A Cascaded Approach for Social Media Text Normalization of Turkish

Dilara Torunoğlu
Dep. of Computer Eng.
Istanbul Technical University
Istanbul, Turkey
torunoglud@itu.edu.tr

Gülşen Eryiğit
Dep. of Computer Eng.
Istanbul Technical University
Istanbul, Turkey
gulsen.cebiroglu@itu.edu.tr

Abstract

Text normalization is an indispensable stage for natural language processing of social media data with available NLP tools. We divide the normalization problem into 7 categories, namely; letter case transformation, replacement rules & lexicon lookup, proper noun detection, deasciification, vowel restoration, accent normalization and spelling correction. We propose a cascaded approach where each ill formed word passes from these 7 modules and is investigated for possible transformations. This paper presents the first results for the normalization of Turkish and tries to shed light on the different challenges in this area. We report a 40 percentage points improvement over a lexicon lookup baseline and nearly 50 percentage points over available spelling correctors.

1 Introduction

With the increasing number of people using micro blogging sites like Facebook and Twitter, social media became an indefinite source for machine learning area especially for natural language processing. This service is highly attractive for information extraction, text mining and opinion mining purposes as the large volumes of data available online daily. The language used in this platform differs severely from formally written text in that, people do not feel forced to write grammatically correct sentences, generally write like they talk or try to impress their thoughts within a limited number of characters (such as in Twitter 140 characters). This results with a totally different language than the conventional languages. The research on text normalization of social media gained speed towards 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 results that morphologically rich languages such as Turkish differ severely from English and the methods tailored for English do not fit for these languages. It is the case for text normalization as well.

Highly inflectional or agglutinative languages share the same characteristic that a unique lemma in these languages may have hundreds of possible surface forms. This increases the data sparsity in statistical models. For example, it's pointed out in Hakkani-Tür et al. (2000) that, it is due to Turkish language's inflectional and derivational morphology that the number of distinct word forms is very large compared to English distinct word size (Table 1). This large vocabulary size is the reason why the dictionary[1] lookup or similarity based approaches are not suitable for this kind of languages. And in addition to this, it is not an easy task to collect manually annotated data which could cover all these surface forms and their related mistakes for statistical approaches.

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

Table 1: Vocabulary sizes for two Turkish and English corpora (Hakkani-Tür et al., 2000)

In this paper, we propose a cascaded approach for the social text normalization (specifically for Tweets) of Turkish language. The approach is a combination of rule based and machine learning components for different layers of normalization, namely; letter case transformation, replacement rules & lexicon lookup, proper noun detection, deasciification, vowel restoration, accent normalization and spelling correction. Following the work of Han and Baldwin (2011), we divided the work into two stages: ill formed word detection

[1]For these languages, it is theoretically impossible to put every possible surface form into a dictionary.

Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM) @ EACL 2014, pages 62–70,
Gothenburg, Sweden, April 26-30 2014.

and candidate word generation. Our contribution is: 1. a new normalization model which could be applied to other morphologically rich languages as well with appropriate NLP tools 2. the first results and test data sets for the text normalization of Turkish.

The paper is structured as follows: Section 2 and 3 give brief information about related work and morphologically rich languages, Section 4 presents our normalization approach and Section 5 the experimental setup, Section 6 gives our experimental results and discussions and Section 7 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 their combination. In these studies, a lexicon lookup is firstly employed for most common usage of slang words, abbreviations etc. and then a machine learning method is employed for the rest. Zhang et al. (2013) uses replacement rules and a graph based model in order to select the best rule combinations. Wang and Ng (2013) uses a beam search decoder. Hassan and Menezes (2013) propose an unsupervised approach which uses Random Walks on a contextual similarity bipartite graph constructed from n-gram sequences. In Han and Baldwin (2011), word similarity and context is used during lexicon lookup. Cook and Stevenson (2009) uses an unsupervised noisy channel model. Clark and Araki (2011) makes dictionary lookup. Liu et al. (2012) uses a unified letter transformation to generate possible ill formed words in order to use them in the training phase of a noisy channel model. Eisenstein (2013) analyzes phonological factors in social media writing.

Others, treating the normalization task as a machine translation (MT) problem which tries to translate from an ill formed language to a conventional one, form also another important group. For example the papers from Kaufmann and Kalita (2010), Pennell and Liu (2011), Aw et al. (2006) and Beaufort et al. (2010) may be collected under this group. Since the emergence of social media is very recent, only the latest 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 statuses contains mentions (@user_name), hashtags (#topic), variant number of emoticons (e.g. :) :@ <3 @>–) and special keywords (RT - retweet, DM - direct message etc.).

Although very rare, there are also some studies on languages other than English and these are mostly for speech recognition and SMS messages , e.g. Panchapagesan et al. (2004) for Hindi TTS, Nguyen et al. (2010) for Vietnamese TTS, Jia et al. (2008) for Mandarin TTS, Khan and Karim (2012) for Urdu SMS. To the best of our knowledge, our study is the first attempt for the normalization of social media data for morphologically rich languages.

3 Morphologically Rich Languages

Morphologically rich languages such as Turkish, Finnish, Korean, Hebrew etc., pose significant challenges for natural language processing tasks (Tsarfaty et al., 2013; Sarikaya et al., 2009). As stated previously, the highly productive morphology of these languages results in a very large number of word forms from a given stem. Table 2 lists only a few (among hundreds of possible) surface forms for the Turkish stem “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)

Table 2: Some surface forms for “ev” (*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 (OOV) rate
4. inflection gap for machine translation[2]

That is why, the normalization methods proposed so far (adapting MT or language models or

[2]Since, the number of possible word surface forms after inflections is very high, the alignment and translation accuracies in these languages are very badly affected.

lexicon lookup approaches) do not seem appropriate for the processing of morphologically rich languages, as in our case for Turkish.

4 The Proposed Architecture

We divide the normalization task into two parts: Ill-formed word detection and candidate generation. Figure 1 presents the architecture of the proposed normalization approach. The following subsections provide the details for both of these two parts and their components.

Before sending the input into these stages, we first use our tokenizer specifically tailored for Twitter for splitting the tweets into meaningful tokens. Our tokenizer is actually the first step of our normalization process since: 1. It intelligently splits the wrongly written word-punctuation combinations (e.g. "a,b" to [a , b]), while leaving "Ahmet'den" (*from Ahmet*) is left as it is since the apostrophe sign is used to append inflectional features to a proper noun.) 2. 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).

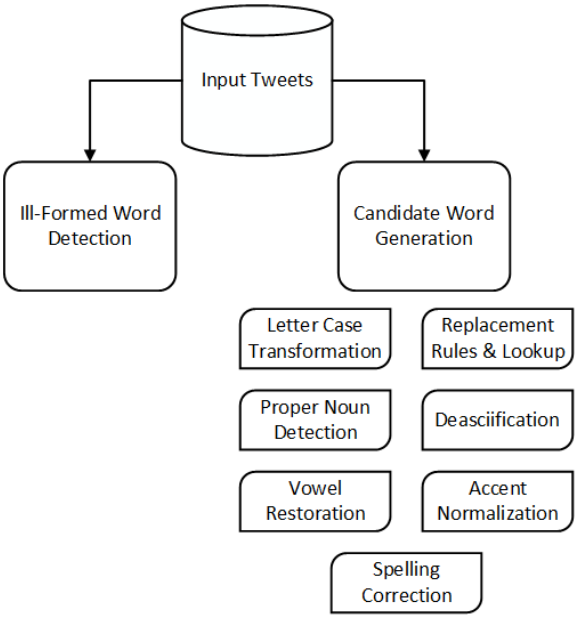

Figure 1: Normalization architecture

4.1 Ill-formed Word Detection

As stated earlier, since it is not possible to use a lexicon lookup table for morphologically rich languages, we use a morphological analyzer (Şahin et al., 2013) and an abbreviation list[3] and a list of 1045 abbreviations for controlling in-vocabulary (IV) words (labeled with a +NC "No Change" label for further use). By this way, we filter all the out-of-vocabulary (OOV) words and transfer them to the candidate generation process. Mentions (@user_name), hashtags (#topic), emoticons (:D) , vocatives ("ahahahaha") and keywords ("RT") are also assumed to be OOV words since we want to detect these and tag them with special labels to be later used in higher-level NLP modules (e.g. POS tagging, syntactic analysis).

4.2 Candidate Generation

In the candidate generation part, we have seven components (rule based or machine learning models) which work sequentially. The outputs of each of these components are controlled by the morphological analyzer and if the normalized form from a component becomes an IV word then the process is terminated and the output is labeled with a relevant tag (provided in Table 3). Otherwise, the candidate generation process continues with the next component over the original input (except for the "Letter Case Transformation" and "Replacement Rules & Lexicon Lookup" components where the input is replaced by the modified output although it is still not an IV word, (see Section 4.2.1 and 4.2.2 for details).

Label	Component
+NC	No Change
+LCT	Letter Case Transformation
+RR	Replacement Rules & Lexicon Lookup
+PND	Proper Noun Detection
+DA	Deasciification
+VR	Vowel Restoration
+AN	Accent Normalization
+NoN	No Suggested Normalization

Table 3: Component Labels

4.2.1 Letter Case Transformation

An OOV token, coming to this stage, may be in one of the 4 different forms: lowercase, UPPERCASE, Proper Noun Case or miXEd CaSe. If the token is in lowercase and does not possess any specific punctuation marks for proper nouns (i.e. ' (apostrophe) or . (period)) , it is directly

[3]obtained from TLA (Turkish Language Association) http://www.tdk.gov.tr/index.php?option=com_content&id=198:Kisaltmalar

transferred to the next stage without any change (e.g. umuttan (*from hope*)). If the token is in Proper Noun Case (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), left untouched (taking the label +NC) and excluded from all future evaluations.

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 abbreviations. These words are labeled with a "+LCT" label after the normalization. If the word does not contain any of these two marks, it is then converted into lowercase form and processed by the morphological analyzer as explained at the beginning of Section 4.2. It should be noted that all words going out from this component towards next stages are transformed into lowercase from this point on.

"ahmet'ten" – Proper Noun

"AHMET'TEN" – Proper Noun

"EACL."- Abbreviation

4.2.2 Replacement Rules & Lexicon Look-up

While normalizing the tweets, we have to deal with the following problems:

1. Slang words
2. Character repetition in interjections
3. Twitter-specific words
4. Emo style writing

We created a slang word lexicon of 272 words. This lexicon contains entries as the following: "kib" for "kendine iyi bak" (*take care of yourself*), "nbr" for "ne haber" (*what's up*). The tokens within the lexicon are directly replaced with their normalized forms.

Repetition of some characters within a word is a very common method to express exclamation in messages, such as in "lütfeeeennnn" instead of "lütfen" (*please*), "çooooooook" instead of "çok" (*very*) and "ayyyyy" instead of "ay" (*oh!*). We reduce the repeated characters into a single character in the case that the consecutive occurrence count is greater than 2.

The usage of Twitter-specific words such as hashtags ("#topic"), mentions ("@user_name"), emoticons (":)"), vocatives ("hahahhah", "höööööö") and keywords ("RT") also causes a host of problems. The recurring patterns in vocatives are reduced into minimal forms during the normalization process, as for "haha" instead of "hahahhah" and "hö" instead of "höööööö".

Emo style writing, as in the example "$eker 4you" instead of "şeker senin için" (*sweety, it's for you*), is another problematic field for the normalization task. We created 35 replacement rules with regular expressions in order to automatically correct or label the given input for Twitter-specific words and Emo style writing. Examples include "\$ $\rightarrow$ ş", "$\epsilon \rightarrow e$", "3 $\rightarrow$ e" and "!$\rightarrow$ i". Through these replacement rules, we are able to correct most instances of Emo style writing.

Our regular expressions also label the following token types by the given specific labels for future reference:

- **Mentions:** Nicknames that refer to users on Twitter are labeled as e.g. *@mention[@dida]*
- **Hashtags:** Hashtags that refer to trending topics on Twitter are labeled as e.g. *@hashtag[#geziparki]*
- **Vocatives:** Vocatives are labeled as e.g. *@vocative[hehe]*
- **Smileys:** Emoticons are labeled as e.g. *@smiley[:)]*
- **Twitter-specific Keywords:** Keywords like "RT", "DM", "MT", "Reply" etc. are labeled as e.g. *@keyword[RT]*

Figure 2 shows the normalized version of a tweet in informal Turkish that could be translated like "*@dida what's up, why don't you call #offended :(*", before and after being processed by this component. Although the word "aramıon" also needs normalization as "aramıyorsun" (*you don't call*), this transformation is not realized within the current component and applied later in the accent normalization component given in Section 4.2.6.

4.2.3 Proper Noun Detection

As previously stated, all OOV words coming to this stage are in lowercase. In this component, our aim is to detect proper nouns erroneously written in lowercase (such as "ahmetten" or "ahmetden") and convert them to proper noun case with correct formatting ("Ahmet'ten" for the aforementioned examples).

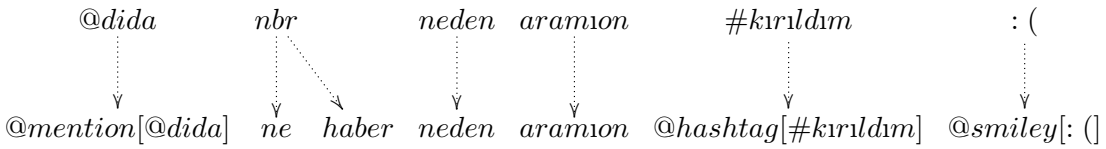

Figure 2: Normalization with Replacement Rules & Lexicon Look-up

For this purpose, we use proper name gazetteers from Şeker and Eryiğit (2012) together with a newly added organization gazetteer of 122 tokens in order to check whether a given word could be a proper noun. Turkish proper nouns are very frequently selected from common nouns such as "Çiçek" (*flower*), "Şeker" (*sugar*) and "İpek" (*silk*). Therefore, it is quite difficult to recognize such words as proper nouns when they are written in lowercase, as the task could not be accomplished by just checking the existence of such words within the gazetteers.

For our proper noun detection component, we use the below strategy:
1. We reduce the size of the gazetteers by removing all words with length ≤ 2 characters, or with a ratio value under our specified threshold (1.5). Ratio value is calculated, according to the formula given in Equation 1, considering the occurrence counts from two big corpora, the METU-Sabancı Treebank (Say et al., 2002) and the web corpus of Sak et al. (2011). Table 4 gives the counts for three sample words. One may observe from the table that "ahmet" occured 40 times in proper case and 20 times in lower case form within the two corpora resulting in a ratio value of 2.0. Since the ratio value for "umut" is only 0.4 (which is under our threshold), this noun is removed from our gazetteers so that it would not be transformed into proper case in case it is found to occur in lowercase form. A similar case holds for the word "sağlam" (healthy). Although it is a very frequent Turkish family name, it is observed in our corpora mostly as a common noun with a ratio value of 0.09.

$$ratio(w_n) = \frac{Occurence\_in\_Propercase(w_n)}{Occurence\_in\_Lowercase(w_n)} \quad (1)$$

2. We pass the tokens to a morphological analyzer for unknown words (Şahin et al., 2013) and find possible lemmata as in the example below. We then search for the longest possible stem within our gazetteers (e.g. the longest stem for "ahmetten" found within the name gazetteer is "ahmet"), and when a stem is found within the gazetteers, the initial letter of the stem is capitalized and the inflectional suffixes after the stem are separated by use of an apostrophe ("Ahmet'ten"). If none of the possible stems is found within the gazetteers, the word is left as is and transferred to the next stage in its original form.

Proper Case	Lowercase	Sense	Ratio
Sağlam=9	sağlam=100	healthy	Ratio=0.09
Umut=40	umut=100	hope	Ratio=0.4
Ahmet=40	ahmet=20	n/a	Ratio=2.0

Table 4: Example of Ratio Values

"ahmet +Noun+A3sg+Pnon+Abl"

"ahmette +Noun+A3sg+Pnom+Loc"

"ahmetten +Noun+A3sg+Pnon+Nom"

4.2.4 Deasciification

The role of the deasciifier is the reconstruction of Turkish-specific characters with diacritics (i.e. ı, İ, ş, ö, ç, ğ, ü) from their ASCII-compliant counterparts (i.e. i, I, s, o, c, g, u). Most users of social media use asciified letters, which should be corrected in order to obtain valid Turkish words. The task is also not straightforward because of the ambiguity potential in asciified forms, as between the words "yasa" (*law*) and "yaşa" (*live*). For this stage, we use the deasciifier of Yüret (Yüret and de la Maza, 2006) which implements the GPA algorithm (which itself is basically a decision tree implementation) in order to produce the most likely deasciified form of the input.

4.2.5 Vowel Restoration

There is a new trend of omitting vowels in typing among the Turkish social media users, in order to reduce the message length. In this stage, we process tokens written with consonants only (e.g. "svyrm"), which is how vowel omission often happens. The aim of the vowel restoration is the generation of the original word by adding vowels into the appropriate places (e.g. "svyrm" to "seviyorum" (*I love*)). We employed a vocalizer (Adalı

and Eryiğit, 2014) which uses CRFs for the construction of the most probable vocalized output.

4.2.6 Accent Normalization

In the social media platform, people generally write like they talk by transferring the pronounced versions of the words directly to the written text. Eisenstein (2013) also discusses the situation for the English case. In the accent normalization module we are trying to normalize this kind of writings into proper forms. Some examples are given below:

"gidicem" instead of "gideceğim"

(*I'll go*)

"geliyonmu?" instead of "geliyor musun?"

(*Are you coming?*)

In this component, we first try to detect the most common verb accents (generally endings such as "-cem, -yom, -çaz" etc.) used in social media and then uses regular expression rules in order to replace these endings with their equivalent morphological analysis. One should note that since in most of the morphologically rich languages, the verb also carries inflections related to the person agreement, we produce rules for catching all the possible surface forms of these accents.

Table 5 introduces some of these replacement rules (column 1 and column 3). As a result, the word "gidcem" becomes "git+Verb+Pos+Fut+A1sg"[4]. We then use a morphological generator and takes the corrected output (if any) "gideceğim" (*I'll go*) for "git+Verb+Pos+Fut+A1sg"[5].

We also have more complex replacement rules in order to process more complex accent problems. To give an example, the proper form of the word "gidiyonmu" is actually "gidiyor musun" (*are you going*) and in the formal form it is the question enclitic ("mu") which takes the person agreement ("-sun" 2. person singular) where as in the accent form the person agreement appears before "mu" as a single letter "gidiyo**n**mu".

[4]Please note that, we also change the last letter of the stem according to the harmonization rules of Turkish: the last letters "bcdg" are changed to "pçtk".

[5]the morphological tags in the table stands for: *+Pos*: Positive, *+Prog1*: Present continuous tense, *+A2sg*: 2. person singular, *+Fut*: Future tense, *+A1sg*: 1. person singular, *+A1pl*: 1. person plural

Accent endings	Correct endings	Morph. Analysis
+iyon	+iyorsun	+Verb+Pos+Prog1+A2sg
+cem	+eceğim	+Verb+Pos+Fut+A1sg
+caz	+acağız	+Verb+Pos+Fut+A1pl

Table 5: Accent Normalization Replacement Rules

4.2.7 Spelling Correction

As the last component of our normalization approach, we propose to use a high performance spelling corrector. This spelling corrector should especially give a high precision score rather than recall since the false positives have a very harming effect on the normalization task by producing outputs with a totally different meaning. Unfortunately, we could not find such a corrector for Turkish. We tested with an MsWord plugin and the spelling corrector of Zemberek (Akın and Akın, 2007) and obtained a negative impact by using both. We are planning to create such a spelling corrector as future work.

If an OOV word couldn't still be normalized at the end of the proposed iterative model (consisting 7 components), it is labeled with a "+NoN" label and left in its original input format.

5 Experimental Setup

In this section we provide information about our used data sets, our evaluation strategy and the used models in the experiments.

5.1 Data Sets

To test our success rates, we used a total of 1,200 tweets aligned and normalized manually. The manual alignment is a one-to-many token alignment task from the original input towards the normalized forms. To give an example, the slang usage "kib" will be aligned to 3 tokens ("kendine iyi bak" (*take care of yourself*)) on the normalized tweet. Although there are cases for many-to-one alignment (such as in "cats,dogs"), these are handled in the tokenization stage before the normalization. We used half of this data set as our validation set during the development of our proposed components and reserved the remaining 600 tweets (collected from a different time slot) as a totally unseen data set for using at the end. Table 6 provides some statistics over these data sets: the number of tweets, the number of tokens and the

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

Table 6: Description of the Data Sets

number of OOV tokens.

Besides the aforementioned datasets, we also had access to a much bigger Twitter data set consisting of 4,049 manually normalized tweets (Eryiğit et al., 2013) (59,012 tokens in total). The only difference of this data set is that the tweets are not aligned on token level as in the previously introduced data sets. That is why, it is not possible to use them for gold standard evaluation of our system. But in order to be able to have an idea about the performance of the previous approaches regarding lexicon lookup, we decided to automatically align this set and create a baseline lexicon lookup model for comparison purposes. (see the details in Section 5.3).

5.2 Evaluation Method

We evaluated our work both for ill formed word detection and candidate generation separately. For ill formed word detection, we provide precision (P), recall (R), f-measure (F) and accuracy (Acc.) scores. For candidate generation, we provide only the accuracy scores (the number of correctly normalized tokens over the total number of detected ill formed words).

5.3 Compared Models

To the best of our knowledge this study is the first attempt for the normalization of Turkish social media data. Since there are only spelling corrector systems available for the task we compared the proposed model with them. In other words, we compared 3 different models with our proposed system:

Model 1 (MsWord) is the model where we use an api for getting the MsWord Turkish spelling suggestions. Although this is not a tool developed for normalization purposes we wanted to see its success on our data sets. We accepted the top best suggestion as the normalized version for the input tokens.

Model 2 (Zemberek) (Akın and Akın, 2007) is also an open source spelling corrector for Turkish.

Model 3 (Lookup Table) is a model that we developed with the aim of creating a baseline lookup approach for comparison. For this purpose, we first used GIZA++ (Och and Ney, 2000) in order to automatically align the normalized tweets (using the 4,049 tweets' data set presented in Section 5.1) and created a lookup table with the produced aligned token sequences. We then used this lookup table to check for the existence of each ill formed word and get its normalized counterpart.

6 Experimental Results

Table 7 and Table 8 gives the results of the ill formed word detection for different systems for the validation set and the test set consecutively. In these experiments, we do not provide the results of the "Lookup Table" model since the ill formed detection part of it is exactly the same with our proposed model. For MsWord and Zemberek we considered each modified word as an ill formed word detected by that system. We can see from the tables that our proposed model has an f-measure of ill formed word detection 0.78. As it is explained in Section 4.1, our ill formed word detection approach is very straightforward and it uses only a morphological analyzer and an abbreviation list in order to detect OOV words. Thus, one may wonder why the scores for the proposed model are not very close to 1 although it outperforms all of its available rivals. This is because, there exists nearly 20% of the ill formed tokens which are not suspended to our morphological filter although they are manually annotated as ill formed by human annotators. This is certainly possible for morphologically rich languages since a word surface form may be the valid analysis of many stems. The ill formed word "çalışıcım" is a good example for this situation. Although this word will be understood by most of the people as the ill formed version of the word "çalışacağım" (*I'm going to work*), it is considered by the morphological analyzer as a valid Turkish word since although very rare, it could also be the surface form of the word "çalış" with additional derivational and inflectional suffixes "çalış+ıcı+m" meaning "*my worker*".

Systems	P	R	F	Acc.
MsWord	0.25	0.59	0.35	0.58
Zemberek	0.21	0.17	0.19	0.21
Proposed Model	0.75	0.81	0.78	0.80

Table 7: Ill Formed Word Detection Evaluation Results on Validation Set

Systems	P	R	F	Acc.
MsWord	0.24	0.19	0.21	0.56
Zemberek	0.11	0.29	0.20	0.11
Proposed Model	0.71	0.72	0.71	0.86

Table 8: Ill Formed Word Detection Evaluation Results on Test Set

Data Set	Systems	Accuracy
Validation Set	MsWord	0.25
	Zemberek	0.21
	Lookup Table	0.34
	Proposed Model	0.75
Test Set	MsWord	0.24
	Zemberek	0.11
	Lookup Table	0.31
	Proposed Model	0.71

Table 9: Candidate Generation Results on Data Sets

Table 9 gives the evaluation scores of each different system for both the validation and test data sets. Although the lookup model is very basic, one can observe from the table that it outperforms both MsWord and Zemberek. Our proposed iterative model obtains the highest scores (75% for validation and 71% for test sets) with a relative improvement of 40 percentage points over the lexicon lookup baseline.

7 Conclusion

In this paper we presented a cascaded normalization model for Turkish which could also be applied to the morphologically rich languages with appropriate NLP tools. The model has two main parts: ill formed word detection and candidate word generation consisting of 7 normalization stages (letter case transformation, replacement rules & lexicon lookup, proper noun detection, deasciification, vowel restoration, accent normalization and spelling correction) executed sequentially one on top of the other one. We present the first and highest results for Turkish text normalization[6] of social media data with a 86% accuracy of ill formed word detection and 71% accuracy for candidate word generation. A morphological analyzer is used for the detection of ill formed words. But we believe the accuracy of this first detection stage may be improved by the addition of a lexicon lookup (before the morphological filter) consisting the most frequent normalization cases extracted from manually normalized data if available. Thus, as a future work we plan to extend our work both on the ill formed word detection and on the creation of a spelling corrector with social web data in focus.

[6]The produced test sets and the Web interface of the Turkish Normalizer is available via http://tools.nlp.itu.edu.tr (Eryiğit, 2014)

Acknowledgment

This work is part of our ongoing research project "Parsing Turkish Web 2.0 Sentences" supported by ICT COST Action IC1207 TUBITAK 1001 (grant no: 112E276). The authors want to thank Turkcell Global Bilgi for sharing the manually normalized data of user comments from the Telecom domain. We also want to thank Ozan Arkan Can for his valuable discussions and helps during the data preparation.

References

Kübra Adalı and Gülşen Eryiğit. 2014. Vowel and diacritic restoration for social media texts. In *5th Workshop on Language Analysis for Social Media (LASM) at EACL*, Gothenburg, Sweden, April. Association for Computational Linguistics.

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source nlp framework for turkic languages. *Structure*.

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for sms text normalization. In *Proc. of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 33–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 770–779, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eleanor Clark and Kenji Araki. 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.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proc. of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 71–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jacob Eisenstein. 2013. Phonological factors in social media writing. In *Proc. of the Workshop on Language Analysis in Social Media*, pages 11–19, Atlanta, Georgia, June. Association for Computational Linguistics.

Gülşen Eryiğit, Fatih Samet Çetin, Meltem Yanık, Tanel Temel, and İyas Çiçekli. 2013. Turksent: A sentiment annotation tool for social media. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 131–134, Sofia, Bulgaria, August. Association for Computational Linguistics.

Gülşen Eryiğit. 2014. ITU Turkish NLP web service. In *Proc. of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden, April. Association for Computational Linguistics.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proc. of the 18th conference on Computational linguistics - Volume 1*, COLING '00, pages 285–291, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proc. of the 49th ACL HLT*, pages 368–378, Portland, Oregon, USA, June. Association for Computational Linguistics.

Hany Hassan and Arul Menezes. 2013. Social text normalization using contextual graph random walks. In *Proc. of the 51st ACL*, pages 1577–1586, Sofia, Bulgaria, August. Association for Computational Linguistics.

Yuxiang Jia, Dezhi Huang, Wu Liu, Shiwen Yu, and Haila Wang. 2008. Text normalization in Mandarin text-to-speech system. In *ICASSP*, pages 4693–4696. IEEE.

Max Kaufmann and Jugal Kalita. 2010. Syntactic normalization of Twitter messages. In *Proc. of the 8th International Conference on Natural Language Processing (ICON 2010)*, Chennai, India. Macmillan India.

Osama A Khan and Asim Karim. 2012. A rule-based model for normalization of sms text. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, pages 634–641. IEEE.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. In *Proc. of the 50th ACL*, pages 1035–1044, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thu-Trang Thi Nguyen, Thanh Thi Pham, and Do-Dat Tran. 2010. A method for vietnamese text normalization to improve the quality of speech synthesis. In *Proc. of the 2010 Symposium on Information and Communication Technology*, SoICT '10, pages 78–85, New York, NY, USA. ACM.

Franz Josef Och and Hermann Ney. 2000. Giza++: Training of statistical translation models.

K Panchapagesan, Partha Pratim Talukdar, N Sridhar Krishna, Kalika Bali, and AG Ramakrishnan. 2004. Hindi text normalization. In *Fifth International Conference on Knowledge Based Computer Systems (KBCS)*, pages 19–22. Citeseer.

Deana Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of sms abbreviations. In *IJCNLP*, pages 974–982.

Muhammet Şahin, Umut Sulubacak, and Gülşen Eryiğit. 2013. Redefinition of turkish morphology using flag diacritics. In *Proc. of The Tenth Symposium on Natural Language Processing (SNLP-2013)*, Phuket, Thailand, October.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2011. Resources for Turkish morphological processing. *Lang. Resour. Eval.*, 45(2):249–261, May.

Ruhi Sarikaya, Katrin Kirchhoff, Tanja Schultz, and Dilek Hakkani-Tur. 2009. Introduction to the special issue on processing morphologically rich languages. *Trans. Audio, Speech and Lang. Proc.*, 17(5):861–862, July.

Bilge Say, Deniz Zeyrek, Kemal Oflazer, and Umut Özge. 2002. Development of a corpus and a treebank for present-day written Turkish. In *Proc. of the Eleventh International Conference of Turkish Linguistics*, Famaguste, Cyprus, August.

Gökhan Akın Şeker and Gülşen Eryiğit. 2012. Initial explorations on using CRFs for Turkish named entity recognition. In *Proc. of COLING 2012*, Mumbai, India, 8-15 December.

Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. 2013. Parsing morphologically rich languages: Introduction to the special issue. *Computational Linguistics*, 39(1):15–22.

Pidong Wang and Hwee Tou Ng. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *Proc. of NAACL-HLT*, pages 471–481.

Deniz Yüret and Michael de la Maza. 2006. The greedy prepend algorithm for decision list induction. In *Proc. of the 21st international conference on Computer and Information Sciences*, ISCIS'06, pages 37–46, Berlin, Heidelberg. Springer-Verlag.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive parser-centric text normalization. In *Proc. of the 51st ACL*, pages 1159–1168, Sofia, Bulgaria, August. Association for Computational Linguistics.