Natural Language Engineering **23** (6): 835–875. © Cambridge University Press 2017
doi:10.1017/S1351324917000134

Social media text normalization for Turkish

GÜLŞEN ERYİĞİT and DİLARA TORUNOĞLU-SELAMET
Department of Computer Engineering, Istanbul Technical University, Istanbul, Turkey
e-mail: gulsen.cebiroglu@itu.edu.tr, torunoglud@itu.edu.tr

(*Received 2 August 2015; revised 18 April 2016; accepted 19 April 2017; first published online 2 June 2017*)

Abstract

Text normalization is an indispensable stage in processing noncanonical language from natural sources, such as speech, social media or short text messages. Research in this field is very recent and mostly on English. As is known from different areas of natural language processing, morphologically rich languages (MRLs) pose many different challenges when compared to English. Turkish is a strong representative of MRLs and has particular normalization problems that may not be easily solved by a single-stage pure statistical model. This article introduces the first work on the social media text normalization of an MRL and presents the first complete social media text normalization system for Turkish. The article conducts an in-depth analysis of the error types encountered in Web 2.0 Turkish texts, categorizes them into seven groups and provides solutions for each of them by dividing the candidate generation task into separate modules working in a cascaded architecture. For the first time in the literature, two manually normalized Web 2.0 datasets are introduced for Turkish normalization studies. The exact match scores of the overall system on the provided datasets are 70.40 per cent and 67.37 per cent (77.07 per cent with a case insensitive evaluation).

1 Introduction

With the increasing number of people using social media services (such as Facebook, Twitter, Youtube, Instagram, Foursquare and Google+), social media data has become a highly attractive target for machine learning and natural language processing (NLP) research employed in many high-impact applications, including social network analysis, information extraction, web mining, opinion mining and brand monitoring. Social media text normalization is an important preprocessing step in parsing this valuable resource. Baldwin and Li (2015) and Melero *et al.* (2015) investigate its impact on some core NLP applications, such as lemmatization, part-of-speech tagging, parsing, named entity recognition (NER) and text-to-speech systems for the case of English and Spanish.

The language used in social media differs severely from formally written texts, for several reasons. People often write in a way that is much more personal and reflects their spoken idiolect, as part of which they are arguably using idiolectic grammar rules (Blevins *et al.* 2016). Moreover, those that are not corporate entities communicating via social media channels do not feel forced to write grammatically

correct words or sentences in a social media context. The distinction that first appeared in the early days of text messaging was linguistically classified by Crystal (2008) as the use of logograms, pictograms, initialisms, omitted letters, nonstandard spellings and shortenings. In social media, it is very common to observe nonstandard words (Baldwin *et al.* 2015) created due to the omission of some letters (especially in platforms that allow messages with a limited number of characters, such as Twitter), phonological variations as in spoken language (Eisenstein 2013a) and nonlexical valence shifters for sentiment intensification (Muhammad, Wiratunga and Lothian 2015) (e.g., the use of repeating letters/characters, multiple exclamation and punctuation marks and capitalization). This situation results in substantially different languages than conventional ones.

The message given in Example 1 is easily understandable by almost the entire young population active in social media and it demonstrates a good example of this trendy writing style.

Example 1

'r u gonna lov meeeee 4ever ?'
(*Are you going to love me forever ?*)

The research on text normalization of social media gained speed toward the end of the last decade and as always, almost all of these elementary studies are conducted on the English language. We know from earlier research that morphologically rich languages (MRLs), such as Finnish, Czech, Korean, Hungarian and many others differ severely from English and methods tailored for English do not fit these languages well. MRLs pose interesting challenges for NLP tasks (e.g., data scarcity, the representation of rich morphological features in different tasks) as is the case with text normalization. The supervised normalization approaches would require a larger amount of manually annotated data in the case of MRLs, due to the data sparsity problem caused by the complexity of the language morphology. Turkish is a morphologically very rich language and its agglutinative nature reveals a very high number of possible suffix combinations. With these characteristics, Turkish is a strong representative of MRLs. The relatively very large number of possible word surface forms in Turkish leads to many more possibilities in the social media domain. In order to give an idea about the nature of the problem in the Turkish language, Table 1 provides an example related to the idiolectic usages of the verb 'yapmak' (*to make*) in future tense under different person agreements. As it is the case in the English 'going to' future construction that undergoes different nonstandard variations with idioletic or rural usages such as 'gonna' or 'gunna' (Labov 1969; Lacoste 2012), the Turkish future suffix (-ecek/-acak) together with different person agreements[1] causes similar but more variant surface forms for Turkish verbs. The valid surface forms for the verb 'yapmak' in future tense are provided in the first row of the table. Although the lemma ('yap' – *make*) remains the same in all items,

[1] First sg – first person singular agreement, second sg – second person singular agreement, third sg – third person singular agreement, first pl – first person plural agreement, second pl – second person plural agreement, third pl – third person plural agreement.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 1. *Some idiolectic usages for the verb 'yapmak' (to make) in future tense under different person agreements*

yapacağım-yapacaksın-yapacak-yapacağız-yapacaksınız-yapacaklar (First sg) (Second sg) (Third sg) (First pl) (Second pl) (Third pl)
yapcam – yapcan – yapcak – yapcaz – yapcanız – yapcaklar
yapçam – yapçan – yapçak – yapçaz – yapçaksınız – yapçaklar
yapıcam – yapıcan – yapıcak – yapıcaz – yapıcaksınız – yapıcaklar
yapacam – yapacan – yapacak – yapacaz – yapacanız – yapacaklar

the idiolectic variations of the affix sequences cause several different surface forms. The same phenomenon may be observed with almost any verb in Turkish, which results in many more different surface forms compared to English for the same reason. One should note that the listed surface forms in Table 1 may still undergo more nonstandard variations with different phenomena (such as omitted vowels or diacritics, spelling errors and many more) occurring in social media.

This article focuses on the social media text normalization of Turkish. Following the work of Han and Baldwin (2011), the normalization process is divided into two stages: ill-formed word detection (IWD) and candidate word generation (CG). The proposed approach is based on partitioning the candidate generation task into seven different subtasks and solving each of them using relevant rule-based, statistical or machine learning components together with an overall cascaded architecture that combines them. This enables the use of appropriate rule-based or machine learning techniques for each different component(also termed layer). This also reduces the need for manually normalized data that is crucial for resource-poor languages, making it possible to generate the training data automatically for some of the layers, such as vowel and diacritic restoration. The proposed seven layers are as follows: (1) letter case transformation, (2) replacement rules & lexicon lookup, (3) proper noun detection, (4) diacritic restoration, (5) vowel restoration, (6) accent normalization and (7) spelling correction. The introduced cascaded approach combines these layers so that they can cooperate with each other and correct the cases that could not be solved independently by a single layer. Each layer makes use of different NLP subcomponents and resources, such as a morphological analyzer, a morphological generator, a language validator (LV) and proper noun gazetteers.

To the best of our knowledge, this article, which is an extension of our recent work (Torunoğlu & Eryiğit 2014), is the first study on the social media text normalization of an MRL and presents the first complete social media text normalization system for Turkish. This article further introduces the following additional material, not included in the prior work: (1) A detailed investigation of the normalization subcategories by analyzing their distributions on real datasets, (2) an improved vowel restoration module capable of doing partial vowel restoration, (3) a novel Turkish spelling corrector, (4) two new data resources for Turkish Web 2.0 normalization studies introduced for the first time in the literature, (5) an improved flow in the cascaded architecture and (6) an in-depth evaluation of the normalization modules and the entire system on the provided datasets. The lexicons and rules used in the

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

rule-based components, and all the NLP subcomponents and language resources used or developed in this article are available via `http://tools.nlp.itu.edu.tr` for reproducibility and further research in the field. The exact match scores of the overall system on the provided datasets are 70.40 per cent and 67.37 per cent (77.07 per cent with a case insensitive evaluation).

The paper is structured as follows: Section 2 and 3 give brief information about related work and the challenges posed by MRLs. Section 4 introduces our proposed architecture that gives details about the IWD and CG stages, and explains our solutions for each of the seven candidate generation layers and their work flow. Section 5 gives the experimental results and discussions and, finally, Section 6 gives the conclusion.

2 Related work

An important part of the previous studies have taken the normalization task either as a lexicon lookup (together with or without replacement rules) or as a statistical problem. There also exist many studies which use a combination of these. In these studies, a lexicon lookup is first employed for the normalization of the most commonly used slang words, abbreviations, etc. and then a machine learning method is employed for the rest. In the following paragraph, we briefly mention the work on English, and we then summarize the work conducted for other languages in the later paragraph of this section. Since the emergence of social media is very recent, only the latest normalization studies are focused on this area and the earlier ones generally work for the text normalization in TTS (text-to-speech), ASR (automatic speech recognition) systems or SMS messages. Social media normalization poses new challenges on top of these, for example, Twitter contains mentions (@user_name), hashtags (#topic), a variety of emoticons (e.g., ‘ :@ ’ , ‘ <3 ’, ‘ @>– ’, ‘ :) ’) and special keywords (*RT* – retweet, *DM* – direct message etc.).

Cook and Stevenson (2009) use an unsupervised noisy channel model, whereas Han and Baldwin (2011) use word similarity and context during lexicon lookup for the normalization of SMS text messages. Clark and Araki (2011) use dictionary lookup and Liu, Weng and Jiang (2012) use a unified letter transformation algorithm to generate possible ill-formed words in order to use them in the training phase of a noisy channel model, both for the social media domain. Zhang *et al.* (2013) use replacement rules and a graph-based model in order to select the best rule combinations for three different domains: Twitter posts, SMS text messages and call-center logs. Similarly, Han, Cook and Baldwin (2013) again use a statistical approach for Twitter and SMS text messages. Wang and Ng (2013) use a beam search decoder, and Hassan and Menezes (2013) propose an unsupervised approach that uses Random Walks on a contextual similarity bipartite graph constructed from n-gram sequences. Both of the works report their results for the social media domain. Eisenstein (2013a) first analyzes the phonological factors in social media writing, and then Yang and Eisenstein (2013) report the highest results on the used datasets using a unified unsupervised statistical model. Li and Liu (2014)

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

report that they outperform these results by reranking the outputs of multiple systems. Xu, Xia and Lee (2015) propose a syllable-based method for tweet normalization; they extend the conventional noisy channel model by incorporating syllables to represent word-to-word transitions at both word and syllable levels. Sridhar (2015) focuses on unsupervised text normalization of tweets using distributed word and phrase representations. In a recent shared task (W-NUT 2015) on the processing of UGC text, participants were asked to normalize English tweets in the normalization subtask. The highest results at W-NUT are obtained by a constrained system (Leeman-Munk, Lester and Cox 2015; Leeman-Munk 2016), which uses two forward neural networks to predict the ill-formed words and their normalized versions. A more detailed analysis of the systems participating in W-NUT may be found in Baldwin *et al.* (2015). Beckley (2015) focuses on a simple approach for Twitter Text Normalization, which includes a modestly sized, partially curated wordlist combined with a reranking scheme. Jin (2015)'s tweet normalization system first generates candidates based on past knowledge and a novel string similarity measurement and then selects a candidate using features learned from training data. Wagner and Foster (2015) focus on learning edit operations for microblog normalization with a generalized perceptron algorithm. Doval Mosquera, Vilares and Gómez-Rodríguez (2015) adapt a Spanish microtext normalization system (Vilares, Alonso and Vilares 2013) to English and apply a combination of two traditional approximations (the spell checking and the automatic speech recognition metaphors) to this task (Kobus, Yvon and Damnati 2008), whereas Min and Mott (2015) use a deep contextual long-short term memory model. Jahjah, Khoury and Lamontagne (2016) use phonetic signatures to match similar words during the task of English tweet text normalization. Akhtar, Sikdar and Ekbal (2015), Supranovich and Patsepnia (2015) and Berend and Tasnadi (2015) use conditional random fields (CRFs) for the same task. Others, treating the normalization task as a machine translation (MT) problem that tries to translate from an ill-formed language to a regular one, form also another important group. For example, the studies on SMS text messages from Aw *et al.* (2006), Beaufort *et al.* (2010), Pennell and Liu (2011) and on Twitter from Kaufmann and Kalita (2010) may be collected under this group. Limsopatham and Collier (2015) also adapt a phrase-based MT system to normalize medical terms in social media messages.

Although fewer than the studies on English, there are also some studies on other languages and these are mostly for speech recognition and SMS text messages, e.g., Panchapagesan *et al.* (2004) for Hindi TTS, Nguyen, Pham and Tran (2010) for Vietnamese TTS, Jia *et al.* (2008) for Mandarin TTS and Khan and Karim (2012) for Urdu SMS. For such languages, the work for social media has recently picked up speed: De Clercq *et al.* (2013) use an MT-based system for SMS and social media messages in the Dutch language, and similarly Saloot, Idris and Mahmud (2014) use an MT-based system for social media messages in Malay. Schulz *et al.* (2016) follow the study of De Clercq *et al.* (2013) and propose a multimodular approach for social media messages in the Dutch language using multiple modules, some of which rely again on MT- and transliteration-based systems. Zhang, Chen and Huang (2014)

use a graph-based approach for Chinese social media text normalization. Qian *et al.* (2015) use a transition-based model for joint segmentation, POS-tagging and normalization for the Chinese language. Duran *et al.* (2015) propose a lexicon-based tool for user-generated content (UGC) normalization in Brazilian Portuguese. In recent years, we observe many studies focusing on social media text normalization for Spanish, which is a morphologically more interesting language (also carrying some similar challenges to Turkish such as the importance of diacritic restoration in social media text normalization). Melero *et al.* (2015) rerank a spelling corrector's outputs for the normalization of Spanish UGC. Alegria *et al.* (2015) present a benchmark for lexical normalization of Spanish social media posts and review the approaches to the TweetNorm_es shared task (Alegria *et al.*, 2013). The best performing approach to this shared task is from Porta and Sancho (2013), who use finite state transducers in order to compile the normalization rules and a 3-gram language model to decode the word graph, and obtain the most probable sequence for each tweet. Another study is conducted by Ageno *et al.* (2013), which searches for similar OOV words in several gazetteers and lexica (Spanish, English, NEs, morphological derivates of Spanish words) using the edit distance measure with several cost matrices: one for keyboard typos, one for phonetic similarity and normal edit distance. Although Spanish, which is a fusional language, has a richer morphology than English, this richness is rather limited compared to agglutinative languages (e.g., Turkish, Korean, Japanese, Hungarian, Finnish). Agglutinative languages clearly tend to have longer words than fusional ones (Egidio et al. 2013) and the spelling correctors (such as Eger *et al.* (2016) for Latin, Silverberga, Kauppinen and Lindén (2016) and Pirinen *et al.* (2010) for Finnish and Oflazer (1996) for Turkish) remain insufficient for the case where most of the errors in the UGC domain cause changes with edit distance larger than two. To the best of our knowledge, our study is the first attempt to tackle the problems of a MRL in the field of social media text normalization.

3 Challenges of morphologically rich languages

As quoted in Tsarfaty *et al.* (2010), 'The term Morphologically Rich Languages is used in the CL/NLP literature to refer to languages in which substantial grammatical information, i.e., information concerning the arrangement of words into syntactic units or cues to syntactic relations, is expressed at word level'. MRLs, such as Turkish, Finnish, Korean, Arabic, Hebrew, etc., pose significant challenges for NLP tasks. Highly inflectional or agglutinative languages share the characteristic that a unique stem in these languages may have hundreds of possible surface forms. This increases the data sparsity in statistical models. For example, it is pointed out in Hakkani-Tür, Oflazer and Tür (2000), that it is due to the Turkish language's inflectional and derivational morphology that the number of distinct word forms is very large compared to English distinct word number (Table 2).

As stated previously, the highly productive morphology of these languages results in a very large number of word forms from a given stem. Table 3 lists only a few (among hundreds of possible) surface forms for the Turkish stem *ev* (house).

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 2. *Vocabulary sizes for two Turkish and English Corpora (Hakkani-Tür et al. 2000)*

Corpus size	Turkish	English
1M words	106,547	33,398
10M words	417,775	97,734

Table 3. *Some surface forms for the Turkish word ev (house)*

Surface form	English
ev	'house'
eve	'to the house'
evde	'at the house'
evdeki	'(which is) at the house'
evdekiler	'those (who are) at the house'
evdekilerde	'at those (who are) at the house'

Sarikaya *et al.* (2009) list the emerging problems as below:

(1) increase in dictionary size;
(2) poor language model probability estimation;
(3) higher out-of-vocabulary rate;
(4) inflection gap for MT.

That is why the normalization methods proposed so far (adapting MT or language models or pure lexicon lookup approaches[2]) do not seem appropriate for the processing of MRLs with poor language resources, as in our case with Turkish.

4 The proposed architecture

Similar to Han and Baldwin[3] (2011), we divide the normalization task into two stages. These are namely (IWD) and stages. As can be understood from the names, IWD is the stage where the input tokens are analyzed in order to select the ones to be normalized, and CG is the stage that generates the normalized output for its inputs coming from the prior stage. The candidate generation stage is then partitioned into seven layers as presented in Figure 1. The following subsections provide the details for both IWD and CG stages, as well as their individual components.

Before sending the input token sequence (e.g., a Tweet or a post) into normalization, we first use our tokenizer composed of regular expressions and specifically

[2] Although our proposed model also uses a small-sized lexicon for replacing most frequently occurring social media patterns (only 286 noncanonical slang words and their normalized forms), these are rather very limited ones compared to the studies which are fully relied on using lexicons for the normalization task.

[3] Han and Baldwin (2011) specify three stages ((1) confusion set generation, (2) ill-formed word detection and (3) candidate word selection) where the first and third steps correspond to our candidate generation layer.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

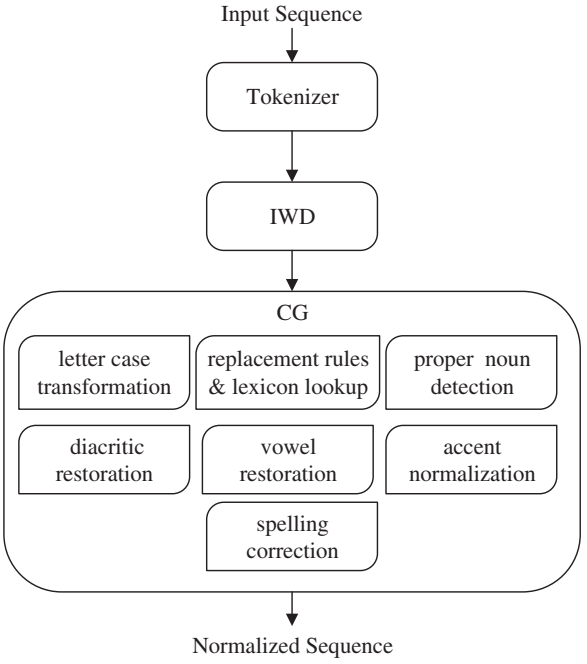

Fig. 1. Normalization architecture.

tailored for Web 2.0 data for splitting the input sequence into meaningful tokens. Our tokenizer is actually the first step of our normalization process. It does special processing for emoticons and consecutive punctuation marks so that they still reside together after the tokenization (e.g., :D or !!!!! are output as they occur). Incorrectly typed punctuations (e.g., 'a,b' to 'a , b') are split into separate tokens, while leaving period signs occurring after title words (such as Dr., Mr., etc.) and apostrophes untouched. The apostrophe which is generally used in English for the possessive clitic ('s) in common nouns or the plural suffix ('s) in proper nouns, may get a long token appended to it in MRLs (consisting of multiple inflectional features); e.g., *Ankara'dakiler* (those which/who are in Ankara).

4.1 Ill-formed word detection (IWD)

As stated earlier, since it is not feasible and efficient to use a lexicon lookup table for storing and checking all possible surface word forms in an MRL, one needs to have an LV that will verify whether a given word belongs to that language. In the case of an MRL, a high-coverage morphological analyzer may be used for this purpose. The aim of a morphological analyzer is to analyze the word surface forms and output their possible lemmas, part-of-speech tags and inflectional/derivational structures in the case of an MRL. If the morphological analyzer is capable of producing a valid output for a given input, this will certify that the given surface form is a valid word in

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

that language. One should keep in mind that, with this approach, all foreign words[4] will also be detected as ill-formed and will be automatically normalized, although this may not be preferable in some cases. Although the language identification task has been considered an almost solved problem for many applications, language detectors fail in the social media context due to phenomena called code-switching (i.e., the usage of multiple languages within the same context or even the same sentence). This article treats the detection and normalization of these as out of scope and only focuses on the normalization of Turkish words. Solorio *et al.* (2014) make an overview of the first shared task on language identification of code-switched data. As future work, our approach may be extended by adapting one of the recent code-switching approaches, such as Alex, Dubey and Keller (2007), Lui, Lau and Baldwin (2014), Eskander *et al.* (2014), Jhamtani, Bhogi and Raychoudhury (2014), Das and Gamback (2013) among many others.

In this stage, the erroneous letter case usages (all UPPER CASE, mIXed case writings) should also be detected and sent to the candidate generation stage for normalization. For the IWD of Turkish, we use a high-coverage morphological analyzer[5] (Eryiğit 2014) and a letter case checker, which detects erroneous letter case usages. All tokens that are selected as *in-vocabulary* (IV) words (labeled with a $+N$ C (No Change) label in the example below) are left untouched and output in their original form. The filtered *out-of-vocabulary* (OOV) words are transferred to the candidate generation stage. Mentions (@user_name), hashtags (#topic), emoticons (:D), vocatives (*ahahahaha*) and keywords (*RT*) are also assumed to be OOV words, since we would like to detect these and tag them with special labels to be later used in higher level NLP modules (e.g., POS tagging or syntactic analysis).

The following example shows an input sentence (*bgn Twitter çok eglenceliiiiiiiii dimi ? :D*) and the labels assigned to each of its components after being processed by the IWD stage. The expected normalized form of the sentence is *'Bugün Twitter çok eğlenceli değil mi ? @smiley[:D]'* that means *'Twitter is very entertaining today, isn't it ? :D'*. The words that are passed directly from IWD are marked in the example below with a check mark (✓). The ones that are filtered to be transferred to the candidate generation (CG) stage are marked with a cross (✗).

Example 2

bgn → ✗→ CG
Twitter → ✓→ NC
çok → ✓→ NC
eglenceliiiiiiiii → ✗→ CG
dimi → ✗→ CG

[4] It is very common in social media posts for foreign words to be used together with words from the language in focus. Excluding these from the normalization is a common problem for most languages.

[5] The lexicon of the morphological analyzer used also contains an abbreviation list of 1,045 abbreviations obtained from the TLA (Turkish Language Association) `http://www.tdk.gov.tr/index.php?option=com_content&id=198:Kisaltmalar` and the most frequently used foreign words (e.g., Twitter, Facebook) adopted into the Turkish language.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

$$\begin{array}{lll} ? \rightarrow & \checkmark \rightarrow & \text{NC} \\ :D \rightarrow & \times \rightarrow & \boxed{\text{CG}} \end{array}$$

4.2 Candidate generation

In order to design an appropriate model for the candidate generation stage, the main nonstandard spellings in Turkish social media texts are investigated[6] and the following trends are observed: the usage of slang words, abbreviations and initialisms, character repetitions in interjections, logograms, social media specific words, wrong letter cases, direct transcription of words as if they were pronounced with an accent, vowel and diacritic omissions and misspelling of words. In order to normalize such nonstandard spellings, seven main candidate generation layers are designed and implemented. These are (refer to Section 5.1 for the distribution of these categories on our test sets) as follows:

(1) replacement rules & lexicon lookup (e.g., *2g2bt* → 'too good to be true');
(2) letter case transformation (e.g., *JOHN'S* → 'John's');
(3) proper noun detection (e.g. *potter* → 'Potter within the context Harry Potter');
(4) diacritic restoration (e.g., *facade* → 'façade');
(5) vowel restoration (e.g., *Twttr* → 'Twitter');
(6) accent normalization (e.g., *gonna* → 'going to');
(7) spelling correction (e.g., *acheive* → 'achieve').

The first two components and the accent normalization module (normalization of phonetic writing in different accents) are mostly solved by rule-based techniques (sometimes making use of statistics); proper noun detection and diacritic and vowel restoration modules are created using a machine learning technique (namely CRFs) and the spelling corrector using a statistical model. Traditional spelling correction techniques rely on the fact that most spelling errors are within a short edit-distance of their correct form (Kukich 1992; Max & Wisniewski 2010). That is why spelling correction needs special treatment in case of MRLs, which by nature consist of longer words resulting in errors in longer edit distances and mostly due to the wrong spellings outside the word lemma (within the affixes) (Ingason *et al.* 2009; Pirinen & Lindén 2014; Pirinen *et al.* 2010). In case of languages having rather shorter words than MRLs, the complete omission of diacritics and vowels would not be a severe problem and could be jointly solved with spelling correction. But with the entrance of social media into our daily lives, diacritic and vowel restoration as well as accent normalization need special treatment for MRLs. In our proposed architecture, we treat these tasks as separate layers.

Since no data resources for Turkish text normalization were available for training purposes during the implementation of this study, we tried to minimize the need for gold-standard normalization data during the design phase by either proposing rule-based techniques or by using machine learning approaches where it would

[6] The main nonstandard spelling types are investigated on Turkish social media data in general and on our validation set which will be introduced in later sections.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

be possible to create the training data automatically. During the analyses of the rule-based modules, a validation set (see Section 5.1 for details) is used to compose the relevant rules. For machine learning (i.e., vowel and diacritic restoration) and statistical models (i.e., spelling correction), the automatically created training data are explained later in the following subsections.

After the individual creation of each module, they are integrated in a cascaded structure (Section 4.2.8) in order to be able to normalize the words which could not be corrected independently by these. The outputs of each component are checked by the LV, and then, if the normalized form from a component becomes an IV word, then the process is terminated (except for the *Letter Case Transformation*, *Replacement Rules & Lexicon Lookup* and *Diacritization* components, where the input is replaced by the modified output although it is still not an IV word; see Section 4.2.8 for details). Otherwise, the candidate generation process continues with the next component of the original input.

4.2.1 Replacement rules & lexicon lookup

The usage of slang words and initialisms (e.g., *kib* for *kendine iyi bak* (take care of yourself) or *nbr* for *ne haber* (what's up), character repetition in interjections (e.g., *lütfeeeennnn* instead of *lütfen* (please) or *çooooooook* instead of *çok* (very)), Web 2.0 specific words, such as hashtags (e.g., *#topic*), mentions (e.g., *@user_name*), emoticons (e.g., *:p*), vocatives (e.g., *ahahhah* or *hööööö*) and keywords (e.g., *RT*), logograms (e.g., *$eker 4you* instead of *şeker senin için* (sweety, it's for you)), emails and URLs are very common in social media texts. In this module, the normalization of these types are realized by the use of replacement rules designed for catching and normalizing the most frequent patterns and by the use of a lexicon composed of most frequently used slang words and initialisms. Our lexicon contains 286 noncanonical slang words and initialisms, and their normalized forms. To give an example, the following words *ltf*, *pls*, *ltfn*, *piliz* all refer to *lütfen* (please) in Turkish. In this module, once the input token is looked up and found within the lexicon, it is directly replaced by its normalized form.

Replacement rules are written as regular expression patterns and handle the normalization of character repetitions, logograms,[7] the detection and tagging of Web 2.0 specific words, emails and URLs. For the normalization of character repetitions, we reduce repeated characters into a single character when the consecutive occurrence count is greater than two. There are additional replacement rules to catch and normalize logograms. Examples include $\$ \rightarrow ş$, $\epsilon \rightarrow e$, $3 \rightarrow e$, $@ \rightarrow a$, $! \rightarrow i$ and $ß \rightarrow b$. The following token types are tagged by the given specific labels for future usage in higher level applications:

[7] Crystal (2008) defines logograms (e.g., 4 → *for*) as 'the use of single letters, numerals, and typographic symbols to represent words, parts of words, or even noises associated with actions'. In the case of Turkish, the use of some special nonalphabetic characters (generally named EMO characters in Turkish social media) instead of Turkish letters is very frequent in social media language: there exist even online EMO converters on the Web. In this manuscript, we use the term logogram by extending its meaning to also cover representation of letters instead of whole words or parts of words.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

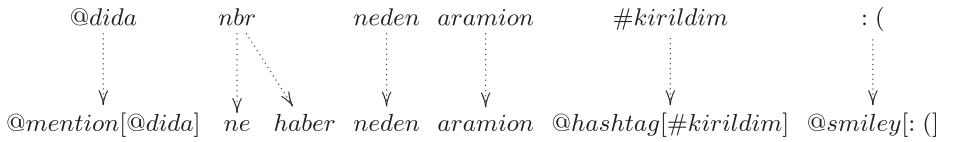

Fig. 2. Normalization with replacement rules & lexicon lookup.

- **Mentions:** Nicknames that refer to users on Twitter are labeled as *@mention[@dida]*.
- **Hashtags:** Hashtags that refer to trending topics are labeled as *@hashtag [#geziparki]*.
- **Vocatives:** Vocatives are labeled as *@vocative[hehe]*.
- **Smileys:** Emoticons are labeled as *@smiley[:)]*.
- **URLs:** URLs are labeled as *@url[http://www.cse.unt.edu/∼ rada/jnle]*.
- **E-mails:** E-mails are labeled as *@email[texline@cup.cam.ac.uk]*.
- **Keywords:** Keywords like *RT*, *DM*, *MT*, *Reply* etc. are labeled as *@keyword [RT]*.

Figure 2 shows the normalized version of a tweet in informal Turkish that could be translated as *'@dida what's up, why don't you call #offended :('*, before and after being processed by this component. Although the word *aramion* also needs to be normalized as *aramıyorsun* (you don't call), this transformation is not realized within the current component and normalized later in the accent normalization phase described in Section 4.2.4.

4.2.2 Letter case transformation

Ill-formed words coming to this stage are treated differently according to their cases (lowercase, UPPERCASE, Proper Noun Case or miXEd CaSe). If the token is in lowercase and does not contain any specific punctuation marks for proper nouns or title words (i.e., an apostrophe ''' or a period '.'), it is directly transferred to the next stage without any change (e.g., *umuttan* (from hope)). If it contains one of these specific punctuation marks (e.g., *john'dan*), it is converted to proper noun case (*John'dan* (from John)) and output as the normalization result. If the token is already in Proper Noun Case and contains an apostrophe sign (e.g., *Umut'tan*), it is accepted as a correct proper noun (even if it does not occur within the morphological analyzer's lexicon or was previously detected as an OOV word) and left untouched.

For UPPERCASE, miXEd CaSe and lowercase words, we convert them into Proper Noun Case if they either contain an apostrophe (''') (which is used in Turkish to separate inflectional suffixes from a proper noun) or a period (.) (which is used formally in Turkish to denote some abbreviations). If the word does not contain either of these two marks, it is then converted to lowercase. It should be noted that all words output from this component toward next stages are transformed into lowercase from this point on.

4.2.3 Proper noun detection

The primary aim of this component is to detect and correct proper nouns erroneously written in lowercase with an inappropriate surface form or that have been lower-

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

cased by our previous model (Section 4.2.2). To give an example, the words *ahmetten* and *ahmetden* are both erroneously written on behalf of the ablative inflected form of a male name *Ahmet* and should be actually written as *Ahmet'ten*. Since these words in their current state would be rejected by the LV, they would be detected as an OOV by IWD and transferred to CG for correction.

Turkish proper nouns are very frequently selected from common nouns. Some samples are given below. The name *umut* that means *hope* in English is a commonly used person name in Turkish. Since the task could not be easily accomplished by just checking the occurrence of such words within gazetteers, it is quite difficult to identify such words as proper nouns when they are written in lowercase.

Example 3

Umut (hope) – a unisex name or a surname

Çiçek (flower) – a female name or a surname

Şeker (sugar) – a female name or a surname

İpek (silk) – a female name or a surname

NER can be basically defined as identifying and categorizing predefined types of named entities, such as person, location and/or organization names. Although NER approaches can achieve nearly human annotation performance on well-formed text, noisy user-generated text presents serious challenges for this task and performance on this domain is still very behind that obtained on well-formed text (Baldwin *et al.* 2015). The task of detecting and correcting ill-formed proper nouns on social media data may actually be regarded as a version of NER where named entities should be first detected and then normalized into a valid form.

In this module, we adapt a NER model (from Şeker and Eryiğit (2012; 2017) reporting the highest results for Turkish NER in the literature) and use it for the normalization of proper nouns. The adapted model based on sequential classification (i.e., CRFs) uses lexical, morphological and contextual features for modeling the morphologically rich nature of the Turkish language. The model uses a morphological analyzer (producing all possible analyses for a given token) and a disambiguator (selecting the most probable analysis from the outputs of the morphological analyzer) in order to assign the relevant morphological features for each token. The same training dataset (Tür, Hakkani-Tür and Oflazer 2003) consisting ∼500K words collected from news articles is used during the training stage. But since this dataset is of a well-formatted nature, in order to remove the influence of the letter case feature (e.g., uppercase, lowercase) used in the original study and to adapt it for our normalization module, we lowercased all the named entities during data preparation before the training stage. As the second step of our adaptation, we developed two new gazetteers for person names and surnames in addition to the original gazetteers used in the original study. These two new gazetteers are the filtered versions[8] of the original person name and surname gazetteers using some threshold values calculated

[8] The size of the gazetteers are initially reduced by removing all the words with length smaller than two characters.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 4. *Example of ratio values*

Proper case	Lowercase	Common noun meaning	Ratio
Sağlam=9	*sağlam*=100	healthy	0.09
Umut =40	*umut* =100	hope	0.4
Ahmet =40	*ahmet* =20	n/a	2.0

(formula given in Equation 1) over two additional corpora: the METU Corpus (Say *et al.* 2002) and the web corpus of Sak, Güngör and Saraçlar (2011). The names having a zero denominator or no occurrence count at all are assigned a very large ratio value so that they are all treated as proper nouns. Table 4 gives the occurrence ratios for three sample words. One may observe from the table that *ahmet* occurred 40 times in proper case form and 20 times in lower case form within the two corpora resulting in a ratio value of 2.0. The aim of this calculation is to detect nouns that are more likely to occur as a proper noun rather than a common noun. With this purpose, we select a threshold value of 1.5 (after an investigation of the gazetteers) above which a ratio would denote that the corresponding noun should be treated as a proper noun. For example, the ratio value for *umut* is only 0.4 (which is below the threshold) and it would not be converted to proper case. A similar situation holds for the word *sağlam* (healthy). Although it is a quite frequent Turkish family name, it is observed mostly as a common noun in our corpora with a ratio value of 0.09.

$$ratio(w_n) = \frac{OccurrenceInPropercase(w_n)}{OccurrenceInLowercase(w_n)} \quad (1)$$

During the testing stage, each token is looked up within the gazetteers. Since the gazetteers contain only lemmatic information, the surface form of an inflected proper noun would never exist in them. In order to derive the appropriate lemma for such words (especially if not written with appropriate orthographic rules), we make use of an unknown-word analyzer (which is distributed with the used morphological analyzer (Eryiğit 2014)) also capable of analyzing the inflected forms of unknown lemmas according to Turkish morphological rules. The unknown-word analyzer has more flexible vowel-consonant harmony rules for unknown words so that it strips all possible suffixes even if they do not phonetically conform to the unknown lemma. For our initial example, both the inputs *ahmetten* and *ahmetden* would be analyzed the same way since the suffixes *-den* and *-ten* are possible surface forms of the ablative case suffix according to consonant harmony.

Example 4

ayşe+Guess+Noun+A3sg+P2sg+Abl

ayşen+Guess+Noun+A3sg+Pnon+Abl

ayşende+Guess+Noun+A3sg+P2sg+Nom

ayşenden+Guess+Noun+A3sg+Pnon+Nom

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

It is very probable that multiple admissible lemmas may be produced at the end of the explained process. For example, the above output (Example 4)[9] is obtained after the analysis of the word *ayşenden*. The possible lemmas proposed by the unknown word analyzer are *ayşe*, *ayşen*, *ayşende*, *ayşenden*. The first two (*Ayşe* and *Ayşen*) are both frequently used Turkish female names and occur within the gazetteers with high ratio values. Both of the lemmas have the ablative case marker tag in the morphological analyzer output, but the first one *ayşe* is inflected with an extra *+n* suffix (second person possessive form), which is a rarely used suffix for proper nouns (meaning 'from your Ayşe'). In this case, although both of the analyses are valid, a human would most probably choose the second one with the lemma *Ayşen* with no possessive marker The inflectional suffixes are then separated by the use of an apostrophe resulting in the corrected word *Ayşen'den* (from Ayşen). In order to assign the most probable lemmas and morphological analyses for such words, the outputs of the morphological analyzer are filtered before being transferred to the morphological disambiguator. For this filtering, the possible lemmas are looked up from the gazetteers, and the longest possible one existing within the gazetteers (e.g., *ayşen* for this example) is accepted as the legal one and only the analyses producing this lemma are kept. If none of the possible lemmas exist within the gazetteers, the word is left as it is and transferred to the next stage in its original form.[10] For the normalization stage, the named entities extracted as the result of the recognition layer are then capitalized and the suffixes added to the assigned lemma are separated with the use of an apostrophe sign added to the surface form.

4.2.4 Accent normalization

As defined by McKean (2005), 'An accent is a manner of pronunciation peculiar to a particular individual location or nation'. In social media, people generally write as they speak by transferring the pronounced versions of words directly to the written text. Eisenstein (2013a) discusses the impact of phonological variation in English social media texts by giving the following Example 5. In this article, our aim is to be able to detect the most frequent accents used in Turkish social media texts and normalize them. Turkish is a language that is almost always pronounced as it is spelled. Due to its rich morphology, the accent problems of Turkish are more severe than the examples given below.

Example 5

lef → 'left'
jus → 'just'

[9] The morphological analysis for the word *ayşenden*: The +Guess label is to reflect that the output belongs to the unknown-word analyzer, +A3sg is for singular nouns, +Pnon stands for no possessive marker, whereas +P2sg for the second person possessive marker, +Nom for the nominative case marker and +Abl for ablative case.

[10] During our NER experiments, we also tested with extra features related to the existence of the tokens within these new gazetteers. But since these are just a subset of the already existing gazetteers within the system, we could not obtain any significant improvement with these additional features.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

wit → 'with'
gonna → *goin* → 'going'
doin → 'doing'
kno → 'know'

As in most MRLs, words have many inflected forms and the number of possible accents becomes very high. While the detection of accent usage is an important issue, another crucial stage is its normalization which could not be realized by just using lookup tables over stored samples. For example, while templates like *going to* in English could be easily resolved by storing their different accents (*goin, gonna*) and using them in replacement rules during normalization with any verb (as in *'I'm gonna dance'*, *'I'm gonna sing'*), the same future tense accents would be a severe issue in the case of Turkish. The tense suffix will be affixed to a stem together with many other possible suffixes (person, polarity, mood and tense markers, including compound tenses) and will conform to many vowel-consonant harmony rules according to the stem to which it is affixed. One should first catch all possible accented forms and then normalize the word by reconstructing the phonological harmony. In the below example, the verb *git* (to go) takes the following suffixes[11]: (1) the negative marker *+mA*, (2) the future tense suffix *+(y)AcAk* and (3) the first person agreement marker *+(H)m*. The morphological analysis of the resulting regular word *gitmeyeceğim* (*git+me+yecek+im*) is *git*+VERB+NEG+FUT+A1SG. It is seen in the example that the accented version *gidmiycem* differs severely from the normalized form *gitmeyeceğim*. Even the stripped stems of the two words (*gid* and *git*) are different from each other due to the consonant lenition.[12] One needs to use a more complex approach in order to normalize such words.

Example 6

gidmiycem instead of *gitmeyeceğim*

(I will not go)

In this component, we focus on normalizing the most common verb accents that Turkish people use in spoken language since these are the ones which have the largest number of different surface forms. As the first step of our solution, we create a replacement rule table similar to the one given in Table 5, consisting of the most frequent verb accent endings, such as *-cem, -yom, -çaz* and their appropriate morphological analyses if they were written in appropriate manner as *-ceğim, -yorum, -cağız*. We then construct regular expressions in order to detect these endings on the input tokens and then replace them with the matched morphological analysis given in the table. As an example, the input *gidmiycem* becomes *gid*+VERB+NEG+FUT+A1SG.

[11] In the generic suffix representation, the letter *A* represents either an *a* or *e* according to the last vowel of the previous morpheme, the letters between parantheses (y) will either appear or not according to the last letter of the previous morpheme and the letter *H* represents either an *ı, i, u* or *ü.*; e.g., *+(y)AcAk* appears as *yecek* in the given example; the last letter then changes into the letter *ğ* according to consonant lenition.

[12] In Turkish, the terminal voiceless consonants *p, ç, t, k* of a stem are replaced by the voiced consonants *b, c, d, ğ* before a vowel-initial suffix.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 5. *Examples of accent normalization replacement rules*

Accent endings	Morph. analysis
+iyon	+VERB+POS+PROG1+A2SG
+cem	+VERB+POS+FUT+A1SG
+caz	+VERB+POS+FUT+A1PL

The morphological tags in the table stand for: +POS: Positive, +PROG1: Present continuous tense, +A2SG: second person singular, +FUT: Future tense, +A1SG: first person singular, +A1PL: first person plural.

As a second step, we use a morphological generator (Eryiğit 2014), to reconstruct the normalized form of the verb. The morphological generator produces a valid Turkish word by applying all the rules of a morphological analyzer in the reverse order (from lexical form through surface form). There exist cases where it is not possible to obtain the correct verb stem by just removing the accented suffixes given in the table. For example, *gid* is the surface form of the verb stem *git* (to go) only after being affixed a morpheme starting with a vowel. For such cases, we also apply the phonological harmony rules to the stem.

There exist also more complex cases where the normalized form is not a single token. In the below Example 7, the proper form of the word *gidiyonmu* is actually *gidiyor musun* (are you going) and in formal writing the interrogative enclitic *mi* (in this case, with the second person singular suffix *-sun*) is written separately from the verb itself. On the other hand, the example shows that the person suffix is directly attached to the verb in the accented version: *gidiyo**N**mu*. For such cases, we apply the generation process twice (for the verbal stem and the enclitic separately). At the end of this process, if a valid output is produced, it is directly accepted as the normalization result. Otherwise, the input word is transferred to the next normalization stage in its original form.

Example 7

'geliyonmu?' instead of *'geliyor musun?'*

(Are you coming?)

4.2.5 Vowel restoration

Twitter played an April Fool's joke[13] announcing that they were not including the vowels in Tweets. Although this was only a joke, nowadays, people have already developed this habit of writing without vowels (Eisenstein 2013b), which was started mainly with the need to fit their messages into 140 characters Tweets or 160 character

[13] The joke was about Twitter shifting to a two-tiered service where the basic free service *Twttr* would only allow using consonants in tweets. For five dollars a month, the premium *Twitter* service would also include vowels. `https://blog.twitter.com/2013/annncng-twttr`

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 6. *Instance representation for the word okuldan*

Curr. Word	Neigh. Ch(−3)	Neigh. Ch(−2)	Neigh. Ch(−1)	Neigh. Ch(+1)	Neigh. Ch(+2)	Neigh. Ch(+3)	Class Label
kldn	_	_	_	k	l	d	o
kldn	_	_	k	l	d	_	u
kldn	_	k	l	d	n	_	null
kldn	k	l	d	n	_	_	a
kldn	l	d	n	_	_	_	null

SMS messages. As a result, the vowel restoration problem is no longer limited to some specific language families such as Semitic languages (Zitouni, Sorensen and Sarikaya 2006; Gal 2002), but rather applicable to social media text normalization in general.

Vowel restoration in Turkish was recently studied by Adalı and Eryiğit (2014). In this work, the authors reported an accuracy of 54.07 per cent over a test set synthetically generated by removing all the vowels in existing valid Turkish texts and using their original form as gold-standard. They use CRFs to train their model again over a synthetically generated training set. The deficiency of the approach is that it only considers the words' consonants and adds vowels between these as in *Twttr* ⇒ 'Twitter', ignoring the already existing vowels within the given input. For example, the input *svyrm* is correctly vowelized as *seviyorum* (I love), but the input *savyrm* is again vowelized as *seviyorum* instead of *savıyorum* (I avoid) by ignoring the occurrence of the letter *a* in the given input.

In order to improve the vowel restoration system, we propose a new partial vowel restorer which will be constrained on the existing vowels in the input. With this purpose, we present an approach similar to the sequence classifier using CRFs introduced by Lafferty, McCallum and Pereira (2001) but this time using a constrained Viterbi algorithm at the decoding stage for CRFs. We use the freely available NLP library MALLET (McCallum 2002) and reimplement its decoding stage in order to implement partial vowel restoration.

Since manual compilation of new training data for partially missing vowels is a costly process, we trained our improved model with the same automatically created training data from Adalı and Eryiğit (2014). Although it would be possible to create such datasets (both for training and testing phases) synthetically by randomly removing some vowels from well-written texts, we prefer not to do so as it would not reflect real world usage statistics.

Table 6 gives the feature representation of the word *okuldan* (from school) for the training stage. As can be seen from the table, the task is treated as a character level sequence classification task where the class labels are the eight possible Turkish vowels *(a, e, ı, i, o, ö, u, ü)*. In this approach, each position between the consonants needs to be classified under one of these classes (plus one *null* class for the case of no vowel). As a result, we have nine possible class labels (last column of Table 6) to assign to each position. Since the occurrence of two consecutive vowels (such as

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

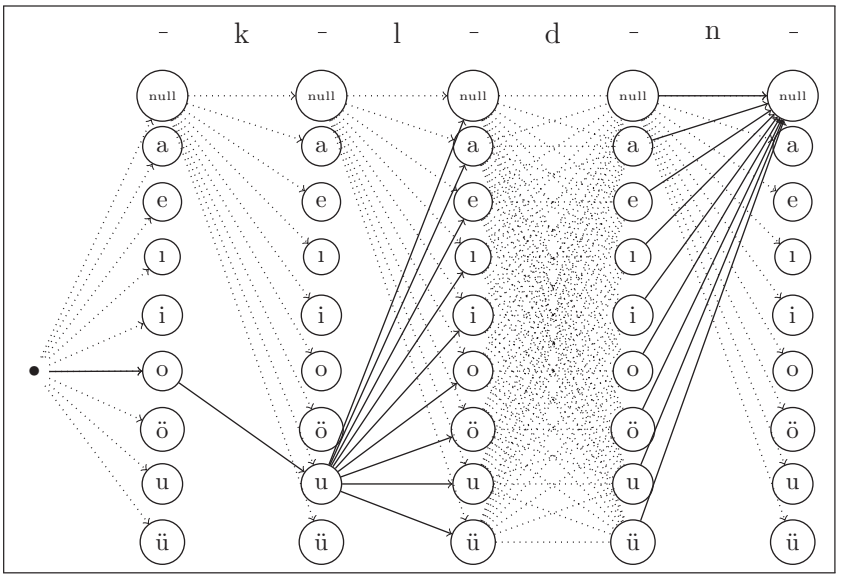

Fig. 3. Trellis for the input *okuldn*.

in the English word 'book') is very rare in Turkish and they are mostly not written without vowels due to produced ambiguities, each position is limited to either 0 or 1 possible vowel addition.[14] This choice also reduces the complexity of the search space and the errors caused by the extra vowel addition. We adopt the same features as the previous work[15]: the consonants of the input token and the neighboring characters within a window of ± 3 characters. The table displays the values of these features for each instance.

At the decoding stage, n possible positions will produce 9^n possible output words. For the input *kldn*, many of these would be valid Turkish words such as *koldan* (from the arm), *kuldan* (from the slave), kıldan (from the hair). The existence of some vowels within the input would undoubtedly help the disambiguation and to find the most probable output. Figure 3 shows the constructed trellis (9^5=59,049 possible paths) for the input *okuldn*. By using the existing vowels as constraints during Viterbi decoding, one may reduce the possible paths to $9^{(5-2)}$=729 (specified with continuous lines on the trellis). In addition to this, we make further pruning after investigating the vowel removal trends used in Turkish social media data. We observe that people do not miss the first and the last letter in the case that it is a vowel (for example, *aşkım* (my love) is never written as *şkm* but it is rather written as *aşkm*). In other words, the omission of vowels occurs most of the time within the word and not in the boundaries. Thus, differing from the previous work, we do

[14] In Turkish, the occurrence of two consecutive vowels is rarely seen: some exceptional cases are words adopted from Arabic and Persian such as *saat* (clock) and some compound words such as *karaağaç* (elm tree).

[15] Adalı and Eryiğit (2014) also test with different word contexts (within a window of ± 2 words) and report no increase in success rates.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 7. *Possible diacritization outputs for the input words cin, kus and sok*

Input	Output 1	Output 2	Output 3	Output 4
cin	*cin* (genie)	*çin* (China)	*çın* (ding)	*cın* (OOV)
kus	*kus* (vomit)	*kuş* (bird)	*küs* (sulk at)	*küş* (OOV)
sok	*sok* (insert)	*şok* (shock)	*sök* (disassemble)	*şök* (OOV)

not try to add vowels to the frontiers. However, if the word starts or ends with a vowel, we use this information at the decoding stage as our constraints. This further reduces the possible number of paths to $9^2 = 81$ in the given example (concentrating only on locating the vowels between the consonants *l* - *d* and *d* - *n*).

4.2.6 Diacritization

Sometimes, because of the nonconformity of the input methods in computers and mobile devices (unlocalized keyboards, inconsistent or badly designed user input software), or due to users' typing habits, many texts in social media are missing diacritical marks. Diacritization (also called 'deasciification' for Turkish) is the task of reproducing Turkish-specific characters with diacritics (*ı, İ, ş, ö, ç, ğ, ü*) from their ASCII-compliant counterparts (*i, I, s, o, c, g, u*). Deasciification is not a straightforward task because of the potential for ambiguity in the asciified form. For example, the inputs *cin*, *kus* or *sok* all consist of two critical Turkish-specific characters belonging to the set given above and this results in $2^2 = 4$ possible outputs (Table 7) for each of them. The given examples carry a high degree of ambiguity so that 3/4 of their deasciified counterparts are valid Turkish words.

Contrary to the Turkish vowel restoration problem which is a contemporary issue, the diacritization problem has always been attractive even before the raise of social media, since the first day of press usage. Thus, there are many studies on Turkish diacritization in the literature, namely Tür (2000), Zemberek (2007), Yüret and de la Maza (2006) and Adalı and Eryiğit (2014). The last study reporting the highest diacritization performance (which is also the appearing that we used in this study[16]) treats the task again as a character-level sequential classification problem. Although Adalı and Eryiğit (2014) adopt exactly the same multiclass classification approach (introduced in the previous subsection) by providing the current letter in focus as a feature to the CRF, they claim that the search space is automatically reduced to that of a binary classification problem.

For example, when the character in focus is the letter *c*, the system will only try to decide between the two labels (*c* and *ç*) as shown in Figure 4, since this is predefined in the system even though the possible class labels are of size 14 (the non-ASCII Turkish letters along with their ASCII counterparts as provided above). Since it is

[16] The mentioned work is implemented under the same research project (TUBITAK 1001 - grant no: 112E276) as this article and directly used in this module.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

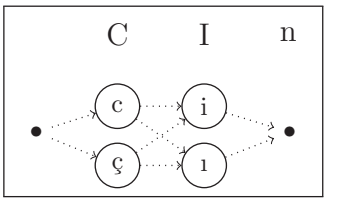

Fig. 4. Trellis for the input *CIn* (the critical letters are capitalized meaning that they may either be written as *c* or *ç* and *i* or *ı*).

very easy for a classifier to learn this from relevant features and provided samples, this information is not provided to the system as an external constraint.

4.2.7 *Spelling correction*

Since the normalization task originates in fact from correcting spelling errors (but mostly made on purpose in the case of social media), a spelling corrector is certainly an indispensable component of a normalizer. Unfortunately, as stated previously in Torunoğlu and Eryiğit (2014), the current performances of the available spelling correctors for Turkish fall behind those for English. The authors state that they tested the impact of two spelling correctors (Akın & Akın 2007; Microsoft 2010) and that they obtained negative impacts on normalization performances due to the low precision scores of the correctors. The spelling corrector to be used in this task should have a special bias for precision over recall since the false positives have a very harmful effect on the normalization task; the erroneous outputs may have a radically different meaning and ruin the sense of the whole sentence.

Torunoğlu-Selamet *et al.* (2016) make a comparison of available spelling correctors for Turkish: the error-tolerant finite-state recognizer (ETFSR) of Oflazer (1996), the spelling corrector of MSWord (Microsoft 2010), the open source Turkish NLP framework Zemberek (Akın & Akın 2007) and four new spelling correction models mainly adapted from Wang *et al.* (2011). In our normalization architecture, we elect to use the best performing model (SC#4) (though this is far from perfect).

Wang *et al.* (2011) propose a fast and accurate approximate string search algorithm (ASS) that keeps track of the frequent mistakes extracted from training data (by use of an Aho-Corasick search tree) and proposes the most probable correction candidates. The method is not directly applicable to an MRL since it uses a vocabulary trie for validating the generated candidates, taking hundreds of megabytes in memory even if we try to place only the most frequently occurring word surface forms. In the used system, this lookup module is replaced by a language model. Figure 5 gives the general flow of the employed system. SC#4 is inspired by Linden and Pirinen (2014), in that it uses a language and an error model together in order to generate candidates. Candidates which are generated by the error model are validated using the language model and the best proposal is the candidate with minimum rule cost and maximum unigram probability. One may refer to the aforementioned references for further details.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 8. *Samples for complex error types*

Input	Normalized output	Modules needed
gdcem	*gideceğim* (I'll go)	Accent normalization + Vowel restoration
cagladan	*Çağla'dan* (from Çağla)	Diacritization + Proper noun detection
gıtmıcem	*gitmeyeceğim* (I won't go)	Accent normalization + Diacritization
$ekeeeeerım	*şekerim* (my sweety)	Replacement rules + Diacritization

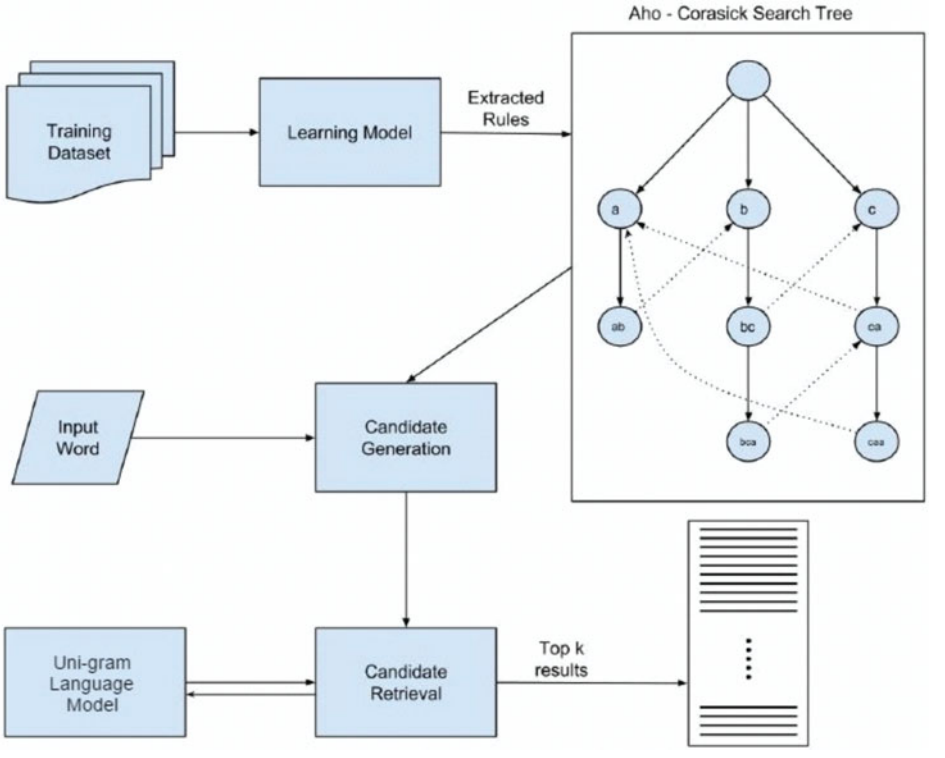

Fig. 5. (Colour online) Adaptation of ASS into Turkish (Torunoğlu-Selamet *et al.* 2016).

4.2.8 The cascaded model

The subsections of Section 4.2 introduced each component used in the candidate generation layer. Although each module performs well on its targeted error types (see Section 5.4 for a detailed evaluation), there exist many cases which may not be normalized by a single module but rather their joint work. Table 8 gives some examples of such complicated errors and the modules that need to cooperate in order to normalize them.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

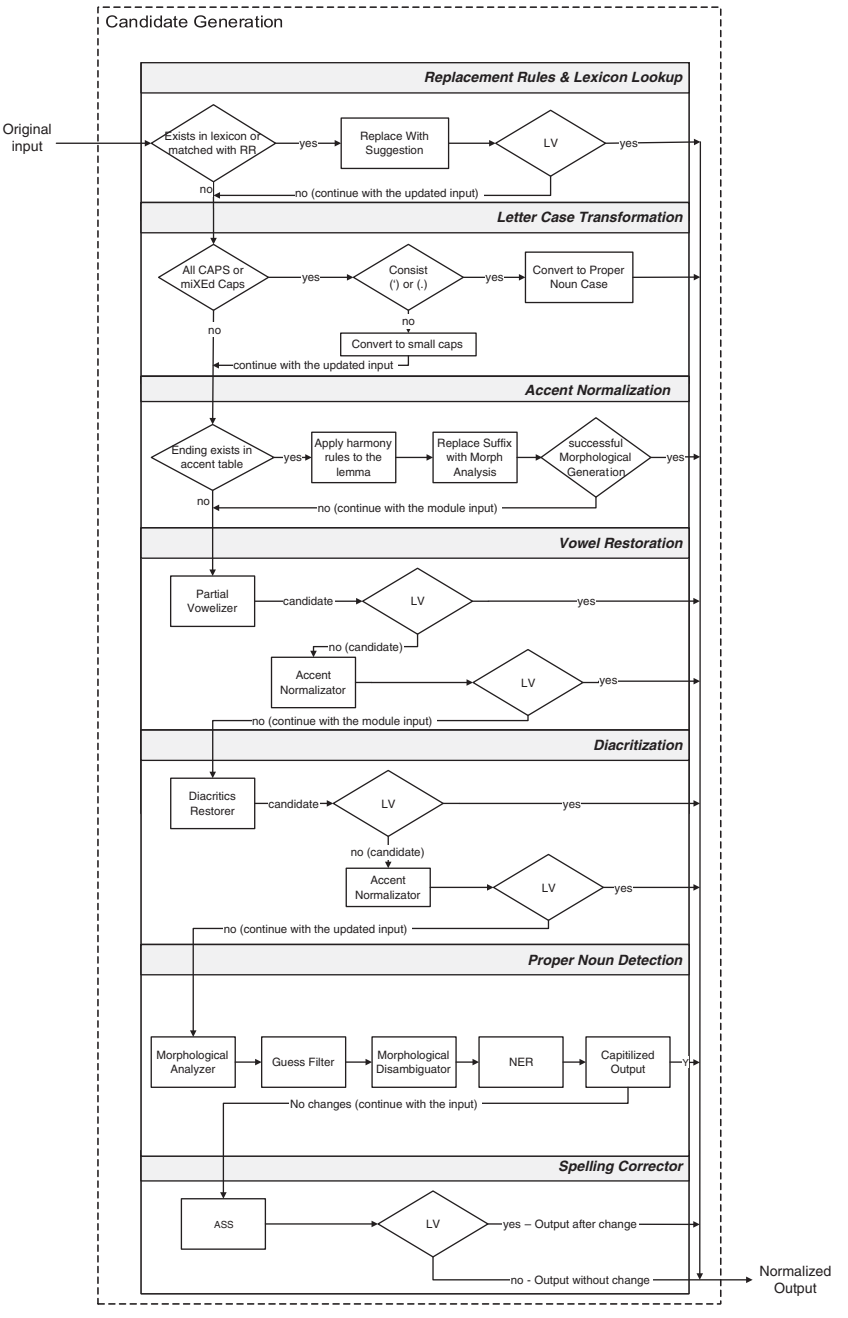

Fig. 6. Cascaded normalization model.

In this article, we introduce a cascaded model to integrate the presented seven normalization components. The design of the system flow is a sophisticated task since each module may affect the behavior of the other ones and a detailed analysis is needed during the design phase. Figure 6 shows the proposed integration of the seven normalization components for the case of Turkish. The figure also attempts to

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

reflect each module's internal structure, as explained in detail in previous sections. Accent normalization, vowel restoration and proper noun detection modules do not change their inputs while transferring to the next module in the event that the inputs are not validated by the LV. The remaining modules transfer their updates to the subsequent modules to be further normalized even if their outputs are not validated yet. Some modules operate multiple times both on the original input and the input updated from the former modules. These are the proper noun detection and accent normalization components.

5 Experimental results & discussions

In this section, we introduce the newly created test datasets collected from social media and provide detailed statistics about the existing normalization errors in them (Section 5.1). We then give the evaluation methodology and the models that we with which we compare (Section 5.2). The experimental results are then presented under two different subsections: (1) IWD performances (Section 5.3) and (2) candidate generation and integrated system performances including the analysis of individual normalization modules, and the error analysis of the entire system (Section 5.4).

5.1 Datasets

Torunoğlu and Eryiğit (2014) use a dataset of 1,200 manually normalized tweets. They use half of this data for validation purposes (as we also do in this study) and the remaining half as the test set. In order to compare our results with the pioneering work, we also test our system on this test set and provide results in the following sections. However, since the motivation in the collection of that rather small dataset was just to serve as a testbed during the design and the development of the normalization framework, these data collected from Twitter within a short time interval did not clearly reflect the natural composition of error types in real data and their usage frequencies in social media in general.

In this article, we introduce two new normalization datasets: one collected from Web 2.0 sentences and one from Twitter. *ITU Web Treebank* (IWT) (Pamay *et al.* 2015) is a recently released corpus manually annotated in multiple layers (normalization, morphology, named entities and syntax).[17] The corpus consists of 5,010 sentences collected in a balanced manner from blogs, customer reviews, forums, social media posts and reader comments on news media. *The Big Twitter dataset* (BTS) (see Title page footnote for further references) constitutes 26,149 manually normalized Turkish tweets extracted between May 10th, 2012 and January 20th, 2014 with keywords related to the telecommunication domain and well-known Turkish telecom companies. The manual normalization strategy was slightly different between the two datasets: In the creation of the IWT, human annotators were asked to normalize the sentences so that they also conform to orthographic conventions

[17] IWT is manually annotated using a new web-based version of the ITU Treebank annotation tool (Eryiğit 2007), which now includes an interface for manual text normalization as well.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 9. *Description of the datasets*

Data sets	# Sentences	# Tokens	# Normalized tokens
Validation set	600	6,322	2,957 (46.8%)
$Test_{small}$	600	6,507	1,821 (28.0%)
BTS	26,149	385,568	57,088 (14.8%)
IWT	5,010	39,152	5,101 (13.0%)

such as the capitalization of a sentence's first letter. This task is limited to the normalization of the existing tokens; the addition of punctuation and changes in sentence structure were not allowed. In this dataset, the Web 2.0 specific writings such as mentions, hashtags, etc. are also tagged manually with specific labels as explained in Section 4.2.1. For IWT, we used the updated version of our ITU Annotation Tool (Eryiğit 2007). Five annotators worked for the annotation of normalization of the new corpora. Before the annotation process started, the annotators had two weeks of supervised training in the new annotation framework, after which they started on annotating actual data. The IWT took four months to develop as the dataset was also annotated in three other areas. The annotators started from raw sentences and they first manually normalized the sentences that have been extracted from the web, and then assigned morphological tags for the tokens, before moving on to the dependency annotation. In this work, only the normalization annotation was used to test the normalization tool's performance. Also, in the beginning of the annotation process each sentence was intended to be annotated by at least two annotators, however, due to budget-related and time-related constraints, most of the sentences could only be annotated by a single annotator. As a result, it was not possible to measure interannotator agreement.

On the other hand, during the normalization of BTS, the human annotators were motivated to correct mostly the tokens which could not be analyzed by the morphological analyzer. As a result, the orthographic rules (e.g., sentence first letter capitalization) are mostly omitted if the used annotation environment (Eryigit *et al.* 2013) for this dataset provided a valid morphological analysis for the input word. This mostly affected the performance of the letter case transformation module discussed in the following sections. Table 9 gives statistics on the used datasets, such as the sentence counts (tweet counts for datasets collected from Twitter), token counts and manually normalized tokens. One may notice from this table that the normalized token counts are nearly a seventh of the whole token count for our two natural datasets (BTS and IWT), whereas it was nearly one-third of $Test_{small}$ and validation sets from Torunoğlu and Eryiğit (2014).

The design proposed in this article (i.e., the categorization of the candidate generation task into different components) is inspired by a manual investigation of normalization errors on the validation set. Thus, in order (1) to analyze the distributions of our decided error categories on real data and (2) to be able to evaluate the success of our candidate generation subcomponents on their targeted error types, we automatically categorized the normalization errors in our test sets

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

into different subsets. Since our aim in this automatic categorization is to obtain a rough distribution of error types, the sets are split in a mutually exclusive way, that is, an error is included in only a single set even if it would fall within the scope of more than one category. For example, the word *msj* is a frequently observed abbreviation for the word *mesaj* (message) in Web 2.0 and is included in our slang word and initialisms lexicon to be normalized directly through the lookup module, the same error might also be handled by the vowel restoration module. In such cases, the error is only included in one set, namely Set_{RR} for this example, explained below.

After the manual annotation, both datasets were aligned at sentence level. Therefore, as with the first task, we semiautomatically aligned the datasets at token level and extracted the normalization changes (erroneous input → normalized form). The changes are then automatically categorized into seven categories as following:

(1) *Replacement rules and lexicon lookup set* (Set_{RR}): In order to extract the normalization errors falling under this category, we select all the inputs existing in our normalization lexicons containing letters outside the Turkish alphabet or multiple occurrences of the same letter either consecutively or as matching our regular expressions such as for catching mentions, hashtags, smileys and vocatives. Although one may argue that the related module would perform excellently with this approach, there exist many cases where the human annotators did not normalize this type of errors as expected. For example, '.....' has often been manually normalized to a single dot '.', whereas the related module would normalize it to '...'. And, of course, there are also cases where the related module could not correctly normalize the incoming errors. To give an example, the input *olum* exists in our slang word and initialisms lexicon to be normalized as *oğlum* (dude), but it is normalized manually as *ölüm* (death) within the related context.

(2) *Letter case transformation set* (Set_{LCT}): In order to extract normalization errors falling under this category, we use both the input word and its manually normalized form. If two words are exactly the same except for letter cases, then the pair is collected under this category.

(3) *Proper noun detection set* (Set_{PND}): If the manually normalized form contains an apostrophe character whereas the original input does not, or they both contain an apostrophe character but the input is not written in proper noun case, then the sample is included in this set.

(4) *Diacritization set* (Set_{DA}): In order to extract normalization errors falling under this category, we first check if the original and normalized words have exactly the same length. If the lengths match and the differing letters are only Turkish-specific characters (*ı, İ, ş, ö, ç, ğ, ü*) and their ASCII-compliant counterparts (*i, I, s, o, c, g, u*), we select the sample as a diacritization error.

(5) *Vowel restoration set* (Set_{VR}): Similar to the previous set, we selected the samples where the only differences in words are the lack of vowels (*a, e, i, ı, o, ö, u, ü)* in the input word.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 10. *The distribution of normalization error types*

	$Test_{small}$		BTS		IWT	
Error Type	#of Error	Ratio (per cent)	#of Error	Ratio (per cent)	#of Error	Ratio (per cent)
Set_{LCT}	318	17.5	11,707	20.5	2,226	43.6
SetRR	888	48.8	5,188	9.1	440	8.6
SetPND	56	3.1	4,210	7.4	292	5.7
SetDA	241	13.2	21,081	36.9	1,407	27.6
SetVR	55	3.0	1,756	3.1	83	1.6
SetAN	64	3.5	4,461	7.8	117	2.3
SetRest	199	10.9	8,658	15.2	536	10.5
Total	1,821	100.0	57,088	100.0	5,101	100.0

(6) *Accent normalization set* (Set_{AN}): All input words ending with the accent suffixes in our list are collected under this set.
(7) *Rest set* (Set_{REST}): All samples not categorized with the above rules, are collected under the 'rest' set. In the individual module evaluation section, we test our spelling corrector on this subset, but one should notice that Set_{REST} contains the noisiest data where it is really hard to obtain the correct form within short edit distances.

Table 10 provides the obtained distributions on the used datasets. DA type errors have the highest portion in BTS, whereas in IWT, it is the LCT type errors. This is due to the abundance of sentences in IWT whose initial letters are required to undergo case transformation, as explained above. DA type errors still constitute the second highest portion in this dataset. Rest type errors have the third rank in both of the datasets. As expected, VR type error ratios are higher in the Twitter-specific dataset (BTS) than the general Web 2.0 data (IWT). The error distribution in $Test_{small}$ supports our concern about the inorganic nature of this small dataset having nearly half of its errors in Set_{RR}.

5.2 Evaluation methodology & compared models

Evaluation on the proposed normalization architecture is performed on four separate tiers: (1) the performance of the IWD, (2) the candidate generation performances (under perfect IWD as by Han and Baldwin (2011)) of the individual components on the automatically categorized subsets, (3) the performance of the entire candidate generation stage (with all the modules integrated) and (4) the overall system performance including IWD and candidate generation together. IWD is evaluated by providing the precision (P), recall (R) and F-measure (F). Candidate generation is evaluated by providing the accuracy scores of exact matches with human annotations. In this evaluation, the candidate generation process is applied to the manually normalized whole word set (as if IWD worked perfectly). The system performance is then similarly measured by the accuracy score; but this time the candidate generation

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

process is applied only to the ill-formed words automatically detected by the IWD stage.

Social media text normalization of Turkish is a very recent research area and, to the best of our knowledge, this is the first academic study making in-depth analyses of the problems and providing solutions. Therefore, there is no existing normalization system that can be compared with the proposed model. In order to have an idea about the performances of the available baseline models, we compare with two Turkish spelling correctors and the Giza++ lookup model generated previously by Torunoğlu and Eryiğit (2014):

- *MSWord*: The first spelling corrector that we compare with is the usage of the MSWord spelling checker (Microsoft 2010) as an automatic spelling corrector system. In order to do that, we developed an automatic spelling corrector that makes use of the MSWord API to obtain the first spelling suggestion (assumed to be the best) for a given input token. If there are no available suggestions for an input word then the input is treated as an IV word accepted by the system.
- *Zemberek*: Zemberek (Akın & Akın 2007) is an open source library for Turkish language processing. The second spelling corrector, similar to the former, again takes the top candidate from the spelling suggester of Zemberek.
- *GizaLookup*: Torunoğlu and Eryiğit (2014) use a statistical MT toolkit entitled GIZA++ (Och & Ney 2003) in order to automatically align normalization errors and their corrections within a small training set of Twitter posts. The dataset used was actually a small portion (4,049 tweets which were available at that time) of the BTS dataset introduced in this article. The aligned errors and their corrections are then stored in a lookup table to be extracted during the candidate generation stage.

5.3 Evaluation of ill-formed word detection

Table 11 gives the IWD results of the systems compared. As stated previously, the morphological analyzer (Eryiğit 2014) used has a very high coverage as might be noticed from the high precision scores (91.74 per cent on $Test_{small}$ and 92.48 per cent on IWT). When we investigate the false positives (the tokens detected by IWD not manually normalized by human annotators) on all of the three used datasets and especially on the BTS dataset (producing a rather low precision score 75.05 per cent when compared to our other two datasets),[18] we see that these are mostly due to the mistakenly left out normalization errors.

As explained previously, for MSWord and Zemberek, each automatically modified word is considered an ill-formed word detected by that system. In order to make a fair comparison, while calculating the performances of MSWord and Zemberek, the special token types explained in Section 4.2.1 (labeled with special labels starting with '@' in our system) are excluded from the evaluations in their favor. The last

[18] The difference of the annotation stage of the BTS dataset compared to other two datasets is discussed in Section 5.1.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 11. *Ill-formed word detection evaluation results*

	$Test_{small}$			BTS			IWT		
	P	R	F	P	R	F	P	R	F
MsWord	87.50	70.53	78.11	69.28	84.12	75.98	88.76	78.89	83.53
Zemberek	20.12	98.80	33.43	23.67	68.88	35.23	19.28	82.17	31.24
$IWD_{@excl}$	87.81	76.25	81.62	73.39	90.82	81.18	92.30	88.19	90.20
IWD	**91.74**	**83.31**	**87.33**	**75.05**	**91.51**	**82.47**	**92.48**	**88.44**	**90.41**

Table 12. *Different examples of real-word normalization errors*

Erroneous writing	Manually normalized to
sinirsiz (calm)	*sınırsız* (unlimited)
aksam (parts)	*akşam* (evening)
su (water)	*şu* (that)
iste (want)	*işte* (at work or here)
asık (hung)	*aşık* (in love)
dondum (I froze)	*döndüm* (I turned)
çalışıcım (my worker)	*çalışacağım* (I will work)

two lines in Table 11 present the scores of the newly proposed IWD system both by excluding ($IWD_{@excl}$) and including these tokens. In these experiments, we do not provide the results of the '*GizaLookup*' model since the IWD part of it is exactly the same as that used in the proposed model.

Normalization errors missed by the IWD module are due to the ambiguity caused by the emergence of words that are still valid (i.e., IV words) even after mistyping. Table 12 gives some examples of such real-word errors for Turkish. The ratio of the errors resulting in an actual Turkish word being measured between 8.86 per cent (for IWT) and 16.69 per cent (for $Test_{small}$) is still very low compared to the interval 25–40 per cent reported by Kukich (1992) as the ratio of errors resulting in an actual English word over the entire spelling errors. This may be due to the morphologically rich nature of the Turkish language where affixation may sometimes help decrease the ambiguities (Eryiğit & Adalı 2004). Although not as severe as in English, more sophisticated models (e.g., a noisy channel model) would be useful for detecting this type of error as in the case of spelling error detection systems.

The proposed straightforward approach in this article obtains an F-measure of 82.47 per cent on BTS and 90.41 per cent on IWT. The system is also evaluated on $Test_{small}$ and obtained an F-score of 87.33 per cent with a 16.33 percentage point improvement on previously reported results ($P = 71$ per cent, $R = 72$ per cent, $F = 71$ per cent) by Torunoğlu and Eryiğit (2014). This increase is due to the improved IWD system introduced in Section 4.1 and also to the used morphological analyzer (Eryiğit 2014), which has higher coverage compared to the previously used analyzer (Şahin, Sulubacak and Eryiğit 2013).

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 13. *Overall performances on OOV words*

Data sets	Ill-formed evaluation (R)	Candidate generation (Acc.)	System performance (Acc.)
$Test_{small}$	83.31	84.07	81.76
BTS	91.51	71.66	70.40 (CaseIns)
IWT	88.44	70.66	67.37

Table 11 reveals that the spelling checker of MSWord performs very close to the proposed model, whereas Zemberek produces very low precision scores by producing a large amount of false positives. The last two lines reveal that the two different evaluations of the proposed model ($IWD_{@excl}$ and IWD) are very close to each other for BTS and IWT datasets. The difference is large on $Test_{small}$ since this dataset has a large ratio of specially labeled tokens as given in Section 5.1.

5.4 Evaluation of candidate word generation and integrated system performance

The results of evaluated candidate generation and system performance on OOV words are provided in Table 13 for the overall datasets and in Table 14 (for BTS's categorized error subsets) and Table 15 (for IWT's categorized error subsets) for separate modules. In these tables, the column *Per-Module Success* provides the performances (in terms of accuracy scores) of the individual candidate generation components introduced in Section 4.2 on each related subset (Section 5.1), that is, the Replacement Rules & Lexicon Lookup module on Set_{RR}, the Letter Case Transformation on Set_{LCT}, the Accent Normalization module on Set_{AN}, the Vowel Restoration module on Set_{VR}, the Diacritization module on Set_{DA}, the Proper Noun Detection on Set_{PND} and, finally, the Spelling Correction module on Set_{REST}. The IWD recall values for each subset are also provided under the column *Ill-formed Detection*. The performance of the integrated candidate generation system is again evaluated on each subset as well as the entire dataset (the last rows of the tables). The results of this evaluation are provided under the column *Candidate Generation* (in terms of accuracy scores). In this evaluation, the system is applied directly to the input words without passing them through the IWD filtering in order to see the candidate generation performance alone. The overall system performances together with IWD and CG is then provided under the last columns of the tables as accuracy scores. As stated previously in Section 5.1, since BTS manual annotation differs from IWT in the sense of capitalization and special labeling of vocatives, mentions, keywords, etc., a case insensitive evaluation accepting all the specially labeled tokens as correct is carried out for all evaluations of BTS. For IWT, a stricter evaluation is performed for each step by calculating the exact match of automatic normalization output with human annotation. The system performance for IWT is provided by both the strict evaluation and the lighter evaluation of BTS (*System Perf.*) in order to compare with the BTS results. The overall system performance (Table 13) is 70.40 per cent on BTS and 67.37 per cent for IWT (77.07 per cent with the lighter

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 14. *Evaluation results of different modules on big twitter set (BTS)*

Test **Sets**	# of words	Per-module success (Acc.)	Ill-formed detection (R)	Candidate generation (Acc.)	System perf.$_{\text{CaseIns}}$ (Acc.)
Set_{RR}	5,188	78.49	96.23	80.59	77.83
Set_{LCT}	11,707	100.00	86.81	77.21	78.00
Set_{AN}	4,461	64.49	98.22	74.36	73.55
Set_{VR}	1,756	84.62	94.87	85.59	81.78
Set_{DA}	21,081	96.84	90.31	90.28	84.25
Set_{PND}	4,210	42.61	88.44	39.34	39.30
Set_{REST}	8,685	22.45	78.99	27.63	25.48

Table 15. *Evaluation results of different modules on ITU Web treebank (IWT)*

Test sets	# of words	Per-module success (Acc.)	Ill-formed detection (R)	Candidate generation (Acc.)	System perf. (Acc.)	System perf.$_{\text{CaseIns}}$ (Acc.)
Set_{RR}	440	85.68	96.81	86.82	86.82	92.50
Set_{LCT}	2,226	89.04	83.09	71.34	77.54	88.14
Set_{AN}	117	62.39	98.29	70.94	70.94	71.79
Set_{VR}	83	73.49	74.69	84.34	69.88	69.87
Set_{DA}	1,407	94.46	90.90	90.76	85.29	86.42
Set_{PND}	292	39.57	61.19	38.31	37.81	44.77
Set_{REST}	536	26.12	78.21	27.61	25.93	27.42

evaluation). As expected, the system performance results are lower than the CG alone due to the false negatives of the IWD process.

When the per-module successes are investigated, the most successful component is the diacritization module, which yields a score of 96.84 per cent on BTS's Set_{DA} and 94.46 per cent on Set_{DA} of IWT. Due to the lighter evaluation of BTS, LCT's performance is 100 per cent as expected for this dataset. As a Twitter-specific trend, the samples in Set_{VR} are much higher in BTS than IWT (1,756 compared to 83). The vowel restoration module is observed to be more successful on BTS (84.62 per cent).[19] As expected, the spelling corrector performances are very low on the subsets Set_{REST} containing the most difficult normalization errors. The success of the replacement rule and lexicon lookup module is higher for IWT (85.68 per cent) than for BTS (78.49 per cent) due to the higher number of specially labeled tokens during manual annotation. The performances of accent normalization are similar for both of the datasets (64.49 per cent for BTS and 62.39 per cent for IWT).

[19] We observed that the newly introduced constrained vowel restoration approach provided a 23.28 per cent improvement over the original unconstrained model from Adalı and Eryiğit (2014). We observe that in BTS, the words which include no vowel at all comprise only 16 per cent of Set_{VR}.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

The proper nouns related to the telecom brands in BTS are less prone to be confused with Turkish common nouns. That is noticeable from the higher per-module success of proper noun detection on BTS (42.61 per cent) than on IWT (39.57 per cent) as well as the IWD performance of 88.44 per cent for BTS's Set_{PND} versus 61.19 per cent for IWT's Set_{PND}. We also tested our newly proposed proper noun detection model over the gold standard named entity annotation of IWT (Şeker & Eryiğit 2017). For measuring the success of the proposed approach, we synthetically created a test dataset from IWT by lowercasing the named entities and then trying to automatically detect and normalize them. As a baseline model, we also tested with a token based simple lookup strategy by replacing the NER recognition layer by a direct lookup from the newly added gazetteers (with threshold approach – Section 4.2.3) and application of the same normalization stages to the words with a recognized lemma. While we obtain an F-measure of 17 per cent with the described baseline model, the precision, recall and F-measure scores of the proposed approach are measured as $P=$ 90 per cent, $R=$57 per cent and $F=$70 per cent. The positive impact of the addition of the new gazetteers to the proposed model is reflected a 4 percentage point improvement.

As may be observed from Table 14 and Table 15, integrated candidate generation (column 5) provides an improvement on some of the stand-alone usages of submodules (column 3). These improvements are due to successful collaboration of different modules in solving complex error types as exemplified in Table 8. For example, in BTS, accent normalization module success on Set_{AN} improves from 64.49 per cent to 74.36 per cent with the help of the other modules. There exist also cases where the performances drop with the integration of the modules. For example, for Set_{DA} in IWT, the performance drops from 94.46 per cent to 90.76 per cent. This is due to the cases where the previous modules in the cascaded flow (Figure 6) ruin the input before the diacritization module takes action. To give an example, the input token *simdi* which is manually normalized to the word *şimdi* (now) is correctly solved by the stand-alone application of the diacritization module. On the other hand, in the integrated model, it is first the vowel restorer which takes the action and converts the input to *simidi* (*simit* in accusative case)[20] which is also a valid Turkish word. Another example is for Set_{LCT} and Set_{PND} in BTS. Since Set_{LCT} contains only letter case usage errors and we make a case-insensitive evaluation for BTS, the per-module success on this set is 100 per cent. The decrease (100 per cent → 77.21 per cent) observed when these words are directly pushed to the candidate generation stage is apparent, since CG will most probably change these inputs by treating them as ill-formed words by default. That is why we see again a slight increase on the system performance in the same line (77.21 per cent → 78.00 per cent) by the addition of IWD which filters out some of the words (100−86.81 = 13.19 per cent) as in-vocabulary words.

As the next step of our investigation, we measure the impact of each candidate generation submodule on the overall system performance. For this investigation, we use an ablation test which measures the overall system performance by leaving out

[20] *simit* is a circular bread, typically encrusted with sesame seeds.

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

Table 16. *The impact of individual modules on whole system performance*

	IWT system perf. (Acc.)	BTS system perf. (Acc.)
All modules	**67.37**	**70.40**
− Replacement rules & lexicon lookup	↓3.96	↓8.03
− Letter case transformation	↑0.19	↓3.31
− Accent normalization	↓0.93	↓8.05
− Vowel restoration	↓0.05	↓3.86
− Diacritization	↓5.70	↓13.84
− Proper noun detection	↓2.41	↓5.39
− Spelling correction	↓0.94	↓3.42

Table 17. *Comparison of system performances with related works*

Systems	Test_{small} (Acc.)	BTS (Acc.)	IWT (Acc.)
MSWord	36.64	41.98	55.34
Zemberek	2.31	6.16	3.37
GizaLookup	31.63	38.44	10.23
$\text{This study}_{@excl}$	72.01	70.40	66.71
This study	**81.76**	**70.40**	**67.37**

a single CG module one at a time. Table 16 gives the results of these experiments. The first line of the table restates the overall system performance with the entirety of modules in action. The performances resulting from leaving out each module compared with that of the integrated system is given in the following lines. We use McNemar's paired test in order to measure the statistical significance of the differences given in the table. The results reveal that all of our modules have significant impact on the overall system (with $p < 10^{-5}$) except for the letter case transformation module for the IWT dataset whose absence causes a slight increase in the scores which are not statistically significant. The spelling correction, although not performing very well on Set_{REST}, has a positive impact on the overall system and its absence causes a drop of almost 1 percentage point in IWT and 3.5 in BTS on the performances. The vowel restoration module having the lowest impact still causes a statistically significant decrease (with $p < 10^{-5}$) with its absence. The effect is obvious even with case-insensitive evaluation on BTS and is due to the early normalization of proper nouns such as *john'dan* (example given in Section 4.2.2) in the cascaded flow, in which case the input word is not touched by further modules.

Table 17 gives the comparison of system performances with the introduced baseline systems. As expected, the proposed approach outperforms all the baselines by large margins. It may be noticed from the table that, although MsWord results are very low on Test_{small} (36.64) and BTS (41.98), the result is much higher for IWT (55.34). This is due to the high proportion of sentence-initial letter corrections in IWT

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

(Table 10 nearly 80 per cent of 2,226 LCT type errors are of this type). Since this type of error is very rare in BTS, the performance remains only at 41.98 per cent on this dataset compared to the 70.40 per cent success of the proposed model. Since the GizaLookup model is composed of a small subset of BTS, its success is higher on this dataset than the others. The success on $Test_{small}$ is 81.76 per cent, which provides an improvement of 11.28 percentage points over the previously reported results (70.48 per cent) (Torunoğlu & Eryiğit 2014).

6 Conclusion & future works

In this article, a cascaded model for social media text normalization of Turkish is presented. The model has two main parts: IWD and CG consisting of seven normalization layers: letter case transformation, replacement rules & lexicon lookup, proper noun detection, diacritization, vowel restoration, accent normalization and spelling correction. Two manually annotated datasets (BTS from Twitter consisting of 57K and IWT from Web 2.0 in general consisting of 5K manually normalized errors) are introduced and used during experimental analysis. An in-depth analysis of error type distributions on these datasets is performed in order to shed light on the nature of social media normalization problems. The proposed normalization approach obtained a 70.40 per cent accuracy score on BTS and 67.37 per cent (77.07 per cent case insensitive comparison) on IWT. The web interface (Eryiğit 2014) of the proposed system (as a SaaS) and all of the used and introduced resources are available for researchers via `http://tools.nlp.itu.edu.tr`.

This pioneering study aimed to investigate and reveal the necessary components for text normalization of a morphologically very rich language, Turkish, and tried to pave the way for further research in the field by providing necessary language resources. The design of a cascaded model is a complicated task and requires heavy manual error analysis since each normalization component affects the behavior of the others. Such situations were exemplified and discussed in detail within the article. However, our error analyses reveal that there is still room for improvement on the orchestration of normalization components. Since the proposed model is kind of a greedy search, the first valid (IV) solution is selected as the output of the system even if a better one may be found by the following components. This may be either by the action of another component prior to the acting one (e.g., the OOV input word *baslamadan* is supposed to be normalized by the diacritic restoration component as *başlamadan* – 'before starting' in the gold standard dataset, however, it is normalized as *basılamadan* – 'without being able to be pushed' by the vowel restoration component) or by the additional action of another component on top of the acted one (e.g., the OOV input word *basabas* is supposed to be normalized by both the diacritic restoration and spelling correction components as *başbaşa* – 'tête-à-tête', however, it is normalized as *başabaş* – 'on equal terms' as the IV output of the diacritic restoration component alone). As future work, we plan to work on more sophisticated integration models of the proposed normalization components which are each proven to have significant impacts on the normalization task. A weighted integration (such as a logarithmic opinion pool (Smith, Cohn and Osborne 2005))

https://doi.org/10.1017/S1351324917000134 Published online by Cambridge University Press

of different modules or module combinations as well as language models (for the reranking of possible outputs) is worthy of investigation for further improvements in the field.

As in all semantic related tasks, the inclusion of wider context into the system decisions is an important topic that deserves investigation for the normalization task as well. For example, our vowel restorer, which takes only the neighboring tokens into account as context information, is unsuccessful in producing the gold-standard human annotation (*güzeli* – 'the beautiful') for the OOV input word 'gzli'. Instead, it produces a closer (in terms of minimum edit distance) solution (*gizli* – 'secret') which may be also correct in the narrower context:

En gzli fen
('the most' 'secret/beautiful' 'science')
'My favorite one is science'

This could only be disambiguated by a human by looking at the previous sentence: 'Which is your favorite lesson?'.

The automatic detection of code-switching between Turkish and foreign languages (especially English, which seems to be the most frequent pair in social media) is an open research topic that has not been explored enough in the literature. Typological differences of Turkish and nonstandard penetration of foreign words into the language under phonological and inflectional changes may result in complex situations for the normalization task. For example, the English word 'servers' could be seen as 'sörvırlar' or 'serverlar' in a Turkish post where the word server is either tried to be spelled in Turkish as it is pronounced (sörvır) and then inflected with the plural suffix -lar, or the reader is assumed to know the pronunciation of the English spelling and then the appearing inflection seems to violate the Turkish vowel harmony rules in the written word (serverlar) but not during its pronunciation. As a result, we believe the automatic detection of code-switching and its normalization is an important research topic which deserves investigation in the future.

Acknowledgments

All proposed sub-modules used in candidate generation layer are either developed during this study or used from recent research outputs (Eryiğit 2014; Torunoğlu & Eryiğit 2014; Adalı & Eryiğit 2014; Torunoğlu-Selamet *et al.* 2016) produced within the same research project, *Parsing Turkish Web 2.0 Sentences* supported by TUBITAK (The Scientific and Technological Research Council of Turkey) 1001 (grant no: 112E276) and part of the ICT COST Action IC1207. The authors would like to thank the project members Mehmet Talha Çakmak, Tugay İlbay, Ozan Arkan Can, Eren Bekar, Kübra Adalı, Umut Sulubacak and Tuğba Pamay for their valuable discussions and help during the development of this work. Two new data sets are introduced in this article: ITU Web Treebank (IWT) (Pamay *et al.* 2015) will be available to the research community by the ITU NLP Group (Istanbul Technical University Natural Language Processing Research Group) via `http://tools.nlp.itu.edu.tr/Datasets`. The authors want to thank Turkcell

Global Bilgi Call Center for sharing the manually normalized Twitter set (BTS) from the telecommunication domain. The data set produced as a result of another R&D project on Sentiment Analysis of Turkish Tweets (TUBITAK-TEYDEB (The Scientific and Technological Research Council of Turkey – Technology and Innovation Funding Programs Directorate) project (grant number: 3120605)) will be provided by *Turkcell Global Bilgi* Information Technology Department for further research. Finally, we want to thank our three anonymous reviewers for insightful comments and suggestions, and Damien Jade Duff for proofreading that helped us improve the final version of the article.

References

Adalı, K., and Eryiğit, G. 2014. Vowel and diacritic restoration for social media texts (LASM) at EACL. In *Proceedings of 5th Workshop on Language Analysis for Social Media*, Gothenburg, Sweden, pp. 53–61.

Ageno, A., Comas, P. R., Padró, L., and Turmo, J. 2013. The TALP-UPC approach to Tweet-Norm 2013. In *Proceedings of the Tweet Normalization Workshop (TWEET-NORM) at SEPLN*, Madrid, Spain, p. 58.

Akhtar, Md S., Sikdar, U. K., and Ekbal, A. 2015. IITP: multiobjective differential evolution based Twitter named entity recognition. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 61–7.

Akın, A. A., and Akın, M. D. 2007. Zemberek, an open source nlp framework for Turkic languages.

Alegria, I., Aranberri, N., Comas, P. R., Fresno, V., Gamallo, P., Padró, L., San Vicente, I., Turmo, J., and Zubiaga, A. 2015. Tweetnorm: a benchmark for lexical normalization of Spanish tweets. *Language Resources and Evaluation* **49**(4): 883–905.

Alegria, I., Aranberri, N., Fresno, V., Gamallo, P., Padró, L., San Vicente, I., Turmo, J., and Zubiaga, A. 2013. Introducción a la tarea compartida tweet-norm 2013: normalización léxica de tuits en Español. In *Proceedings of the Tweet Normalization Workshop (TWEET-NORM) at SEPLN*, Madrid, Spain, pp. 1–9.

Alex, B., Dubey, A., and Keller, F. 2007. Using foreign inclusion detection to improve parsing performance. In *Proceedings of EMNLP-CONLL*, Prague, Czech, pp. 151–60.

Aw, A., Zhang, M., Xiao, J., and Su, J. 2006. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL*. Morristown, NJ, USA, pp. 33–40.

Baldwin, T., Kim, Y.-B., de Marneffe, M. C., Ritter, A., Han, B., and Xu, W. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: twitter lexical normalization and named entity recognition. In *Proceedings of ACL-IJCNLP 2015*, Beijing, China, p. 126.

Baldwin, T., and Li, Y. 2015. An in-depth analysis of the effect of text normalization in social media. In *Proceedings of NAACL*, Denver, Colorado, pp. 420–9.

Beaufort, R., Roekhaut, S., Cougnon, L.-A., and Fairon, C. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of ACL '10,* Stroudsburg, PA, USA, pp. 770–9.

Beckley, R. 2015. Bekli: a simple approach to Twitter text normalization. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 82–6.

Berend, G., and Tasnádi, E. 2015. Uszeged: correction type-sensitive normalization of English tweets using efficiently indexed n-gram statistics. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 120–5.

Blevins, T., Kwiatkowski, R., Macbeth, J., McKeown, K., Patton, D., and Rambow, O. 2016. Automatically processing tweets from gang-involved youth: towards detecting loss and aggression. In *Proceedings of COLING*. Osaka, Japan, pp. 2196–206.

Clark, E., and Araki, K. 2011. Text normalization in social media: progress, problems and applications for a pre-processing system of casual English. *Procedia-social and Behavioral Sciences* **27**: 2–11.

Cook, P., and Stevenson, S. 2009. An unsupervised model for text message normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity at NAACL-HLT*, Stroudsburg, PA, USA, pp. 71–8.

Crystal, D. 2008. *Txtng: The gr8 db8*. OUP Oxford, New York.

Das, A., and Gambäck, B. 2013. Code-mixing in social media text: the last language identification frontier. *Traitement Automatique des Langues (TAL): Special Issue on Social Networks and NLP* **54**(3): 65–79.

De Clercq, O., Desmet, B., Schulz, S., Lefever, E., and Hoste, V. 2013. Normalization of Dutch user-generated content. In *Proceedings of Recent Advances in Natural Language Processing*, Hissar, Bulgaria, pp. 179–88.

Doval Mosquera, Y., Vilares, J., and Gómez-Rodríguez, C. 2015. Lysgroup: adapting a Spanish microtext normalization system to English. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 99–105.

Eger, S., *et al*. 2016. A comparison of four character-level string-to-string translation models for (OCR) spelling error correction. *The Prague Bulletin of Mathematical Linguistics* **105**(1): 77–99.

Egidio, Y. M. O. F. P., and Coupé, M. C. 2013. A quantitative and typological approach to correlating linguistic complexity. In *Proceedings of the 5th Conference on Quantitative Investigations in Theoretical Linguistics*, University of Leuven, pp. 71–5.

Eisenstein, J. 2013a. Phonological factors in social media writing. In *Proceedings of the Workshop on Language Analysis in Social Media*, Atlanta, Georgia: Association for Computational Linguistics, pp. 11–9.

Eisenstein, J. 2013b. What to do about bad language on the internet. In *Proceedings of NAACL-HLT*, Atlanta, Georgia, pp. 359–69.

Eryiğit, G. 2007. ITU treebank annotation tool. In *Proceedings of Workshop on Linguistic Annotation (LAW) at ACL*, Prague, Czech, pp. 117–20.

Eryiğit, G. 2014. ITU Turkish NLP web service. In *Proceedings of the Demonstrations at EACL*, Gothenburg, Sweden, pp. 1–8

Eryiğit, G., and Adalı, E. 2004. An affix stripping morphological analyzer for Turkish. In *Proceedings of the International Conference on Artificial Intelligence and Applications*, Inssbruck, pp. 299–304.

Eryigit, G., Cetin, F. S., Yanık, M., Temel, T., and Çiçekli, I. 2013. Turksent: a sentiment annotation tool for social media. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse at ACL*, Sofia, Bulgaria, pp. 131–4.

Eskander, R., Al-Badrashiny, M., Habash, N., and Rambow, O. 2014. Foreign words and the automatic processing of Arabic social media text written in roman script. In *Proceedings of the 1st Workshop on Computational Approaches to Code Switching at ACL*, Doha, Qatar, pp. 1–12.

Gal, Y. 2002. An HMM approach to vowel restoration in Arabic and Hebrew. In *Proceedings of Workshop on Computational Approaches to Semitic Languages at ACL*, Stroudsburg, PA, USA, pp. 1–7.

Hakkani-Tür, D. Z., Oflazer, K., and Tür, G. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proceedings of COLING* Stroudsburg, PA, USA, pp. 285–91.

Han, B., and Baldwin, T. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of ACL-HLT*, Portland, Oregon, USA, pp. 368–78.

Han, B., Cook, P., and Baldwin, T. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)* **4**(1): 5:1–27.

Hassan, H., and Menezes, A. 2013. Social text normalization using contextual graph random walks. In *Proceedings of ACL*, Sofia, Bulgaria, pp. 1577–86.

Ingason, A. K., Jóhannsson, S. B., Rögnvaldsson, E., Loftsson, H., and Helgadóttir, S. 2009. Context-sensitive spelling correction and rich morphology. In *Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA)*, Odense, Denmark, pp. 231–4.

Jahjah, V., Khoury, R., and Lamontagne, L. 2016. *Word Normalization using Phonetic Signatures*, pp. 180–5. Cham: Springer International Publishing.

Jhamtani, H., Bhogi, S. K., and Raychoudhury, V. 2014. Word-level language identification in bi-lingual code-switched texts. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation*, Phuket,Thailand, pp. 348–57.

Jia, Y., Huang, D., Liu, W., Dong, Y., Yu, S., and Wang, H. 2008. Text normalization in Mandarin text-to-speech system. In *Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4693–6. IEEE, Las Vegas.

Jin, N. 2015. Ncsu-sas-ning: candidate generation and feature engineering for supervised lexical normalization. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 87–92.

Kaufmann, M., and Kalita, J. 2010. Syntactic normalization of Twitter messages. In *Proceedings of the 8th International Conference on Natural Language Processing (ICON)*, Chennai, India, pp. 1–7

Khan, O. A., and Karim, A. 2012. A rule-based model for normalization of sms text. In *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI)*, Athens, Greece, pp. 634–41.

Kobus, C., Yvon, F., and Damnati, G. 2008. Normalizing sms: are two metaphors better than one? *Proceedings of COLING*, Manchester, UK, pp. 441–8.

Kukich, K. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)* **24**(4): 377–439.

Labov, W. 1969. *A Study of Non-Standard English*, Educational resources information center. ERIC Clearinghouse for Linguistics, Washington. D.C.

Lacoste, V. 2012. *Phonological Variation in Rural Jamaican Schools*, Creole language library. John Benjamins Publishing Company, Amsterdam.

Lafferty, J., McCallum, A., and Pereira, F. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning* (ICML), San Francisco, CA, USA, pp. 282–9.

Leeman-Munk, S., Lester, J., and Cox, J. 2015. Ncsu_sas_sam: deep encoding and reconstruction for normalization of noisy text. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 154–61.

Leeman-Munk, S. P. 2016. *Morphosyntactic Neural Analysis for Generalized Lexical Normalization*. Ph.D. thesis, North Carolina State University.

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

Limsopatham, N., and Collier, N. 2015. Adapting phrase-based machine translation to normalise medical terms in social media messages. In *Proceedings of EMNLP*, Lisbon, Portugal, pp. 1675–80.

Liu, F., Weng, F., and Jiang, X. 2012. A broad-coverage normalization system for social media language. In *Proceedings of ACL*, Stroudsburg, PA, USA, pp. 1035–44.

Lui, M., Lau, J. H., and Baldwin, T. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics* **2**: 27–40.

Max, A., and Wisniewski, G. 2010. Mining naturally-occurring corrections and paraphrases from Wikipedia's revision history. In *Proceedings of LREC*, Valletta, Malta, pp. 3143–8.

McCallum, A. K. 2002. *Mallet: A Machine Learning for Language Toolkit*. http://mallet.cs.umass.edu.

McKean, E. 2005. *The New Oxford American Dictionary*, vol. 2. New York: Oxford University Press.

Melero, M., Costa-Jussà, M. R., Lambert, P., and Quixal, M. 2016. Selection of correction candidates for the normalization of Spanish user-generated content. *Natural Language Engineering* **22**(1): 135–61.

Microsoft. 2010. *Microsoft Word, Version 10.0*. Microsoft.

Min, W., and Mott, B. 2015. Ncsu_sas_wookhee: a deep contextual long-short term memory model for text normalization. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 111–9.

Muhammad, A., Wiratunga, N., and Lothian, R. 2015. Context-aware sentiment analysis of social media. In *Advances in Social Media Analysis*, Switzerland, pp. 87–104.

Nguyen, T.-T., Thi, P., Thanh T., and Tran, D.-D. 2010. A method for Vietnamese text normalization to improve the quality of speech synthesis. In *Proceedings of the 2010 Symposium on Information and Communication Technology*, New York, NY, USA, pp. 78–85.

Och, F. J., and Ney, H. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* **29**(1): 19–51.

Oflazer, K. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics* **22**(1): 73–89.

Pamay, T., Sulubacak, U., Torunoğlu-Selamet, D., and Eryiğit, G. 2015. The annotation process of the ITU web treebank. In *Proceedings of LAW Workshop at NAACL*, Denver, Colorado, pp. 95–101.

Panchapagesan, K., Talukdar, P. P., Krishna, N. S., Bali, K., and Ramakrishnan, A. G. 2004. Hindi text normalization. In *Proceedings of the 5th International Conference on Knowledge Based Computer Systems*, India, pp. 19–22.

Pennell, D., and Liu, Y. 2011. A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of the International Joint Conference on Natural Language Processing*, Chiang Mai, Thailand, pp. 974–82.

Pirinen, T. A., and Lindén, K. 2010. Finite-state spell-checking with weighted language and error models. In *Proceedings the Workshop on Creation and Use of Basic Lexical Resources for Less-Resourced Languages at LREC*, Valetta, Malta, pp. 13–8.

Pirinen, T. A., and Lindén, K. 2014. State-of-the-art in weighted finite-state spell-checking. In *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing*, Kathmandu, Nepal, pp. 519–32.

Porta, J., and Sancho, J.-L. 2013. Word normalization in Twitter using finite-state transducers. In *Proceedings of the Tweet Normalization Workshop (TWEET-NORM) at SEPLN*, Madrid, Spain, pp. 49–53.

Qian, T., Zhang, Y., Zhang, M., Ren, Y., and Ji, D. 2015. A transition-based model for joint segmentation, pos-tagging and normalization. In *Proceedings of EMNLP*, Lisbon, Portugal, pp. 1837–46.

Şahin, M., Sulubacak, U., and Eryiğit, G. 2013. Redefinition of Turkish morphology using flag diacritics. *Proceedings of the 10th Symposium on Natural Language Processing (SNLP-2013)*, Pukhet, Thailand, pp. 1–8.

Sak, H., Güngör, T., and Saraçlar, M. 2011. Resources for Turkish morphological processing. *Language Resources and Evaluation* **45**(2): pp. 249–61.

Saloot, M. A., Idris, N., and Mahmud, R. 2014. An architecture for Malay tweet normalization. *Information Processing & Management* **50**(5): pp. 621–33.

Sanches Duran, M., Volpe Nunes, M. das Graças, and Avanço, L. 2015. A normalizer for UGC in Brazilian Portuguese. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 38–47.

Sarikaya, R., Kirchhoff, K., Schultz, T., and Hakkani-Tur, D. 2009. Introduction to the special issue on processing morphologically rich languages. *IEEE Transactions on Audio, Speech, and Language Processing* **17**(5): 861–2.

Say, B., Zeyrek, D., Oflazer, K., and Özge, U. 2002. Development of a corpus and a treebank for present-day written Turkish. In *Proceedings of the 11th International Conference of Turkish Linguistics*, Northern Cyprus.

Schulz, S., Pauw, G. De, Clercq, O. De, Desmet, B., Hoste, V., Daelemans, W., and Macken, L. 2016. Multimodular text normalization of Dutch user-generated content. *ACM Transactions on Intelligent Systems and Technology* **7**(4): 1–22.

Şeker, G. A., and Eryiğit, G. 2012. Initial explorations on using CRFs for Turkish named entity recognition. In *Proceedings of COLING 2012*, Bombay, India, pp. 2459–74.

Şeker, G., and Eryiğit, G. 2017. Extending a CRF-based named entity recognition model for Turkish well formed text and user generated content. *Semantic Web Journal* **8**(5): 625–42.

Silfverberg, M., Kauppinen, P., and Lindén, K. 2016. Data-driven spelling correction using weighted finite-state methods. In *Proceedings of the Workshop on Statistical NLP and Weighted Automa*, Berlin, Germany, pp. 51–9.

Smith, A., Cohn, T., and Osborne, M. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of ACL*, Ann Arbor, Michigan, USA, pp. 18–25.

Solorio, T., Blair, E., Maharjan, S., Bethard, S., Diab, M., Ghoneim, M., Hawwari, A., AlGhamdi, F., Hirschberg, J., Chang, A., and Fung, P. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the 1st Workshop on Computational Approaches to Code Switching at ACL*, Doha, Qatar, pp. 62–72.

Sridhar, R., and Kumar, V. 2015. Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing at ACL*, Denver, Colorado, pp. 8–16.

Supranovich, D., and Patsepnia, V. 2015. Ihs_rd: lexical normalization for English tweets. *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 78–81.

Torunoğlu, D., and Eryiğit, G. 2014. A cascaded approach for social media text normalization of Turkish. In *Proceedings of the 5th Workshop on Language Analysis for Social Media at EACL*, Gothenburg, Sweden, pp. 62–70.

Torunoğlu-Selamet, D., Bekar, E., Ilbay, T., and Eryiğit, G. 2016. Exploring spelling correction approaches for Turkish. In *Proceedings of the 1st International Conference on Turkic Computational Linguistics at CICLING*, Konya, pp. 7–11.

Tsarfaty, R., Seddah, D., Goldberg, Y., Kübler, S., Candito, M., Foster, J., Versley, Y., Rehbein, I., and Tounsi, L. 2010. Statistical parsing of morphologically rich languages (SPMRL): what, how and whither. In *Proceedings of the 1st Workshop on Statistical Parsing of Morphologically Rich Languages at NAACL-HLT*, Stroudsburg, PA, USA, pp. 1–12.

Tür, G. 2000. *A Statistical Information Extraction System for Turkish*. PhD Thesis, Department of Computer Engineering and the Institute of Engineering and Science of Bilkent University, Ankara.

Tür, G., Hakkani-Tür, D., and Oflazer, K. 2003. A statistical information extraction system for Turkish. *Natural Language Engineering* **9**(2): 181–210.

Vilares, J., Alonso, M., and Vilares, D. 2013. Prototipado rápido de un sistema de normalización de tuits: una aproximación léxica. In *Proceedings of the Tweet Normalization Workshop (TWEET-NORM) at SEPLN*, Madrid, Spain, pp. 39–43.

Wagner, J., and Foster, J. 2015. Dcu-adapt: learning edit operations for microblog normalisation with the generalised perceptron. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL*, Beijing, China, pp. 93–8.

Wang, P., and Ng, H. T. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *Proceedings of NAACL-HLT*, Atlanta, Georgia, pp. 471–81.

Wang, Z., Xu, G., Li, H., and Zhang, M. 2011. A fast and accurate method for approximate string search. In *Proceedings of ACL-HLT*, Stroudsburg, PA, USA, pp. 52–61.

Xu, K., Xia, Y., and Lee, C.-H. 2015. Tweet normalization with syllables. In *Proceedings of ACL-IJCNLP*, Beijing, China, pp. 920–8.

Yang, Y., and Eisenstein, J. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of EMNLP*, Seattle, Washington, USA, pp. 61–72.

Yüret, D., and De La Maza, M. 2006. The greedy prepend algorithm for decision list induction. In *Proceedings of the 21st International Conference on Computer and Information Sciences*, Berlin, Heidelberg, pp. 37–46.

Zhang, C., Baldwin, T., Ho, H., Kimelfeld, B., and Li, Y. 2013. Adaptive parser-centric text normalization. In *Proceedings of ACL*, Sofia, Bulgaria, pp. 1159–68.

Zhang, Q., Chen, H., and Huang, X. 2014. Chinese-English mixed text normalization. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, New York, NY, USA, pp. 433–42.

Zitouni, I., Sorensen, J., and Sarikaya, R. 2006. Maximum entropy based restoration of Arabic diacritics. In *Proceedings of COLING-ACL*, Stroudsburg, PA, USA, pp. 577–84.