

## Neural text normalization for Turkish social media

Item Type	Conference contribution
Authors	Goker, Sinan;Can, Burcu
Citation	Göker, S. and Can, B. (2018) Neural text normalization for Turkish social media, 2018 3rd International Conference on Computer Science and Engineering (UBMK), 20-23 September, 2018, Sarajevo, Bosnia-Herzegovina.
DOI	<a href="https://doi.org/10.1109/ubmk.2018.8566406">10.1109/ubmk.2018.8566406</a>
Publisher	IEEE
Download date	2026-04-15 06:41:31
License	<a href="https://creativecommons.org/licenses/by-nc-nd/4.0/">https://creativecommons.org/licenses/by-nc-nd/4.0/</a>
Link to Item	<a href="http://hdl.handle.net/2436/623730">http://hdl.handle.net/2436/623730</a>

# Neural Text Normalization for Turkish Social Media

Sinan GÖKER

Havelsan A.Ş.

Department of Computer Engineering

Hacettepe University

Ankara, Turkey

sinangoker@hacettepe.edu.tr

Burcu CAN

Department of Computer Engineering

Hacettepe University

Ankara, Turkey

burcucan@cs.hacettepe.edu.tr

**Abstract**—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 due to its informal 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. In this study, we apply two approaches for Turkish text normalization: Contextual Normalization approach using distributed representations of words and Sequence-to-Sequence Normalization approach using neural encoder-decoder models. As the approaches applied to Turkish and also other languages are mostly rule-based, additional rules are required to be added to the normalization model in order to detect new error patterns arising from the change of the language use in social media. In contrast to rule-based approaches, the proposed approaches provide the advantage of normalizing different error patterns that change over time by training with a new dataset and updating the normalization model. Therefore, the proposed methods provide a solution to language change dependency in social media by updating the normalization model without defining new rules.

**Index Terms**—text normalization, distributed representation, encoder-decoder, unsupervised learning, long short-term memory (LSTM), deep learning

## I. INTRODUCTION

Social media has become one of the most irreplaceable parts of our lives. People share almost every idea, thought and dream of their own through it and 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 data convenient for processing becomes limited due to spelling mistakes, misuses, common current abbreviations, and structural disorders, which have 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 the studies to convert the data into a normal form, the erroneous texts are required to be corrected. This task is called text normalization.

Text normalization studies provide successful solutions to correct erroneous 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 techniques are generally rule-based. 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 paper, we introduce methods that are different from conventional methods so that the proposed methods result in more accurate outputs as being more sensitive to changes in error patterns due to normalizing over distributed representations of words and also normalizing over orthographic patterns that are captured automatically by an encoder-decoder neural network architecture.

Word representations are learned through the contextual similarities between words, as Firth suggests<sup>1</sup>. 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.

In this study, we use word2vec [2] 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. Therefore, words in similar contexts have similar word representations. Noisy social media data in Turkish language is used as input and each erroneous word is matched with its canonical candidate words by using its word representation. Here we assume that erroneous 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 erroneous forms because they all mean the same.

<sup>1</sup>“You shall know a word by the company it keeps” [1]

As a second normalization approach, we present an encoder-decoder model to learn the error rules automatically. Encoder-decoder architecture using recurrent neural networks [3] is 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.

The paper is organized as follows: Section II addresses the related work, section III describes the proposed normalization methods, section IV presents the experimental results, and section V concludes the paper with future goals.

## II. RELATED WORK

Different features have been utilized in text normalization by using different methods. Many methods have used the similarity between orthographic features of words to capture their canonical forms. For this purpose, different edit distance measures have been used, such as longest common subsequence and edit distance. Additionally, contextual features have also been utilized to capture semantic relations between noisy and canonical forms of words.

Hasan and Menezes [4] utilize both contextual and lexical features. The authors 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. 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.

Sönmez and Özgür [5] introduce another graph-based method that uses grammatical features in addition to contextual and lexical features. The contextual and grammatical features are encoded in a graph, where the relative positions of words and their part-of-speech (PoS) tags are encoded.

Sridhar [6] introduces another unsupervised text normalization algorithm that makes of distributional features of words and phrases. Unlike the previous work, contextual features are learned via neural word embeddings by using the continuous bag-of-words model (CBOW) of word2vec [2] and the neural network architecture by Collobert et al. [7]. A lexicon that consists of noisy and canonical word pairs is constructed by making use of the distance between neural word embeddings and by filtering out some of the candidates to find the best canonical candidate for each word. Contextual normalization approach that we apply for Turkish is based on the distributed

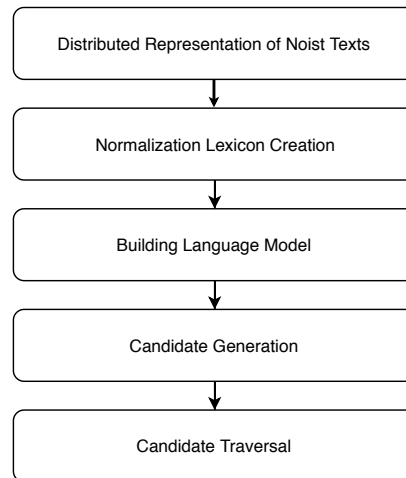


Fig. 1. Methodology of Contextual Normalization Approach.

representations of words and the lexical edit distance measure proposed by Sridhar [6].

Yang and Eisenstein [8] 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 parameters of the model only for the observed n-grams while doing gradient-based updates.

Ikedo, Shindo, and Matsumoto [9] 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 and decoder architecture. In order to overcome the need of a large corpus required by the 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.

Sutskever, Vinyals, and Le [3] propose a method by using deep recurrent neural networks for the machine translation task as a sequence-to-sequence learning problem. They use long short-term memory neural networks (LSTMs) to map each source sentence to its target sentence in another language. They achieve successful results especially on long sentences.

There is not much work on Turkish social media normalization. One approach is proposed by Torunoğlu and Eryiğit [10]. Their method is rule-based and the authors aim to correct the given text by applying manually defined rules as a pipeline process until obtaining the canonical form of a given noisy word. Therefore, their model is supervised and language-dependent because of the manually defined rules.

## III. METHODOLOGY

In this study, noisy social media text (i.e. Twitter tweets) is used as input and a corrected form of the noisy text is generated by performing two different approaches. Both approaches are compared on a validation set in order to evaluate their accuracies.

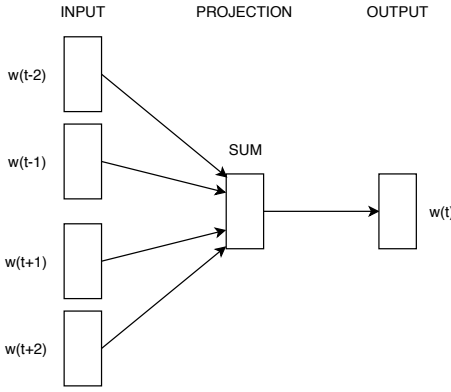


Fig. 2. Continuous Bag-of-Words (CBOW) architecture [2].  $w(t)$  denotes the center word that is guessed through the contextual words:  $w(t-2)$ ,  $w(t-1)$ ,  $w(t+1)$ , and  $w(t+2)$  for a window size of 2 in this example.

Here, we apply two different neural methods for text normalization. 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 a neural encoder-decoder model that uses bidirectional LSTM (bi-LSTM) architecture [11] that learns normalization rules from a large set of noisy and canonical word pairs. Both methods are described below.

### A. Contextual Normalization Approach

Contextual normalization approach is performed by using distributed representations of noisy words. For the distributed representations, we use the neural word embeddings. The main steps of this approach are shown in Fig 1.

Learning the distributed representation of the 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 syntactic features, that is called an embedding. Here, CBOW (Continuous Bag-of-Words) approach of word2vec [2] is used to learn the word representations (see Fig. 2). In the 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 the training. As a result, a vector space is created for the social media text where each word is represented by a word embedding.

We use a corpus of manually collected news [12] that consists of 184 million words to generate the lexicon. Once we select 2 million canonical word types by finding each unique word in the news archive, we retrieve the nearest  $n^2$  neighbours of each canonical word by using the pre-trained word embeddings. To this end, we use the cosine similarity

between two word embeddings  $u$  and  $v$ :

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

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, lexical similarity cost (SimCost) between two words  $w_1$  and  $w_2$  is calculated similar to Hassan and Menezes [4]:

$$\text{SimCost}(w_1, w_2) = \frac{\text{LCSR}(w_1, w_2)}{\text{ED}(w_1, w_2)} \quad (2)$$

Here, LCSR denotes the longest common subsequence ratio [13] where LCS denotes the longest common subsequence and ED is the edit distance:

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

In this study, consonant skeleton structure is used to select a more significant letter sequence that can express the noisy words. Differently from Sridhar [6], as Turkish is agglutinating, two additional consonant skeleton structures representing the relation between stems of words are used to calculate the edit distance for the lexical similarity cost. Since stem is located in the beginning of a word in agglutinating languages, the additional consonant skeleton structures are suggested. To this end, we eliminate the vowels in words and computed the edit distance between different types of consonant skeletons of the two words.

Once the lexical similarity cost is calculated, the cost value is stored in the normalization lexicon with its correspondent pair. Some example noisy and canonical word pairs with their similarity costs are given in Table I. The candidate canonical forms for the noisy word *düşüncm* (*my opinion*) are given in Table II according their lexical similarity costs. 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) = undef ined). Algorithm 1 describes the process for generating the normalization lexicon.

After the lexicon is generated, we build a bigram language model for the transition probabilities between consecutive words in order to perform a sentence level text normalization. We train the bigram language model on the same news corpus dataset [12] and the resulting bigram probability values are stored in the transition lexicon to be used in the last step of the Viterbi algorithm.

Sentence level text normalization is performed word by word by using Viterbi algorithm in order to choose the normalized candidate sentence that has the minimum cost. 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. The transition costs between any candidate canonical forms of two consecutive noisy words is

<sup>2</sup>We use different thresholds to choose the nearest neighbours. The results for different values of  $n$  are given in Section IV.

---

**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  $L$ 

```

1: for  $w \in W$  do
2:   for  $v \in V$  do
3:     if ( $v \notin W$ ) then
4:       compute Cosine Similarity( $w,v$ )
5:       store top  $K$  neighbours in map  $M(w, v)$ 
6:     end if
7:   end for
8: end for
9: for  $w \in W$  do
10:  for  $m \in M$  do
11:    compute SimCost( $w,m$ )
12:    push  $m \rightarrow (w, SimCost(w, m))$  into lexicon  $L$ 
13:  end for
14: end for

```

---

TABLE I  
EXAMPLE NOISY-CANONICAL WORD PAIRS IN THE NORMALIZATION LEXICON

Noisy Form (n)	Canonical Form (c)	SimCost(n,c)
batıl	batıl	-13.8
kaciramaz	kaçıramaz	0.34
ogret	öğret	0.41
patlycam	patlyorum	1.69
çikarıl	çikarıldığı	1.97
sevinmicem	sevinemedim	2.48
kutliyim	kutlayayım	2.48
sevinmicem	sevinemedim	2.48
ölmедđi	ađlamadıđı	2.48
sevinmicem	sevinemedim	2.77
karsilikli	karşılıkli	2.89
oturmayi	çöpü	13.8

retrieved from the bigram language model. Here, we use the negative logarithm of the bigram probabilities for the transition costs. Viterbi cost for all candidates belonging to each word of the sentence with the candidates of the previous word is calculated. The minimum cost is stored in each time step. This process is applied to all words by adding up the Viterbi costs cumulatively (see Fig 3). The final Viterbi cost becomes the cost of the candidate canonical sentence. A backtrace 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.

### B. Sequence-to-Sequence Normalization Approach

Sequence-to-Sequence normalization approach is performed by using a neural encoder-decoder model. According to the recent studies, encoder-decoder models have been effective especially for the machine translation task [3], [14], [15], [16]. In this study, we adopt LSTM networks for the encoder-

TABLE II  
AN EXAMPLE WORD AND ITS CANDIDATE CANONICAL FORMS

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

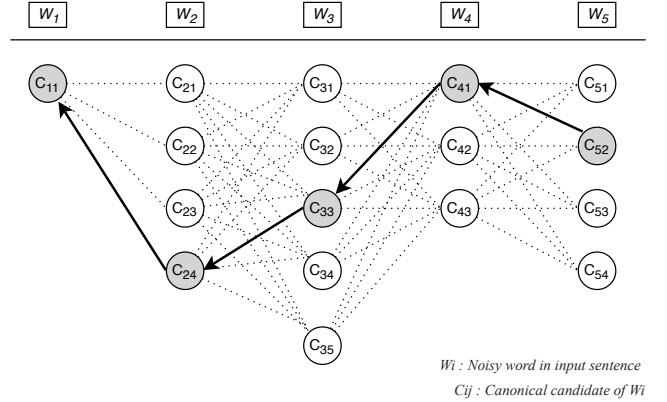


Fig. 3. Candidate Traversal by using Viterbi Algorithm

decoder model in order to use deep learning techniques for a higher accuracy in the normalization.

In this study, 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, encoder-decoder model is trained to learn how to normalize a given noisy text in the training phase. Therefore, any type of lexical changes attempted in the canonical forms is learned in training by inducing the error types automatically, differently from the rule-based approaches that use manually constructed normalization rules. For further predictions, a noisy text is given to the encoder LSTM as input. The encoder outputs a fixed-length state vector which is then used as input vector for the decoder LSTM. Finally, the decoder predicts an output as canonical text by using the parameters learned in training.

## IV. EXPERIMENTS AND RESULTS

### A. Data and Implementation Details

We use a set of 1200 manually collected tweets [10] that are manually normalized for evaluation purposes. Table III shows the details of the dataset. Since the contextualized normalization approach is fully unsupervised, we use both training and testing datasets for training. However, the encoder-decoder model uses half of the tweets for training and half of the data for testing purposes in a supervised setting.

NewsCor [12] that comprises of manually collected news archives from three major newspapers in Turkish is used for

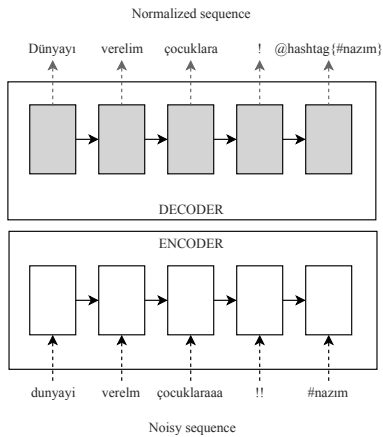


Fig. 4. Sequence-to-sequence Normalization Architecture by using Encoder - Decoder Model (EDM)

TABLE III  
TWITTER DATASET USED FOR TRAINING AND TESTING [10]

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

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 their canonical forms. The canonical words required to create the normalization lexicon are obtained from the NewsCorpus. The corpus consists of 184 million word types and 212 million word tokens. Assuming that the corpus contains mostly formally written text, we created a list of unique words and used the list for generating the normalization lexicon. Table IV shows the details of the news corpus.

We use RMSProp optimizer that is mainly used for the recurrent neural networks. We use categorical cross-entropy for the loss function, and we use softmax for the activation function. With 256 character embedding dimension and 20 batch size, the encoder-decoder model is trained for 25 epochs. The change of loss for both training and validation is given in Fig. 5 according to the training epochs. The graph shows that there is no overfitting and the validation loss also continues to drop gradually in time.

### B. Evaluation

We compare the two neural-based approaches against to Microsoft Word, Zemberek [17], a lookup table method [10], and the rule-based cascaded approach [10]. The experimental results are given in Table V. The accuracy is defined as the ratio of correctly normalized words to the total number of words to be normalized in the test set. As the results show, the two proposed approaches in this paper outperforms all other models. The sequence-to-sequence approach gives the highest accuracy for Turkish social media normalization.

We also test with different values of  $n$  to select the nearest neighbours (see Section III-A) of each word in the contextualized normalization approach while generating the lexicon.

TABLE IV  
TURKISH NEWSPAPER CORPUS DETAILS [12]

NewsCor Corpus	Words	Tokens	Types
	184M	212M	2.2M

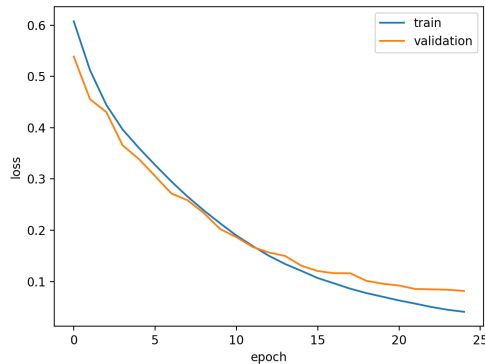


Fig. 5. EDM Train/Validation loss function over epochs

Results for  $n = 25$  and  $n = 100$  are given in Table VI. 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. 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%. 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. 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.

Sample normalization outputs of the two approaches are given in Table VII and Table VIII. The results show that the contextualized normalization approach is better at finding the repetitive letters such as *dersleriüi-dersleri*, whereas the sequence-to-sequence model is better at correcting the abbreviations such as *gsye-Galatasaray'a* that require more edit operations.

In the results, it is observed that for some cases, the normalization process leads to similar errors. In the Contextual Normalization approach, abbreviation normalization is one of the frequent erroneous results. The reason is possibly that the

TABLE V  
NORMALIZATION RESULTS

Model	Accuracy (%)
MsWord	25
Zemberek	21
Lookup Table [10]	34
Rule-based Cascaded Approach [10]	71
Contextual Normalization Approach	<b>72.18</b>
Sequence-to-sequence Normalization Approach	<b>74.80</b>

TABLE VI  
CONTEXTUAL NORMALIZATION RESULTS WITH PARAMETERS

KNN	Consonant Skeleton	Accuracy(%)
n=25	word-wide	39.21
	first 5 characters	39.21
	first 3 consonants	<b>49.48</b>
n=100	word-wide	72.14
	first 5 characters	72.15
	first 3 consonants	<b>72.18</b>

relationship between the abbreviations and their expansions cannot be adequately represented by lexical or cosine similarity.

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

## V. CONCLUSION AND FUTURE WORK

In this paper, 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 learn the canonical form of a given noisy word. The results show that the proposed approach outperforms other models although the proposed model in this paper is fully unsupervised. The latter method adopts an encoder-decoder model with a bi-LSTM architecture. The model does not use contextual information, but instead it learns the error patterns via a large number of noisy-canonical word pairs. The results show that sequence-to-sequence learning gives the highest scores for the normalization task compared to other models.

We aim to experiment on other languages in the future since both models are language independent. Including more contextual information by performing sentence-level normalization in the encoder-decoder model is also left as a future goal.

## REFERENCES

- [1] J. R. Firth, *A synopsis of linguistic theory 1930-1955*. Oxford: Blackwell, 1957.
- [2] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of ICLR Workshop*, 2013.
- [3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

TABLE VII  
EXAMPLE NORMALIZATION OUTPUTS OF THE CONTEXTUAL NORMALIZATION

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

TABLE VIII  
EXAMPLE NORMALIZATION OUTPUTS OF THE SEQUENCE-TO-SEQUENCE NORMALIZATION

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

- [4] H. Hassan and A. 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)*, vol. 1, 2013, pp. 1577–1586.
- [5] C. Sonmez and A. Ozgur, "A graph-based approach for contextual text normalization," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 313–324.
- [6] V. K. R. 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*, 2015, pp. 8–16.
- [7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, Nov. 2011.
- [8] Y. Yang and J. Eisenstein, "A log-linear model for unsupervised text normalization," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 61–72.
- [9] T. Ikeda, H. Shindo, and Y. Matsumoto, "Japanese text normalization with encoder-decoder model," in *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, 2016, pp. 129–137.
- [10] D. Torunoğlu and G. Eryiğit, "A cascaded approach for social media text normalization of turkish," in *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, 2014, pp. 62–70.
- [11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [12] H. Sak, T. Güngör, and M. Saraçlar, "Turkish language resources: Morphological parser, morphological disambiguator and web corpus," in *Advances in natural language processing*. Springer, 2008, pp. 417–427.
- [13] I. D. Melamed, "Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons," *arXiv preprint cmp-lg/9505044*, 1995.
- [14] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
- [15] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.
- [16] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
- [17] A. A. Akin and M. D. Akin, "Zemberek, an open source nlp framework for turkic languages," *Structure*, vol. 10, pp. 1–5, 2007.