Engineering Science and Technology, an International Journal 35 (2022) 101192

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Graph-based Turkish text normalization and its impact on noisy text processing

Seniz Demir [a,*], Berkay Topcu [b]

[a] Department of Computer Engineering, MEF University, Istanbul, Turkey
[b] Artificial Intelligence and Analytics Solutions, Turkcell Technology, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 5 August 2021
Revised 5 May 2022
Accepted 31 May 2022
Available online 23 June 2022

Keywords:
Text normalization
Turkish
Graph-based representation
Noisy text

ABSTRACT

User generated texts on the web are freely-available and lucrative sources of data for language technology researchers. Unfortunately, these texts are often dominated by informal writing styles and the language used in user generated content poses processing difficulties for natural language tools. Experienced performance drops and processing issues can be addressed either by adapting language tools to user generated content or by normalizing noisy texts before being processed. In this article, we propose a Turkish text normalizer that maps non-standard words to their appropriate standard forms using a graph-based methodology and a context-tailoring approach. Our normalizer benefits from both contextual and lexical similarities between normalization pairs as identified by a graph-based subnormalizer and a transformation-based subnormalizer. The performance of our normalizer is demonstrated on a tweet dataset in the most comprehensive intrinsic and extrinsic evaluations reported so far for Turkish. In this article, we present the first graph-based solution to Turkish text normalization with a novel context-tailoring approach, which advances the state-of-the-art results by outperforming other publicly available normalizers. For the first time in the literature, we measure the extent to which the accuracy of a Turkish language processing tool is affected by normalizing noisy texts before being processed. An analysis of these extrinsic evaluations that focus on more than one Turkish NLP task (i.e., part-of-speech tagger and dependency parser) reveals that Turkish language tools are not robust to noisy texts and a normalizer leads to remarkable performance improvements once used as a preprocessing tool in this morphologically-rich language.

1. Introduction

Recent years have witnessed a phenomenal growth of user generated data coming from a wide spectrum of web sources. Many people have been sharing their thoughts and feelings over different platforms more frequently than ever and information of varying modalities has been flowing over the web every day. The evolution of interpersonal communication with the advent of social media and online forums [25] contributes to that rapid volume growth. Soon after, this web content has received much attention by language researchers who recognize the potential of harvesting user generated texts for various commercial applications such as opinion mining [57,75], marketing intelligence [71,3], and language identification [68]. However, for language processing tasks that rely on well structured texts and predictable language constructs, this user generated content is often unsuitable as data. The newly shaped communication preference of millions of users has elicited the birth of a new social language which does not comfort to spelling and grammar rules of standard written languages. For instance, user generated noisy texts often contain initialisms (e.g., 'AMA' for 'ask me anything'), phonetic substitutions (e.g., '4' for 'for'), and non-standard words (e.g.,'comin' for 'coming'). Many factors affect the nature of this non-standard language and the way in which it is changing, including the characteristics of social media platforms, the age, gender, and educational level of users, and social influences between groups of users [5,23,30].

User generated content has opened a new research avenue for speech and language processing communities, namely handling unstructured and noisy real texts [73]. The variety at different linguistics levels from morphological and grammatical formations to semantics of conversations in user generated texts makes analysis of data with traditional techniques more challenging and use of previous language tools (e.g., named entity recognizers [66,48], text to speech synthesizers [63], dependency parsers [27,79], and

* Corresponding author.
E-mail addresses: demirse@mef.edu.tr (S. Demir), berkay.topcu@turkcell.com.tr (B. Topcu).

https://doi.org/10.1016/j.jestch.2022.101192
2215-0986/

part-of-speech taggers [31,61]) less effective. For instance, consider the Turkish sentence "iii geceler allah rahatlik wersin". Once all non-standard words (underlined) are converted to their standard forms (i.e., "İyi geceler, Allah rahatlık versin."), the sentence can be translated into English as "*Good night, God gives comfort.*". Unfortunately, popular translators Google Translate and Microsoft Bing Translator, cannot properly handle this sentence and translate it as "*iii nights god peace wersin*" and "*iii Nights god relax*". Two particular approaches are widely used to address the adherence of language tools to well structured texts [69]: i) the adaptation of a tool to user generated content by retraining with clean and noisy texts [31,44] and ii) the correction of noisy texts (i.e., normalization) before being processed despite the loss of some valuable information (e.g., style and phonological variation for sociolinguistic analysis [49]).

Previous normalization research has addressed the task using different unsupervised and supervised approaches including noisy channel [16,8] and machine translation [64,52] models. Rule-based and graph-based [40,72] approaches have also been proposed as unsupervised solutions to the problem. Recent research efforts have shifted towards capturing contextual word similarities using embeddings and applying various deep-learning normalization models to user generated content. Recurrent neural networks [80] and sequence to sequence networks [41] were some prominent architectures explored in this direction. Various natural languages have been in focus of earlier studies (e.g., English [51], Dutch [14], and Slovene [52]), but the effectiveness of proposed solutions varied significantly from one language to another. Unfortunately, Turkish normalization research is not yet mature enough with a limited number of studies [24,36,62], and the lack of common evaluation benchmarks and publicly available resources are two obstacles that slow down the advancements in the field.

This article presents our research work on normalizing Turkish noisy texts where we address lexical normalization by substituting each non-standard word token in a sentence with its standard rendering. We define a word token as non-standard if it does not have a dictionary entry or its surface form cannot be obtained by following Turkish morphotactics and ortographic realizations. These non-standard words are often intentionally generated for various reasons (e.g., the length limit in Twitter or general conventions in SMS messages) and normalizing them is far beyond what spell checkers can perform. All sorts of missing (bi→bir*(one)*) or repetitive characters (güzellllllll→güzel*(beautiful)*), phonetic substitutions (capon→japon*(Japanese)*), typos/misspellings (e.g., geliyroum→geliyorum*(I am coming)*), single word abbreviations (tmm→tamam*(ok)*), and slang words (efso→efsane*(legend)*) are addressed by this work[1]. We consider only one-to-one word transformations [50] (e.g., poroje→proje *(project)*) and leave one-to-many (e.g., nzm→ne zaman *(when)*) transformations untouched during normalization.

Turkish is an agglutinative language and its productive inflectional and derivational morphology enables the generation of hundreds of distinct surface forms from the same word stem [37]. A large dictionary due to the proliferation of surface forms and typically long sequences of morphemes make it unfeasable to use a lookup table for detecting non-standard Turkish words in user generated texts. Moreover, due to the richness in the surface forms of Turkish words, multiple standard words might appear as possible normalization candidates for a single ill-formed word. This inevitably results in the need for a sophisticated selection approach. In addition, some language-specific issues make developing an unsupervised normalization solution very challenging. For instance, omitting diacritic signs from words is a normalization issue frequently observed in user generated Turkish texts. Using a deasciifier might not solve all issues since some Turkish words only differ in terms of diacritic signs (e.g., asıl (*principal*) ↔ asil (*noble*), koy (*put*) ↔ köy (*village*)). Therefore, the semantic contribution of a word to the sentence should be examined in order to determine whether the word is written by omitting diacritic signs and hence should be normalized. Finally, Turkish is a low-resourced language and the lack of large-scale annotated datasets hinders rapid development of supervised normalization studies despite immense practical needs. All these factors thus lead to a rather complex normalization task in Turkish.

In order to develop a generalizable unsupervised Turkish normalizer, we follow a methodology that was successfully practiced before, where non-standard words in noisy texts are first identified and then restored to appropriate standard words [39,26]. Our normalizer benefits from a morphological analyzer and a named entity recognizer in identifying non-standard words. Our work has a modular design consisting of different subnormalizers, each with its own focus of interest. Lists of normalization candidates produced by these subnormalizers are merged into a single list where candidate corrections are sorted according to a similarity score. Our work then uses the standard word with the highest similarity score for normalization. We argue that contextual and lexical similarities between a non-standard word and its appropriate correction are of great importance for normalization and thus benefit from these similarities in selecting the candidate correction that yields the highest similarity. A bipartite graph acquired from a corpus of clean and noisy texts is used to model contextual appearances of words, and random walks are performed over this graph in order to determine standard words that appear in similar contexts with the word to be normalized. Our approach utilizes a context tailoring methodology in cases where a complete context match is not found while exploring the graph. In order to identify lexically similar candidates, character-based edit operations are applied to non-standard words as our best effort to capture lexical variations in a very broad range.

The representational power of graphs has been widely used in language processing research for many years [56]. Several successful studies have effectively explored similarities and revealed regularities in the application domain by utilizing nodes and edges to represent concepts and entities. Graphs have been adapted to various problems such as plagiarism detection [60], text summarization [6], word sense disambiguation [45], and text normalization [40]. However, graph-based solutions are often costly in terms of computation time, complexity, and scalability. In order to address normalization over our bipartite graph in a computationally acceptable fashion, we use our context tailoring approach to select the starting node(s) of random walks, benefit from contextual similarities in order to restrict transitions between nodes, and put some restrictions on random walks (e.g., the number of maximum steps that can be taken).

The contributions of this work, which is strongly improved over our earlier work [19] in several directions, can be summarized as:

- We propose a graph-based normalization methodology that melds contextual and lexical similarities between non-standard and standard words using a novel context-tailoring approach.
- We achieve the state of the art performance (79.40% f-measure and 95.73% accuracy) on a test corpus of 1430 Turkish tweets as compared to other reference Turkish normalizers and provide an in-depth performance analysis of our normalizer over this corpus.

[1] In all of the examples, a non-standard word (left) and its standard form (right) are separated by the → symbol.

- We conduct the first-ever extrinsic evaluations on Turkish normalization by comparing the outputs of two language processing tools (i.e., a part-of-speech tagger and a dependency parser) before and after normalization. We report an f-measure improvement of 19.25% in part-of-speech tagging and 33.07% in dependency parsing which reveals that normalization eliminates the need for application targeted retraining up to some degree.

The rest of this article is organized as follows. 2 discusses related work on text normalization. 3 describes our normalizer in detail and 4 presents our intrinsic and extrinsic experiments. Finally, 5 concludes the article.

2. Related Work

Some pioneering normalization works defined the problem as a noisy channel model, where a standard word is assumed to be corrupted while being transferred through a noisy channel [42,8]. In this model, conditional probabilities of all candidate corrections given the non-standard word are computed and the one with the highest probability is selected for correction. A Hidden Markov Model (HMM) was proposed by Choudhury et al. [12] in order to characterize the noisy channel where SMS messages are distorted with intentional shortenings and unintentional typos. In another noisy channel approach [16], a mixture model was used to capture all observed word formations in SMS messages. Linguistic observations specific to each word formation category (e.g., stylistic variations and prefix clippings) were modeled explicitly by a distinct model and these models were combined in a Naive Bayes formulation. A log-linear model [76] which allows arbitrary features was applied to characterize relations between standard and non-standard words in an unsupervised noisy channel approach. Feature weights were trained by sequential Monte Carlo in a maximum-likelihood framework in order to overcome large label space, and the local context was handled via a language model.

Previous research also handled normalization as a machine translation problem from non-standard to standard words [64]. The work presented in [2] used a phrase-based statistical machine translation model to normalize English SMS texts at the token level, whereas the work on normalizing Slovene tweets [52] used a character-level statistical translation system. The combination of token level and character level translations in a cascaded framework [14] was shown to be successful in normalizing Dutch user generated content. Despite their effectiveness, machine translation approaches require large amount of parallel training data which is hard and time-consuming to gather (if not readily available) from constantly evolving user generated content.

Some previous works handled the normalization task as a two-step process. In the first step, standard word candidates for a non-standard word were identified in order to build a confusion set and the best candidate from this set was selected in the following decoding step. Both in candidate selection and decoding steps, different approaches were explored. In one of the earlier works [35], the IBM-similarity function, where the longest common subsequence between two strings are normalized by the edit distance of their consonant skeletons, was utilized to select candidate words from a dictionary. In the decoding step, an n-gram language model was used to find the most probable path over the confusion network built for the sentence that contains non-standard words. A cognitively-driven normalization system incorporated letter transformation [51], visual priming, and string/phonetic similarity approaches in order to obtain the set of candidate corrections for a non-standard word [50]. The best candidate was selected by using local contexts in Viterbi decoding. Another work [39] benefited from character and phonemic edit distances to select candidate words from a dictionary and ranked them based on a trigram language model. All n-gram contexts obtained from a large Web corpus were used to select the best candidate from the identified set by matching contexts of most-frequent n-grams with the non-standard word's left and right contexts.

A significant amount of research focused on word representations to capture domain-specific contextual similarities between non-standard and standard words [17]. These representations were induced from linear or non-linear embeddings (e.g., neural networks) [46]. As a consequence of the success of deep learning in several language tasks, recurrent neural networks [80,13], convolutional neural networks [78], and sequence to sequence learning [41] were applied to text normalization. In a recent work [55], one of the most popular contextual language models, namely BERT, was shown to be effective in normalizing user-generated noisy texts. The normalization task was defined as a token mapping between non-standard and standard words, and for this purpose two new word piece alignment methods were developed. Moreover, the language model was trained with noisy texts (only 3000 sentences) and then fine tuned with their standard forms. Evaluation studies revealed that the system achieved competitive results as compared to other studies that rely on large-scale datasets or external modules such as normalization lexicons.

Our work is closely related to the line of research that utilizes graphs for capturing contextual similarities between non-standard and standard words. In [72], contextual similarities were modeled via a word-association graph and their lexical similarities were measured via longest common subsequence ratio and edit distance metrics. Our work differs from this work in various aspects such as not using part-of-speech information in identifying contextual similarities and processing all words within a sentence at the same time. We benefited from the graph representation proposed in [40], where words and their contexts (defined in terms of n-grams) were represented as nodes in different bipartites. In that work, random walks were performed over the graph to induce a normalization lexicon, which was later used to generate word confusion sets for non-standard words during normalization. The main drawback of the approach is that it generates the same confusion set for a word no matter in which context it appears. Moreover, contexts that differ only in one word (even with the same part-of-speech) are treated as if they are two dissimilar contexts without a single common word. Four key features make our normalizer unique: i) a context tailoring approach for deciding from where to start random walks, ii) different contextual and lexical similarity measures, iii) no limit on the space of extracted n-grams, and iv) an on-the-fly normalization rather than word lattices.

Only a few recent studies focused on Turkish noisy text normalization. The cascaded approach [24] used seven normalization layers (e.g., vowel restoration and accent normalization) to generate a single candidate for a non-standard word, where each layer handles a particular kind of normalization edit type seen in noisy texts. The system incorporated rule-based techniques, machine learning approaches, and statistical models, and was evaluated on different datasets. Similarly, a rule-based normalization approach was used in a recent work [43] where recurrent character, deasciification, and deaccenting problems are mainly targeted. Distributed representations of words were utilized in [36] to capture contextual similarities between words, and these similarities then formed the basis of a normalization lexicon induced from a collection of news. Words in a noisy text were normalized according to this lexicon via a Viterbi decoder. Moreover, an encoder-decoder recurrent neural network model was trained on a Twitter dataset in order to induce error types automatically and to normalize noisy texts given as input. In another work [77], a variety of techniques (e.g., lexical similarity and language model based contextual similarity)

was used to handle different Turkish normalization issues. One recent work focused only on diacritic restoration as the normalization edit type [62]. Given a non-standard word, correction candidates were produced via a decision list and then sorted according to a similarity score computed over distributed word representations. A machine translation based approach was also explored in Turkish normalization [15]. Non-standard words were initially normalized using orthographic character replacement rules, and then translated into standard forms via a character level MT system (both statistical MT and neural MT). Finally, these words were restored to proper letter cases via a recaser. Unfortunately, the current state of the art is still far from providing a clear picture of the problem and in need of many additional results since performances of these Turkish specific normalization systems were evaluated on a diverse set of test corpora (mostly tweet corpora). Our approach significantly differs from these research efforts in two ways. First, our work has the ability to produce more than one normalization candidate for a non-standard word, and hence can also be used as a spell checker depending on user needs. Second, our work can normalize a non-standard word by relying totally on contextual features or lexical features of the word. The decision of which features should be used and how they should be weighted relative to each other is left to users.

3. Turkish Text Normalizer

Our work separates normalization task into three consecutive steps [39,18]: i) the detection of non-standard words, ii) the generation of candidate corrections for each identified non-standard word, and iii) the selection of best candidates for normalization.

3.1. Non-standard Word Detection

For deploying a non-standard word detection method, fully relying on external resources like well-constructed word lexicons or tree structures might be a good choice for languages with less complicated word formations such as English [40] and Malay [67]. However, building such resources with wide coverage is not feasible for the morphologically rich language Turkish. Moreover, it is also challenging to distinguish non-standard words that should be left unnormalized such as named entities [39]. Inline with previous research [74,62], we first preprocess and tokenize a given sentence into word tokens, and then send these tokens to a Turkish morphological analyzer. All words that cannot be decomposed into a sequence of morphemes by the analyzer are identified as non-standard. The morphological analyzer explores all possible ways of forming a sequence that consists of a word stem from its lexicon and suffixes as guided by morphotactics and ortographic rules. If a word is not ill-formed, the analyzer returns at least one possible morpheme sequence. From among all words that are identified as non-standard, we eliminate those that are recognized as proper nouns by a Turkish named entity recognizer. Although applying this approach to noisy texts is not error-free (e.g., some proper nouns are identified as non-standard words), it is more applicable to Turkish than a standard lexicon lookup approach.

3.2. Candidate Word Generation

At the core of our candidate generation approach lies three general characteristic relations between a non-standard word and its normalization equivalence. We develop a graph-based subnormalizer to capture the first two relations and a transformation-based subnormalizer to capture the third relation given below:

1. *Shared Contexts:* A non-standard word, no matter in which context it is used, can be substituted by its correct normalized form without any loss in meaning or any degrade in integrity. Thus, contexts shared by non-standard words and their normalization equivalences are important clues that should be explored. For instance, substituting non-standard words ‘dogru’ and ‘dormak’ by their well-formed corrections in the following sentence (*Need to ask the right question to get the right answer*) does not cause any meaning loss:

   $\begin{bmatrix}\text{dogru}\\ \downarrow\\ \text{doğru}\end{bmatrix}$ cevap almak için

   $\begin{bmatrix}\text{dogru}\\ \downarrow\\ \text{doğru}\end{bmatrix}$ soruyu

   $\begin{bmatrix}\text{dormak}\\ \downarrow\\ \text{sormak}\end{bmatrix}$ lazım

2. *Contexts of Non-Standard Words:* A non-standard word might be replaced with different standard words depending on its context. Thus, the context of a non-standard word is important and a “one-fits-all” lexicon lookup approach is not generally applicable. For instance, the non-standard word ‘aaba’ in the first sentence (*I wonder if my mother will also like my school?*) is normalized to the word ‘acaba’ *(I wonder if)* whereas it is normalized as ‘araba’ *(car)* in the second sentence *(Driving car is a great pleasure for me)*:

   $\begin{bmatrix}\text{aaba}\\ \downarrow\\ \text{acaba}\end{bmatrix}$ annem de okulumu sevecek mi?.

   $\begin{bmatrix}\text{aaba}\\ \downarrow\\ \text{araba}\end{bmatrix}$ sürmek benim için büyük keyif

3. *Character-Based Transformations:* Non-standard words and their invocabulary forms are lexically or phonetically similar. Therefore, character-based transformations (i.e., insertion, deletion, or substitution) can be applied to a word in order to produce its non-standard forms. These edit operations might be phonetic, graphemic, typographic, etc. [50]. For instance, the non-standard word ‘ailelerne’ can be produced from two different standard words as follows:

   ailelere *(to families)* $\overset{\text{inserting'n'}}{\Rightarrow}$ ailelerne

   ailelerine *(to their families)* $\overset{\text{deleting'i'}}{\Rightarrow}$ ailelerne

3.2.1. Graph-based Subnormalizer

Our graph-based subnormalizer extracts words and their contexts from an unannotated raw text collection (both clean and noisy texts) and builds a bipartite graph (**Corpora_Graph**) that reflects their co-occurrences in the collection[2]. All n-gram word sequences (might contain zero or more non-standard words) are extracted from the collection, and all words at the center of these sequences are represented as **word nodes** in the first bipartite and their filtered contexts (obtained by removing the center word) are represented as **context nodes** in the second bipartite. A word node and a context node are connected with an undirected edge if the word appears in that context and the edge weight represents the co-occurrence count of the pair. For instance, 1 shows a Corpora_Graph constructed from a set of 5-gram sequences. The word nodes Word_2, Word_4, and Word_5 represent non-standard Turkish words whereas nodes Word_1 and Word_3 represent standard words. A context node might be connected to more than one word node (e.g., Cntx_1) and such contexts are “shared contexts” between different words. To guarantee that every word has a context no matter

[2] This graph is built only once before being used for normalization.

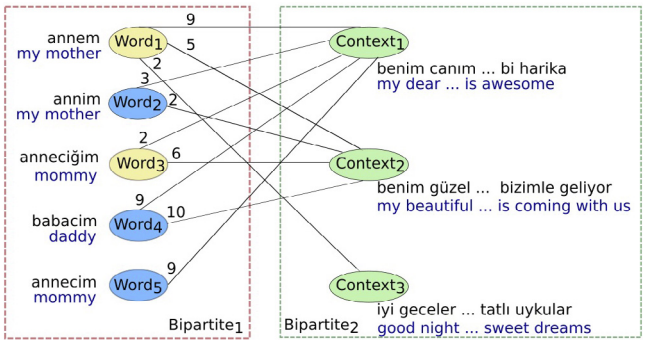


Fig. 1. Corpora_Graph built from a set of 5-gram word sequences.

where it appears in the sentence, we introduce the symbol ∅ to represent missing words. For instance, consider the sentence "bugünn oğlumu gösterisi için hazırlıyıp fotoğraflarını çekeceğim" *(Today, I will prepare my son for his show and take his photos)*. Since no words precede the first word ('bugünn'), missing word symbols are used to represent this fact in its filtered context: [∅, ∅,oğlumu, gösterisi] ([∅, ∅,*my son*, *his show*]).

We perform a number of random walks over the graph in order to find standard words that are contextually similar to a non-standard word. The node from where a random walk starts **(starting node)** is of great importance in a dense graph with so many nodes and connections if the number of steps that can be taken is limited. It is straightforward to select the node representing non-standard word as the starting node (if exists). However, this approach would fail in utilizing valuable information about the word context. As one of our novelties, we follow a comprehensive selection approach to decide where to start random walks and how to initially select the first transition from a starting node:

- If the non-standard word appears in the graph, its connected context nodes are identified. If any of these neighbours is the same with the filtered context of the word, that context node forms the only starting node. Otherwise, the context of the non-standard word is tailored in turns until similar contexts are found from among neighbour context nodes. At each turn, all possible contexts obtained by relaxing the restriction put by a single word in tailored contexts of the previous turn are used. The context nodes that are found to be similar form the set of starting nodes. However, if no context is found to be similar until the tailoring threshold is reached, the word node corresponding to the non-standard word is identified as the only starting node.
- If the word does not appear in the graph, its filtered context is compared to all contexts in the second bipartite. If the same context is found, that node is identified as the starting node. Otherwise, context nodes that are found to be similar to tailored contexts form the set of starting nodes.

In this work, tailoring corresponds to compromising from context content by relaxing constraints put by contained words. The relaxation of a word constraint is achieved by replacing the word with the symbol ∅, which in turn can match to any word (standard or non-standard). For instance, assume that 2 shows some context nodes (Contexts (a)-(e)) of a graph and the filtered context of a non-standard word is Context (1). This filtered context does not directly match to any of the contexts in the graph. However, if we tailor this context by relaxing restrictions put by a single word, Contexts (2)-(5) would be obtained and these tailored contexts would be found similar to Contexts (a), (b), (c), and (e). Moreover,

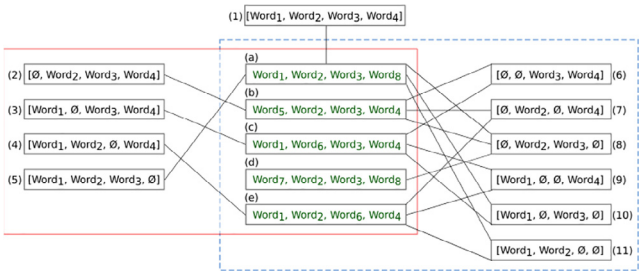


Fig. 2. Similar contexts.

tailoring the filtered context by relaxing any two words would produce less restricted contexts (Contexts (6)-(11)) which are similar to a larger set of contexts (Contexts (a)-(e)) in the graph. Context tailoring enables us to explore a larger number of similar contexts, but the number of relaxed words reduces their similarities to the filtered context. Nonetheless, exploring contexts that share only a single word with the filtered context (the upper limit of word relaxation) still poses a contextual restriction and guides the selection of a proper starting node for walks. Moreover, the number of relaxed words is limited to eliminate obtaining contexts with a similarity below a certain threshold.

Our approach identifies either no starting node[3] or selects a single word node or at least one context node as the starting node(s). For instance, 1 presents glosses of non-standard words with their contexts in the first column and lists the matching filtered or tailored contexts of these words in the second column along with the identified starting node(s) when the graph in 1 is used. Our approach performs a fixed number of random walks from each starting node, where each walk consists of the same number of sequential steps taken from one bipartite to another. A walk ends when a standard word is reached or the maximum number of steps is exhausted without reaching a standard word. All reached standard words within a predetermined edit distance to the non-standard word are treated as contextually similar. In cases where there are more than one possible transition from a node, we benefit from edge weights of the bipartite graph in order to favour transitions between words and contexts that co-occur frequently. We first compute transition probabilities from a node to all of its neighbours. From among the nodes with the highest computed probability, we then randomly select a node to visit next. The transition probability (TP_{ab}) and transition frequency ($TFreq_{ab}$) from node a to node b are computed as follows:

$$\text{(i) } TP_{ab} = \frac{TFreq_{ab}}{\sum_{x \in Neighs(a)} TFreq_{ax}}$$

$$\text{(ii) } TFreq_{ab} = Edge\_weight_{ab} \text{ (word node } a \Rightarrow \text{context node } b)$$

$$\text{(iii) } TFreq_{ab} = \frac{Edge\_weight_{ab}}{Freq(b)} \text{ (context node } a \Rightarrow \text{word node } b)$$

where $Edge\_weight_{ab}$ corresponds to the weight of the edge connecting nodes a and b. We penalize transitions to a common word by taking into account its occurrence frequency ($Freq(b)$) in the collection from which the graph is built. However, no penalty is applied to transitions to contexts shared by many words. By dividing $TFreq_{ab}$ to the summation of all transition frequencies from node a to its neighbours, we restrict TP_{ab} to the range [0,1]. Moreover, the degree of connections that each node has and weights associated with these connections significantly effect transition probability calculations. Transition probabilities[4] computed over the graph shown in 1 are listed in 2 (from word to context nodes)

[3] In such cases, no walk is performed.
[4] S1 and S2 are $\sum$ of $TFreq_{1x}$ and $\sum$ of $TFreq_{2x}$, respectively.

Table 1
Starting node examples.

Non-Standard Word&Cntx	Matching Cntx → Starting Node(s)
"benim canım *annim* bi harika"	[benim, canım, bi, harika]→ $Cntx_1$
"benim cefakar *annim* seni seviyorum"	[benim, ∅, ∅, ∅] → $Cntx_1$, $Cntx_2$
"dün yine *babacim* konuşmaya geldi"	None → $Word_4$
"sonunda güzel *kuziiiinimmmm* yemeklerle geliyor"	[∅, güzel, ∅, geliyor] → $Cntx_2$

Table 2
Transition probabilities from word to context nodes.

	$Cntx_1$	$Cntx_2$	$Cntx_3$
$Word_1$	9/16	5/16	2/16
$Word_2$	3/5	2/5	0
$Word_3$	2/8	6/8	0
$Word_4$	9/19	10/19	0
$Word_5$	1	0	0

and 3 (from context to word nodes). According to these probabilities, the following are possible random walks of at most 4 steps[5]:

- $Word_2 \Rightarrow Cntx_1 \Rightarrow Word_1$ (**two steps**)
- $Word_2 \Rightarrow Cntx_1 \Rightarrow Word_4 \Rightarrow Cntx_2 \Rightarrow Word_3$ (**four steps**)
- $Word_4 \Rightarrow Cntx_2 \Rightarrow Word_2 \Rightarrow Cntx_1 \Rightarrow Word_5$ (**four steps**)

Here, the first and second walks end at nodes $Word_1$ and $Word_3$, respectively. If any of them is performed, either the standard word 'annem' or the standard word 'anneciğim' is identified as contextually similar to the non-standard word 'annim'. However, the third walk ends at the non-standard word node $Word_5$ since the maximum number of steps is taken without reaching a standard word. In our work, the maximum number of steps in a random walk is predetermined. All walks starting from a context node have one fewer steps to guarantee a termination at a word node.

3.2.2. Transformation-based Subnormalizer

One important observation is that there are some commonalities (e.g., phonetic and typographic) between non-standard words and their standard forms. Therefore, a non-standard word can be obtained by applying a number of edit operations (e.g., insertion or substitution) to the surface form of a standard word and the local context is not of considerable utility in modeling these lexical and stylistic variations at the character-level [50,51]. We argue that producing a list of lexically similar correction candidates is a great addition to identifying contextually similar words in normalization and hence follow a language-specific character-based transformation methodology to identify standard words that might be the original form of a non-standard word. Given a lexically noisy variant, our work produces a list of lexically similar standard words in four steps. At each step, a different transformation is independently applied to an ill-formed word and a list of normalization candidates are obtained. These lists are then merged into a single list of lexically similar candidates.

Although some characters should be written with diacritics (e.g., ğ and ç) in Turkish, the tendency of using ascii characters while writing on the web results in omitted diacritic signs. In order to eliminate the problematic use of such characters, we apply diacritics reconstruction (deasciification) to words consisting of only ascii characters [1] by using a Turkish deasciifier[6]. The deasciifier does not change the word formation but rather performs some substitutions from the set {i→ı, o→ö, u→ü, c→ç, g→ğ, s→ş, I→İ, O→Ö, U→Ü, C→Ç, G→Ğ, S→Ş} if directed by the language model. Unfortunately, deasciification has inherent ambiguity issues and is not free of errors since different standard Turkish words might differ exactly in one diacritic sign such as asıl (*principal*) ↔ asil (*noble*) and çam (*pine*) ↔ cam (*glass*). As a first step, we identify newly produced standard word with non-ascii characters after deasciification as lexically similar to the non-standard word.

Typing characters of a word in the wrong order is a typographic error frequently observed in noisy texts (e.g., 'sednen' instead of 'senden' *(from you)*). In the second step, we generate all variants of a non-standard word by swapping each character with its neighbour characters (i.e., one and two next/previous characters) and filter out non-standard words from among all generated words using a morphological analyzer. The remaining words are then added to the list of lexically similar words.

Turkish users on online platforms are prone to write some verb patterns that they are accustomed to use while speaking spontaneously. In order to correct the improper use of such suffixes (e.g., tense and person suffixes), we define a number of substitution rules to cover different patterns that we observe in user generated texts. In the third step, we apply these rules to produce the standard form of a non-standard verb if it matches one of the considered patterns. The generated word is finally added to the list of lexically similar words. 4 presents some verb examples before and after applying the given substitution rules.

Finally, we address other kinds of misspellings by utilizing a word suggester which can produce corrections for a non-standard word. For this purpose, we use Zemberek NLP toolkit[7] which offers several features such as tokenization, morphological analysis, normalization, and spell checking (word suggestion). We specify an edit distance and add all suggestions within this distance to the non-standard word to the list of lexically similar words. For instance, for the non-standard word 'etmiyon', the word suggester produces several standard words including 'etmiyor[1], eğmiyor[2], etmeyin[2], yetmiyor[2]' once the edit distance is set to two[8].

3.3. Candidate Word Selection

Although any candidate word produced by a subnormalizer can be used as a standalone correction, we meld contextually similar and lexically similar candidate words into a single list in order to bring different perspectives together. If no candidate correction is produced for a non-standard word, that word is left as is in the text. If there is only one candidate correction for a non-standard word, using that standard word is the only option. However, in cases with more than one candidate, we first compute a similarity score (Sim_{ab}) between the non-standard word *a* and each of its candidate corrections [72] by weighting contextual and lexical similarities of these words (both in the range of [0–1]):

(i) $Sim_{ab} = \lambda_1 \times CntxSim_{ab} + \lambda_2 \times LexSim_{ab}$

(ii) $CntxSim_{ab} = \frac{\min_{\forall x \in X} HT_{ax}}{HT_{ab}}$

(iii) $LexSim_{ab} = EditScr_{ab}$

where X is the set of standard word nodes reached during random walks. To assign a contextual similarity score to a candidate word *b*, hitting time (HT_{ab}) is used. In a random walk, hitting time refers to the minimum number of steps taken from a starting node of the word *a* to the ending node that represents the word *b*. Since our approach performs more than one walk from the same starting node, an ending node might be reached more than once. In these

[5] Any of these walks might be possible in a real scenario depending on actual word frequencies in a collected corpora.
[6] https://github.com/ahmetb/turkish-deasciifier-java
[7] https://github.com/ahmetaa/zemberek-nlp
[8] Edit distances are shown as superscripts.

Table 3
Transition probabilities from context to word nodes.

	$\mathbf{Word}_1$	$\mathbf{Word}_2$	$\mathbf{Word}_3$	$\mathbf{Word}_4$	$\mathbf{Word}_5$
$\mathbf{Cntx}_1$	$\frac{9}{Freq(1)*\ S1}$	$\frac{3}{Freq(2)*\ S1}$	$\frac{2}{Freq(3)*\ S1}$	$\frac{9}{Freq(4)*\ S1}$	$\frac{9}{Freq(5)*\ S1}$
$\mathbf{Cntx}_2$	$\frac{5}{Freq(1)*\ S2}$	$\frac{2}{Freq(2)*\ S2}$	$\frac{6}{Freq(3)*\ S2}$	$\frac{10}{Freq(4)*\ S2}$	0
$\mathbf{Cntx}_3$	1	0	0	0	0

Table 4
Verb patterns.

Non-Standard Verb	Substitution Rule	Standard Verb
gelicen	...-icen/+eceksin	geleceksin (*You'll come*)
bakıyodur	...-yodur/+yordur	bakıyordur (*Is/Are looking*)

cases, the minimum hitting time is taken. EditScr_{ab} is the editing score between words a and b. A sentence written by using a keyboard with ascii character set cannot contain non-ascii characters and any word in this sentence inherently misses all necessary diacritic signs (all at once). We argue that in such cases, a word should not be penalized for every single missing sign but rather the deasciification step needs to be treated as a single edit operation. Thus, a lexically similar candidate generated by the deasciifier has an editing score of one. The editing score of a candidate produced by swapping characters or by the word suggester equals to one over the number of edit operations between non-standard and standard words. Words generated by substitution rules all have an editing score of one. The same lexically similar candidate might be produced in more than one step. In such cases, the highest computed score is used as its editing score.

Our work selects a candidate word with the highest similarity score as the best correction. However, if more than one such candidate exist, we benefit from sophisticated string similarity measures [32]. We compute the similarity (StrSim_{ab}) of the words a and b using the Longest Common Subsequence Ratio (LCSR_{ab}) [53] and Edit Distance (ED_{ab}) [47] measures:

$$\text{(i) } StrSim_{ab} = \frac{LCSR_{ab}}{ED_{ab}}$$

$$\text{(ii) } LCSR_{ab} = \frac{LCS_{ab}}{\max(Length_a, Length_b)}$$

where LCS_{ab} corresponds to the length of the longest common subsequence between words a and b. In the literature, different edit distance (ED_{ab}) computations are explored (e.g., consonant skeletons of words [40] and reduced character repetitions [81]). Here, we only eliminate character repetitions before computing edit distances. From among the candidates with highest similarity score (Sim_{ab}), the one with the highest string similarity (StrSim_{ab}) is selected for correction. But, if no distinction can be made using string similarities, we randomly pick a correction candidate. Please note that our work is open to integrating new subnormalizers as long as each subnormalizer produces a candidate list along with individual scoring. For instance, one subnormalizer that can possibly be added in the future might generate candidate words that are phonetically similar to a given non-standard word [81]. In addition, our approach is flexible enough to change the similarity computation by incorporating different criterion or weighting scheme[9]. Finally, our work might also be used as a word suggester to produce more than one correction candidate.

[9] If a new subnormalizer is incorporated into the system, its similarity scoring should be added to the weighted formula and the computed score needs to be in the range of [0–1].

4. Experiments

In this work, we performed several intrinsic and extrinsic evaluations to assess relevant aspects of our performance on real data. In order to build our bipartite graph, we compiled a corpus of both clean (∼6 GB of newspaper articles) and noisy texts. Noisy texts consist of tweets collected using Twitter Streaming API from April to October 2015 (∼11 GB), and 20 million Turkish tweets retrieved from a publicly available corpus[10]. We employed an in–house language identifier to select tweets that are written completely in Turkish. Each tweet was passed through a preprocessing step with several language-independent regular expressions for cleaning. For instance, some of these expressions removed tweet specific terms (e.g., hashtags, user mentions, and emoticons), special keywords (e.g., RT and DM) and punctuations from tweets. The remaining 121,466,753 tweets (∼9 GB) were finally tokenized into 1,583,254,116 tokens (with 7,401,321 distinct tokens). We experimented with both 3-gram and 5-gram word sequences that are extracted from this corpus and obtained two graphs with different characteristics as shown in 5. A graph where words are connected to more contexts on average would help us to explore a more diverse set of contexts while tailoring the context of a non-standard word. Additionally, once a step is to be taken from a context node, having less word connections would result in a more discriminative selection (i.e., less frequent context-word pairs). Thus, our normalizer is built on top of the 5-gram graph and performs 300 random walks of at most ten steps from each of the starting nodes. The edit distance used to filter out some corrections produced by random walks (3.2.1) or suggested by the word suggester (3.2.2) equals to two. Finally, tailoring threshold is set to three. Our normalizer uses the morphological analyzer of NUVE NLP Library[11] and a Turkish named entity recognizer [59] in order to detect non-standard words.

In this study, we compared our work with the popular spell checker/corrector MS-Word, the spell checker of Zemberek Toolkit, the publicly available Turkish normalizer (T-Norm) [24], and the machine translation based normalizer (MT-Norm) [15] to show what the current state is. We selected T-Norm since it also divides the normalization task into two subtasks (non-standard word detection and candidate generation) and performs non-standard word detection using a morphological analyzer. We selected MS-Word since it offers non-standard word detection and candidate generation as a bundle. We considered the first spelling suggestion of MS-Word as its best correction. Zemberek was used since it not only analyzes a word morphologically but also determines whether a word is a proper noun or not. However, we only compared our non-standard word detection performance with the Zemberek spell checker since our work uses its word suggester in finding lexically similar words. We finally selected MT-Norm since it does not utilize a separate non-standard word detection mechanism.

We gathered our test corpus by retrieving 715 tweets via streaming API between January-March 2016 and randomly selecting 715 tweets from Tweets Corpus [70]. All tweets were first preprocessed and normalized by two native speakers to obtain gold

[10] http://www.kemik.yildiz.edu.tr/
[11] https://github.com/hrzafer/nuve

Table 5
3-gram and 5-gram bipartite graphs.

Graph	Bipartite	# Nodes	Highest Degree	Avg. Degree	Highest Edge Weight	Avg. Edge Weight
3-gram	Word	6019 K	8637 K	78.09	1094 K	1.86
	Context	237062 K	48 K	1.98		
5-gram	Word	5123 K↓	14445 K↑	114.56↑	58 K↓	1.18↓
	Context	567078 K↑	2547↓	1.03↓		

corrections and their disagreements were resolved by a third native speaker. The corpus consists of 15,697 tokens and 2,880 non-standard words (18.3%). Only one non-standard word was identified in 768 of these sentences (∼53.7%). Multiple non-standard words were found in the remaining 662 sentences (∼46.3%) and 48.8% of the time, consecutive non-standard words appeared in the same sentence. Different error types (i.e., normalization categories) were observed in our test data such as typographical errors (e.g., 'rehberg' ↔ 'rehber'), repetitions (e.g., 'aaagabeey' ↔ 'ağabey'), and phonetical transformations (e.g., 'ueroya' ↔ 'avroya').

4.1. Non-Standard Word Detection

Non-standard word detection, the first step in processing a noisy sentence, has the potential to negatively affect the overall performance due to false positives (i.e., standard words that are wrongly identified as non-standard) and false negatives (i.e., non-standard words that are wrongly identified as standard). To handle the ambiguous nature of Turkish morphology, we use external tools to determine whether a word is standard or not. In this experiment, the non-standard word detection performance, shown in 6, indeed corresponds to the combined performance of these tools on noisy texts. The evaluation of T-Norm is not provided since it is not possible to assess the stand-alone performance of the internal detection module and proper nouns are handled in the candidate generation phase rather than the word detection phase. The precision is computed as the ratio of correctly identified non-standard words to all words that are identified as non-standard whereas the recall is computed as the ratio of correctly identified non-standard words to all correct non-standard words.

From among 2,880 non-standard words identified by human annotators in gold data, our normalizer wrongly identified 190 non-standard words as standard in 144 sentences. In most of these cases, the word is actually a standard word like its correction in gold data. However, once the contextual content of the sentence is taken into account, the word should be substituted by its correction. Our normalizer determined 200 standard words as non-standard in 193 sentences. This is mostly due to proper nouns not detected by the named entity recognizer. Some of these words even exhibit strong indicators of being a proper noun such as separated suffixes from the noun with an apostrophe (Türkiye'den). In a few cases, vocatives (e.g., oouvvv) and abbreviations (e.g., dk. for dakika (*minute*)) are wrongly detected as non-standard words.

Our performance is very close to that of MS-Word but the results demonstrated that our normalizer mainly suffers from false positives and has an almost equal false negatives performance with MS-Word. However, Zemberek had a very high false negatives rate and a low f-measure as compared to our normalizer. Although evaluation results highlighted potential for further improvements, our non-standard word detection results seemed satisfactory. In the future, we plan to develop a more sophisticated method to overcome the morphologically rich nature of Turkish.

4.2. Context-free Normalization

Although our normalizer exploits the context of a non-standard word as much as possible, it can yet generate correction candidates in the absence of local context. This can be achieved by first performing random walks from the node that represents the non-standard word in the bipartite graph (if exists) and then enriching contextually similar candidate list with lexically similar candidates. In our second experiment, we evaluated the effectiveness of our normalizer in such cases. We randomly selected 400 non-standard words from our test corpus and asked two native speakers to normalize these words without knowing the context where they appear. The speakers were told to trust their first intuitions and their disagreements were resolved by asking a third speaker to intuitively choose one of the two proposed corrections. All selected words were then normalized as stand-alone words by our normalizer and the reference systems T-Norm and MS-Word. 7 presents our results where we compute the precision as the ratio of correctly normalized words to all normalized words, and the recall as the ratio of correctly normalized words to all words that require normalization. These results show that our normalizer received the highest scores as compared to other systems.

Since our normalizer might produce more than one candidate correction (if exists), we also computed agreement scores between our top n candidates and gold reference data as shown in 8. The table also presents the agreement scores of our normalizer with the reference systems. The agreement score shows the percentage of cases where the normalization candidate of a reference system or the gold data appears in our top n candidates. We observed that our system produces the gold correction for a non-standard word in 65.25% of the time (261 cases). But once top five candidate corrections produced by our normalizer are considered, the gold correction appears among them in 87.25% of the time (349 cases). A similar increase was observed in comparisons to other systems as well. For instance, the correction candidate produced by T-Norm is what our normalizer also suggests in 40.25% of the time but the suggestion of T-Norm appears in our top five correction suggestions in 59.25% of the time. It should be noted that our agreement with gold data was higher than our agreements to other systems in all top n suggestions. The results demonstrated that even if the right correction of a non-standard word is not the one selected by our normalizer, it might be in top n corrections that it produces. In most of such cases, no contextually similar candidate was found and from among several lexically similar candidates with the same similarity score, one correction was randomly picked. Overall, these findings supported the proposition that our normalizer might be used as an effective Turkish spell checker/word suggester.

If we follow context-free normalization approach for all non-standard words contained in the underlying graph, this exclusive system capability enables us to induce a normalization lexicon from the graph. In our earlier work [20], we leveraged this approach to construct the first publicly available Turkish normalization lexicon. To validate its effectiveness as a look-up table,

Table 6
Non-standard word detection performance.

	False Pos.	False Neg.	Precision	Recall	F-Measure
MS-Word	152	184	94.66	93.61	94.13
Zemberek	89	638	96.18	77.85	86.05
Our Normalizer	*200*	*190*	*93.08*	*93.40*	*93.24*

Table 7
Context-free normalization performance.

	Precision	Recall	F-Measure
MS-Word	57.99	56.25	57.11
T-Norm	52.99	48.75	50.78
Our Normalizer	*67.79*	*65.25*	*66.50*

Table 8
Agreement scores between top n candidate words and reference data.

Top	Gold Correction	MS-Word	T-Norm
1	*65.25*	47.25	40.25
2	*80.25*	62.25	51.25
3	*84.25*	69.25	56.25
4	*86.25*	71.25	58.25
5	*87.25*	75.25	59.25

we also developed a straightforward normalizer where the lexicon is used to build word confusion sets for non-standard words and a Viterbi decoder along with a language model is used to select best correction alternatives from these sets. The following are some entries from that normalization lexicon:

- yaralanının→yaralının(*the wounded*)|yaralanın(*get wounded*)
- goca→loca(*lodge*)|gonca(*bud*)|koca(*husband*)

4.3. Context-aware Normalization

In this work, we benefit from contexts of non-standard words as our best effort to overcome complicated cases. To assess the performance of our normalizer in detecting non-standard words and properly correcting them when local contexts are available, we carried out a series of experiments. As described in 4.1, non-standard word detection might negatively affect the system performance. To measure its impact, we also assessed the stand-alone performance of our candidate generation and selection approaches where gold word detection is applied (i.e., the normalizer is asked to correct only manually identified non-standard words in test data). 9 presents the results of these experiments where normalization performances of the reference systems on test corpus are also given. We computed precision and recall as described in 4.2 using gold corrections, and the accuracy score as the ratio of correctly normalized words and standard words that left untouched to all words in the test set.

Our performance varied according to whether gold word detection is applied (+GWD) or not (-GWD). Our approach for discovering non-standard words (with an f-measure of 93.24%) degraded system performance by approximately 3.1% f-measure and 2.2% accuracy. We also observed that the importance given to contextual and lexical similarities (i.e., λ_1 and λ_2) in selecting best candidate also has an effect on the performance (0.28%-0.35% f-measure and 0.02%-0.01% accuracy). Once both contextual and lexical similarities are of the same importance (EW: Equal weights), the performance was observed to be slightly lower than the optimal result (OW: Optimal weights) that we achieved by changing λ_1 and λ_2 in the range [0,1]. MS-Word had the worst performance on all aspects. Although the spell checker was slightly better than our normalizer in detecting non-standard words (given in 6), its correction performance was markedly degraded on our test corpus. This might be due to the fact that its main design goal is to correct misspellings in formal texts. Once compared to T-Norm, our normalizer achieved remarkably higher results in all metrics, such as an improvement of 7.16% in f-measure and an improvement of 1.41% in accuracy (-GWD+EW). MT-Norm had the closest performance to our normalizer. Although we achieved a higher precision score, MT-Norm which does not use a separate non-standard word detection mechanism normalized more words and achieved a higher recall score than our normalizer (-GWD+EW). However, this was changed in gold word detection scenario (+GWD+EW) where we eliminated the negative impact of missing or incorrect detections. We also performed paired t-test (with $p < 0.05$) to measure the statistical significance of the improvement that we achieved. The results revealed that our normalizer, though open to further developments, significantly improves on the previous state-of-the-art systems (with $p - values < 0.0402$).

Our experimental results given in 10 revealed that neither contextually similar candidates (the first and second rows) nor lexically similar candidates (the third and fourth rows) alone are adequate to achieve the optimal performance. Although our normalizer produced contextually similar candidates in slightly less than half of the cases (42.71–45.17%), most of the time the correct standard word is one of these candidates (85.77–85.37%). On the other hand, our normalizer produced lexically similar candidates in majority of the cases (even in some cases with no contextually similar candidate). But, relying only on lexical similarities resulted in significant performance drops. These results validated that blending contextual similarities (57.02% f-measure) and lexical similarities (70.55% f-measure) together indeed increases our overall system performance (79.40% f-measure).

We also evaluated our performance in terms of whether the correct normalization is in the top n candidates produced by our system or not. As shown in 3a and 3b, the f-measure score was increased if top 2 candidates are considered. In cases without gold word detection, the optimal f-measure score was increased from 79.68% to 85.20%, whereas an f-measure increase from 82.87% to 89.60% was achieved with gold word detection. Although similar increases were obtained with top 3, top 4 and top 5 candidates, the amount of increase after top 3 suggestions was not significant. Finally, we explored individual effects of different system specifications on the performance. Due to space limitations, we only reported our results when an edit distance of three was rather used in filtering out candidates. As shown in 3c and 3d, exploring words that are less similar in terms of edit distance had a slight negative impact on performance. Once the results shown in 3a and 3c are compared for the top candidate, the measured optimal f-measure without gold word detection was decreased from 79.68% (+ED:2) to 78.69% (+ED:3). A similar drop from 82.87% (+ED:2) to 81.79% (+ED:3) was observed with gold word detection once the results given in 3b and 3d are compared for the top candidate.

4.4. Normalization Impact on Downstream Language Applications

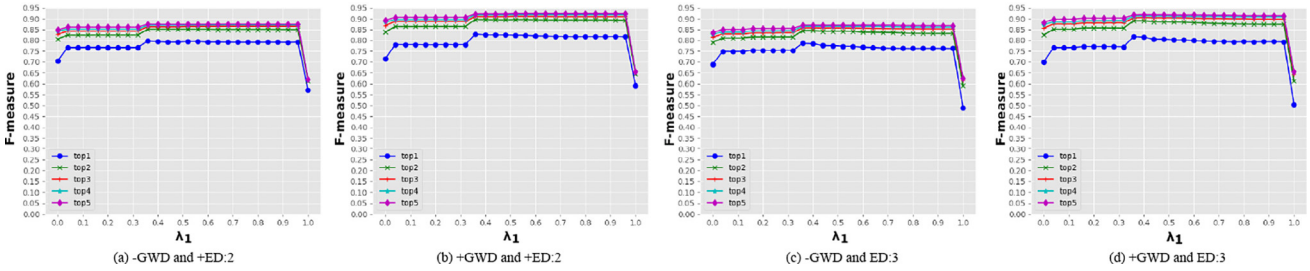
Transforming noisy user generated texts into a form more akin to formal texts that various language applications trained on is

Table 9
Context-aware normalization performances.

	Setting	Precision	Recall	F-Measure	Accuracy
MS-Word		58.85	62.22	60.49	92.08
T-Norm		70.35	74.24	72.24	94.32
MT-Norm		78.63	80.14	79.38	95.49
Our Normalizer	*-GWD+EW*	*79.92*	*78.89*	*79.40*	*95.73*
	-GWD+OW	*80.20*	*79.17*	*79.68*	*95.75*
	+GWD+EW	*83.25*	*81.81*	*82.52*	*97.94*
	+GWD+OW	*83.60*	*82.15*	*82.87*	*97.95*

Table 10
The effect of contextual and lexical similarities

	Setting	Precision	Recall	F-Measure
$\lambda_1 = 1 \& \lambda_2 = 0$	*-GWD*	85.77	42.71	57.02
	+GWD	85.37	45.17	59.08
$\lambda_1 = 0 \& \lambda_2 = 1$	*-GWD*	71.07	70.03	70.55
	+GWD	72.20	70.80	71.49
$\lambda_1 = 0.5 \& \lambda_2 = 0.5$	*-GWD*	79.92	78.89	79.40
	+GWD	83.25	81.81	82.52



(a) -GWD and +ED:2 (b) +GWD and +ED:2 (c) -GWD and ED:3 (d) +GWD and ED:3

Fig. 3. System performance with edit distance = 2 (a-b) and with edit distance = 3 (c-d).

motivated by foreseen performance increase. Although it is widely accepted that general normalization or application-oriented normalization [79] eventually has an effect on downstream language applications, only a few recent work has consolidated this experimentally [4,34]. To our best knowledge, none is known on the sole impact of normalization on Turkish language processing. In order to obtain answers to the research question as to what extent normalization supports Turkish language applications in achieving their best possible performance, we studied two applications with different focus of interest. In part-of-speech tagging, each word by itself is morphologically analyzed whereas a deeper analysis of all words as a whole is performed in dependency parsing. Moreover, part-of-speech tagger and dependency parser are representatives of applications that suffer from accuracy drops on collections from different genres due to their sensitivity to data variations [54].

Here, we followed the methodology applied in earlier research works [79,4]. Original test tweets were processed by both applications to obtain baselines for further comparisons. In addition, manually normalized forms of these tweets were processed for gold standards. Although manual normalization overcomes accuracy drops due to non-standard nature of data, the output might still suffer from the overall application performance (i.e., the highest measured performance on clean texts). Since our goal here is not to assess the real performance of an application but rather the impact of normalization on its performance, we argue that this is an appropriate strategy for setting gold standards. For a detailed analysis, we investigated two different experimental setups [33]. In the first setup, non-standard words in the test corpus were identified and normalized by our normalizer and reference systems, whereas in the second setup our normalizer corrected only manually identified non-standard words (with gold word detection). In both cases, the corpus was processed by part-of-speech tagger and dependency parser, and the performance scores were compared with baseline and gold standard scores. To the best of our knowledge, our work is the first that has performed a comprehensive evaluation of the sole impact of normalization on processing noisy texts in Turkish NLP applications using the same normalizer.

4.4.1. Part-of-Speech Tagger

Part-of-speech (POS) tagger assigns syntactic categories (e.g., noun, adjective, and verb) to words in a sentence [9,65]. In order to provide a universal inventory of categories, a recent effort on developing cross-linguistically consistent treebank annotation for many languages [58] has defined a fixed POS tag set. POS tagging is proved to be useful for text analysis in different language processing tasks such as language modeling [28]. However, Turkish POS tagging is a complex task due to inherent morphological level ambiguity. In this experiment, we used the NUVE NLP Library[12] to assign syntactic categories to all words in our test corpus. For performance evaluations, we only focused on three frequently seen POS tags in Turkish texts, and defined the precision and recall computing as a variation of the methods used in [79,4]:

$$Precision_{NAV} = \frac{|NAV \cap NAV_{gold}|}{|NAV|} \; Recall_{NAV} = \frac{|NAV \cap NAV_{gold}|}{|NAV_{gold}|}$$

where NAV and NAV_{gold} are the sets of nouns, adjectives, and verbs in the sentence to be compared and the gold standard.

11 presents the results of this experiment, where the first row shows the baseline performance of the tagger (i.e., the performance

[12] https://github.com/hrzafer/nuve

Table 11
The impact of normalization on POS tagger.

Setting	Precision	Recall	F-Measure
Unnormalized	73.88	78.36	76.05
MS-Word	90.74	94.96	92.80
T-Norm	92.25	95.26	93.73
MT-Norm	93.28	96.07	94.65
Our Normalizer (-GWD)	93.91	96.72	95.30
Our Normalizer (+GWD)	94.43	97.35	95.87

Table 12
The impact of normalization on dependency parser.

Setting	Precision	Recall	F-Measure
Unnormalized	53.02	53.76	53.39
MS-Word	71.25	70.95	71.10
T-Norm	76.66	76.17	76.42
MT-Norm	85.86	86.25	86.05
Our Normalizer (-GWD)	86.28	86.65	86.46
Our Normalizer (+GWD)	89.26	89.59	89.42

on unnormalized test corpus). The performance scores of our reference systems are given in the second, third, and fourth rows. The tagger performance without gold word detection is presented in the fifth row, whereas the best performance achieved with gold word detection is shown in the sixth row. From this experiment, we can conclude that normalization contributed to tagging accuracy by an f-measure of at least 16.75%, and the improvement achieved by our normalizer was more significant (19.25% f-measure improvement) than those of reference systems. Moreover, a little gain in f-measure (0.57%) was achieved once gold non-standard word detection is available. Finally, the paired t-test analysis showed that the improvement that our normalizer achieved in recall was statistically significant ($p < 0.05$) over all reference systems except MT-Norm.

4.4.2. Dependency Parser

Dependency parser analyzes grammatical structure of a sentence and identifies dependency relations (syntactic or semantic) between constituent words (head and dependent) in order to produce its dependency tree [11,29]. In the literature, successful attempts have been reported on parsing for various languages [21,38]. The biggest challenge in Turkish dependency parsing, which has received comparatively little attention over the past [22,10], is the need for establishing dependency relations between sublexical units (i.e., inflectional groups) rather than words. New initiatives on Turkish parsing were introduced as part of recent cross-linguistically consistent treebank annotation research efforts [58]. In this experiment, we trained a graph-based parser [7] for Turkish and used it to process our test sentences. To facilitate a common evaluation methodology [4], three kinds of dependency relations were considered, and precision and recall scores were measured in comparison with gold data as:

$$Precision_{SOV} = \frac{|SOV \cap SOV_{gold}|}{|SOV|} \; Recall_{SOV} = \frac{|SOV \cap SOV_{gold}|}{|SOV_{gold}|}$$

where SOV and SOV_{gold} are the sets of subject, object, and verb dependencies in the sentence to be compared and the gold standard, respectively.

12 summarizes the experimental results. The parser performance was observed to be significantly improved from 53.39% to 86.46% f-measure once our normalizer was used. Although f-measure improvements of at least 17.71% were achieved with reference systems, our normalizer contributed most to the parser performance. However, the results revealed that non-standard word detection had a negative effect of 2.96% f-measure on parser performance. The improvement observed in parsing was much higher than what we achieved in tagging. We argue that this difference is due to the complexity of the task being handled. Since parsing is a more complex task where each word individually has a potential to affect several dependency relations, it is less robust to differences between unnormalized and normalized data than part-of-speech tagging. Similar to POS tagging task, the measured difference in terms of recall between our normalizer and the reference systems, T-Norm and MS-Word, was statistically significant under the paired t-test with $p < 0.05$. Both of these experiments validated that text processing stages that depend on the output of these applications might greatly benefit from normalization.

5. Conclusion and Future Work

This article describes our normalization approach which is based on the idea that contextual and lexical similarities between non-standard words and their standard forms need to be explored for appropriate normalization. To encode contextual similarities, a bipartite graph is constructed from a large collection of clean and noisy texts. Random walks performed over the graph by traversing between words and their contexts are used to identify contextually similar candidates for non-standard words. Our novel context tailoring approach benefits from local contexts of non-standard words in determining starting node(s) of these walks. Character-based edit operations are used to produce lexically similar candidates for non-standard words. Our approach selects the best normalization alternative from among all identified candidates via a weighted scoring mechanism. We conducted several experiments in order to assess the performance of our Turkish normalizer. Intrinsic evaluations on a test corpus of 1430 tweets revealed that our system achieves state-of-the art results, and melding contextual and lexical similarities during normalization performs better than relying on either similarity alone. The experiments where we explored the effect of normalization on Turkish downstream applications showed that more accurate results are obtained with normalization, and the more complex the task being handled by the application the more performance gain is achieved.

Although our work is a significant advance in the state-of-the-art, it has some room for future improvements. One major area is to improve our non-standard word detection approach which cannot properly handle abbreviations, slang words, and proper nouns at all times. Unfortunately, false negatives and false positives negatively affect the functioning of our normalizer since leaving a non-standard word as is or changing an already standard word results in poor precision and recall. Another interesting research direction is to better guide the randomness in graph traversal. Please recall that, our graph is dense and the average degree of word nodes is high. There are some times where the right standard word, despite being in the graph, cannot be reached during traversals. This is due to limitations taken mandatorily for performance issues and random choices that are made from among several transition possibilities. A special care should be taken to design a sophisticated graph traversal strategy that would yield a better performance. Our character-based transformations might produce so many candidates with the same similarity score in some cases. We believe that enhancing our approach by modeling the task as a character-level sequence to sequence learning problem would improve the performance. One important future work to investigate is the application of our approach to languages besides Turkish. We expect that the graph-based subnormalizer can be easily trained for a new language and many of the techniques that are carried out by the transformation-based subnormalizer can be adapted to or replaced with other language-dependent techniques for that language. As future work, we also plan to explore alternative ways of melding

all identified similar candidates in a single list and to incorporate a phonetic-subnormalizer so as to produce standard words that sound like the non-standard word in focus. Finally, we plan to cover many-to-many word normalizations.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Arslan, Deasciification approach to handle diacritics in turkish information retrieval, Information Processing & Management 52 (2016) 326–339.
[2] A. Aw, M. Zhang, J. Xiao, J. Su, A phrase-based statistical model for sms text normalization, in: Proceedings of the 21st COLING Conference, 2006, pp. 33–40.
[3] S. Balan, J. Rege, Mining for social media: Usage patterns of small businesses, Business Systems Research Journal 8 (2017) 43–50.
[4] T. Baldwin, Y. Li, An in-depth analysis of the effect of text normalization in social media, in: Proceedings of the 14th NAACL Conference: Human Language Technologies, 2015, pp. 420–429.
[5] D. Bamman, J. Eisenstein, T. Schnoebelen, Gender identity and lexical variation in social media, Journal of Sociolinguistics 18 (2014) 135–160.
[6] R.C. Belwal, S. Rai, A. Gupta, A new graph-based extractive text summarization using keywords or topic modeling, Journal of Ambient Intelligence and Humanized Computing 12 (2021) 8975–8990.
[7] B. Bohnet, Very high accuracy and fast dependency parsing is not a contradiction, in: Proceedings of the 23rd COLING Conference, 2010, pp. 89–97.
[8] E. Brill, R.C. Moore, An improved error model for noisy channel spelling correction, in: Proceedings of the 38th ACL Conference, 2000, pp. 286–293.
[9] B. Can, A. Üstün, M. Kurfali, Turkish pos tagging by reducing sparsity with morpheme tags in small datasets, in: Proceedings of the 19th CICLing Conference, 2018, pp. 320–331.
[10] O. Cetinoglu, J. Kuhn, Towards joint morphological analysis and dependency parsing of turkish, in: Proceedings of the 2nd International Conference on Dependency Linguistics, 2013, pp. 23–32.
[11] Choi, J.D., 2012. Optimization of Natural Language Processing Components for Robustness and Scalability. Ph.D. thesis. University of Colorado Boulder..
[12] M. Choudhury, R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, A. Basu, Investigation and modeling of the structure of texting language, International Journal on Document Analysis and Recognition 10 (2007) 157–174.
[13] G. Chrupala, Normalizing tweets with edit scripts and recurrent neural embeddings, in: Proceedings of the 52nd ACL Conference, 2014, pp. 680–686.
[14] O.D. Clercq, S. Schulz, B. Desmet, E. Lefever, V. Hoste, Normalization of dutch user-generated content, in: Proceedings of the 9th RANLP Conference, 2013, pp. 179–188.
[15] Çolakoğlu, T., Sulubacak, U., Tantuğ, A.C., 2019. Normalizing non-canonical Turkish texts using machine translation approaches, in: Proceedings of the 57th Annual Meeting of the Assoc. for Computational Linguistics: Student Research Workshop, pp. 267–272..
[16] Cook, P., Stevenson, S., 2009. An unsupervised model for text message normalization, in: Proceedings of the 4th Workshop on Computational Approaches to Linguistic Creativity, pp. 71–78..
[17] Costa Bertaglia, T.F., Volpe Nunes, M.d.G., 2016. Exploring word embeddings for unsupervised textual user-generated content normalization, in: Proceedings of the 2nd W-NUT Workshop, pp. 112–120..
[18] J. Cotelo, F. Cruz, J. Troyano, F. Ortega, A modular approach for lexical normalization applied to Spanish tweets, Expert Systems with Applications 42 (2015) 4743–4754.
[19] Demir, S., 2016. Context tailoring for text normalization, in: Proceedings of Text Graphs at NAACL-HLT: the 10th Workshop on Graph-based Methods for Natural Language Processing, pp. 6–14..
[20] Demir, S., Tan, M., Topcu, B., 2018. Turkish normalization lexicon for social media, in: Proceedings of the 17th Computational Linguistics and Intelligent Text Processing Conference, pp. 418–429..
[21] T. Dozat, C.D. Manning, Deep biaffine attention for neural dependency parsing, CoRR (2016). abs/1611.01734.
[22] Durgar El-Kahlout, İ., Akin, A.A., Yilmaz, E., 2014. Initial explorations in two-phase Turkish dependency parsing by incorporating constituents, in: Proceedings of the 1st Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages, pp. 82–89..
[23] J. Eisenstein, B. O'Connor, N.A. Smith, E.P. Xing, Diffusion of lexical change in social media, PLoS ONE 9 (2014) 113–114.
[24] G. Eryigit, D. Torunoglu-Selamet, Social media text normalization for Turkish, Natural Language Engineering 23 (2017) 835–875.
[25] A. Farzindar, D. Inkpen, Natural Language Processing for Social Media, Second Edition., Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2018.
[26] E. Flint, E. Ford, O. Thomas, A. Caines, P. Buttery, A text normalisation system for non-standard English words, in: Proceedings of the 3rd Workshop on Noisy User-generated Text, 2017, pp. 107–115.
[27] J. Foster, O. Cetinoglu, J. Wagner, J. Le Roux, J. Nivre, D. Hogan, J. van Genabith, From news to comment: Resources and benchmarks for parsing the language of web 2.0, in: Proceedings of 5th International Joint Conference on Natural Language Processing, 2011, pp. 893–901.
[28] L. Galescu, E.K. Ringger, Augmenting words with linguistic information for n-gram language models, in: Proceedings of the Eurospeech Conference, 1999, pp. 2171–2174.
[29] P. Gamallo, M. Garcia, Dependency parsing with finite state transducers and compression rules, Information Processing & Management 54 (2018) 1244–1261.
[30] Garimella, A., Mihalcea, R., 2016. Zooming in on gender differences in social media, in: Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media, pp. 1–10..
[31] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, N.A. Smith, Part-of-speech tagging for twitter: Annotation, features, and experiments, in: Proceedings of the 49th ACL Conference: Human Language Technologies, 2011, pp. 42–47.
[32] W.H. Gomaa, A.A. Fahmy, Article: A survey of text similarity approaches, International Journal of Computer Applications 68 (2013) 13–18.
[33] R. van der Goot, R. van Noord, G. van Noord, A taxonomy for in-depth evaluation of normalization for user generated content, in: Proceedings of the 11th LREC Conference, 2018, pp. 684–688.
[34] R. van der Goot, B. Plank, M. Nissim, To normalize, or not to normalize: The impact of normalization on part-of-speech tagging, in: Proceedings of the 3rd Workshop on Noisy User-generated Text, 2017, pp. 31–39.
[35] Gouws, S., Hovy, D., Metzler, D., 2011. Unsupervised mining of lexical variants from noisy text, in: Proceedings of the First Work. on Unsupervised Learning in NLP, pp. 82–90..
[36] S. Göker, B. Can, Neural text normalization for Turkish social media, in: 3rd International Conf. on Computer Science and Engineering (UBMK), 2018, pp. 161–166.
[37] D.Z. Hakkani-Tür, K. Oflazer, G. Tür, Statistical morphological disambiguation for agglutinative languages, Computers and the Humanities 36 (2002) 381–410.
[38] Hall, J., Nivre, J., 2008. A dependency-driven parser for german dependency and constituency representations, in: Proceedings of the Workshop on Parsing German, pp. 47–54..
[39] B. Han, P. Cook, T. Baldwin, Lexical normalization for social media text, ACM Transactions on Intelligent Systems and Technology (TIST) 4 (2013) 1–27.
[40] H. Hassan, A. Menezes, Social text normalization using contextual graph random walks, in: Proceedings of the 51st ACL Conference, 2013, pp. 1577–1586.
[41] Ikeda, T., Shindo, H., Matsumoto, Y., 2016. Japanese text normalization with encoder-decoder model, in: Proceedings of the 2nd Workshop on Noisy User-generated Text, pp. 129–137..
[42] M.D. Kernighan, K.W. Church, W.A. Gale, A spelling correction program based on a noisy channel model, in: Proceedings of the 13th COLING Conference, 1990, pp. 205–210.
[43] A.T. Koksal, O. Bozal, E. Yürekli, G. Gezici, #turki$hTweets: A benchmark dataset for Turkish text correction, Findings of the Association for Computational Linguistics: EMNLP 2020 (2020) 4190–4198.
[44] Kong, L., Schneider, N., Swayamdipta, S., Bhatia, A., Dyer, C., Smith, N.A., 2014. A dependency parser for tweets, in: Proceedings of the EMNLP Conference, pp. 1001–1012..
[45] N. Koppula, B. Padmaja Rani, K.S. Rao, Graph based word sense disambiguation, in: S.C. Satapathy, V.K. Prasad, B.P. Rani, S.K. Udgata, K.S. Raju (Eds.), Proceedings of the First International Conference on Computational Intelligence and Informatics, Springer Singapore, 2017, pp. 665–670.
[46] Kumar, V., Sridhar, R., 2015. Unsupervised text normalization using distributed representations of words and phrases, in: Proceedings of the NAACL Conference: Human Language Technologies, pp. 8–16..
[47] V.I. Levenshtein, Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10 (1966) 707.
[48] C. Li, Y. Liu, Improving named entity recognition in tweets via detecting non-standard words, in: Proceedings of the 53rd ACL-IJCNLP Conference, 2015, pp. 929–938.
[49] W. Ling, C. Dyer, A.W. Black, I. Trancoso, Paraphrasing 4 microblog normalization, in: Proceedings of the EMNLP Conference, 2013, pp. 73–84.
[50] F. Liu, F. Weng, X. Jiang, A broad-coverage normalization system for social media language, in: Proceedings of the 50th ACL Conference, 2012, pp. 1035–1044.
[51] F. Liu, F. Weng, B. Wang, Y. Liu, Insertion, deletion, or substitution?: Normalizing text messages without pre-categorization nor supervision, in: Proceedings of the 49th ACL Conference, 2011, pp. 71–76.
[52] N. Ljubešić, T. Erjavec, D. Fišer, Standardizing tweets with character-level machine translation, in: Proceedings of the 15th CICLing Conference, 2014, pp. 164–175.
[53] I.D. Melamed, Bitext maps and alignment via pattern recognition, Computational Linguistics 25 (1999) 107–130.
[54] A. Mukherjee, S. Kübler, M. Scheutz, Creating pos tagging and dependency parsing experts via topic modeling, in: Proceedings of the 15th EACL Conference, 2017, pp. 347–355.

[55] Muller, B., Sagot, B., Seddah, D., 2019. Enhancing bert for lexical normalization, in: Proceedings of the 5th Workshop on Noisy User-generated Text, pp. 297–306..
[56] V. Nastase, R. Mihalcea, D.R. Radev, A survey of graphs in natural language processing, Natural Language Engineering 21 (2015) 665–698.
[57] Neri, F., Aliprandi, C., Capeci, F., Cuadros, M., By, T., 2012. Sentiment analysis on social media, in: Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 919–926..
[58] J. Nivre, M.C. de Marneffe, F. Ginter, Y. Goldberg, J. Hajic, C.D. Manning, R.T. McDonald, S. Petrov, S. Pyysalo, N. Silveira, R. Tsarfaty, D. Zeman, Universal dependencies v1: A multilingual treebank collection, in: Proceedings of the 10th LREC Conference, 2016, pp. 1659–1666.
[59] E. Okur, H. Demir, A. Özgür, Named entity recognition on twitter for turkish using semi-supervised learning with word embeddings, in: Proceedings of the 10th LREC Conference, 2016, pp. 549–555.
[60] A.H. Osman, N. Salim, M.S. Binwahlan, R. Alteeb, A. Abuobieda, An improved plagiarism detection scheme based on semantic role labeling, Applied Soft Computing 12 (2012) 1493–1502.
[61] O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, N.A. Smith, Improved part-of-speech tagging for online conversational text with word clusters, in: Proceedings of the NAACL Conference: Human Language Technologies, 2013, pp. 380–390.
[62] Z. Ozer, I. Ozer, O. Findik, Diacritic restoration of turkish tweets with word2vec, Engineering Science and Technology, an International Journal 21 (2018) 1120–1127.
[63] Pennell, D., Liu, Y., 2012. Evaluating the effect of normalizing informal text on TTS output, in: 2012 IEEE Spoken Language Technology Workshop (SLT), pp. 479–483..
[64] D.L. Pennell, Y. Liu, Normalization of informal text, Computer Speech & Language 28 (2014) 256–277.
[65] M. Pota, F. Marulli, M. Esposito, G.D. Pietro, H. Fujita, Multilingual pos tagging by a composite deep architecture based on character-level features and on-the-fly enriched word embeddings, Knowledge-Based Systems 164 (2019) 309–323.
[66] Ritter, A., Clark, S., Mausam, Etzioni, O., 2011. Named entity recognition in tweets: An experimental study, in: Proceedings of the EMNLP Conference, pp. 1524–1534..
[67] M.A. Saloot, N. Idris, R. Mahmud, An architecture for malay tweet normalization, Information Processing & Management 50 (2014) 621–633.
[68] N. Sarma, S.R. Singh, D. Goswami, Influence of social conversational features on language identification in highly multilingual online conversations, Information Processing & Management 56 (2019) 151–166.
[69] S. Schulz, G.D. Pauw, O.D. Clercq, B. Desmet, V. Hoste, W. Daelemans, L. Macken, Multimodular text normalization of dutch user-generated content, ACM Transactions Intelligent Systems Technology 7 (2016) 61:1–61:22.
[70] T. Sezer, Tweets corpus: Building a corpus by social media, Journal of Milli Egitim Education and Social Sciences 210 (2016) 621–633.
[71] S. Shayaa, S. Ainin, N.I. Jaafar, S.B. Zakaria, S.W. Phoong, W.C. Yeong, M.A. Al-Garadi, A. Muhammad, A. Zahid Piprani, Linking consumer confidence index and social media sentiment analysis, Cogent Business & Management 5 (2018) 1–12.
[72] C. Sönmez, A. Özgür, A graph-based approach for contextual text normalization, in: Proceedings of the EMNLP Conference, 2014, pp. 313–324.
[73] R. Sproat, A.W. Black, S. Chen, S. Kumar, M. Ostendorf, C. Richards, Normalization of non-standard words, Computer Speech & Language 15 (2001) 287–333.
[74] D. Torunoğlu, G. Eryiğit, A cascaded approach for social media text normalization of turkish, in: Proceedings of the 5th Workshop on Language Analysis for Social Media, 2014, pp. 62–70.
[75] C. Yang, H. Zhang, B. Jiang, K. Li, Aspect-based sentiment analysis with alternating coattention networks, Information Processing & Management 56 (2019) 463–478.
[76] Y. Yang, J. Eisenstein, A log-linear model for unsupervised text normalization, in: Proceedings of the EMNLP Conference, 2013, pp. 61–72.
[77] S. Yildirim, T. Yildiz, An unsupervised text normalization architecture for turkish language, Research in Computing Science 90 (2015) 183–194.
[78] S. Yolchuyeva, G. Németh, B. Gyires-Tóth, Text normalization with convolutional neural networks, International Journal of Speech Technology 21 (2018) 589–600.
[79] C. Zhang, T. Baldwin, H. Ho, B. Kimelfeld, Y. Li, Adaptive parser-centric text normalization, in: Proceedings of the 51st ACL Conference, 2013, pp. 1159–1168.
[80] H. Zhang, R. Sproat, A.H. Ng, F. Stahlberg, X. Peng, K. Gorman, B. Roark, Neural models of text normalization for speech applications, Computational Linguistics 45 (2019) 293–337.
[81] J. Zobel, P. Dart, Phonetic string matching: Lessons from information retrieval, in: Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval, 1996, pp. 166–172.