NEURAL TEXT NORMALIZATION FOR TURKISH SOCIAL MEDIA

TÜRKÇE SOSYAL MEDYA İÇİN NÖRAL METİN NORMALİZASYONU

Sinan GÖKER

Asst. Prof. Dr. Burcu CAN BUĞLALILAR

Supervisor

Submitted to Graduate School of Science and Engineering of

Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

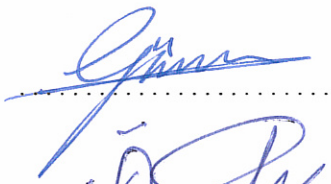
in Computer Engineering

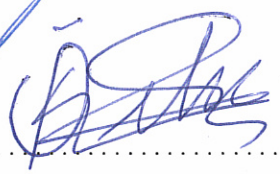
06/2018

This work named **"Neural Text Normalization for Turkish Social Media"** by **Sinan GÖKER** has been approved as a thesis for the Degree of **MASTER OF SCIENCE IN COMPUTER ENGINEERING** by the below mentioned Examining Committee Members.

Prof. Dr. Pınar KARAGÖZ
Head ..... 

Asst. Prof. Dr. Burcu CAN BUĞLALILAR
Supervisor ..... 

Asst. Prof. Dr. Gönenç ERCAN
Member ..... 

Asst. Prof. Dr. Özkan KILIÇ
Member ..... 

Asst. Prof. Dr. Adnan ÖZSOY
Member ..... 

This thesis has been approved as a thesis for the Degree of **MASTER OF SCIENCE IN COMPUTER ENGINEERING** by Board of Directors of the Institute for Graduate School of Science and Engineering.

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU
Director of the Institute of
Graduate School of Science and Engineering

YAYINLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

☑ **Tezimin/Raporumun tamamı dünya çapında erişime açılabilir ve bir kısmı veya tamamının fotokopisi alınabilir.**
(Bu seçenekle teziniz arama motorlarında indekslenebilecek, daha sonra tezinizin erişim statüsünün değiştirilmesini talep etseniz ve kütüphane bu talebinizi yerine getirse bile, tezinin arama motorlarının önbelleklerinde kalmaya devam edebilecektir.)

☐ **Tezimin/Raporumun ..... tarihine kadar erişime açılmasını ve fotokopi alınmasını (İç Kapak, Özet, İçindekiler ve Kaynakça hariç) istemiyorum.**
(Bu sürenin sonunda uzatma için başvuruda bulunmadığım taktirde, tezimin/raporumun tamamı her yerden erişime açılabilir, kaynak gösterilmek şartıyla bir kısmı ve ya tamamının fotokopisi alınabilir)

☐ **Tezimin/Raporumun ..... tarihine kadar erişime açılmasını istemiyorum, ancak kaynak gösterilmek şartıyla bir kısmı veya tamamının fotokopisinin alınmasını onaylıyorum.**

☐ **Serbest Seçenek/Yazarın Seçimi**

16 / 07 / 2018

Sinan Göker

ETHICS

In this thesis study, prepared in accordance with the spelling rules of Institute of Graduate School of Science and Engineering of Hacettepe University,

I declare that

- all the information and documents have been obtained in the base of the academic rules.
- all audio-visual and written information and results have been presented according to the rules of scientific ethics
- in case of using others works, related studies have been cited in accordance with the scientific standards
- all cited studies have been fully referenced
- I did not do any distortion in the data set
- and any part of this thesis has not been presented as another thesis study at this or any other university.

18/06/2018

Sinan GÖKER

ABSTRACT

Neural Text Normalization for Turkish Social Media Texts

Sinan GÖKER

Master of Science,Computer Engineering Department

Supervisor: Asst. Prof. Dr. Burcu CAN BUĞLALILAR

June 2018, 77 pages

Social media has become a rich data source for natural language processing tasks with its worldwide use; however, it is hard to process social media data directly in language studies due to its unformatted nature. Text normalization is the task of transforming the noisy text into its canonical form. It generally serves as a preprocessing task in other NLP tasks that are applied to noisy text and the success rate gets higher when studies are performed on canonical text.

In this study, two neural approaches are applied for Turkish text normalization task: Contextual Normalization approach using distributed representations of words and Sequence-to-Sequence Normalization approach using encoder-decoder neural networks. As the conventional approaches applied to Turkish and also other languages are mostly domain specific, rule-based or cascaded, they are already becoming less efficient and less successful due to the change of the language use in social media. Therefore the proposed methods provide more comprehensive solution that are not sensitive to the language change in social media.

Keywords: text normalization, distributed representation, word2vec, encoder, decoder, semantic, unsupervised learning, long short-term memory (LSTM), deep learning

ÖZET

Türkçe Sosya Medya Metinleri için Nöral Metin Normalizasyonu

Sinan GÖKER

Yüksek Lisans,Bilgisayar Mühendisliği

Danışman: Yrd. Doç. Dr. Burcu CAN BUĞLALILAR

Haziran 2018, 77 sayfa

Sosyal medya, dünya çapında yaygın kullanımı ile doğal dil işleme çalışmaları için zengin bir veri kaynağı haline gelmiştir; Bununla birlikte, kuralsız metinlerde oluşan doğası nedeniyle dil çalışmalarında sosyal medya verilerini doğrudan kullanabilmek oldukça zordur. Hatalı yazılmış bir metni doğru yazılmış haline dönüştürme işlemine metin normalleştirme denir. Metin normalleştirme çoğunlukla diğer doğal dil işleme çalışmalarında ön hazırlık işlemi olarak görev alır ve metinlerin doğru yazılmış halleri üzerinden yapılan çalışmalarda başarı oranı daha yüksek olur.

Bu çalışma kapsamında Türkçe metin normalleştirme görevi için iki farklı yaklaşım uygulanmaktadır: Kodlayıcı-kod çözücü (encoder-decoder) yapay sinir ağları modeli kullanılarak diziden diziye (sequence-to-sequence) normalleştirme yaklaşımını ve sözcüklerin dağıtık temsilleri (distributed representation of words) kullanılarak bağlamsal normalleştirme yaklaşımı ile metin normalleştirme görevi gerçekleştirilmiştir. Türkçeye ve diğer dillere uygulanan mevcut yaklaşımlar çoğunlukla alana yönelik, kural tabanlı ya da kademeli normalleştirme kurallarının izlendiği çalışmalar olduğundan, sosyal medyada dil kullanım alışkanlığının değişmesi bu çalışmaların verimini ve başarı oranını düşürmektedir. Bu nedenle önerilen

yöntemler sosyal medyada dil kullanımındaki değişikliklerden etkilenmeyen daha kapsamlı bir çözüm sunmaktadır.

Anahtar Sözcükler: metin normalizasyonu, dağıtılmış gösterim, word2vec, kodlayıcı, kod çözücü, anlamsal, gözetimsiz öğrenme, LSTM, derin öğrenme

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my master thesis supervisor Asst. Prof. Dr. Burcu Can Buğlalılar for her precious advices, guidance and her everlasting patience. Since I started working with her, I have learned priceless information from her.

I would also like to thank my thesis committee members for accepting to be in my thesis committee.

In addition, I would like to thank Zeynep Aydoğu, for all her love, support and patience. I also want to thank to my dear friends Atilla Suncak and Mehmet Özgen for providing me with continuous encouragement throughout my years of study and through the process of researching and writing this thesis. I would also like to thank to my all friends and colleagues for good wishes.

I would finally like to express my appreciate to my beloved family for believing and supporting me for all my life.

CONTENTS

Page

FIGURES

Page

TABLES

1. INTRODUCTION

1.1. Overview

Social media occupies an important place in our lives. People share almost every idea, thought and dream of their own through it and thus the amount of produced content on those platforms is still on the increase. From this aspect, social media has become a rich and highly valuable resource for natural language processing (NLP) and machine learning researchers.

Although the amount of social media content increases, the amount of clean data is limited due to spelling mistakes, misuses, typo errors, common current abbreviations, and structural disorders, which have a negative effect on NLP studies for social media content. Moreover, every age creates its own usage of natural language on social media. Therefore, the common errors in writing also changes from one generation to another. In order to increase the success rate of converting the data into correct form, the noisy texts are required to be corrected. The task of transforming noisy[1] text into its canonical[2] form is called text normalization. Some text normalization examples are given below.

Example: The noisy text

Günlerden bir gün yinee bu bnkta oturuyodum.

is normalized as :

Günlerden bir gün yine bu bankta oturuyordum.(One day, I was sitting on this bench again.)

Example: The noisy text

Karincalar yagmurda telas icinde kosusturuyorlardi.

is normalized as :

Karıncalar yağmurda telaş içinde koşuşturuyorlardı. (Ants were running around in a hurry in the rain.)

Text Normalization is the preliminary stage for the NLP application on social media, because applying it as a preprocessing in the tasks such as Sentiment Analysis [8], Automatic Speech

[1] In text normalization task, the word 'noisy' stands for a wrongly written word.
[2] In text normalization task, the word 'canonical' stands for the correct form of a word.

Recognition (ASR) [9], Speech Processing [10], Morphological Disambiguation [11] provides a high success ratio. However, in absence of text normalization, the success ratio is relatively lower such as Morphological Disambiguation [12], Sentiment Analysis [13] studies. Consequently, increasing success rate of normalization task provides more successful results in other NLP tasks and machine learning fields.

1.2. Motivation

Text normalization provides successful solutions to correct noisy text, make the abbreviations understandable, and normalize specific uses (hashtag, mention, link, etc.) for social media. The output data from this process provides a more meaningful format, which makes the text normalization task an important process for other NLP applications.

Many text normalization techniques applied to social media data are insufficient and inefficient because those are mostly rule-based techniques that are generally manually defined. This is because of the change of language use in time in social media, which brings together new error patterns. New writing styles are introduced in different human generations that need to be handled differently. Such rule-based methods require too much labor for updating the rules in time. As conventional techniques cannot follow those patterns, the success rate of the techniques declines in time.

In this thesis, we introduce methods that are different from the other conventional methods so that no static rules are defined. Since there is no rule dependency, the methods support normalization of different error types due to normalizing over both distributed representations of words and orthographic patterns that are captured automatically by an encoder-decoder neural network architecture. The only thing that is expected for the system to diagnose the different error patterns is to train the system with the data set containing the new errors.

Word representations are learned through the contextual similarities between words, as Firth suggests[3]. Each word is represented by a feature vector that bears any lexical, semantic, or syntactic feature of the word in vector space. Therefore similar words tend to have similar word representations and they will be closer to each other in the vector space.

[3] "You shall know a word by the company it keeps"[14]

Word2vec [15] is used to learn the neural word representations that is a prediction-based model that learns the word representations using the contextual information of each word without counting the co-occurrences in this study. Therefore, words in similar contexts have similar word representations. Noisy social media data in Turkish language is used as input and each noisy word is matched with its canonical candidate words by using its word representation. Here we assume that noisy forms of a word will also have similar word representations. For example, *yapıcam*, *yapacam*, and *yapcam* (meaning *I will do*) will be all in similar contexts and therefore all will be having similar word representations. Once having the word representation of one of those incorrect forms, it will lead to the canonical form *yapacağım* that has a similar representation to the noisy forms because they all mean the same.

As a second normalization approach, we present an encoder-decoder model to learn the error rules automatically. Encoder-decoder architecture for recurrent neural networks [16] is a model mainly used for sequence-to-sequence learning tasks, such as machine translation, text summarization, etc. The encoder network takes a sequence input (i.e. a text in source language in machine translation task) and outputs the encoded input which is represented by a feature vector. Once the input is encoded, it is given as input to the decoder network that transforms/decodes the encoded input into the actual input or the intended input (i.e. the text in target language in machine translation task). Here, a fixed-length internal feature vector is learned, which represents the structural relation between the input and output. Therefore, the transformation from source to target is performed in a rule independent way for further predictions with the new input data. In this study, as for the text normalization, the source is the noisy text and the target is the canonical form of the noisy text.

This study is the first application of proposed text normalization approaches to Turkish social media, although both of methods are applied to other languages. Both approaches are newly applied methods for Turkish text normalization studies due to the fact that they are made up of rule-independent methods, and both of them achieves significant normalization results for text normalization of Turkish social media. The first application of neural methods for Turkish text normalization has been done in this study to the best of our knowledge.

1.3. Research Questions

- Can a rule-independent text normalization method be applied to social media data in Turkish?
- Can a rule independent method have as high accuracy as a conventional rule-based method in text normalization?
- How successful are neural normalization methods in agglutinative languages such as Turkish?

1.4. Thesis Structure

The structure of the thesis is given as follows:

Chapter 2 describes background information of the proposed approaches from two different aspects of normalization: Surface form normalization and semantic form normalization.

Chapter 3 reviews the methodology used in text normalization. This chapter also argues the strong and weak sides of those studies.

Chapter 4 explains the algorithms, methodology steps and structures of the proposed approaches in detail by giving examples.

Chapter 5 represents the evaluations, statistical results of the approaches in detail. This chapter also contains comparisons of proposed approaches with other baseline studies. We end this chapter with an error analysis.

Finally, **Chapter 6** concludes the thesis with a brief summary and discussion on the proposed approaches, contributions to literature and gives potential future work.

2. BACKGROUND

In this section, we give background information to prepare the reader for the rest of the thesis. The study has been performed by using two different approaches which are related to two methods: Surface Form Normalization deals with normalizing the text value by using its orthographic features while Semantic Normalization utilizes the contextual features. This chapter provides brief information on Surface Form Normalization and Semantic Normalization. Methods used for Surface Form normalization are explained in Section 2.1. and Semantic Normalization is described in Sections 2.2.

2.1. Surface Form Normalization

Surface Form Normalization is a normalization method that uses morphological and orthographic features of the given text. The main goal is to analyze noisy text orthographically and then to predict the canonical form as a result of several calculations. The methods in this category either use some distance metrics between noisy and canonical forms or directly learn the rules applied when creating noisy form of a given text.

2.1.1. Edit Distance (ED)

This metric is used for quantifying how dissimilar two texts are. Total number of deletion, insertion and substitution edit operations between those two texts are computed to measure the distance between them.

The edit distance between $a = a_1 \dots a_n$ and $b = b_1 \dots b_m$ where a and b are two strings and $a_1 \dots a_n$ and $b_1 \dots b_m$ are their letters is given as d_{mn} as follows:

$$
\textit{ED(a,b)} = \begin{cases} d_{i0} = \sum_{k=1}^{i} w_{del}(b_k), & \text{for } 1 \le i \le m \\ d_{0j} = \sum_{k=1}^{j} w_{ins}(a_k), & \text{for } 1 \le j \le n \\ d_{ij} = \begin{cases} d_{i-1,j-1} & \text{for } a_j = b_i \\ min \begin{cases} d_{i-1,j} + w_{del}(b_i) \\ d_{i,j-1} + w_{ins}(a_j) \\ d_{i-1,j-1} + w_{sub}(a_j, b_i) \end{cases} & \text{for } a_j \neq b_i \end{cases}, & \text{for } 1 \le i \le m, 1 \le j \le n \end{cases} \tag{1}
$$

Here, i and j are the indexes of the edit distance matrix which is created by applying the above rules. d_{i0} denotes the first column, d_{0j} denotes the first row and d_{ij} denotes rest of the matrix. At the end of process, the value of d_{mn} is the edit distance between the two strings.

2.1.2. Longest Common Subsequence (LCS)

Longest common subsequence is the longest common substring of the given strings.

An example longest common subsequence between two Turkish words is given in Figure 2.1.

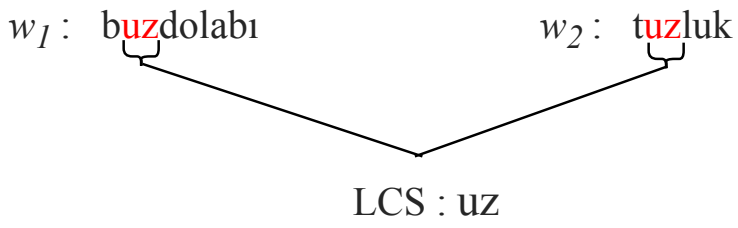


Figure 2.1. LCS example on two strings

The LCS between $a = a_1 \dots a_n$ and $b = b_1 \dots b_n$ where a and b are two strings and $a = a_1 \dots a_n$ and $b = b_1 \dots b_n$ are their letters is given as $LCS(a_i, b_j)$ as follows:

$$
LCS(a_i, b_j) = \begin{cases} \emptyset, & \text{if } i = 0 \text{ or } j = 0 \\ LCS(a_{i-1,b_{j-1}}) \frown a_i, & \text{if } a_i = b_j \\ longest(LCS(a_i, b_{j-1}), LCS(a_{i-1}, b_j)), & \text{if } a_i \neq b_j \end{cases} \tag{2}
$$

2.1.3. Longest Common Subsequence Ratio (LCSR)

LCSR denotes the longest common subsequence ratio [17] between two words. This ratio is calculated by dividing the LCS by the maximum length of the given sequences.

The LCSR between two words w_1 and w_2 is calculated as follows:

$$LCSR(w_1, w_2) = \frac{LCS(w_1, w_2)}{MaxLength(w_1, w_2)} \tag{3}$$

2.1.4. Lexical Similarity Cost (SimCost)

Lexical similarity cost (SimCost) is the orthographic similarity metric between two words and is calculated by dividing the LCSR by the ED (see Section 2.1.1.) similar to the study of Hassan and Menezes [5].

The SimCost between two words w_1 and w_2 is defined as follows.

$$SimCost(w_1, w_2) = \frac{LCSR(w_1, w_2)}{ED(w_1, w_2)} \tag{4}$$

2.1.5. Encoder – Decoder

In addition to the distance metrics used for finding the canonical form of a noisy string, encoder-decoder models are also applied for learning the conversion rules between noisy and canonical forms. The neural encoder-decoder architecture is basically a sequence-to-sequence model. It represents the relation between source and target sequences as a state vector by matching the features of the sequences.

In deep learning, encoder-decoder model is performed by two neural networks: encoder neural network and decoder neural network. Encoder network maps the features of the source sequence to a fixed-length state vector that contains the weights of the relations between the source and the target, and decoder network generates the target sequence from the weight vector.

The architecture of the Encoder-Decoder model is given in Figure 2.2.

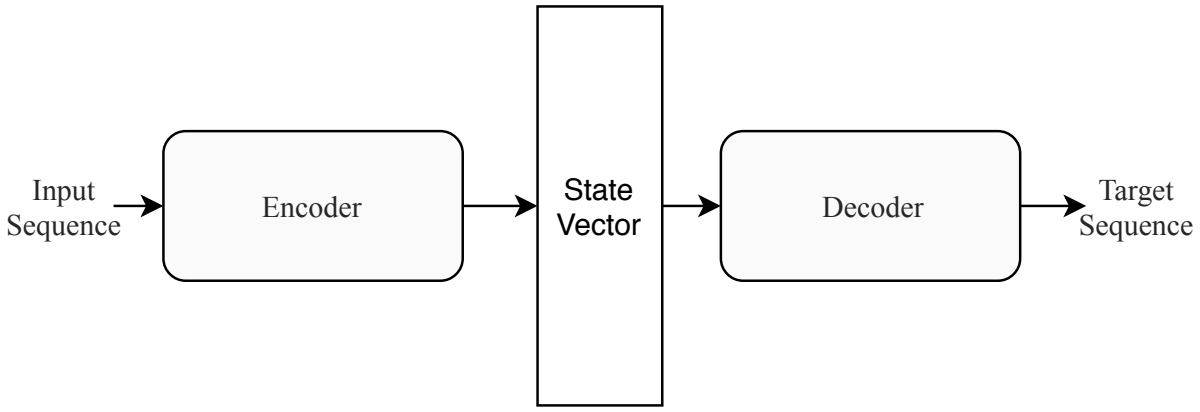


Figure 2.2. Encoder Decoder Architecture

2.1.6. Recurrent Neural Network

Recurrent neural networks (RNN) are used for processing sequential information differently from the conventional artificial neural networks (ANN). RNNs are recurrent neural networks as they perform a task by processing each element of the sequence consecutively. All inputs (and also outputs) are processed independently in conventional ANNs. RNNs, on the other hand, perform the task on a word in a sentence by considering the previous words. Therefore, RNNs gain the ability of memorizing information from the previous steps. This gives RNNs the ability of learning the transformations within the context.

The basic architecture of a Recurrent Neural Network is given in Figure 2.3.

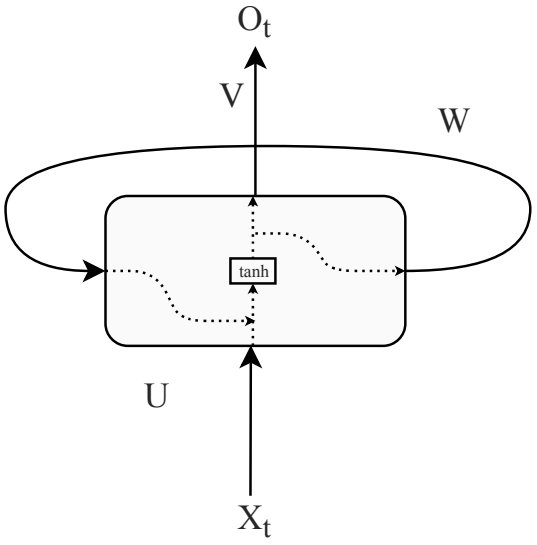


Figure 2.3. Recurrent Neural Network Architecture

An RNN takes a sequence of inputs $X_1, X_2, X_3, \ldots, X_n$ where X_t is the input in time step t. h_t is the hidden state in time step t and it acts as the memory in the neural network. Hidden state carries previous information and is calculated by the previous hidden state with the input in time step t. The calculation of a hidden state is given as follows:

$$h_t = f(UX_t + Wh_{t-1}) \tag{5}$$

The non-linear function f is used for calculating hidden state in time step t which is called activation function. $tanh$, $sigmoid$ and $ReLu$ are commonly used activation functions to provide non-linearity in RNNs [18]. O_t is the output in time step t. U, V, W are the parameters that are used by RNN for all time steps. The calculation to predict a word with its previous states is given as follows:

$$O_t = softmax(Vh_t) \tag{6}$$

2.1.7. Long-Short Term Memory (LSTM)

Long-Short Term Memory (LSTM) network is a kind of specialized RNN. Theoretically, RNN networks are capable of handling long-term dependencies, however in practice, several studies (e.g. Hochreiter[19] and Bengio, et al. [20]) show that they are inadequate for calculating more than a few states by considering the previous term dependencies.

LSTMs, introduced by Hochreiter and Schmidhuber [21], have been developed for handling this problem and become a frequently used networks in many studies. They are formed as chain sequences to be used recurrently like RNNs. Therefore, they can take a sequence as input and process it by considering the previous ones. There are memory cells in LSTMs as a solution for the long-term dependency problem. Those cells are used in all previous states through the sequence.

LSTMs have four different specialized neural network layers (input gate, forget gate, update gate, output gate) while RNNs have only one simple activation layer such as $tanh$.

- Input gate (g_i) is the neural network layer with a non-linear function that determines stochastically which value to be updated at that time step.

- Forget gate (g_f) determines which value to be transfered to the next state.
- Update gate (g_u) updates the cell state by considering the information in input and forget gates.
- Output gate (g_o) determines the output which is a hidden state for the next step.

The basic architecture of an LSTM and gates are given in Figure 2.4.

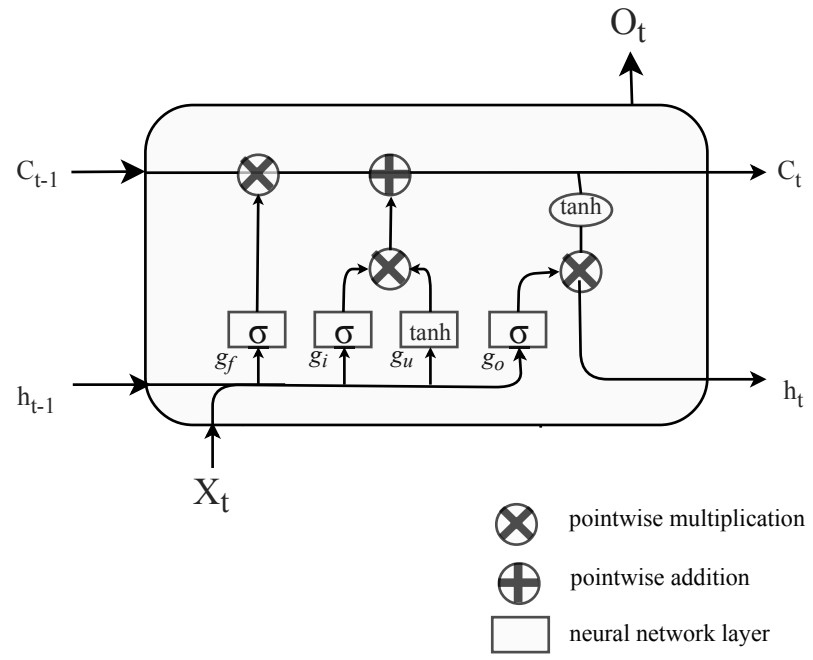


Figure 2.4. Long-Short Term Memory Neural Network Architecture

2.2. Semantic Form Normalization

Semantic Form Normalization is a normalization method that considers the semantic features of a given noisy text. The main goal is to analyze the noisy text and by determining the semantic similarities to extract the canonical candidates.

2.2.1. Distributed Representation of Words

The distributional hypothesis in linguistics is that words that occur in similar contexts tend to have similar meanings (Harris, 1954) [22]. In NLP field, distributed models are used

to represent semantic similarities in vector space. Therefore, Vector Space Model (VSM) is one of the most important methods for this aspect. In deep learning studies, distributed neural network based representation of words are generated by training the networks with the co-occurrence probabilities of the words with their contextually similar ones.

VSM was developed for the SMART information retrieval system Salton [23]. It is the vectorial representation of texts that is generally used in NLP tasks that require semantic information. Each dimension of the vector represents a term that can be a letter, a word or any subsequence of the input text. The value of a dimension is non-zero if a term occurs in the input text. Representing a text in vectorial form enables to apply algebraic calculations.

There are some kinds of VSM such as Latent Semantic Analysis (LSA) and Singular Value Decomposition (SVD). LSA is an indexing technique between documents and the terms that they contain. SVD is a matrix factorization method that represents a matrix as a product of matrices.

Distributed representation of the input text d_j is defined such as:

$$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j}) \tag{7}$$

2.2.2. CBOW (Continuous Bag-of-Words)

CBOW (Continuous Bag-of-Words) is a distributed representation architecture which is one of the approaches of word2Vec introduced by Mikolov [15]. In CBOW architecture, the bag of surrounding words are used to predict the center word where the distributed representations of words occurring in similar contexts tend to be also similar at the end of the training. As a result, a vector space is created where each word is represented by a word embedding.

CBOW architecture is given in Figure 2.5. for $w(t)$ which denotes the center word to be guessed through the contextual words: $w(t-2)$, $w(t-1)$, $w(t+1)$, and $w(t+2)$ for a window size of 2.

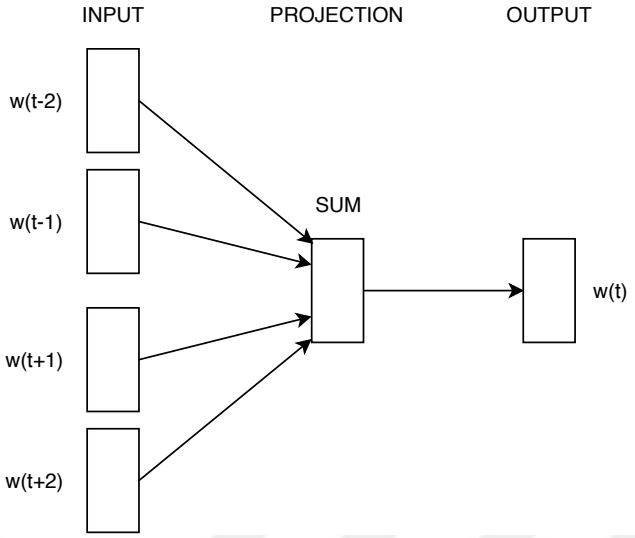


Figure 2.5. Continuous Bag-of-Words (CBOW) Architecture

2.2.3. Cosine Similarity

Cosine Similarity is a metric that is often used to compute the relational similarity of two vectors (i.e. distributed representations), u and v, by cosines of the angle between them. The similarity or comparison between two texts is formulated by cosine function. If the two vectors are same, then the value of cosine similarity equals to 1. However, the cosine value is 0 when the vectors are completely unrelated. The cosine similarity is computed as follows:

$$cos(u, v) = \frac{\sum_{i=1}^{D} u_i \times v_i}{\sqrt{\sum_{i=1}^{D} {u_i}^2 + \sum_{i=1}^{D} {v_i}^2}} \tag{8}$$

3. RELATED WORK

In this chapter, several NLP studies related to Text Normalization task are reviewed. Within this scope, different features have been utilized in text normalization by using different methods. It has been found out that many methods have used the similarity between orthographic features of words in order to capture their canonical forms. For this purpose, several distance metrics have been used such as longest common subsequence and edit distance. Additionally, contextual features have also been utilized in order to capture semantic relations between noisy and canonical forms of words. The literature that contains some important studies related to Text Normalization is briefly explained in two sub-sections as supervised and unsupervised models below.

3.1. Literature Review on Supervised Text Normalization

In supervised approaches, the model is being trained with a labeled dataset that contains inputs with their corresponding targets. Thus, the model learns how to relate an input and its target after sufficient training. In text normalization task, supervised models are trained to learn the way of transforming noisy text into canonical forms over annotated datasets. We review supervised text normalization studies in this section.

Torunoğlu and Eryiğit [1] propose a cascaded text normalization approach for social media text in Turkish. The proposed method is supervised and language-dependent. The main idea is to normalize the given text by applying manually defined rules as a pipeline process until obtaining the canonical form. On the other hand, as the change of language use in time in social media, manually defined rules will be incapable of normalizing. Because conventional techniques cannot follow the further new error patterns, therefore the success rate declines eventually. The system architecture is shown in Figure 3.1.

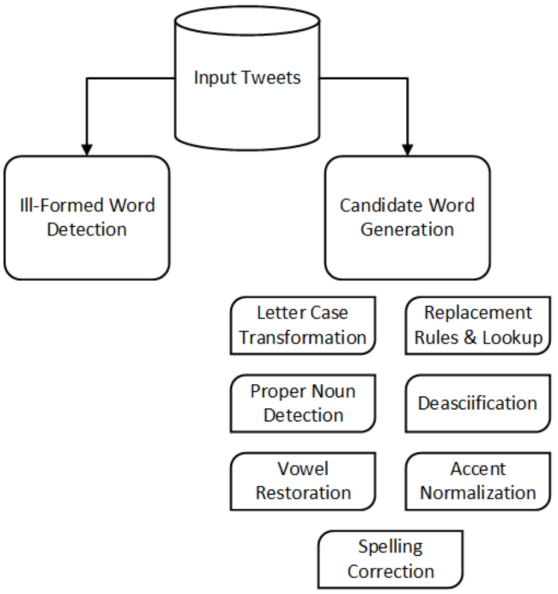


Figure 3.1. Cascaded Text Normalization architecture. [1]

Sutskever et al. [16] propose a sequence-to-sequence learning method by using deep recurrent neural networks for machine translation problem (see in Figure 3.2.). They use long short-term memory neural networks (LSTMs) in order to map each source sentence to its target sentence in another language. They achieve successful results especially on long sentences.

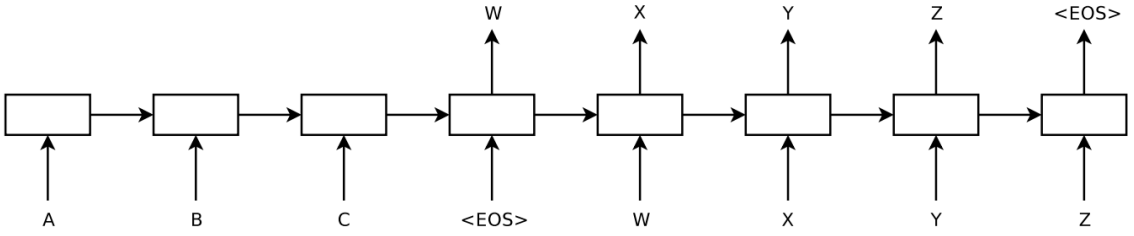


Figure 3.2. Sequence-to-sequence machine translation architecture.

Ikeda, et al. [2] introduce a character-based neural encoder-decoder model for Japanese text normalization. Recurrent Neural Network (RNN) based Gated Recurrent Unit (GRU) neural networks are used for the encoder-decoder architecture (see in Figure 3.3.). In order to overcome the need of a large corpus required by neural encoder-decoder model, they artificially create a large scale data by augmenting their dataset by applying various edit

operations on the canonical forms to obtain their noisy forms. Therefore, a large amount of dataset that involves noisy and canonical word pairs is built for training purposes.

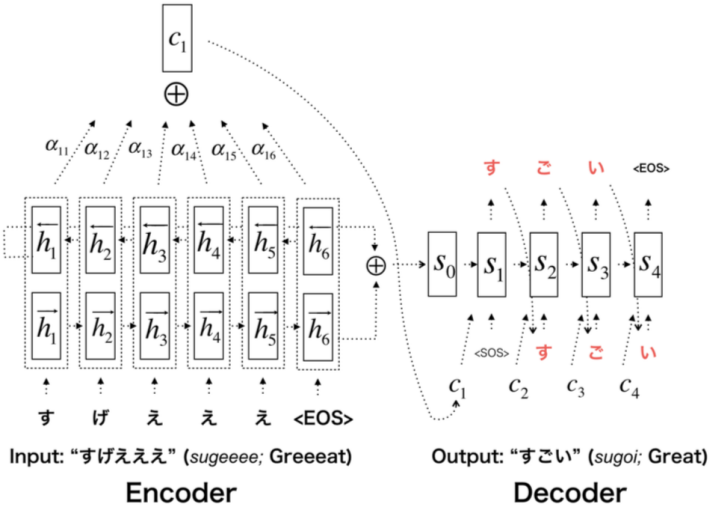


Figure 3.3. The encoder-decoder architecture for Japanese text normalization. [2]

Li and Liu [24] proposed a non-standard word (NSW) detection method. This method has two approaches using NSW detection results for named entity recognition in social media texts. One of the approaches embraces a pipeline strategy while the other one deals with a joint decoding.

Aw et al. [3] proposed a phrase-based statistical normalization approach on Short Message Service (SMS) normalization for English texts. They assumed SMS as a language variation of English and defined the SMS normalization as machine translation task of SMS language to the English language. The approach offers a solution for both words and phrases. They focus on three major cases while defining rules: replacement of noisy words, removal of slang words and insertion of auxiliary verb and subject pronoun. The rules based on grammar and orthographic variations are manually defined and used in the study. An example of replacement rules in SMS normalization is shown in Figure 3.4..

Substitution	Deletion	Insertion
u -> you	*m*	*are*
2 → to	*lah*	*am*
n → and	*t*	*is*
r → are	*ah*	*you*
ur →your	*leh*	*to*
dun → don't	*1*	*do*
man → manches-ter	*huh*	*a*
no → number	*one*	*in*
intro → introduce	*lor*	*yourself*
wat → what	*ahh*	*will*

Figure 3.4. Most common replacement rules used for SMS Normalization [3]

Clark and Araki [4] proposed a supervised rule-based approach that contains a manually compiled and verified dictionary that can store phrases for normalizing social media texts in English. They named the system as Casual English Conversion System (CSCE). The dictionary contains casual English phrases, their normalized forms and replacement types such as abbreviation, misspelling, punctuation etc. They defined phrase matching rules over the dictionary and used the rules automatically in normalization. The system architecture of proposed method is shown in Figure 3.5.

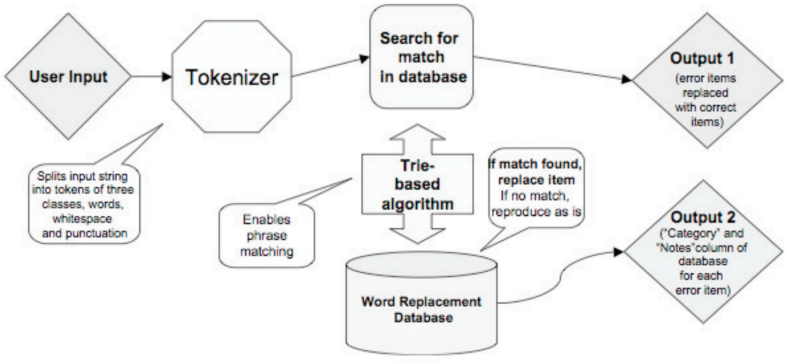


Figure 3.5. System architecture of Casual English Conversion System (CSCE) [4]

3.2. Literature Review on Unsupervised Text Normalization

In unsupervised approaches, the model is being trained with a unlabeled dataset. An unlabeled dataset does not contain any association or explanation of source and target texts, it

only contains raw data. Thus, different methods are used to learn the association between input and its target such as clustering, rule definitions, etc. In text normalization task, unsupervised models are trained to learn the way of transforming noisy text into canonical forms on an unannotated dataset by itself. We review unsupervised text normalization studies in this section.

Hasan and Menezes [5] utilize both contextual and lexical features of the text. They use Random Walks on a bipartite graph that is built based on the contextual similarity where a set of nodes represents the contexts and another set corresponds to noisy and canonical words (see in Figure 3.6.). A normalization lexicon is generated through random walks on the bipartite graph. The most suitable candidate words are chosen according to the longest common subsequence and edit distance.

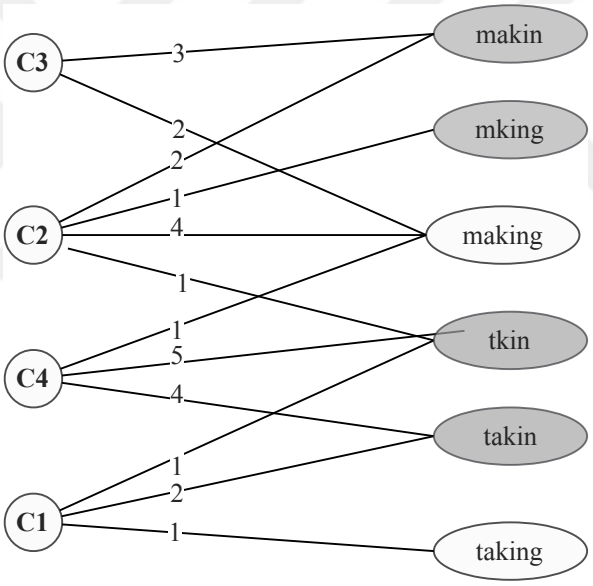


Figure 3.6. Bipartite Graph Representation, edge weight is the co-occurrence count of a word and its context. [5]

Sönmez and Özgür [6] introduce another graph-based method for the normalization of the social media text. The proposed approach uses grammatical features in addition to contextual and lexical features. The contextual and grammatical features are encoded in a graph (see Figure 3.7.), where the relative positions of words and their part-of-speech (PoS) tags are encoded.

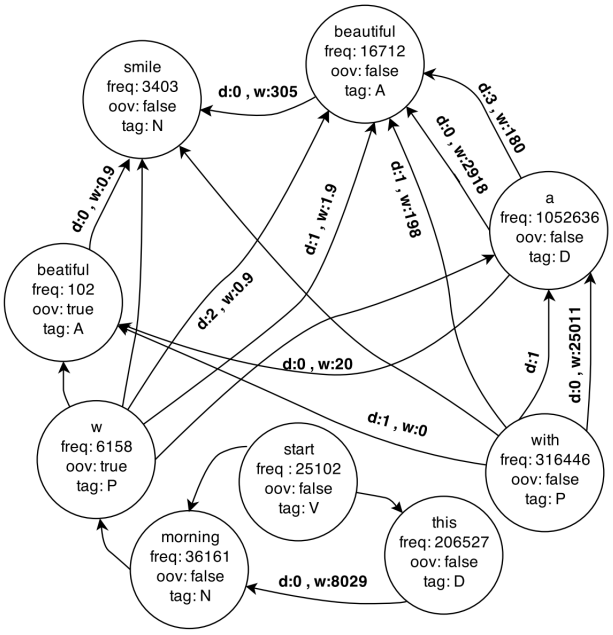
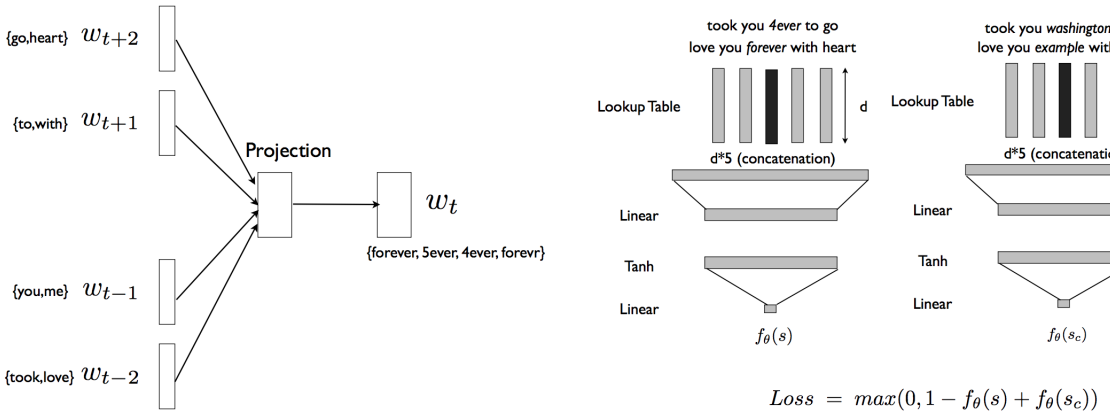


Figure 3.7. Word Association Graph for a sample sentence. [6]

Sridhar [7] introduces an unsupervised text normalization algorithm that makes use of distributional features of words and phrases. Differently from the other related studies, contextual features are learned via neural word embeddings by using continuous bag-of-words model of word2Vec [15] and the neural network architecture by Collobert et al. [25] (see in Figure 3.8.). A lexicon that consists of noisy and canonical word pairs is constructed by using of the distance between neural word embeddings and by filtering out some of the candidates to find the best canonical candidate for each word. One of the methods that we apply for Turkish is based on this study and more details about this method is given in Chapter 4.



(a) Continuous Bag-of-Words Architecture (b) Neural Network Architecture

Figure 3.8. Distributed representation architectures for text normalization of two different approaches. [7]

Yang and Eisenstein [26] propose a log-linear model that scores source and target strings in an unsupervised framework. A language model is combined with the log-linear model to compute the weight gradients only for the observed n-grams while doing gradient-based updates.

3.3. Discussion

In this chapter, we reviewed the previous work on text normalization in two sub-sections as supervised and unsupervised approaches. Supervised models deal with text normalization task by using labeled data, while unsupervised models perform normalization with unlabeled data.

There are numerous text normalization studies on English in the literature. Most of the work done on social media data is also in English whereas there is not much work on English social media normalization. There are numerous text normalization studies on English in the literature. Most of the studies that are performed on social media data are also in English. Even there is small amount of text normalization studies are performed on agglutinative languages such as Turkish, however they are mostly rule-based.

As the widespread use of neural approaches, text normalization studies, like many studies in the NLP field, also use the advantages of Artificial Neural Networks (ANN) to learn and

model non-linear and complex relationships. By considering the text normalization studies in the literature, neural approaches become as common as rule-based approaches and offer solutions to with different advantages, such that the enable normalization without requiring any manually defined rules.

In supervised normalization approaches, a great amount of labeled data is required for a successful learning process. Finding labeled data for text normalization over social media data require too much labor. For languages other than English, this process becomes even more difficult. Since the error patterns in the social media language change over time, the labeled data must also be updated so that the system can maintain the same success. Unsupervised normalization approaches do not encounter such problems in these studies because they work with unlabeled data.

Conventional social media normalization approaches are insufficient because they use mostly rule-based techniques that are generally manually defined. The change of language use over time in social media results new error patterns. Such rule-based methods require too much labor for updating the rules in time. As conventional techniques cannot follow those patterns, the success rate of the techniques declines in time.

The review of these studies will constitute a reference point for performing a neural text normalization method presented in the following chapter.

4. METHODOLOGY AND IMPLEMENTATION

In this chapter, the two approaches that are proposed in this thesis for social media text normalization are explained in detail. One of the approaches is Contextual Normalization which uses distributed representations of words and the other one is Sequence-to-Sequence Normalization which uses encoder-decoder neural networks. Noisy social media texts (i.e. Twitter tweets) are used as input and normalized forms of the noisy texts are generated as output for each normalization method. First method uses the pre-trained neural word embeddings to make use of distributional features of noisy words to learn their canonical forms through context by including semantic features. The latter is based on an encoder-decoder model that uses bidirectional LSTM (bi-LSTM) architecture [21] that learns normalization rules from a large set of noisy and canonical word pairs.

In order to perform text normalization operations, training is performed in both approaches. For contextual normalization, word2vec [15] model is trained on Twitter dataset in order to learn word embeddings. For sequence-to-sequence normalization, encoder and decoder neural networks are trained to learn how to normalize error patterns of noisy words. Once both models are trained, each system can perform normalization process as they have the information to generate or extract canonical forms for further noisy texts.

4.1. Contextual Normalization Approach

Contextual normalization approach is a method which considers semantic relations of texts. As being trained by unannotated dataset, it is an unsupervised method. The training operation is required for creating a lexicon to extract candidate canonical words for each word in the noisy text. By using a dynamic programming algorithm, the most likely sequence of candidates are selected as the normalized output text. The main steps of this approach are shown in Figure 4.1.

4.1.1. Distributed Representation of Noisy Text

Learning distributed representation of social media text is the first step of this approach. The main idea is to represent the words as a feature vector that bears lexical, semantic and

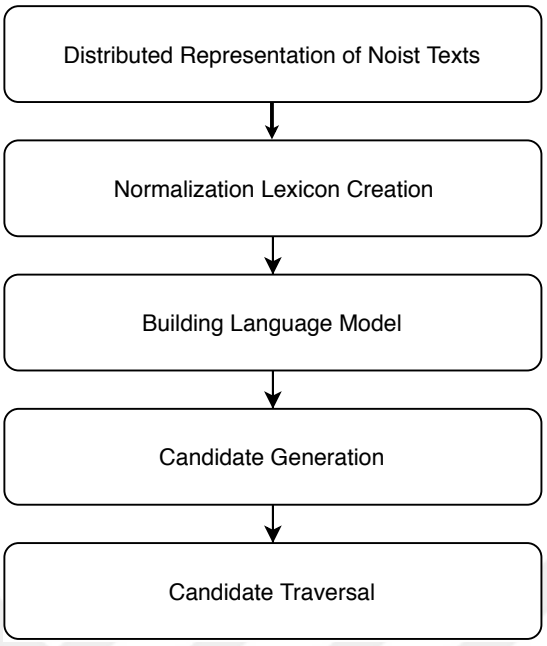


Figure 4.1. Methodology of Contextual Normalization Approach.

syntactic features in the same representation (i.e. word embedding). Word2vec [15] is the model that is used to represent texts and their semantic relations.

CBOW (Continuous Bag-of-Words) architecture of word2vec is used to learn the word representations (see Section 2.2.2.). In CBOW architecture, the bag of surrounding words are used to predict the center word where the word representations of words occurring in similar contexts tend to be also similar at the end of training. As a result, a vector space is created for the social media text where each word is represented by a word embedding.

A word embedding model has been created to retrieve the nearest noisy words for any word. When querying this model with a canonical word, a list of noisy words having the highest semantic similarity scores (Cosine similarity) is retrieved. An example of k-nearest possible noisy words for a canonical Turkish word is given in Table 4.1.

4.1.2. Normalization Lexicon Creation

Normalization Lexicon is a map that contains canonical-noisy word pairs with their lexical similarities (SimCost). For this aspect, we use a corpus of news archive [27] (NewsCor) that consists of manually collected 184 million words to generate the lexicon. Once we select 2 million unique canonical words by performing several calculations on NewsCor, we retrieve

Table 4.1. An example canonical word and its 25-nearest possible noisy words

Canonical Word (c)	Noisy Word(n)	CosineSimilarity(c,n)
şarkı (*song*)	şarkıyı	0.814
	şarkılar	0.743
	şarkıları	0.703
	sarkı	0.689
	şarkılarını	0.664
	şarkıymış	0.650
	şarkıdır	0.647
	şarkıı	0.642
	şarkının	0.636
	parçayı	0.631
	şarkıda	0.629
	sarki	0.621
	sarkılar	0.615
	yazganın	0.606
	şarkımı	0.603
	müziklerin	0.589
	türküyü	0.588
	şarkısı	0.585
	şarkıdan	0.582
	şarkılardan	0.580
	şarkısını	0.580
	müzikler	0.570
	şarkyı	0.575
	şarkıya	0.572
	karaokede	0.572

the nearest n[1] neighbours of each canonical word by using the pre-trained word embeddings. To this end, we use cosine similarity (see Equation 8) between two word embeddings u and v in order to find the contextual similarity between the canonical and every noisy word vector in the vector space.

Eventually, 43 million unique canonical-noisy word pairs are gathered that will be used as the lexicon. Those pairs are swapped as noisy-canonical pairs. Finally, SimCost(w_1, w_2) (see Eqaution 4) between two words w_1 and w_2 is calculated, the cost value is stored in

[1]We use two different threshold values (n=25, n=100) in order to choose the nearest neighbours. The results for different values of n are reviewed in Chapter 5..

the normalization lexicon with its correspondent pair. Algorithm 1 describes the process of generating the normalization lexicon.

Algorithm 1: The Algorithm for Generating the Normalization Lexicon

Input: Unique canonical word list W

Input: Word embeddings of noisy word vocabulary V

Input: Number of nearest neighbours K

Output: Normalization lexicon L

for $w \in$ W **do**
  for $v \in$ V **do**
    if ($v \notin$ W) **then**
      compute Cosine Similarity(w,v)
      store top K neighbours in map $M(w, v)$
    end if
  end for
end for
for $w \in$ W **do**
  for $m \in$ M **do**
    compute SimCost(w,m)
    push $m \rightarrow (w, SimCost(w, m))$ into lexicon L
  end for
end for

Some example noisy and canonical word pairs with their similarity costs are given in Table 4.2.

Table 4.2. Example Noisy-Canonical Word Pairs in the Normalization Lexicon

Noisy Form (n)	**Canonical Form (c)**	**SimCost(n,c)**[2]
batil	batıl	undefined
kaciramaz	kaçıramaz	0.34
ogret	öğret	0.41
patlıycam	patlıyorum	1.69
çıkarıl	çıkarıldığı	1.97
sevinmicem	sevinemedim	2.48
kutliyim	kutlayayım	2.48
sevinmicem	sevinemedim	2.48
ölmedği	ağlamadığı	2.48
sevinmicem	sevinemedim	2.77
karsilikli	karşılıklı	2.89
oturmayi	çöpü	0

Because of the nature of agglutinative languages, two words can have a larger edit distance even though they share the same stem. To this end, elimination of the vowels in words is performed in order to create *consonant skeletons* of those two words to compute the edit distance between them. The consonant skeleton model was used for detecting and normalization of SMS shortcuts, abbreviations or erroneous texts in different studies such as Sridhar [7] and Cuevas [28].

In this study, consonant skeleton structure is used to select a more significant letter sequence that express noisy words. Differently from Sridhar [7]; as Turkish is agglutinative, two additional consonant skeleton structures representing the relation between the stems of the words are used in this thesis in order to calculate the edit distance for lexical similarity cost. Since stem is located in the beginning of a word in agglutinative languages, the additional consonant skeleton structures are developed with the goal of focusing the stem of

[2]In SimCost calculation, the lowest similarity value of 13.8 and the highest similarity value is -13.8 are set if the word skeletons are completely different($SimCost(n, c) = 0$) or identical($SimCost(n, c) = undefined$). When consonant skeletons of two words are identical $ED(n, c)$ is calculated as 0 and $SimCost(n, c)$ is calculated as $undefined$

the words considering the structure of agglutinative languages. Three different consonant skeleton structures that are proposed for this study are described in following section.

4.1.2.1. Word-wide Consonant Skeleton

In the first consonant skeleton structure used in this study, we eliminate all the vowels of two words to find the edit distance for lexical similarity calculation (see Figure 4.2.).

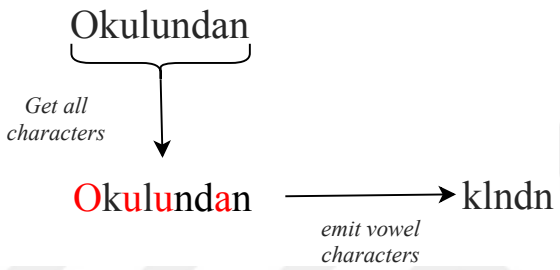


Figure 4.2. Word-wide Consonant Skeleton conversion

Word-wide Consonant Skeleton conversion examples are given in Table 4.3.

Table 4.3. Word-wide Consonant Skeleton Structure examples

Word	Word-wide
yurt (*country*)	yrt
dışında (*out of*)	dşnd
yaşayan (*the one who lives*)	yşyn
Türk (*Turkish*)	Trk
vatandaşları (*citizens*)	vtndşlr

4.1.2.2. First 5 Character-based Consonant Skeleton

For the second consonant skeleton method of this study, we eliminate only vowels in first five letters of two words to find edit distance for lexical similarity calculation (see Figure 4.3.).

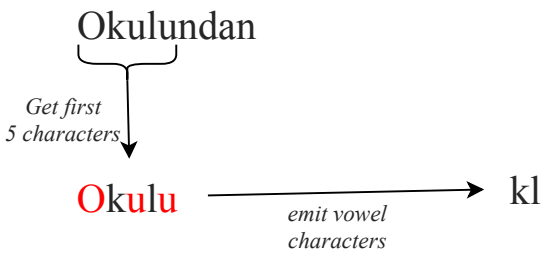


Figure 4.3. First 5 Character-based Consonant Skeleton conversion

First 5 Character-based Consonant Skeleton conversion examples are given in Table 4.4.

Table 4.4. First 5 Character-based Consonant Skeleton Structure examples

Word	First 5 Character
gastronomi (*gastronomy*)	gstr
yemek (*food*)	ymk
kültürü (*culture of*)	klt
hakkındaki (*the thing about*)	hkk
kitaplar (*books*)	ktp

4.1.2.3. First 3 Consonants-based Skeleton

For the last consonant skeleton structure, we eliminate only vowels in first three consonant letters of two words to find the edit distance for lexical similarity calculation (see Figure 4.4.).

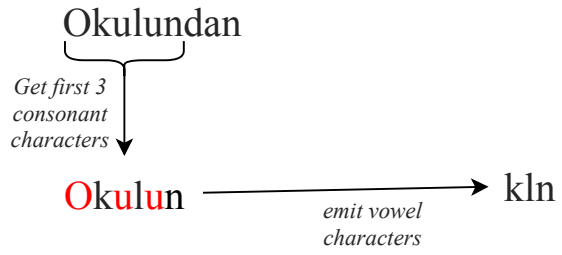


Figure 4.4. First 3 Consonants-based Skeleton conversion

First 3 Consonants-based Skeleton conversion examples are given in Table 4.5.

Table 4.5. First 3 Consonants-based Skeleton Structure examples

Word	**First 3 Consonants**
sınavın (*of exam*)	snv
akabinde (*immediately after*)	kbn
öğrenciler (*students*)	ğrn
tartışma (*arguement*)	trt
başlattılar (*they started*)	bşl

4.1.3. Building the Language Model

Language model is a probability distribution representation of words.This model is used to determine arbitrary occurrences of words in a sequence.

In this study, we used a bigram language model for the transition probabilities between consecutive words in a sentence in order to perform a sentence level text normalization. We train the bigram language model on the same news corpus dataset [27] and the resulting bigram probability values are stored in the transition lexicon to be used in the last step of the Viterbi algorithm (see Section 4.1.4.). The bigram language model over a sequence $P(w_i, \ldots, w_n)$ is calculated as follows:

$$P(w_i, \ldots, w_n) = \prod_{k=1}^{n} P(w_k|w_{k-1}) \tag{9}$$

Bigram probabilities of a sample word pair list are given in Table 4.6.

Table 4.6. Bigram probabilities of arbitrary word pairs

w_{i-1}	w_i	$P(w_i\|w_{i-1})$
rapor (*report*)	hazırlandığını *prepare*)	0.15126
serbest (*free*)	piyasada *market*)	0.10302
peşinde (*behind*)	koşup *run*)	0.07521
çalıştığı (*work*)	kurumları *institution*)	0.00391
yeni (*new*)	cihazın *device*)	0.01009
gerçekçi (*realist*)	yeni *new*)	0.00543

4.1.4. Candidate Extraction

Sentence level text normalization is performed word by word by using Viterbi algorithm in order to choose the normalization having the minimum cost. As the first step, we extract canonical candidates for each word in the input sentence (see Figure 4.5.) by using the normalization lexicon created before (see Section 4.1.2.). The candidate canonical forms for the noisy word *düşüncm* (*My opinion*) are also given in Table 4.7. according to their lexical similarity costs.

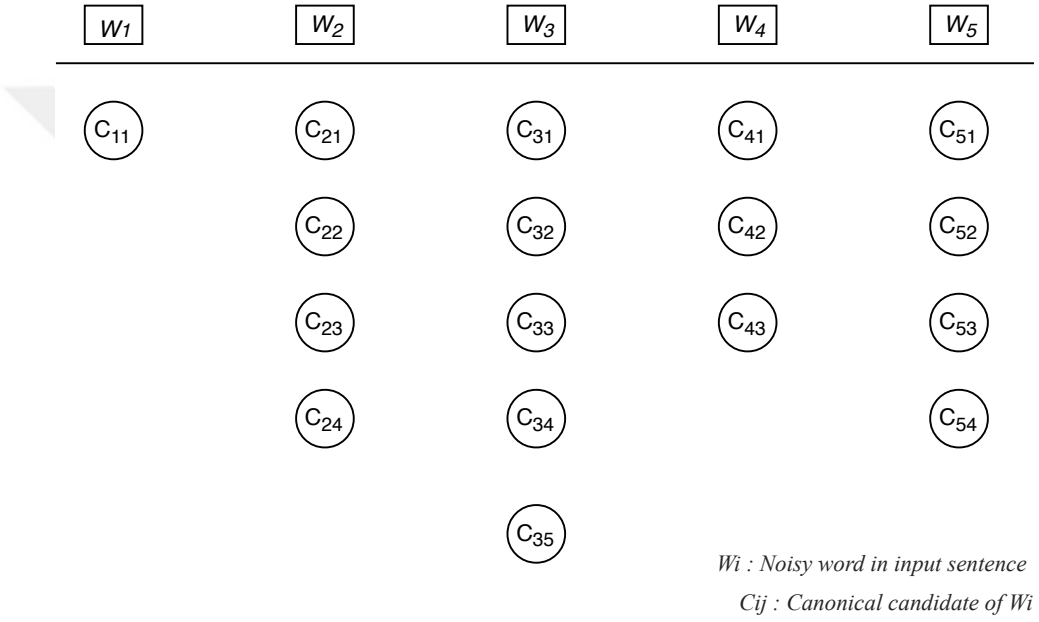


Figure 4.5. Extracted Candidates for an input sentence

Each canonical candidate word and its SimCost value (which will be used for the emission cost) is retrieved from the normalization lexicon for each noisy word.

Table 4.7. An example word and its candidate canonical forms

Word (n)	Candidates (c)	SimCost(n,c)
düşüncm	düşüncem (*my opinion*)	-13.8
	düşüncen (*your opinion*)	0.18
	düşüncemiz (*our opinion*)	0.98
	düşünceniz (*your opinion*)	1.57
	şansım (*my chance*)	2.19
	düşüncelerimi (*my opinions*)	2.39
	dışımda (*except me*)	2.49

Algorithm 2 describes the process of candidate extraction and initialization for an input sentence.

Algorithm 2: The Algorithm for Candidate Initialization of a Sentence

Input: Sentence S ($w_1 ... w_n$)

Input: Normalization lexicon L

1: **for** $w_i \in$ S **do**
2: Candidate List C $\leftarrow$ L(w_i)
3: **for** $c \in$ C **do**
4: set text $candidate.text \leftarrow c.text$
5: set emission probability $candidate.p_e \leftarrow c.SimCost$)
6: add $candidate \rightarrow w.candidateList$
7: **end for**
8: **end for**

Finally, all candidate canonical words for each word of the noisy input sentence has been extracted for the last step; Candidate traversal by Viterbi Algorithm.

4.1.5. Candidate Traversal

Candidate traversal is the process of choosing the most likely candidates of each noisy word in order to create the normalized output text. For this aspect, Viterbi algorithm, dynamic programming algorithm, has been applied in order to normalize a given input sentence by

performing the calculations of emission and transition probabilities of input words with their related candidates.

Transition probability between the candidate canonical forms of two consecutive noisy words are retrieved from the bigram language model (see Section 4.1.3.). Negative logarithm of the bigram probabilities are calculated for obtaining transition probabilities in order to find the normalization path with the minimum cost.

Emission probability is the lexical similarity value of the candidate with its correspondent noisy word. All emission probabilities of each pair have been calculated during lexicon creation (see Section 4.1.2.). Negative logarithm of the lexical similarity costs are calculated for obtaining emission probabilities.

Viterbi cost is the summation that is calculated cumulatively for each word in a sentence with the candidates of the previous word. The minimum cost is stored on that candidate in each iteration. The final Viterbi cost is the cost of the candidate canonical sentence. A backtrace process is performed to obtain the Viterbi path with the minimum cost. The candidate path is accepted as the most likely normalized sentence for the noisy input sentence(see Figure 4.6.).

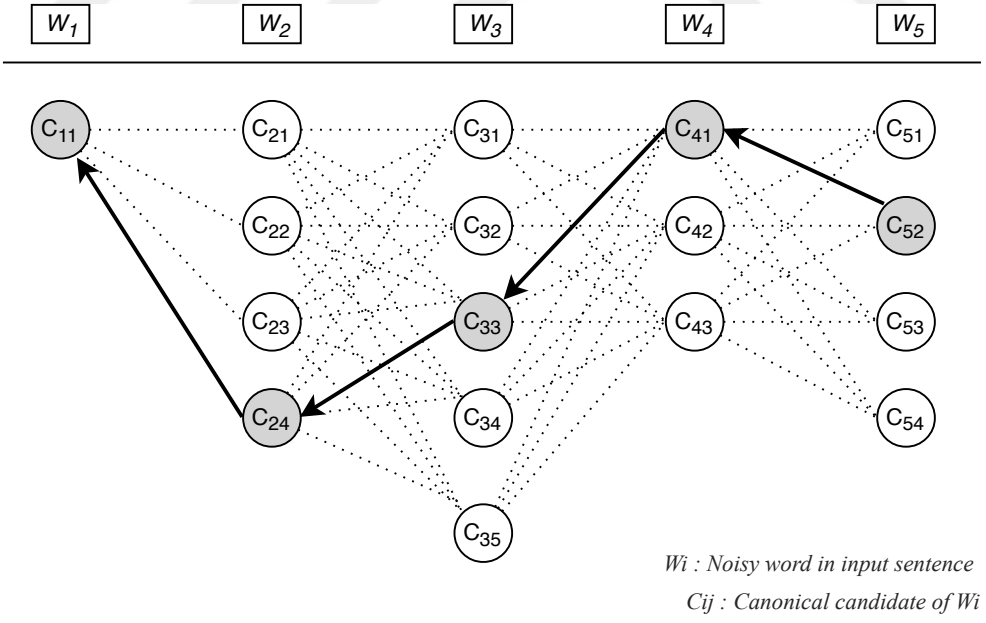


Figure 4.6. Candidate Traversal using Viterbi Path

Algorithm 3 describes the candidate traversal process by using the Viterbi algorithm.

Algorithm 3: The Algorithm for Candidate Traversal by the Viterbi Algorithm

Input: Sentence S ($w_1 ... w_n$)

Input: Normalization lexicon L

Input: Transition lexicon T

Output: Normalized Sentence N # *Candidate Traversal*

for $w_i \in$ S **do**
  if ($i == 1$) **then**
    for $c_i \in w_i$.candidates **do**
      $c_i.viterbiCost \leftarrow 1$
    end for
  end if
  for $c_{cur} \in w_i$.*candidates* **do**
    for $c_{prev} \in w_{i-1}$.*candidates* **do**
      $p_{trans} \leftarrow T(c_{cur}, c_{prev})$
      $p_{trans} \leftarrow -1 * log(p_{trans})$
      $tmp \leftarrow p_{trans} + c_{prev}$.*viterbi*
      if $tmp < c_{cur}$*viterbi* **then**
        c_{cur}.*viterbi* $\leftarrow tmp$
        c_{cur}.*backPointer* $\leftarrow c_{prev}$
      end if
    end for
    $p_{emmit} \leftarrow L(c_{cur}, w_i)$.*SimCost*
    c_{cur}.*viterbi* $\leftarrow c_{cur}$.*viterbi* $+ p_{emmit}$
  end for
end for
$minViterbi \leftarrow 999, c_{min} \leftarrow nil$
for $c_j \in w_n$.*candidates* **do**
  if c_j.*viterbi* $< minViterbi$ **then**
    $minViterbi \leftarrow c_j$.*viterbi*
    $c_{min} \leftarrow c_j$
  end if
end for
$viterbiBacktrace(c_{min})$

Algorithm 4 describes the Viterbi backtrace.

Algorithm 4: The Algorithm of Viterbi Backtrace

Output: Normalized Sentence N
Output: Minimum Viterbi Candidate c_{min}

```
Function viterbiBacktrace
t ← n                                          # Unpack Viterbi
N ← newarray[n + 1]
while t > 0 do
    N[t] ← c_min
    c_min ← c_min.backPointer
    t ← t − 1
end while
```

4.1.6. Limitations

Contextual normalization approach has a procedure of normalizing a sentence word by word considering semantic and lexical similarities. In the normalization lexicon creation phase, each noisy word token is matched with a single canonical word token. Therefore phrases cannot be considered a single token in this approach. Phrases and phrase abbreviations are often normalized as counting in a single word such as *iyi akşamlar*, *kib*, *aeo*. This is the main limitation of this approach and it slightly decreases the success rate of the approach.

4.2. Sequence-to-Sequence Normalization Approach

Sequence-to-Sequence normalization approach is a method that is able to consider morphological and orthographic relations between the texts. Encoder-decoder model is used in this approach by using LSTM neural networks. In this approach, neural encoder-decoder architecture is trained on an annotated dataset, which makes this approach a supervised learning, in order to learn how to normalize the noisy words into canonical forms (see in Figure 4.7.).

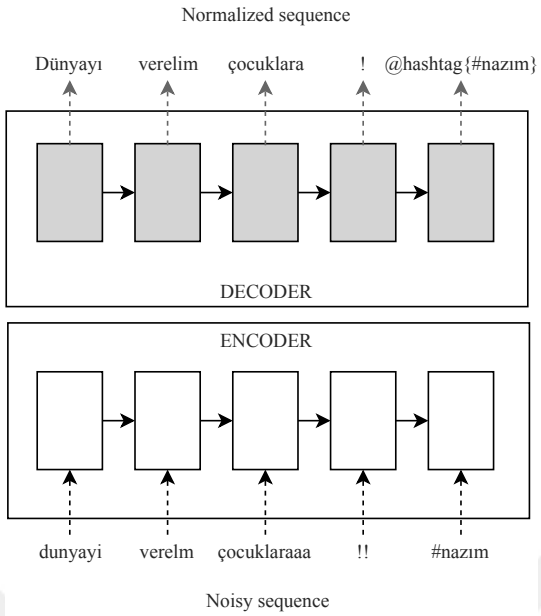


Figure 4.7. Sequence-to-sequence Normalization Architecture by using Encoder - Decoder Model (EDM)

The main steps of this approach are shown in Figure 4.8.

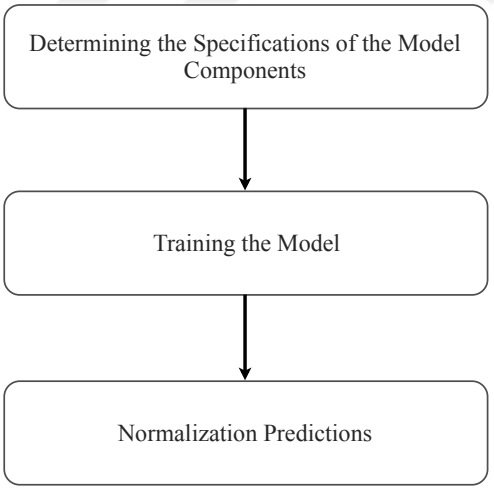


Figure 4.8. Methodology of Sequence-to-sequence Normalization Approach

4.2.1. Determining the Specifications of the Model Components

According to the recent studies, using encoder-decoder model has been shown effective especially in the field of machine translation [16], [29], [30], [31]. We adopt long-short term

memory (LSTM) networks for the encoder-decoder model in order to use deep learning techniques for a higher accuracy in the normalization with sequence-to-sequence normalization approach.

The configuration that produces the most accurate results for the model parameters epoch, batch size, unit, activation function, optimizer function and loss function parameters is obtained by performing different experiments. Different parameter values used in the experiments are shown below.

4.2.1.1. Epoch

Epoch is the parameter that represents number of iterations performed on training data. The experiments are repeated for the values of 10, 15, 20, 25 and 30 in training and evaluation of the model. The parameter that achieves the most accurate result is selected as epoch value of encoder-decoder model.

4.2.1.2. Batch Size

Batch size is the parameter that represents number of samples per gradient update. The experiments are repeated for the values of 20, 30, and 40 in training and evaluation of the model. The parameter that achieves the most accurate result is selected as batch size value of encoder-decoder model.

4.2.1.3. Unit

Unit represents dimension of the memory cells in encoder and decoder LSTMs. The experiments are repeated for the values of 64, 128 and 256 in training and evaluation of the model. The parameter that achieves the most accurate result is selected as unit value of encoder-decoder model.

4.2.1.4. Activation Function

Activation function is used for calculating the output of hidden state. The experiments are repeated for the activation functions of *Sigmoid*, *Tanh*, *Softmax*, *ReLU* in training and evaluation of the model.

The *Sigmoid* is basically a logistic (S-shaped) function. The range of function outputs exist in 0 to 1 for all input values. General formula of *sigmoid* function is given as follows:

$$\sigma(x) = \frac{1}{1 + e^x} \tag{10}$$

Sigmoid activation function curve is given in Figure 4.9.

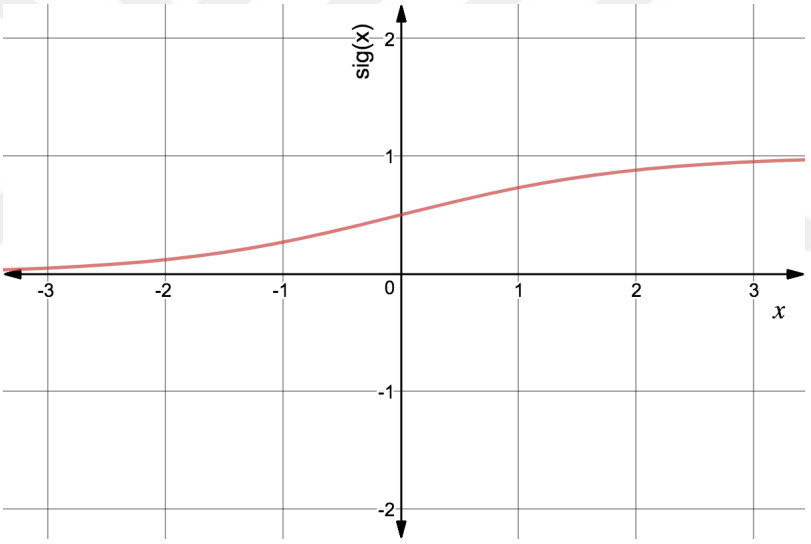


Figure 4.9. *Sigmoid* activation function

The *Tanh* function is mainly used for classification tasks betweet two classes. The negative inputs will be mapped to negative and the zero inputs will be mapped near zero in *tanh* function. General formula of *tanh* activation function is defined as follows:

$$\tanh(x) = \frac{sinh(x)}{cosh(x)} \tag{11}$$

Tanh activation function curve is given in Figure 4.10.

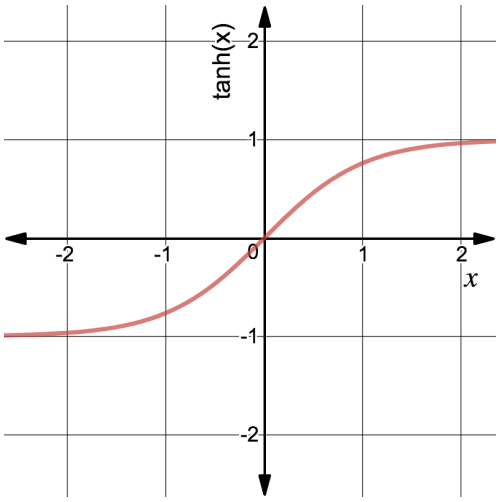


Figure 4.10. *Tanh* activation function

Softmax activation function is a generalized logistic activation function. It is used in the output layer of a neural network for the purpose of multi-class classification. The function normalizes a K-dimensional vector x into a K-dimensional vector $Softmax(x)$. General formula of *Softmax* activation function for $j = 1 \cdots K$ is given below:

$$Softmax(x)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{12}$$

ReLU(Rectified Linear Unit) activation function is a half rectified function. The function gives zero when x is less than zero and the function gives x when x equals or greater than zero. Negative inputs given to the *ReLU* activation function turns the value into zero and positive part of function is a linear function. General formula of *ReLU* function is given as follows:

$$ReLU(x) = \max(0, x) \tag{13}$$

ReLU activation function curve is given in Figure 4.11.

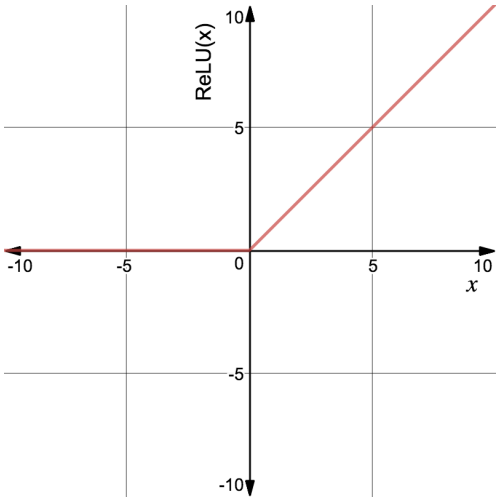


Figure 4.11. *ReLU* activation function

The activation function to calculate output of a hidden state in decoder LSTM that achieves the most accurate result is selected as activation function of encoder-decoder model.

4.2.1.5. Loss Function

Loss function is used for representing the error of the model. In optimization problems the main goal is minimizing the loss function. Loss function is also used to optimize the model in neural networks. Some of widely used loss functions Mean Squared Error, Categorical Cross Entropy and Cosine Proximity are tested in the experiment for training and evaluation of the model.

4.2.1.6. Optimizer Function

Optimizer function is used for minimizing error of the model. It performs a parameter update for each input and target pairs in training phase [32]. The optimizer functions SGD (Stochastic Gradient Descent), RMSProp (Root Mean Square Propagation) [33], Adadelta [34] and Adam (Adaptive Moment Estimation) [35] are tested in the experiments for training and evaluation of the model.

4.2.1.7. Model Configuration

The LSTM's are configured as follows:

- **Unit** represents the dimension of the memory cells in an LSTM. It specifies the dimensions of long-term dependencies. In this study, the unit is 256.
- **Activation function** is used for calculating the output of a hidden state. In this study, we used *Softmax*.
- **Loss function** is used for calculating the error of the model. In this study, we used *Categorical Cross Entropy*.
- **Optimizer function** is used for minimizing the loss function. In this study, we used *RMSProp*.

There has been some configurations on the specifications of the encoder-decoder model as follows:

- **Epoch** is the number of iteration performed with the same data during the training phase. In this study, the epoch is 25.
- **Batch size** is number of samples per gradient update. In this study, the batch size is 20.

Keras is a deep learning framework in Python programming language. It is widely used by researchers to develop applications for neural network based studies. In this study, the encoder-decoder model is implemented by using Keras. The neural layer representation of the model is given in Figure 4.12.

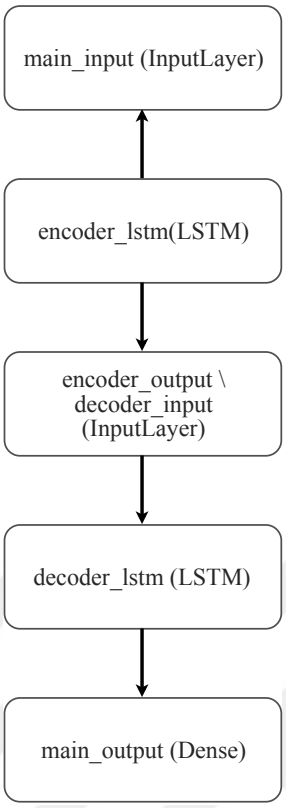


Figure 4.12. Neural Layer Representation of Encoder Decoder Model

4.2.2. Training the Model

In this approach, we train the encoder-decoder model on a Twitter dataset that includes noisy texts with their correspondent canonical forms. Encoder LSTM takes the noisy text as input and creates a fixed-length state vector to be used as input for the Decoder LSTM which is supposed to output the exact canonical form of the input noisy text. Therefore, any type of lexical changes attempted in the canonical forms is learned in the training phase by inducing the error types automatically, differently from the rule-based approaches that use manually constructed normalization rules.

Since each sentence in training set is processed word by word, the approach provides a word level normalization. An input character array is created by collecting all unique characters present in noisy sentences of the training set and used as the dimension of encoder input data that is used in training phase. Similarly a decoder input data and a decoder target data arrays are created by using canonical sentences of the training set target. The schema of EDM is given in Figure 4.13.

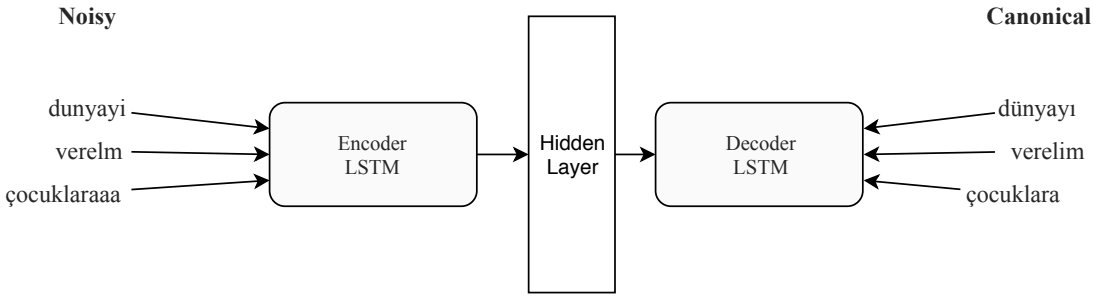


Figure 4.13. EDM training process representation

The encoder-decoder model is trained with following parameters:

- Character embedding dimension is 256
- Batch size is 20
- Number of epochs is 25

The change of loss for both training and validation is given in Figure. 4.14. according to the training epochs.

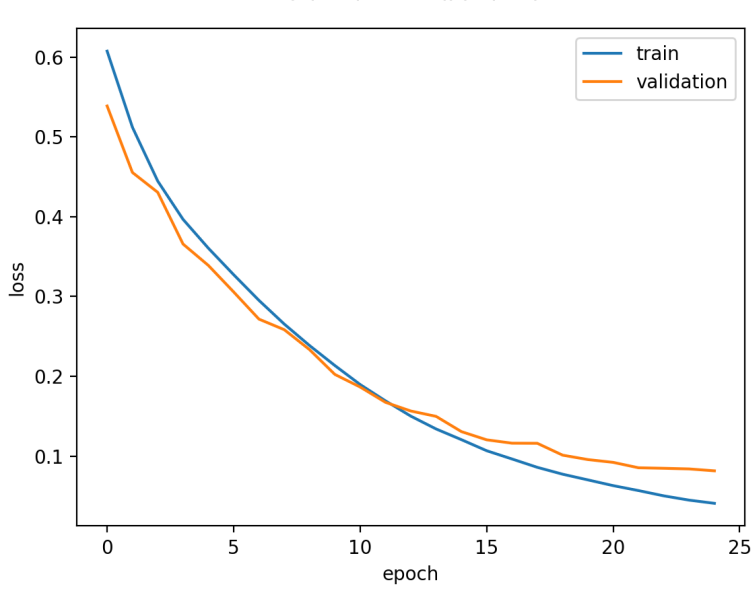


Figure 4.14. EDM Train/Validation loss function over epochs

The graph shows that there is no overfitting and the validation loss also continues to drop in time.

4.2.3. Normalization Predictions

Prediction phase is the main goal of this approach. After training is completed, this neural system is ready for further normalization operations. A noisy text is given to Encoder LSTM as input. Following to this, Encoder outputs a fixed-length state vector which is then used as input vector for Decoder LSTM. Finally, Decoder predicts an output as canonical text by using the parameters learned in training.

4.2.4. Limitations

Sequence-to-sequence normalization approach has a procedure of word level normalization that utilizes morphological and orthographic relations. Therefore a sentence is normalized without considering relations between words in sentence. Even the approach is trained and evaluated with the same dataset for sentence level normalization, the success rate of sentence level normalization was quite low because of the requirement of training the encoder decoder model with a large scale sentence base annotated data set. This is the main limitation of this approach and it slightly decreases the success rate of the approach.

5. EXPERIMENTAL ANALYSIS

5.1. Datasets

We use a set of 1200 tweets [1] that are manually collected and normalized for evaluation purposes. For sequence-to sequence normalization approach, we need to train EDM. Therefore, half of the dataset is used for training. Since the contextualized normalization approach is fully unsupervised, we learned distributed representation of the tokens in the collected tweets. For evaluation purposes for both approaches, the other half of the dataset is used for obtaining the accuracy of the models (see details in Table 5.1.)

Table 5.1. The Details of the Twitter Dataset used for Training and Testing

Data Sets	**Tweets**	**Tokens**	**OOV[1] Words**
Training Set	600	6,322	2,708
Test Set	600	7,061	2,192

NewsCor [27] that comprises of manually collected news archives from three major newspapers in Turkish is used for learning the word embeddings. Since the corpus is composed of news content, it is expected that most of the words in it are written in canonical form. The canonical words that are required to create the normalization lexicon are obtained by processing the Newscor corpus(see Section 4.1.2.).

The corpus consists of 184 million words, 212 million tokens and 2.2 million types (unique tokens). Assuming that the corpus contains mostly formally written text, we created a list of unique words and used the it for generating the normalization lexicon. Table 5.2. shows the details of the NewsCor.

Table 5.2. Turkish Newspaper Corpus Details

	Words	**Tokens**	**Types**
NewsCor Corpus	184M	212M	2.2M

[1] OOV stand for Out of Vocabulary

5.2. Evaluation Metrics

In this section, we briefly explain the used evaluation metrics for the proposed approaches. The metrics we used are accuracy, precision, recall and F1 score. Those metrics are defined as follows:

- **True positives (TP)**: These are the cases in which we predicted right, and it is actually right.
- **True negatives (TN)**: These are the cases in which we predicted wrong, and it is actually wrong.
- **False positives (FP)**: These are cases in which we predicted right, but it is actually wrong (Also known as a "Type I error.")
- **False negatives (FN)**: These are the cases in which we predicted wrong, but it is actually right (Also known as a "Type II error.")

Here are the details of evaluation metrics that we used in the study:

- **Accuracy** is the ratio of all correct predictions to input samples.

$$Accuracy = \frac{TP + TN}{Total} \tag{14}$$

- **Precision** is the ratio of correct positive predictions to all positive predictions. For text normalization task, the precision can be calculated as the ratio of number of *correctly normalized words* to number of *normalized words*.

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

- **Recall** is the ratio of correct positive predictions to all samples that should have been identified as positive. For text normalization task, the recall value can be calculated as the ratio of number of *correctly normalized words* to number of *words require normalization*.

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

- **F1 score** is the harmonic mean of precision and recall. It tries to find the balance between precision and recall.

$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{17}$$

5.3. Experiments

We compare the proposed neural approaches with MS Word spell checker, Zemberek [36] Normalizer, a lookup table method [1] and the rule-based cascaded approach [1].

MsWord is the model that Microsoft Word's Turkish spell checker is used to obtain the suggestions of spelling. However the intended purpose of MS Word is spell checking for the Microsoft Word application, it is used as a baseline model to compare success rates. The best suggestions gathered from Microsoft Word's spell checker are counted as normalization output for given inputs. Microsoft Word's spell checker is also used in many studies in NLP field as a standart spell checker such as Bernstein et. al. [37], Torunoğlu and Eryiğit [1], Demir [38], Doush et. al. [39].

Zemberek [36] is an open source framework that contains NLP tools for Turkish such as morphologic analyzer, tokenizer, stemmer, lemmatizer, normalizer, etc. In this study the normalizer tool of Zemberek that provides basic spell checking and normalization suggestion is used for Zemberek Normalizer Model. Zemberek is also used in many studies in Turkish such as Çöltekin [40], Boynukalin [41], Torunoğlu and Eryiğit [1], Demir [38],

Lookup table and Rule-base cascaded models are developed by Torunoğlu and Eryiğit [1] for purpose of text normalization on Turkish social media. Since the proposed normalization methods are evaluated by using the test dataset, the results of MsWord, Zemberek, Lookup table and Rule-based cascaded models are obtain from the study [1] and compared with ours.

The experimental results are evaluated by the accuracy evaluation metric and given in Table 5.3. The accuracy metric is defined as the ratio of correctly normalized words to the total number of words to be normalized in the test set. Within our knowledge, the study [1] is the only publicly available text normalization study on Turkish social media and it achieves

significant text normalization results on Turkish social media. Consequently we selected the study and compared our approaches with it.

Table 5.3. Normalization Results

Model	***Accuracy (%)***
Ms Word	25
Zemberek Normalizer [36]	21
Lookup Table[1]	34
Rule-based Cascaded Approach [1]	71
Contextual Normalization Approach	**72.18**
Sequence-to-sequence Normalization Approach	**74.80**

As the results show, the two proposed approaches in this paper outperform all other models. The sequence-to-sequence approach gives the highest accuracy for Turkish social media normalization.

The fidelity of contextual normalization approach is measured by using precision, recall, F1 score and accuracy metrics (see Section 5.2.). Consonant skeleton and the value of n to select the nearest neighbours ($n = 25$ and $n = 100$) are evaluated and given in Table 5.4.

Table 5.4. Contextual Normalization Results with Parameters

KNN	**Consonant Skeleton**	***Precision (%)***	***Recall (%)***	***F1 (%)***	***Accuracy(%)***
n=25	word-wide	**55.22**	70.41	61.92	39.22
	first 5 characters	54.37	73.11	62.36	39.21
	first 3 consonants	54.35	**73.29**	**62.42**	**49.48**
n=100	word-wide	52.81	79.79	63.55	72.13
	first 5 characters	52.81	79.81	63.56	72.15
	first 3 consonants	**52.83**	**79.84**	**63.58**	**72.18**

The results show that larger values of n give a higher normalization accuracy since it increases the probability of finding the correct canonical candidate among all candidates.

We also perform different experiments by changing the length of the consonant skeleton for extracting the candidates among the nearest neighbours while using the edit distance to generate the lexicon. The results show that for the 25 nearest neighbours, the consonant skeleton length has a high impact on the accuracy while there is no significant change in precision, recall and F1 values. When we extract the candidates by computing the lexical similarity cost on only the first 3 consonants, the accuracy improves by around 10% with an accuracy of 49.48% and recall increases by around 3%. However, the consonant skeleton length does not have much effect on the accuracy when we use the first 100 nearest neighbours. The highest accuracy is obtained for $n = 100$ and when the first 3 consonants are used for the lexical similarity cost. Changing the consonant skeleton for $n = 100$ and 25 neareast neighbours causes a slight change on precision and recall values except for recall value of word-wide skeleton for $n = 25$ neareast neighbours. This can be a sign that rather than syntactic, mostly semantically related neighbours are captured in the top neighbours of a word and it makes it hard to find the candidate canonical form in 25 words compared to 100 words. Therefore, the length of the consonant skeleton has a higher impact for a less number of neighbours.

In sequence-to-sequence normalization approach, the most accurate result are obtained by performing different experiments. The configuration of encoder-decoder model is tested with different values of unit, activation function, optimizer function and loss function parameters and fixed values of epoch and batch size parameters. The effects of changing the configuration values are compared and the most accurate parameter configuration is selected as model configuration (see Section 4.2.1.7.)

SGD, RMSProp, Adadelta and Adam are used as optimizer function options. Mean Squared Error, Categorical Cross Entropy and Cosine Proximity are used as loss function options. Tanh, Sigmoid, Softmax and ReLU are used as activation function options. The parameter configurations of optimizer functions and loss functions corresponding to each activation function are given below. The values *batch size = 20*, *unit = 128* and *epoch = 15* are constant for these experiments.

The experiments of *Tanh* activation function configuration are given in Table 5.5.

Tanh		Loss Function		
		Mean Squared Error	Categorical Cross Entropy	Cosine Proximity
Optimizer Function	SGD	0	0	0
	RMSProp	19.28	0	27.93
	Adadelta	0	0	4.5
	Adam	26.42	0	**32.05**

Table 5.5. Sequence-to-Sequence Normalization Accuracies of Tanh Activation Function Configuration

The experiments of *Sigmoid* activation function configuration are shown in Table 5.6.

Sigmoid		Loss Function		
		Mean Squared Error	Categorical Cross Entropy	Cosine Proximity
Optimizer Function	SGD	0	0	0
	RMSProp	18.25	**36.43**	31.72
	Adadelta	0	0.07	0
	Adam	0.03	3.85	17.55

Table 5.6. Sequence-to-Sequence Normalization Accuracies of Sigmoid Activation Function Configuration

The experiments of *Softmax* activation function configuration are given in Table 5.7.

Softmax		Loss Function		
		Mean Squared Error	Categorical Cross Entropy	Cosine Proximity
Optimizer Function	SGD	0	0	0
	RMSProp	23.88	**43.33**	34.25
	Adadelta	0	2.33	0
	Adam	1.57	31.2	39.45

Table 5.7. Sequence-to-Sequence Normalization Accuracies of Softmax Activation Function Configuration

The experiments of *ReLU* activation function configuration are shown in Table 5.7.

ReLU		Loss Function		
		Mean Squared Error	Categorical Cross Entropy	Cosine Proximity
Optimizer Function	SGD	0.07	0	0
	RMSProp	**25.7**	0	9.45
	Adadelta	0	0	3.08
	Adam	20.38	0.35	2.53

Table 5.8. Sequence-to-Sequence Normalization Accuracies of ReLU Activation Function Configuration

The results show that Softmax activation function, RMSProp optimizer function and Categorical Cross Entropy loss function parameter configuration achieves the most accurate result for Sequence-to-sequence Normalization. After the activation function, optimizer function and loss function experiments main configuration of the model is determined by testing different batch size and unit values for the selected configuration. The experiments of batch size and unit configurations for Softmax activation function, RMSProp optimizer and Categorical Cross Entropy loss function for 15 epochs are given in Table 5.9.

		Units			
		64	128	256	512
Batch Size	10	41.1	60.68	63.82	59.4
	20	25.5	43.33	**64.8**	57.97
	30	20.05	37.43	49.18	50.3
	40	15.88	29.58	31.8	33.23

Table 5.9. Sequence-to-Sequence Normalization Accuracies of Units and Batch Size

The results show that batch size value of 20 and unit value of 256 parameter configuration achieves the most accurate result for Sequence-to-sequence Normalization. As a result of selected configurations tested with different epoch values are shown in Table 5.10.

	Epoch				
	10	15	20	25	30
Model Accuracy (%)	34.92	64.8	66.35	**74.8**	71.58

Table 5.10. Sequence-to-Sequence Normalization Accuracies over Epochs

Experiments show that the most accurate model configurations parameters for sequence-to-sequence normalization approach are Softmax activation function, Categorical Cross Entropy loss function, RMSProp optimizer function, 265 character embedding dimension, batch size value of 20 and epoch value of 25.

Some normalization output examples of the two approaches are given in Table 5.11. and Table 5.12.. The results show that the contextualized normalization approach is better at finding the repetitive letters such as *dersleriii-dersleri*, whereas the sequence-to-sequence model is better at correcting the abbreviations such *gsye-Galatasaray'a* with more drastic lexical changes.

Table 5.11. Example Normalization Outputs of The Contextual Normalization

Input (noisy) word	Output (normalized) word
asagdaydı	aşağıdaydı
adanayı	Adana'yı
derssleriii	dersleri
yalniz	yalnız
iliski	ilişki

Table 5.12. Example Normalization Outputs of The Sequence-to-Sequence Normalization

Input (noisy) word	Output (normalized) word
candir	candır
saol	sağol
gsye	Galatasaray'a
tanisiyim	tanışayım
foto	fotoğraf

Unified Normalization method is the third normalization method which is used to combine the two normalization approach we proposed in this study. In this method, the normalization lexicon created in the contextual normalization approach is used to train encoder-decoder model in the sequence-to-sequence normalization by using canonical and noisy word pairs (see Section 4.1.2.).

However sequence-to-sequence normalization method is a supervised method itself, it is performed as an unsupervised method in unified normalization method. In this case, the large-sized data set required to train the encoder-decoder model was automatically obtained from normalization lexicon and annotated data is not required for method. However experiments of unified normalization method are repeated with different dataset combinations by using lexical similarity threshold values and percentages of correct and misspelled word pairs in the dataset, a significant result is not achieved. As more extensive studies are required to ensure

that the method can achieve successful results, the results of this method are not included in the normalization results.

5.4. Error Analysis

In the study, it is observed that for some cases, the normalization process produces some similar errors. In the Contextual Normalization approach, abbreviation normalization is one of the frequent error type. Since the relationship between the abbreviations and their expansions can not be adequately represented by lexical similarity or cosine similarity metric, normalized words contain incorrect results for the abbreviation.

Since the lexical similarities between stems with different inflections can be quite close, suffix parts could be malformed even if the correct words are proposed. Although inflectional form normalization is also encountered in the Contextual Normalization approach, it is the most common error type in Sequence-to-Sequence Normalization approach.

6. CONCLUSION

6.1. Conclusion

In this thesis, two neural-based text normalization approaches are proposed for the normalization task on Turkish social media data. One method makes use of pre-trained neural word embeddings in order to include the contextual information to capture the canonical form of a given word. The results show that the proposed approach outperforms other models although the proposed model in this study is fully unsupervised. The latter method adopts an encoder-decoder model with a bi-LSTM architecture in a supervised framework. This model does not make use of any contextual information, but instead it learns the error patterns through a large number of noisy-canonical word pairs. Therefore, the lexical features play an important role in this model.

The two proposed approaches in this study outperforms all other models When we compare the results of our approaches with the other text normalization studies on Turkish social media. The results show that sequence-to-sequence normalization approach gives 74.8% accuracy which is the highest score of the text normalization studies that we compared. while contextual normalization approach achieves over 72% accuracy as the second highest score.

This study shows that using contextual and orthographic features of text and neural normalization approaches may provide similar or higher success rates than traditional rule-dependent studies. Higher accuracies for agglutinative languages such as Turkish can be obtained by extending the context of used data sets, context vectors, and artificial neural networks. By using rule-independent techniques, the dependency of the methodology on error pattern changes is eliminated.

From the point of answering research questions, the study shows that a rule-independent text normalization method can be applied to social media data in Turkish. It is also clearly performed that a rule independent method have as high accuracy as a conventional rule-based method in text normalization. However the success ratio of neural normalization methods are significant for Turkish, it is necessary to repeat the process for other agglutinative languages in order to more strongly answer the research question 'How successful are neural normalization methods in agglutinative languages such as Turkish'.

6.2. Future Research Directions

We aim to experiment on other languages in the future since both models are language independent. This study shows that using neural normalization approaches

Including more contextual information by performing sentence-level normalization in the encoder-decoder model is also left as a future goal.

A morphological analyzer is planned to included in the contextual normalization approach in the future to ensure that the similarities between noisy and canonical words are detected with greater success.

A noisy word detection system is planned to add as the preliminary stage of text normalization as a future work. In this way, by providing the initial determination of noisy words made will increase the success ratio.

A APPENDIX NORMALIZATION OUTPUTS

Table 1.1. Contextual Normalization Outputs

Noisy Tweet	**Normalized Tweet**
Doktor 90 dakikanın kaç saat yaptığını bilmiyor , bu adam doktor olduysa benim meslek sahibi olmucam diye bir korkum yok	Doktor 90 dakikanın kaç saat yaptığını bilmiyor , bu adam doktor olduysa benim meslek sahibi olmayacağım diye bir korkum yok
Spiker Gs savunmasındaki boşluklar diyor =)) stattaki boşlukları söyle .	Spiker gsnin savunmasındaki boşluklar diyor =)) stattaki boşlukları söyle .
@pirdika foto da hoş olmuş hani ;))	@pirdika fotoğraf da hoş olmuş hani ;))
adamın dibi nası oluoo biri anlatabilir miii bana lütfennnn ama lütfnnnn	Adamın dibi nasıl oluyor biri anlatabilir mi bana lütfen ama lütfen
emre aydın aşk acısı temalı şarkılarının yerine kenan doğulunun şans meleğimi gibi bi şarkı yapsın bi kere de	Emre Aydın aşk acısı temalı şarkılarının yerine Kenan doğulunun şans meleğimi gibi bir şarkı yapsın bir kere de
ilk aşkıma söylediğim şarkıı ahaahahhhhh	Ilk aşkıma söylediğim şarkımızın ahaahahhhh
sen fotoğrafnı beğenirsin o seninkini beğenmez ahaha çok büyük eziklik ahahah zaten hiç yakışıklı değildi dimi bende öyle düşünmüştüm ahahah	Sen fotoğrafı beğenirsin o seninkini beğenmez ahaha çok büyük eziklik hahaha zaten hiç yakışıklı değildi dimi bende öyle düşünmüştüm hahaha
#FenerbahceliOlmak İÇİN ÖNCE ADAM OLMAK GEREKİR ...	#FenerbahceliOlmak için önce adam olmak gerekir ...
@dionysosx çok ihmal ediyosun beni , usb'yide salı günü giyeceğin pantolanun cebine sok , bu haftada unutursan bilmiyorum artık , hadi kib	@dionysosx çok ihmal ediyosun beni , usb'yide salı günü giyeceğin pantolanun cebine sok , olabilir haftada unutursan bilmiyorum artık , hadi kib
hani bazı anlar vardır insanın nefesi kesilir soluksuz kalır , kalbi hızlı hızlı atar ... işte biz buna aşk dedik yardım etmedik , adam öldü .	Hani bazı anlar vardır insanın nefesi kesilir soluksuz kalır , kalbi hızlı hızlı atar ... işte biz buna aşk dedik yardım etmedik , adam öldü .
Kuzey Güneyin senaristide amma gelgit akıl ha Adam gibiler bi bakmşn cibilliyetszn teki olmuş Kafasına düzenli aralklrla saksı mı düşüyo ne ?	Kuzey güneyin senaristi de amma gelgit akıl ha adam gibiler bir bakmışın cibiliyetsizin teki olmuş kafasına düzenli aralıklarla saksı mı düşüyor zaman ?

Table 1.2. Sequence-to-Sequence Normalization Outputs

Noisy Word	**Normalized Word**
canim	canım
candir	candır
saol	sağol
foto	fotoğraf
yea	ya
anlatılcak	anlatılacak
bi	bir
nerden	nereden
cüneyit	cüneyt
oglum	oğlum
teomanın	Teoman'ın
bjk	Beşiktaş
gs	Galatasaray
yerrr	yer
diyo	diyor
sanıyodu	sanıyordu
etmicem	etmeyeceğim
tanisiyim	tanışayım
saygidan	saygıdan
tanisalim	tanışalım
slm	selam
istiyosan	istiyorsan
izmirde	İzmir'de
alacğım	alacağım
eglenceli	eğlenceli
msj	mesaj
deiştiriyo	deiğştiriyor

REFERENCES

[1] Dilara Torunoğlu and Gülsen Eryiğit. A cascaded approach for social media text normalization of turkish. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pages 62–70. **2014**.

[2] Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. Japanese text normalization with encoder-decoder model. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 129–137. **2016**.

[3] AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 33–40. Association for Computational Linguistics, **2006**.

[4] Eleanor Clark and Kenji Araki. Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. *Procedia-Social and Behavioral Sciences*, 27:2–11, **2011**.

[5] Hany Hassan and Arul Menezes. Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1577–1586. **2013**.

[6] Cagil Sonmez and Arzucan Ozgur. A graph-based approach for contextual text normalization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 313–324. **2014**.

[7] Vivek Kumar Rangarajan Sridhar. Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 8–16. **2015**.

[8] Helena Gomez, Darnes Vilarino, Grigori Sidorov, and David Pinto Avendano. Cicbuapnlp at semeval-2016 task 4-a: Discovering twitter polarity using enhanced embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 145–148. **2016**.

[9] Ahmed Ali, Preslav Nakov, Peter Bell, and Steve Renals. Werd: Using social text spelling variants for evaluating dialectal speech recognition. *arXiv preprint arXiv:1709.07484*, **2017**.

[10] Jerome R Bellegarda and Christof Monz. State of the art in statistical methods for language and speech processing. *Computer Speech & Language*, 35:163–184, **2016**.

[11] Nasser Zalmout, Alexander Erdmann, and Nizar Habash. Noise-robust morphological disambiguation for dialectal arabic. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 953–964. **2018**.

[12] Carlos Amaral, Dominique Laurent, André FT Martins, Afonso Mendes, Cláudia Pinto, et al. Design and implementation of a semantic search engine for portuguese. In *LREC*. **2004**.

[13] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10. **2010**.

[14] John R. Firth. *A synopsis of linguistic theory 1930-1955*. Oxford: Blackwell, **1957**.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of ICLR Workshop*. **2013**.

[16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112. **2014**.

[17] I Dan Melamed. Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. *arXiv preprint cmp-lg/9505044*, **1995**.

[18] Trideep Rath. *Word and Relation Embedding for Sentence Representation*. Ph.D. thesis, Arizona State University, **2017**.

[19] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91:1, **1991**.

[20] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, **1994**.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, **1997**.

[22] Zellig Harris. *Distributional Structure*. Word, **1954**.

[23] Gerard Salton. The smart retrieval system—experiments in automatic document processing. **1971**.

[24] Chen Li and Yang Liu. Improving named entity recognition in tweets via detecting non-standard words. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 929–938. **2015**.

[25] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, **2011**. ISSN 1532-4435.

[26] Yi Yang and Jacob Eisenstein. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 61–72. **2013**.

[27] Haşim Sak, Tunga Güngör, and Murat Saraçlar. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *Advances in natural language processing*, pages 417–427. Springer, **2008**.

[28] Gems Cuevas, Jedd Gopez, Nicco Nocon, and Peter Suministrado. Norm: A text normalization system for filipino shortcut texts using the dictionary substitution approach, **2014**.

[29] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, **2014**.

[30] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, **2015**.

[31] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, **2014**.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, **2016**. `http://www.deeplearningbook.org`.

[33] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, **2012**.

[34] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, **2012**.

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, **2014**.

[36] Ahmet Afsin Akın and Mehmet Dündar Akın. Zemberek, an open source nlp framework for turkic languages. *Structure*, 10:1–5, **2007**.

[37] Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 313–322. ACM, **2010**.

[38] Seniz Demir. Context tailoring for text normalization. In *Proceedings of TextGraphs-10: the Workshop on Graph-based Methods for Natural Language Processing*, pages 6–14. **2016**.

[39] Iyad Abu Doush and Ahmed M Al-Trad. Improving post-processing optical character recognition documents with arabic language using spelling error detection and correction. *International Journal of Reasoning-based Intelligent Systems*, 8(3-4):91–103, **2016**.

[40] Cagri Cöltekin. A freely available morphological analyzer for turkish. In *LREC*, volume 2, pages 19–28. **2010**.

[41] Zeynep Boynukalin. Emotion analysis of turkish texts by using machine learning methods. *Middle East Technical University*, **2012**.

CURRICULUM VITAE

Credentials

Name,Surname:	Sinan GÖKER
Place of Birth:	Gölbaşı, Turkey
Marital Status:	Single
E-mail:	sinangoker12@gmail.com
Address:	Computer Engineering Dept., Hacettepe University Beytepe-ANKARA

Education

BSc. :	Computer Engineering Dept., Dokuz Eylul University, Turkey
MSc. :	Computer Engineering Dept., Hacettepe University, Turkey

Foreign Languages

English

Work Experience

Software Engineer at Etiya Bilgi Teknolojileri (2012-2017)
Software Engineer at Havelsan (2017-Present)

Areas of Experiences

NLP, Machine Learning, Text Normalization,
Unsupervised Learning

Project and Budgets

-

Oral and Poster Presentations

-

PUBLICATIONS

Göker, S., Can, B. ”Neural Text Normalization for Turkish Social Media” *Computer Science and Engineering (UBMK), 2018 International Conference on. IEEE*, **2018**.

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
THESIS/~~DISSERTATION~~ ORIGINALITY REPORT

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
TO THE DEPARTMENT OF COMPUTER ENGINEERING

Date: 16/07/2018

Thesis Title / Topic: Neural Text Normalization for Turkish Social Media

According to the originality report obtained by ~~myself~~/my thesis advisor by using the *Turnitin* plagiarism detection software and by applying the filtering options stated below on 16./07./18. for the total of ..68..... pages including the a) Title Page, b) Introduction, c) Main Chapters, d) Conclusion sections of my thesis entitled as above, the similarity index of my thesis is 10.. %.

Filtering options applied:

1. Bibliography/Works Cited excluded
2. Quotes excluded / ~~included~~
3. Match size up to 5 words excluded

I declare that I have carefully read Hacettepe University Graduate School of Sciene and Engineering Guidelines for Obtaining and Using Thesis Originality Reports; that according to the maximum similarity index values specified in the Guidelines, my thesis does not include any form of plagiarism; that in any future detection of possible infringement of the regulations I accept all legal responsibility; and that all the information I have provided is correct to the best of my knowledge.

I respectfully submit this for approval.

Date and Signature

Name Surname: SİNAN GÖKER
Student No: N13227378
Department: COMPUTER ENGINEERING
Program: MASTER OF SCIENCE
Status: ☒ Masters ☐ Ph.D. ☐ Integrated Ph.D.

ADVISOR APPROVAL

APPROVED.

Asst. Prof. Dr. Burcu Can Buglalilar

(Title, Name Surname, Signature)