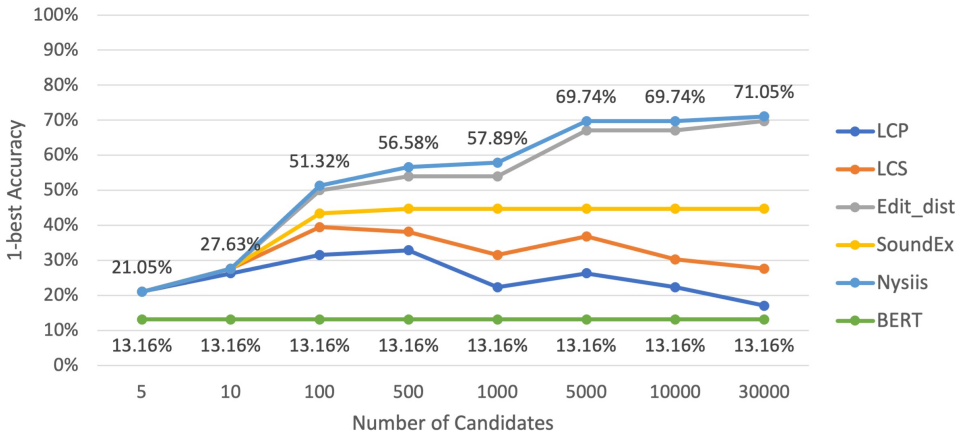

Figure 4.7. Ranking Performance of $BERT_{large}$ on Error 3

Error 10 and Error 12 showed very similar trend in the results as well, with LCS showing the dominant ranking performance with the baseline BERT showing the second highest performance. Figure 4.8 and Figure 4.9 show the ranking performance on Error 12 with *n* candidates generated with the $BERT_{base}$ and $BERT_{large}$ model, respectively. For Error 12, LCS was the highest performing method, showing highest accuracy in 100 candidates and comparable performance in 500 and 1,000 candidates. LCS showed a slight decreasing trend in performance

with the increase in the number of candidates after 100 candidates, and the other methods also showed a decreasing trend in performance with the increased number of candidates. The highest ranking method still outperformed the baseline BERT, but the baseline showed second highest performance with other methods all showing very poor performance after 10 candidates.

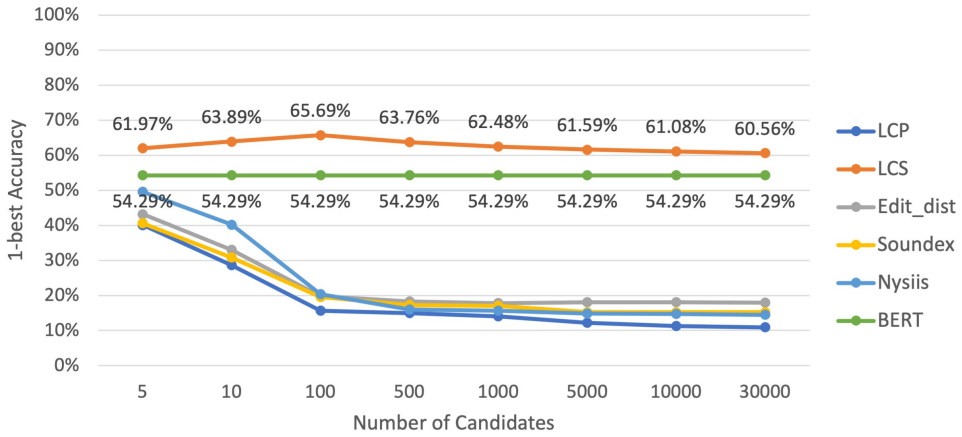

Figure 4.8. Ranking Performance of $BERT_{base}$ on Error 12

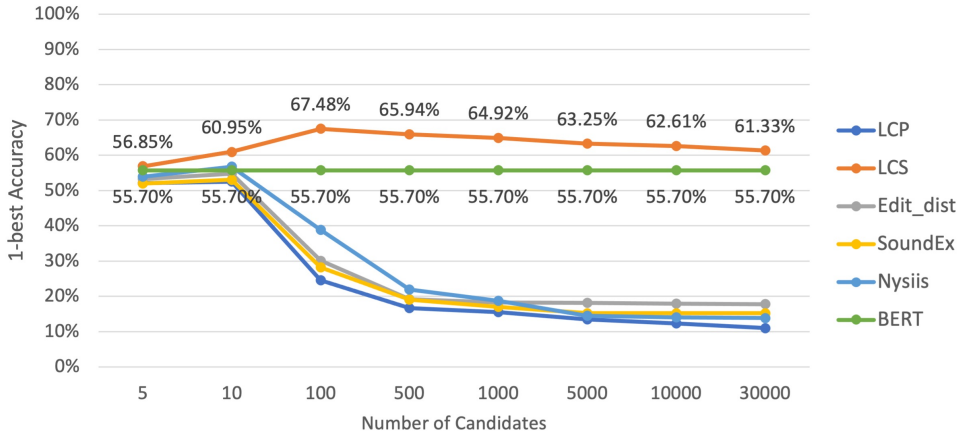

Figure 4.9. Ranking Performance of $BERT_{large}$ on Error 12

In Error 8 and Error 13, LCS was also the highest performing method, but Soundex showed the second highest performance with the baseline BERT showing one of the lowest performances. For example, Figure 4.10 and Figure 4.11 show the ranking performance on Error 8 with *n* candidates generated with the $BERT_{base}$ and $BERT_{large}$ model, respectively. For Error 8,

LCS performed much higher than the baseline BERT, and showed highest performance from 5000 candidates and comparable performance at 1000 candidates.

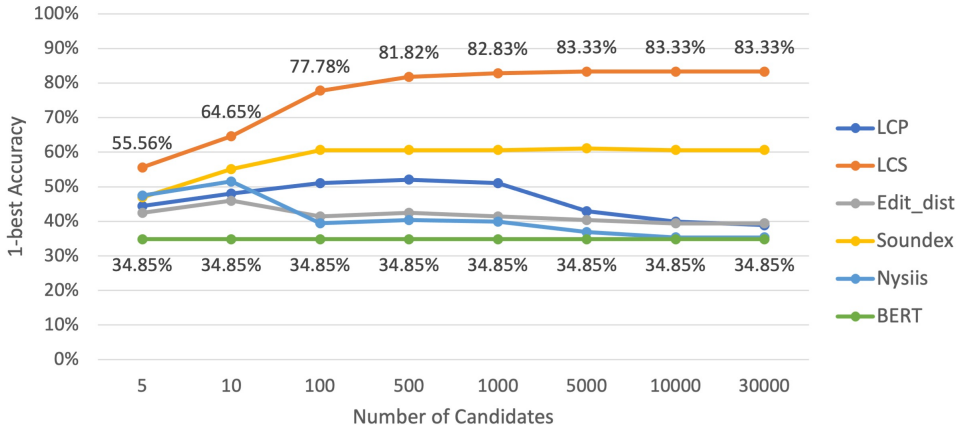

Figure 4.10. Ranking Performance of $BERT_{base}$ on Error 8

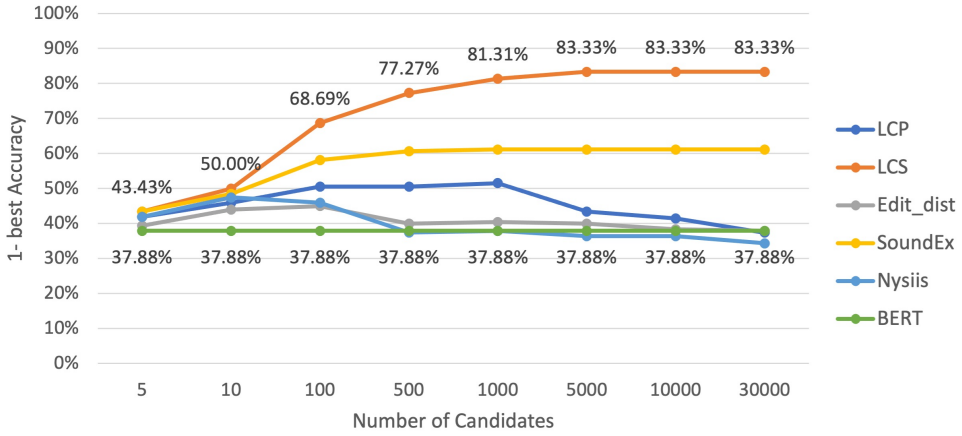

Figure 4.11. Ranking Performance of $BERT_{large}$ on Error 8

Error 9 did not show a similar trend with the results of other error types. Figure 4.12 and Figure 4.13 show the ranking performance on Error 9 with *n* candidates generated with the $BERT_{base}$ and $BERT_{large}$ model, respectively. For Error 9, LCP and LCS showed the highest performance among the ranking methods with the LCP showing the highest accuracy for both models. Edit_dist was the lowest accuracy ranking method for both models. LCP showed highest

accuracy in 30,000 candidates but showed comparable performance in 5,000 and 10,000 candidates. The ranking methods, LCP, LCS, and Soundex outperformed the baseline BERT.

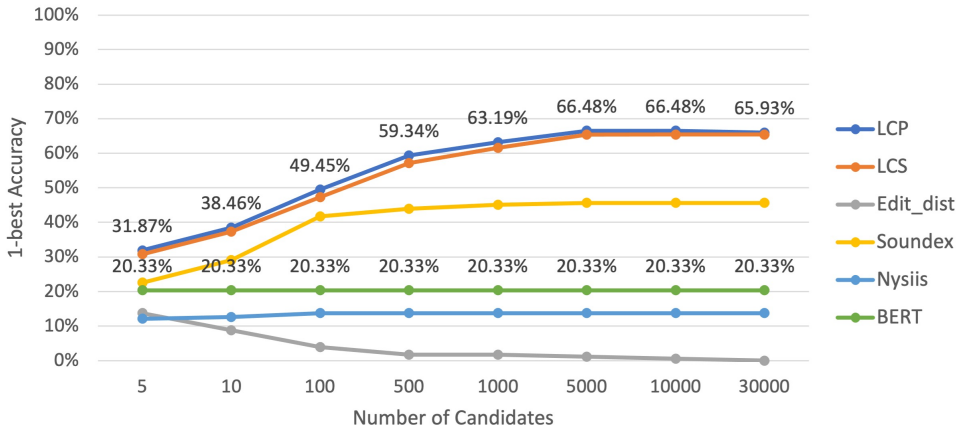

Figure 4.12. Ranking Performance of $BERT_{base}$ on Error 9

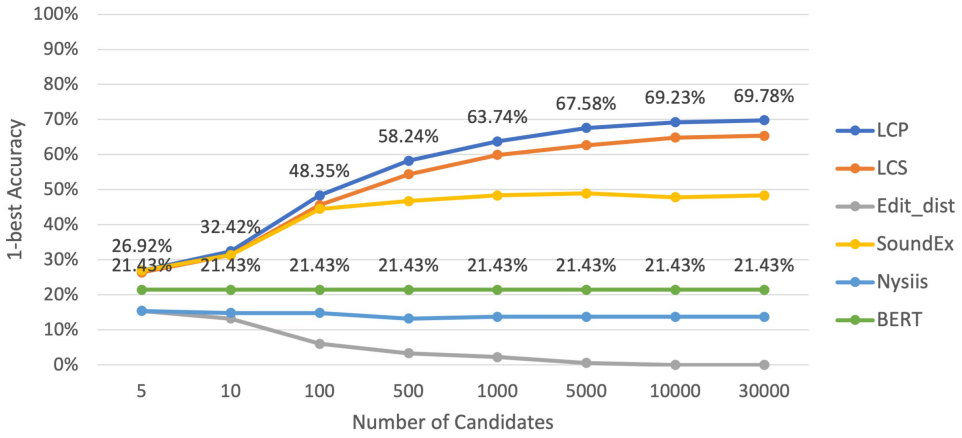

Figure 4.13. Ranking Performance of $BERT_{large}$ on Error 9

Figure 4.14 and Figure 4.15 show the ranking performance on Error 14 with *n* candidates generated with the $BERT_{base}$ and $BERT_{large}$ model, respectively. For Error 14, Soundex showed the highest accuracy in $BERT_{base}$ and LCS showed the highest accuracy in $BERT_{large}$.

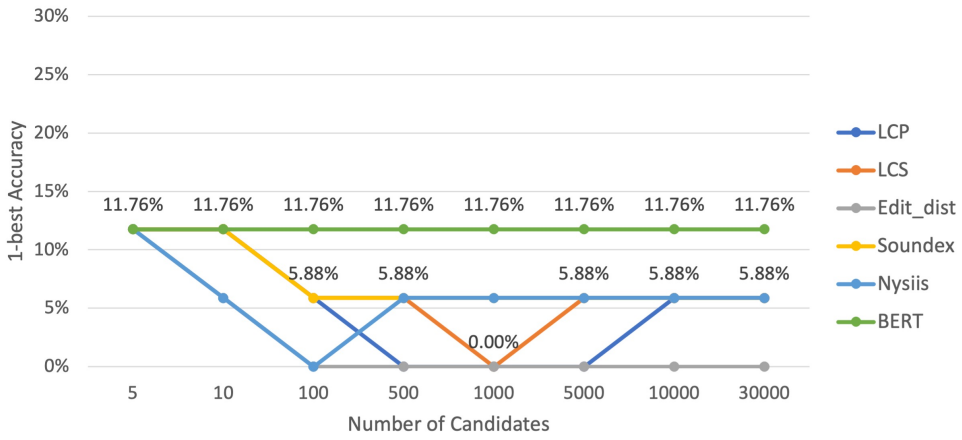

Figure 4.14. Ranking Performance of $BERT_{base}$ on Error 14

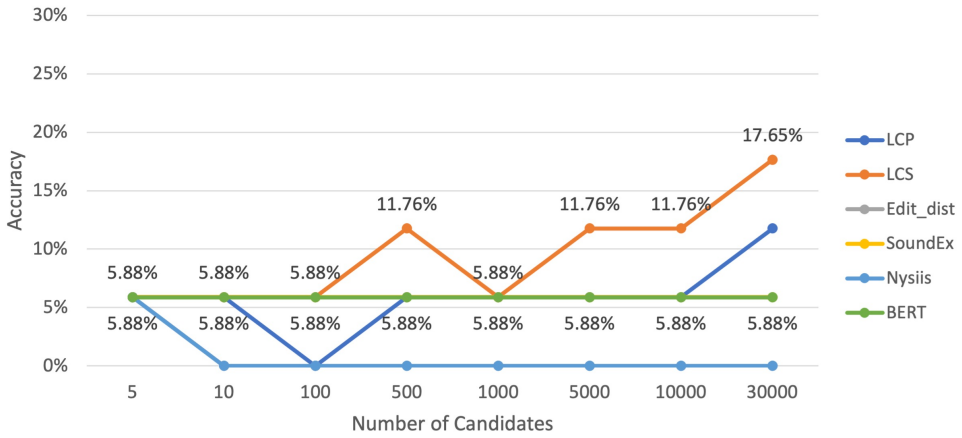

Figure 4.15. Ranking Performance of $BERT_{large}$ on Error 14

4.2.2 Candidate ranking performance with k-best accuracy

The ranking performance of the six ranking methods, including the BERT basic ranking, were evaluated with *k*-best accuracy for *1*-best, *5*-best, and *10*-best results for each *n* number of candidates *(n=10, 1,000, 30,000)*. Ranking was performed on candidates generated with the $BERT_{large}$ model and on ten different error types. The k-best accuracy score is calculated for each experiment. In the result tables, the ranking method with the highest performance in each k-best of each n candidates are indicated with a single underline, with the highest accuracy in each

k-best indicated in bold, and the highest accuracy over all the results is indicated with a double underline.

Table 4.2, Table 4.3, and Table 4.4 shows the k-best ranking results on Error 1, Error 7, and Error 11, respectively. For Error 1 and Error 11, the highest performance was shown in the k-best results of 30,000 candidates, while Error 7 showed highest performance in 1,000 candidates. With all the ranking methods showing significant increase in accuracy with the increased k, an increase can be seen from 77.42% in 1-best to the highest accuracy of 93.55% in 10-best for Error 7.

Table 4.2. *K-best Ranking Results on Error 1*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	44.22	53.06	53.06	47.62	65.99	68.03	42.18	57.82	59.86
LCS	48.30	52.38	53.06	57.14	72.11	76.19	48.30	65.99	69.39
Edit_dist	52.38	53.06	53.06	67.35	78.23	81.63	**70.75**	**81.63**	**83.67**
Soundex	41.50	51.70	53.06	41.50	53.06	60.54	42.18	52.38	58.50
Nysiis	41.50	52.38	53.06	45.58	65.31	66.67	42.18	59.18	63.95
(Baseline) BERT	29.25	46.26	53.06	29.25	46.26	53.06	29.25	46.26	53.06

Table 4.3. *K-best Ranking Results on Error 7*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	55.91	60.22	60.22	64.52	88.17	90.32	58.06	80.65	91.40
LCS	53.76	60.22	60.22	39.78	68.82	79.57	24.73	43.01	51.61
Edit_dist	55.91	60.22	60.22	**77.42**	**90.32**	**93.55**	**77.42**	88.17	90.32
Soundex	52.69	60.22	60.22	62.37	83.87	87.10	62.37	83.87	87.10
Nysiis	58.06	60.22	60.22	69.89	84.95	88.17	64.52	84.95	88.17
(Baseline) BERT	30.11	52.69	60.22	30.11	52.69	60.22	30.11	52.69	60.22

Table 4.4. *K-best Ranking Results on Error 11*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	27.92	30.55	30.55	48.93	54.65	56.09	56.32	56.80	57.04
LCS	26.49	30.07	30.55	48.21	49.64	51.79	53.94	55.61	56.32
Edit_dist	28.16	30.55	30.55	51.31	52.74	53.46	**57.76**	**60.62**	**60.86**
Soundex	27.68	30.55	30.55	35.80	53.22	55.61	35.08	53.22	57.28
Nysiis	26.49	30.55	30.55	35.56	49.88	53.70	21.72	41.29	50.84
(Baseline) BERT	15.27	25.30	30.55	15.27	25.30	30.55	15.27	25.30	30.55

Table 4.5 shows the k-best ranking results on Error 3. In comparison to Error 7, although, Edit_dist was not the highest performing method in Error 3, it was the second highest performing method with performance very close to the first highest performing method, Nysiis. The results show that Nysiis also showed high performance in Error 7 as well. The highest performance was seen in 30,000 candidates and the highest accuracy increased from 71.05% to 88.16% with 5-best results. In LCP, there was a significant increase in accuracy from 17.11% in 1-best to % 78.95% in 10-best.

Table 4.5. *K-best Ranking Results on Error 3*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	26.32	30.26	30.26	23.68	56.68	60.53	17.11	59.21	78.95
LCS	27.63	30.26	30.26	31.58	60.53	60.53	27.63	69.74	81.58
Edit_dist	28.95	30.26	30.26	55.26	63.16	63.16	69.74	86.84	86.84
Soundex	27.63	30.26	30.26	44.74	57.89	61.84	44.74	59.21	67.11
Nysiis	28.95	30.26	30.26	59.21	63.16	63.16	**71.05**	**88.16**	**88.16**
(Baseline) BERT	13.16	25.00	30.26	13.16	25.00	30.26	13.16	25.00	30.26

Table 4.6 and Table 4.7 shows the k-best ranking results on Error 8 and Error 13, respectively. In Error 8, although the highest performance over all the results are shown in 30,000 candidates from LCS, the 1,000 candidate results of the method show very close performance, and the results show that the performance of the overall ranking methods are highest in 1,000

candidates with as well. Similarly in Error 13, although the highest performance over all the results are shown in 1,000 candidates from LCS, the performance of the overall ranking methods are shown highest in 10 candidates. In both error types, most of the ranking methods except for LCS, show the highest results in 10 candidates for 1-best accuracy and a decreasing trend in performance with the increase in the number of candidates. While Error 13 did not show high performance, Error 8 showed high performance with the 1-best accuracy score of 83.33% and the 10-best accuracy of 90.91%.

Table 4.6. *K-best Ranking Results on Error 8*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	51.01	64.14	64.14	47.47	69.19	74.75	37.37	67.68	71.72
LCS	63.13	64.14	64.14	82.83	89.39	90.40	**83.33**	**89.90**	**90.91**
Edit_dist	44.95	63.64	64.14	40.40	50.51	51.52	37.88	37.88	47.98
Soundex	55.05	63.13	64.14	61.11	76.77	79.80	61.11	76.77	79.80
Nysiis	50.00	63.64	64.14	36.87	45.96	48.48	34.34	46.46	47.47
(Baseline) BERT	37.88	57.07	64.14	37.88	57.07	64.14	37.88	57.07	64.14

Table 4.7. *K-best Ranking Results on Error 13*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	22.22	30.56	31.94	17.78	33.89	39.72	11.11	19.44	30.00
LCS	27.22	31.11	31.94	**29.44**	**45.00**	**48.61**	18.61	29.72	36.11
Edit_dist	13.06	27.50	31.94	12.50	18.89	22.78	11.94	15.00	17.78
Soundex	21.67	31.39	31.94	19.44	27.22	34.72	19.44	26.94	31.11
Nysiis	16.39	24.72	31.94	14.17	23.89	25.56	10.56	14.72	21.94
(Baseline) BERT	15.56	27.50	31.94	15.56	27.50	31.94	15.56	27.50	31.94

Table 4.8 and Table 4.9 shows the k-best ranking results on Error 10 and Error 12, respectively. In Error 10, although the highest performance over all the results are shown in 30,000 candidates from LCS, the results show that the performance of the overall ranking

methods are highest in 10 candidates. Similarly in Error 12, although the highest performance over all the results are shown in 1,000 candidates from LCS, the performance of the overall ranking methods are shown highest in 10 candidates as well. In both error types, all the ranking methods except for LCS, show the highest results in 10 candidates with a decreasing trend in performance with the increase in the number of candidates including 1-best results. Error 12 showed an increase in highest accuracy from 65.81% in 1-best to 79.39% in 10-best.

Table 4.8. *K-best Ranking Results on Error 10*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	32.87	54.63	56.94	29.63	39.81	43.52	21.30	39.35	42.59
LCS	50.93	55.56	56.94	**64.81**	74.54	**76.85**	**64.81**	**75.00**	**76.85**
Edit_dist	23.15	48.61	56.94	13.43	26.39	37.50	10.65	15.28	23.15
Soundex	33.80	55.09	56.94	25.93	31.02	33.80	25.93	31.02	32.41
Nysiis	16.67	50.93	56.94	14.81	18.06	18.98	14.81	17.59	17.59
(Baseline) BERT	38.89	52.31	56.94	38.89	52.31	56.94	38.89	52.31	56.94

Table 4.9. *K-best Ranking Results on Error 12*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	27.27	70.68	72.86	14.47	20.74	23.05	11.01	17.03	19.08
LCS	**65.81**	72.09	72.86	63.89	**75.42**	**79.39**	61.33	70.55	72.98
Edit_dist	32.65	71.19	72.86	17.80	25.10	29.96	17.80	22.79	24.58
Soundex	28.94	70.42	72.86	15.62	25.74	28.43	15.24	21.38	23.82
Nysiis	43.02	71.57	72.86	14.85	40.46	70.42	13.83	18.18	62.10
(Baseline) BERT	55.70	70.17	72.86	55.70	70.17	72.86	55.70	70.17	72.86

Table 4.10 shows the k-best ranking results on Error 9. The highest performance was shown in 30,000 candidates but comparable performance was shown in 1,000 candidates as well. In the highest performing method, LCP, the highest accuracy increased from 69.78% in 1-best to 86.81% in 10-best.

Table 4.10. *K-best Ranking Results on Error 9*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	40.66	42.31	42.31	63.74	74.73	75.82	**69.78**	**84.62**	**86.81**
LCS	38.46	41.76	42.31	59.89	71.98	74.18	65.38	79.67	82.97
Edit_dist	10.99	36.26	42.31	1.65	3.85	6.04	0.00	2.20	3.85
Soundex	37.36	42.31	42.31	48.90	65.38	72.53	48.90	66.48	72.53
Nysiis	14.84	29.67	42.31	13.74	15.93	15.93	13.74	15.38	15.38
(Baseline) BERT	21.43	36.81	42.31	21.43	36.81	42.31	21.43	36.81	42.31

Table 4.11 shows the k-best ranking results on Error 14. Even in Error 14, the highest performance increased from 17.65% to 35.29% with 1-best to 10-best. In 30,000 candidates, the accuracy increased from 0% to 35.29% with increased k-best.

Table 4.11. *K-best Ranking Results on Error 14*

Candidate Num. (N)	10			1000			30000		
K-best (K)	1	5	10	1	5	10	1	5	10
LCP	5.88	11.76	11.76	5.88	5.88	11.76	11.76	23.53	29.41
LCS	5.88	11.76	11.76	5.88	11.76	23.53	**17.65**	23.53	29.41
Edit_dist	0.00	5.88	11.76	0.00	11.76	11.76	0.00	**29.41**	**35.29**
Soundex	5.88	11.76	11.76	5.88	11.76	11.76	5.88	11.76	17.65
Nysiis	0.00	11.76	11.76	0.00	17.65	23.53	0.00	23.53	**35.29**
(Baseline) BERT	5.88	11.76	11.76	5.88	11.76	11.76	5.88	11.76	11.76

Figure 4.16 shows the highest accuracy score of the k-best results compared to the two baselines, BERT and MoNoise, on the different error types. BERT refers to the first baseline model BERT, and BERT+R refers to the presented method in BERT candidate generation with additional ranking performed, each representing 1-best, 5-best, and 10-best results. MoNoise_Gold refers to the results of the second baseline model MoNoise with gold error detection, which assumes that correct error detection is performed. Therefore, the gold error detection results of MoNoise has been selected for comparison since the presented method assumes that error detection has been performed. The results show that the BERT+R methods

outperformed the baseline BERT on all error types. In terms of the baseline MoNoise, the 1-best results outperformed the baseline on Error 1 and showed equivalent or comparable performance on Error 9 and 14. The 5-best results outperformed the baseline on Error 1, 9, 14 and showed comparable performance on Error 3, 7, and 8. The 10-best results outperformed the baseline on Error 1, 7, 9, 14 and showed comparable performance on Error 3 and 8.

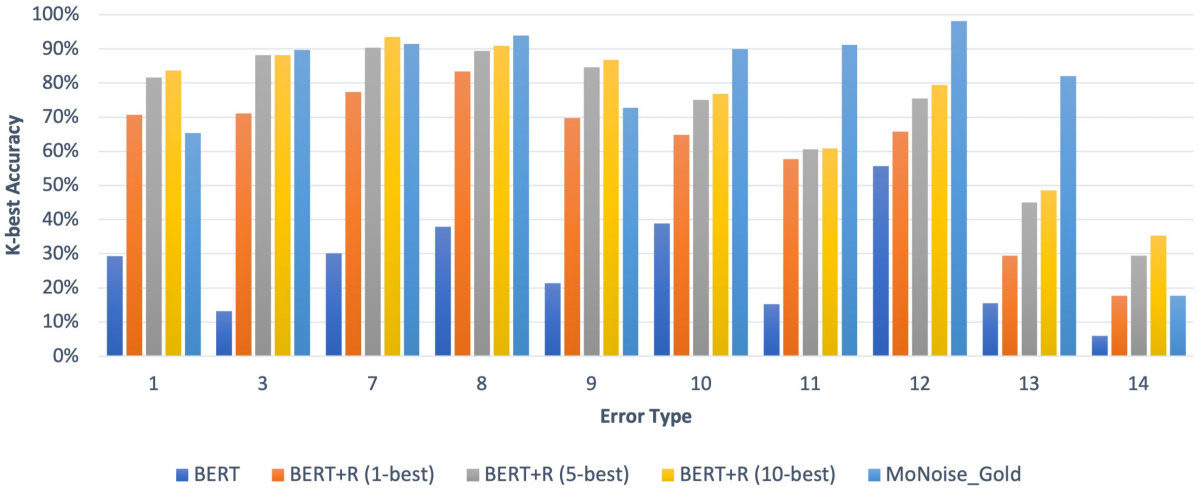

Figure 4.16. Summarization of Ranking Performance based on Error Type

4.2.3 Discussion

The results comparing the $BERT_{base}$ and $BERT_{large}$ model show that the two models show very similar performance and trend in candidate ranking as well. In general, the $BERT_{base}$ model showed slightly higher performance in smaller number of candidates and the $BERT_{large}$ model showed slightly higher performance in larger number of candidates. Overall, all the methods except for the baseline BERT showed very similar performance in the two models. The baseline BERT ranking performance was shown to increase in all error types with the $BERT_{large}$ model.

The 1-best accuracy results showed that different ranking methods were more effective and less effective based on different error types. However, in all the error types, all ranking methods showed very similar performance in 5 candidates, and similar performance until 10

candidates as well except for the baseline BERT method. Also, most methods have shown the highest performance in the 1,000 to 30,000 candidate range.

For interpretation into the different error types, Error 1, 7, and 11 showed a similar trend with Edit_dist as the highest performing method and baseline BERT as the lowest. In Error 1, typographical error, the high performance of Edit_dist was expected since the error type tends to require minor edits. However, this was not expected in Error 7 and 11, repetition error and phonetic transformation error respectively, since both errors include shortening of words which could lead to large edit distance. The high performance of phonetic distances, Soundex and Nysiis, in Error 7 are also understandable since the phonetic similarity remains with repeated characters, e.g. please $\rightarrow$ pleaseee. However, the higher performance of Edit_distance, LCP, and LCS rather than phonetic distances in Error 11 was interesting considering that it is the phonetic transformation error. This result suggests that the phonetic transformation to the human eye could be not entirely in line with phonetic algorithms.

Edit_dist also showed high performance in Error 3, spelling error. With Error 3, it was intuitable that Edit_dist would show high performance since spelling errors usually contain minor edits. However, the performance of Nysiis provided an interesting aspect, since it suggests that spelling errors often maintain close phonetic similarities with the original word. In comparison to Error 1, typographical error, which showed low performance in Nysiis, the high performance shown in Error 3 suggests that this is due to the fundamental differences in the two error types. Since Typographical errors are usually due to accidentally typed characters, the errors are often completely irrelevant to the original character in terms of similarity but rather based on the keys in close proximity on the keyboard. However, in spelling errors, the errors are often based on what the user thinks the spelling is, which will often lead to the errors sharing similar phonetic information with the original word, thus leading to high performance in phonetic distance. Although these two error types, Error 1 and 3, may seem very similar on the surface, the performance of Nysiis has been able to reveal this fundamental difference between the two error types.

Error 10 and 12 shared a similar trend with LCS as the highest performing method with the baseline BERT showing second highest performance. The dominant performance of LCS was unexpected in Error 10, shortening other error, since many characters were expected to be omitted

for shortening purposes. The high performance of LCS was expected in Error 12, regular transformation error, since the error generally maintains similar structure to the original word with minor or partial omissions. However, the low performance of Edit_dist was not expected. Another interesting result was that all the other methods except for LCS and BERT started to dive in performance after 10 candidates, leaving the two methods dominant. Also, this particular error type showed higher performance in lower candidate numbers.

One of the unique results from these two error types was that the baseline BERT ranking method showed the second highest performance, since the baseline BERT showed relatively low performance in all the other error types. Inferring from the significantly high results of the baseline BERT on Error 12, one interpretation could be that this is due to the majority of the errors including the most frequently used vocabulary in sentences such as pronouns (e.g. you) and prepositions (e.g., with, about, before). Since the BERT model masks the error word and only takes the surrounding context into account, the high performance on error words that include high-frequency words is understandable.

LCS was the highest performing method in Error 8 and 13 as well. However, in these two error types, Soundex was the second highest performing method rather than the baseline BERT. The dominant performance of LCS was unexpected in Error 8, shortening vowels error, since many characters (i.e. vowels) were expected to be omitted for shortening purposes. The high performance of LCS was expected in Error 13, slang error, since the error generally maintains some similarity with the original word, e.g. brother $\rightarrow$ bra. However, the overall ranking performance on Error 13 is very low, thus an improvement on the ranking method is required.

Error 9 did not show a similar trend with the other error types, with LCP showing highest performance and Edit_dist lowest. The highest performance of LCP was expected, since Error 9, shortening end error, generally maintains the same prefix with the rest of the word omitted, e.g. favorite $\rightarrow$ fav. The low performance of Edit_dist is also expected since the error word is often shortened significantly. Since Error 14, unknown error was defined by van der Goot et al. (2018) as errors unknown to the annotators or errors with disagreement across annotators, the errors do not maintain a consistent pattern and only contain 17 errors. Thus, this error type can be disregarded in terms of ranking performance.

In all error types, the highest ranking method outperformed the baseline BERT method, showing that using additional ranking methods is capable of improving the ranking performance of the BERT-generated candidates. Meanwhile, the error types 10 and 12 suggest that in certain types of errors, the ranking performed by the BERT model is already highly sufficient, especially in Error 12, regular transformation error.

The k-best accuracy results showing accuracy increase up to 61.84%, reveals that even though the 1-best accuracy does not show substantially high performance, the correct candidate does indeed exist in the 5-best or 10-best candidates, which shows high potential with further performance enhancement through different or more complex ranking methods.

The overall result in comparison with the baselines, BERT and MoNoise, represented in Figure 4.16 shows that the additional ranking method applied on the candidates generated through the BERT model has outperformed the baseline BERT ranking on all error types. Although, the proposed method has not been able to reach the state-of-the-art performance of the baseline MoNoise with gold error detection, the 10-best results have been able to outperform or show comparable performance on 6 out of 10 different error types.

CHAPTER 5. CONCLUSION

5.1 Conclusion and Analysis

This study suggests a simple normalization method that utilizes the state-of-the-art pre-trained language model, BERT, for candidate generation and simple ranking methods for candidate ranking based on different error types. This study has investigated the performance of different stages of normalization based on the different types of errors. To the best of my knowledge, this is the first study to apply the BERT model to the candidate generation of normalization without fine-tuning. Based upon the error taxonomy categorized by van der Goot et al. (2018), the candidate generation performance of the BERT model and the candidate ranking performance of five different ranking methods were compared to the baseline models, BERT and MoNoise.

The results of the BERT model performance on candidate generation revealed the potential of BERT in being applied to generating candidates for normalization. Further results on candidate ranking showed that in certain types of errors, the basic ranking performed by the BERT model already shows high performance, especially in regular transformation errors. In performing additional ranking on the candidates generated through the BERT model, the results have been able to significantly outperform the first baseline BERT ranking. Although performance over the second baseline MoNoise was not substantial, the 10-best results were able to show comparable performance on 6 out of 10 different error types.

The goal of this study is to enlighten on the potential that the BERT model holds in normalization and the prospect of a simple BERT-infused normalization method that can be improved through different ranking methods on different error types, as well as guidance for users performing normalization to preprocess informal text data based on different types of errors it contains. Table 5.1 shows the best performing ranking method for each error type. For datasets including more typographical, spelling, repetition, and phonetic-transformation errors (Error 1, 3, 7, 11), edit distance would be recommended for use in ranking and should avoid using the BERT basic ranking. For datasets including more shortening-vowels, shortening-end, shortening-other,

regular-transformation, and slang errors (Error 8, 9, 10, 12, 13), the use of the longest common subsequence would be recommended for use and to refrain from using edit distance. If one would like to utilize a single ranking method on all error types, longest common subsequence would be recommended for use, since it is the highest performing method among the different error types. As a simple normalization method, this method could also be provided as an alternative to complex and high-computing methods when performing normalization in low resource environments.

Table 5.1. *Best Performing Ranking Method based on Error Type*

Type	**Error Description**	**Example**	**Ranking Method**	**Cand Num**
Error 1	Typographical error	can't → ca'nt	Edit_dist	30000
Error 3	Spelling error	neighbors → neighbours	Nysiis/Edit_dist	30000
Error 7	Repetition error	please → pleaseeeee	Edit_dist	1000
Error 8	Shortening vowels error	people → ppl	LCS	1000/30000
Error 9	Shortening end error	favorite → fav	LCP/LCS	30000
Error 10	Shortening other error	because → bc	LCS	1000/30000
Error 11	Phonetic transformation error	though → tho	Edit_dist	30000
Error 12	Regular transformation error	going → goin	LCS	1000
Error 13	Slang error	thanks → thx	LCS	1000
Error 14	Unknown error	they → nem	LCS	30000

5.2 Future Work

From the findings of this study, many aspects for prospective improvement lies in the proposed method. The following are a few of the recommendations on future research for improvement:

In this study, five simple ranking methods were investigated in performing ranking on the candidates generated through the BERT model. With the potential seen through these five ranking methods and the k-best accuracy results, additional ranking methods and combinations of simple ranking methods should be further investigated for performance enhancement. The high candidate generation results of the BERT model on the different types of errors also suggest that with the appropriate ranking method, there is great potential for improvement. A different

variation of edit distance, Damerau–Levenshtein distance, could be one potential ranking method to be explored. Damerau-Levenshtein includes the transposition of two adjacent characters in its edit operations compared to the Levenshtein distance used in this study. In allowing the swapping of adjacent characters, Damerau-Levenshtein could enhance the performance of edit distance and the phonetic algorithms that use edit distance for calculation. The results of (Christanti & Naga, 2018) showed that using Damerau-Levenshtein distance improved the accuracy over the Levenshtein distance.

One of the critical limitations of using the BERT model for candidate generation was the incapability to handle special characters in words and 1:N or N:1 replacements. However, in terms of the four error types that the BERT model was incapable of handling due to these factors, the Most-Frequent Method (MFR) (van der Goot & Çetinoğlu, 2021) could be a suitable solution. In MFR, normalization is performed by replacing the error word with the most frequent replacement pair seen in training. For the errors not in the training data, the error word is returned. This simple method has shown relatively high performance of an ERR score of 61.88% on the *LexNorm2015* dataset and is also highly suitable for the error types that need handling outside of the BERT model. Therefore, incorporating this method alongside the BERT-based method could potentially be a solution for this limitation of BERT in normalization.

Another limitation of incorporating the BERT model to normalization was that the model could only perform the MLM task on one masked word at a time. However, demonstration of performing the MLM task with the BERT model on multiple masked tokens can be found in the BERT Language Model Demo presented by the Bar Ilan NLP Lab (Bar Ilan NLP Lab, n.d.). Modifying the BERT model and applying the multiple mask token of BERT for the normalization of multiple tokens would be worth investigating.

Also, applying other variations of the BERT model such as DistilBERT (Sanh et al., 2020), which compared to the $\text{BERT}_{\text{base-uncased}}$ model contains 40% fewer parameters with 60% faster speed and preserves over 95% of the performance at the same time, could further enhance the performance of BERT candidate generation.

Considering the BERT model has a multilingual model that supports 104 languages, another future aspect would be to incorporate the multilingual BERT model for the normalization of multiple languages.

REFERENCES

Akhtar, M. S., Sikdar, U. K., & Ekbal, A. (2015). Iitp: Hybrid approach for text normalization in twitter. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (p. 106-110).

Alegria, I., Aranberri, N., Fresno, V., Gamallo, P., Padró, L., Vicente, I. S., … Zubiaga, A. (2013). Introducción a la tarea compartida tweet-norm 2013: Normalización léxica de tuits en español. *Tweet-Norm@SEPLN*, 1–9.

Aw, A., Zhang, J., M.and Xiao, & Su, J. (2006). A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions* (pp. 33–40).

Baldwin, T., de Marneffe, M.-C., Han, B., Kim, Y.-B., Ritter, A., & Xu, W. (2015). Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (pp. 126–135).

Bar Ilan NLP Lab. (n.d.). *Bert language model demo - bar ilan nlp lab.* `https://nlp.biu.ac.il/~ohadr/bert/demo?text=London\%20is\%20a\%20city\%20in\%20the\%20country\%20of\%20---.&word1=&word2=`. (Accessed: 2021-03-23)

Beaufort, R., Roekhaut, S., Cougnon, L.-A., & Fairon, C. (2010). A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of the 48th Annual Meeting of the ACL (ACL 2010)* (pp. 770–779).

Beckley, R. (2015). Bekli:a simple approach to twitter text normalization. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (pp. 82–86).

Berend, G., & Tasnádi, E. (2015). Uszeged: Correction type-sensitive normalization of english tweets using efficiently indexed n-gram statistics. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (pp. 120–125).

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 111–119. Retrieved from `https://doi.org/10.1023/A:1010933404324` doi: 10.1023/A:1010933404324

Brill, E., & Moore, R. C. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting on association for computational linguistics (acl '00)* (pp. 286–293). Hong Kong.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, *19*(2), 263–311. Retrieved from https://www.aclweb.org/anthology/J93-2003

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., & Basu, A. (2007). Investigation and modeling of the structure of texting language. In *International Journal on Document Analysis and Recognition (IJDAR)* (Vol. 10, pp. 157–174).

Christanti, V. M., & Naga, D. S. (2018, apr). Fast and accurate spelling correction using trie and damerau-levenshtein distance bigram. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *16*(2), 827. Retrieved from https://doi.org/10.12928\%2Ftelkomnika.v16i2.6890 doi: 10.12928/telkomnika.v16i2.6890

Chrupala, G. (2014). Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)* (pp. 680–684).

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)* (pp. 1–18).

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of ACL*.

Contractor, D., Faruquie, T. A., & Subramaniam, L. V. (2010). Unsupervised cleansing of noisy text. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)* (pp. 189–196).

Cook, P., & Stevenson, S. (2009). An unsupervised model for text message normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity (CALC '09)* (pp. 71–78).

Cucerzan, S., & Brill, E. (2004). Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)* (pp. 293–300).

De Clercq, O., Schulz, S., Desmet, B., & Hoste, V. (2014). Towards shared datasets for normalization research. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*. ACL.

Erjavec, T., Fišer, D., Č ibej, J., Špela Arhar Holdt, Ljubešić, N., & Zupan, K. (2017). *Cmc training corpus janes-tag 2.0*. Retrieved from `http://hdl.handle.net/11356/1123` (Slovenian language resource repository CLARIN.SI)

Eryiğit, G., & Torunoğ-Selamet., D. (2017). Social media text normalization for turkish. *Natural Language Engineering*, *23*(6), 835–875.

Foster, J., Çetinoğlu, O., Wagner, J., Roux, J. L., Hogan, S., Nivre, J., ... van Genabith, J. (2011). #hardtoparse: Pos tagging and parsing the twitterverse. In *2011 AAAI Workshop* (pp. 421–432).

Gao, J., Li, X., Micol, D., Quirk, C., & Sun, X. (2010). A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)* (pp. 358–366).

Gouws, S., Hovy, D., & Metzler, D. (2011). Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First Workshop on Unsupervised Learning in NLP* (pp. 82–90).

Gouws, S., Metzler, D., Cai, C., & Hovy., E. (2011). Contextual bearing on linguistic variation in social media. In *Proceedings of the ACL-11 Workshop on Language in Social Media.* Association for Computational Linguistics.

Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks*. Berlin, Heidelberg: Springer.

Han, B. (2014). *Improving the utility of social media with natural language processing*. Unpublished doctoral dissertation, Melbourne, Australia.

Han, B., & Baldwin, T. (2011). Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 368–378).

Han, B., Cook, P., & Baldwin, T. (2012). Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning 2012 (EMNLP-CoNLL 2012)* (pp. 421–432).

Han, B., Cook, P., & Baldwin, T. (2013). Lexical normalization for social media text. In *ACM Transactions on Intelligent Systems and Technology* (Vol. 4).

Hassan, H., & Menezes, A. (2013). Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)* (pp. 1577–1586).

Hinton, G., Deng, L., Yu, D., Dahl, G., rahman Mohamed, A., Jaitly, N., ... Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine*, *26*(6), 82–97.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780. Retrieved from `https://doi.org/10.1162/neco.1997.9.8.1735` doi: 10.1162/neco.1997.9.8.1735

Jelinek, F. (1997). *Statistical methods for speech recognition*. Cambridge, MA, USA: MIT Press.

Jin, N. (2015). Ncsu-sas-ning: Candidate generation and feature engineering for supervised lexical normalization. In *Proceedings of the Workshop on Noisy User-generated Text* (pp. 87–92).

Kobus, C., Yvon, F., & Damnati, G. (2008). Normalizing sms: are two metaphors better than one? In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)* (pp. 441–448).

Kumar, A., Makhija, P., & Gupta, A. (2020). User generated data: achilles' heel of bert. *arXiV preprint arXiv:2003.12932v2*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the ICML* (pp. 282–289).

Leeman-Munk, S., Lester, J., & Cox, J. (2015). Ncsu_sas_sam: Deep encoding and reconstruction for normalization of noisy text. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (p. 154-161).

Levandowsky, M., & D., W. (1971). Distance between sets. In *Nature* (Vol. 234, pp. 34–356).

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklady*, *10*(8), 707–710. (Original in Russian in Dokl. Akad. Nauk SSSR 163, 4, 845-848, 1965)

Li, C., & Liu, Y. (2012a). Improving text normalization using character-blocks based models and system combination. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)* (pp. 1587–1602).

Li, C., & Liu, Y. (2012b). Normalization of text messages using character- and phone-based machine translation approaches. In *Proceedings of 13th Interspeech.*

Li, C., & Liu, Y. (2014). Improving text normalization via unsupervised model and discriminative reranking. In *Proceedings of the ACL 2014 Student Research Workshop* (pp. 86–93).

Li, C., & Liu, Y. (2015). Joint pos tagging and text normalization for informal text. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)* (pp. 1263–1269).

Ling, W., Dyer, C., Black, A. W., & Trancoso, I. (2013). Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)* (pp. 73–84).

Liu, F., Weng, F., & Jiang, X. (2012). A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)* (pp. 1035–1044).

Liu, F., Weng, F., Wang, B., & Liu, Y. (2011a). Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 71–76).

Liu, F., Weng, F., Wang, B., & Liu, Y. (2011b). Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 359–367).

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., & Wang, P. (2020). K-bert: Enabling language representation with knowledge graph. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020).*

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiV preprint arXiv:1907.11692*.

Ljubešić, N., Erjavec, T., Miličević, M., & Samardžić, T. (2017a). *Croatian twitter training corpus ReLDI-NormTagNER-hr 2.0.* Retrieved from `http://hdl.handle.net/11356/1170` (Slovenian language resource repository CLARIN.SI)

Ljubešić, N., Erjavec, T., Miličević, M., & Samardžić, T. (2017b). *Serbian twitter training corpus ReLDI-NormTagNER-sr 2.0.* Retrieved from `http://hdl.handle.net/11356/1171` (Slovenian language resource repository CLARIN.SI)

Ljubešić, N., Erjavec, T., & Fišer, D. (2014). Standardizing tweets with character-level machine translation. In *Computational Linguistics and Intelligent Text Processing* (pp. 164–175).

Ljubešić, N., Zupan, K., Fišer, D., & Erjavec, T. (2016). Normalising slovene data: historical texts vs. user-generated content. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)* (pp. 146–155).

Lourentzou, I., Manghnani, K., & Zhai, C. (2019). Adapting sequence to sequence models for text normalization in social media. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 13, pp. 335–345).

Min, W., & Mott, B. (2015). Ncsu_sas_wookhee: A deep contextual long-short term memory model for text normalization. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (pp. 111–119).

Muller, B., Sagot, B., & Seddah, D. (2019). Enhancing bert for lexical normalization. In *Proceedings of the 2019 EMNLP Workshop W-NUT: The 5th Workshop on Noisy User-generated Text* (pp. 297–306).

Nguyen, D. Q., Vu, T., & Nguyen, A. T. (2020). Bertweet: A pre-trained language model for english tweets. *arXiV preprint arXiv:2005.10200v1*.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., & Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT 2013* (pp. 380–390).

Pennell, D., & Liu, Y. (2011a). A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of 5th International Joint Conference on Natural Language Processing* (pp. 974–982).

Pennell, D., & Liu, Y. (2011b). Toward text message normalization: Modeling abbreviation generation. In *Proceedings of 2011 IEEE International Conference on Natural Language Processing (ICASSP '11)* (pp. 5364–5367).

Radivchev, V., & Nikolov, A. (2019). Nikolov-radivchev at semeval-2019 task 6: Offensive tweet classification with bert and ensembles. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)* (pp. 691–695).

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Shannon, C. E. (1948). A mathematical theory of communication. In *Bell Syst. Tech. J.* (Vol. 27, pp. 379–423, 623–656).

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts., C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1631–1642). Association for Computational Linguistics.

Sridhar, V. K. R. (2015). Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of NAACL-HLT 2015* (pp. 8–16). Association for Computational Linguistics.

Supranovich, D., & Patsepnia, V. (2015). Ihs_rd: Lexical normalization for english tweets. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (pp. 78–81). Retrieved from `https://doi.org/10.1145/567752.567774` doi: 10.1145/567752.567774

Toutanova, K., & Moore, R. C. (2002). Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02)* (pp. 144–151). Philadelphia, USA.

van der Goot, R. (2019). Monoise: A multi-lingual and easy-to-use lexical normalization tool. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 201–206).

van der Goot, R., & Çetinoğlu, O. (2021). Lexical normalization for code-switched data and its effect on pos-tagging. *arXiV preprint arXiv:2006.01175v2*.

van der Goot, R., Plank, B., & Nissim, M. (2017). To normalize, or not to normalize: The impact of normalization on part-of-speech tagging. In *Proceedings of the 3rd Workshop on Noisy User-generated Text* (pp. 31–39).

van der Goot, R., Üstün, A., Ramponi, A., & Plank, B. (2020). Massive choice, ample tasks(machamp):a toolkit for multi-task learning in nlp. *arXiV preprint arXiv:2005.14672v2*.

van der Goot, R., & van Noord, G. (2017). Monoise: Modeling noise using a modular normalization system. *arXiV preprint arXiv:1710.03476*.

van der Goot, R., van Noord, R., & van Noord, G. (2018). A taxonomy for in-depth evaluation of normalization for user generated content. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (pp. 684–688). Miyazaki, Japan: European Language Resources Association (ELRA).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 6000–6010).

Wagner, J., & Foster, J. (2015). Dcu-adapt: Learning edit operations for microblog normalization with the generalised perceptron. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text* (pp. 93–98).

Wang, P., & Ng, H. T. (2013). A beam-search decoder for normalization of social media text with application to machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)* (pp. 471–481).

Xu, K., Xia, Y., & Lee, C.-H. (2015). Tweet normalization with syllables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (pp. 920–928).

Xue, Z., Yin, D., & Davison, B. D. (2011). Normalizing microtext. In *Proceedings of the AAAI-11 Workshop on Analyzing Microtext* (pp. 74–79).

Yang, Y., & Eisenstein, J. (2013). A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)* (pp. 61–72).

Yao, L., Mao, C., & Luo, Y. (2019). Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Zhang, C., Baldwin, T., Ho, H., Kimelfeld, B., & Li, Y. (2013). Adaptive parser-centric text normalization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)* (pp. 1159–1168).

Zhang, J., Pan, J., Yin, X., Li, C., Liu, S., Zhang, Y., ... Ma, Z. (2020). A hybrid text normalization system using multi-head self-attention for mandarin. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

Zhu, C., Tang, J., Li, H., Ng, H. T., & Zhao, T. (2007). A unified tagging approach to text normalization. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)* (pp. 688–695).