An Unsupervised Text Normalization Architecture for Turkish Language

Savaş Yıldırım and Tuğba Yıldız

Department of Computer Engineering, Faculty of Engineering, Istanbul Bilgi University
Santral İstanbul,Eyüp, 34060 Istanbul, Turkey
savasy@bilgi.edu.tr
tugba.dalyan@bilgi.edu.tr

Abstract. A variety of applications on the problem of short-text messages require text normalization process that transforms ill-formed words into standard ones. Recently, many successful approaches have been applied to text normalization especially for social media text. Since each natural language has its own difficulties and barriers, we need to design an architecture to normalize short text messages in Turkish language which has an morphologically rich agglutinative structure. The model proceeds from simple solutions towards more complicated and sophisticated ones to reduce time complexity. A variety of techniques from lexical similarity to n-gram language modeling have been evaluated by exploiting several resources such as high quality corpus, morphological parser and dictionaries. We demonstrate that unsupervised text normalization architecture adapting both lexical and semantic similarity for Turkish domain has shown efficient results that might contribute to other studies.

Key words: lexical normalization, short text message, microblog, text preprocessing

1 Introduction

Micro-blogging social environment provides large volume of data on which many applications with regard to natural language processing (NLP) can easily be applied. According to reliable resources [1], %75 percent of Internet users are also active social media users. *Facebook* has over 1 billion users, *Twitter* has about 500 million ones. Moreover this trends are dramatically increasing over time. It is claimed that 30 billion devices will connect to Internet in the next decade. That creates an huge amount of social data, namely big data, and gives both challenge and possibility to the scientist and experts in all fields.

Usage of Internet eventually shapes the communication style and more importantly use of language. Texting Language is a new phenomenon for the field of NLP. Significantly, new generation starts to build their own language codes,

[1] http://www.techradar.com/news

as in Internet slang. Recently many NLP studies have been on Micro-blogging social media and texting language. Disaster detection, sentiment analysis, event discovery are among the most popular studies. In texting language, the quality of the language can vary from slang language to high-quality text. Therefore text processing tool requires a reliable normalization phase to understand the meaning of the messages. Short-text messages highly include typos, abbreviation, acronym, phonetic substitution, number substitution, use of interjection etc. All these raise an important barrier to be solved. Text normalization process is, thus, more important step in the preprocessing phase of the many studies.

There are a variety of studies regarding normalization especially in the most-studied languages such as English, German, and Spanish. Our primary motivation is to apply text normalization approach to a morphologically rich language, Turkish. Although it is one of the less-studied languages and its language resources are quite limited, Internet usage in Turkey is significantly higher than expected, which necessitates such preprocessing tools then. Each language has its own challenges due to grammar, lexicon, culture, and also political environment. For instance, some phonetic substitution such as number substitution as one of the best phenomenon in English, is not widely used in Turkish language. Having a rich morphological structure of a language turns out the problem a bit harder for our case.

With various experiments, we observed the structure of Texting Language in Turkish and we figured out many phenomena. Our normalization phases are based on those findings. Sufficient volume of both Twitter data and newswire corpus are mined to extract the rules, to conduct the experiment and to test. After morphological analysis of a given tweet, the system works on ill-formed tokens. A corpus and a pre-defined dictionary are exploited to build a look-up dictionary, namely IV, since that using a simple traditional dictionary cannot match many cases especially for agglutinative languages and special domains. The frequent terms occurring in the corpus are added to the IV list. A variety of assumptions are experimentally compiled and applied. Furthermore lexical similarity functions are used to correctly normalize the ill-formed tokens. Some cases such as missing a letter, using ASCII counterpart of a letter are easily recovered. Finally, an n-gram language model utilizing corpus evidence are applied to improve system performance. The architecture is designed in a manner of increasing degree of time complexity

2 Related Works

A variety of sources such as tweets, SMS messages, blogs, etc. have been used for normalization with different approaches; machine translation models, dictionary-based approach, language models, finite state transducers, and cascaded methods. One of the important attempts [1] proposes syntactic normalization with combining two steps; they first process the tweets to remove noise and feed them with statistical machine translation (SMT). Other studies have also used MT models for normalization [2–4].

Another approach which is an unsupervised model, with using noisy channel model, is presented by [5, 6] for normalization. The study [7] designs a normalization system that integrates enhanced letter transformation, visual priming, and string/phonetic similarity for SMS and Twitter data sets.

The study [8] proposed rule-based approach for normalization task. In [9], parser-centric view of normalization is presented. In study [10], a classifier is proposed to detect ill-formed words and produce their canonical lexical forms in standard English based on morphophonemic similarity. Word similarity and context are used to select the correct candidate. In [11], they present that dictionary-based approach outperforms state-of-the-art performance. The extension of these studies is proposed with significantly expanded experimentation in [12]

Although Tweet normalization have been widely and generally applied for English, other languages such as German, Spanish, Malay, French etc. have been also supported [13–18]. In Turkish, a cascaded approach has been applied to short text messages [19]. They propose the model by categorizing the problem into seven steps: letter case transformation, replacement rules & lexicon lookup, proper noun detection, deasciification, vowel restoration, accent normalization and spelling correction.

3 Methodology and Experimental Setup

3.1 Social Media Language

As social media continues to ruin the language, the normalization task is getting challenging. It might share the same problems with spell checking but they differ in that lexical variants in short text messages are often intentionally generated due to the need to save characters, for social identity, or due to the some convention. To normalize an ill-formed word, first we build an in-vocabulary (IV) list. If a word is both not a member of IV and not parsed by a parser, it is considered as ill-formed words.

For building a look-up IV, we exploit some resources; a morphological analyzer, a dictionary and a big reliable corpus. Under some circumference, morphological analyzer cannot distinguish correct words from ill-formed words. Some proper nouns cannot be handled by the parser then. Some trendy places, organizations, actors cannot be either detected by morphological parsed or found in standard dictionary. To build a broader dictionary, we exploit a reliable corpus that are mostly compiled from news texts that have been supervised or edited by experts. It consists of acceptable level of formal language. Most frequent terms or correctly parsed terms in it are retrieved and considered a reliable IV terms. Therefore almost all surface forms of a word can be represented by IV. A high coverage of IV might reduce the OOV rate as in all other morphologically rich language such as Fin, Turkish.

In study [20], they discuss the need for vocabulary size. For a comparison, they checked a English corpus consists of texts from the New York Times magazine. Whereas there are fewer than 200.000 different word forms in the 40

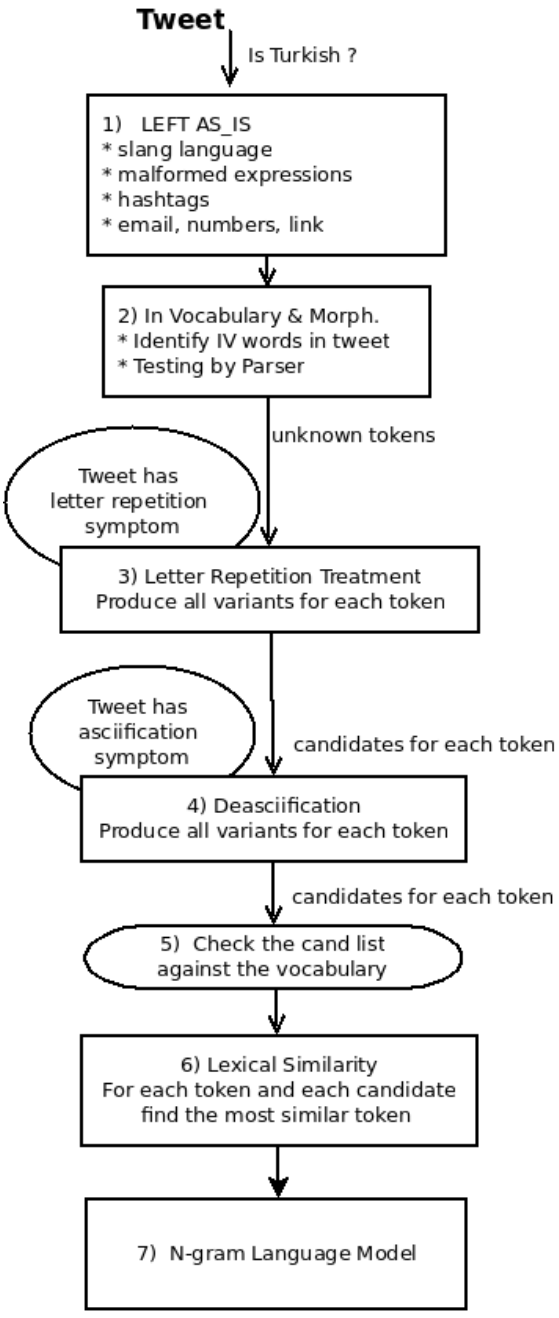

Fig. 1. Architecture of Normalization system

million word English corpus, the corresponding values for *Finnish, Estonian, Turkish* corpora of the same size exceed 1.8 million, 1.5 million and 1 million words, respectively. When we build a broader vocabulary, we reach the 2.4 M unique tokens out of 500 million token sized corpus from which all surface forms of the words are taken.

If a given token is not in IV and cannot parsed by parser, many steps are evaluated to detect the problems and to find a solution. However, we left some particular word untouched as listed below.

- By a slang dictionary with the size of ∼200, all slang words are left as-is.
- 1 or 2 sized words are also left.
- a word mixed with numbers such as *tr12* are left.
- all hash-tags
- all e-mails, all numbers, digits are detected and left
- tokens containing some non-latin5 characters
- all emoticons are left as-is

All remaining words are passed to the next phases of system. According to unidentified words, we figured out some phenomena in Turkish Texting Language. This depicts how social media users use their language to communicate. The facts are as follows:

- Number substitution such as *2nite/(tonight)* as in English is not a case for Turkish Texting Language. Just few cases can be observed for the phonetic substitution.
- Intentionally using ASCII character set is widely spread due to save time and the reason preventing the message uncorrupted in other system that does not support *latin-5* character set. Turkish has own accented character set **(çöşüığ)**. In the mobile phones, to type **s** instead of **ş** is more easier. ASCII character set is widely used than Turkish accents. Replacing ASCII characters by their Turkish counterpart are called *deasciifying*.
- Clipping or shortening the messages to save time. *(Geliyorum− >geliyom, gelyom : koşacağım− > koşcam, koşcm)*
- The Turkish language encompasses a diverse range of accents and dialects as in other languages. This provides a wide range of pronunciation. Those pronunciations and dialects can reshape short text messages and their style. Writing in short text is akin to spoken language. People want to text as if they talk.
- *Interjection* are frequently used as in many languages. The words are lengthened by repeating the letters. When some syllabus is needed to stress, the repetition is used then.
- The suffix (-da) has two kind of usage in the grammar. First is to code locative case. Second meaning is *"also": "Ben de istiyorum" (I also want to)*. In the latter case, a space must intervened between the word and the *da/de* suffix as in the example. Same holds true for the suffix *ki (the meaning is "that is")*. In both cases, users can miss or do not care putting space between words. Therefore the word is left in the unidentified status and cannot be captured by the both morphology analyzer and the vocabulary.

3.2 Normalization Process

We develop different solutions for some of the problems listed above. The Figure 1 depicts high-level architecture of the system. In the *left-as-is* process, some tokens are left as it is written and tagged by the system. Slang terms, numbers, emails and other type of tokens are simply detected. Remaining ill-formed tokens are passed the next process. The further treatment processes are letter repetition, asciification, lexical similarity and n-gram modeling. After each treatment, token is checked against the IV. All the phases are ordered by their degree of complexity.

3.3 Letter repetition treatment

We generate a set of variants by varying the repetitive letters for a given token. For instance, for a token **"gitttikkkkk" (we have gone)**, after reducing the repetition number of a repetitive letter two as in *(gittikk)*, we produce all variants as in the list **(gittik, gitikk, gittik, gitik)**. 2^n variants are produced for a given token that includes n different repetitive letters. In this case the term **"gittik"** is found in IV and accepted. If more than one candidates captured in variant list, decision is made by checking term frequency. This solution is applied in each phase. If none of them is not in IV, we apply lexical similarity to find the closest IV word to any of those variant list.

3.4 Turkish Accents vs. ASCII counterpart

Most short messages have this phenomena since that it is more easy to type ASCII counterpart of the Turkish accents. Some studies called that phenomenon as *asciification*. So the solution is called *deasciification*. The studies immediately apply this approach for any given potential token. However, before applying this process, we need to detect a tweet has this kind of asciification sympthom or not. Instead of working on individual words, we would look entire tweet instead. Our assumption is that if a user types only ASCII letters, s/he does not use any Turkish accents in any position, then deasciification process is needed and applied. Likewise users who types Turkish accent tends not to asciify at all. If a token does not contain any Turkish accent and potential ASCII counterpart of Turkish accents, it is considered suspicious tweet in terms of asciification. This approach dramatically reduces program complexity and prevent applying unnecessary operations and wasting time. The computed probability of that a proper tweet contains any Turkish accent is 92%, where a proper tweet refers to contains at least n tokens, where n is 4. It is, then, said a Tweet (n>=4) is in Turkish language, it most likely contains Turkish accents.

Turkish characters and their ASCII counterparts are *"ğüşıöç"* and *"gusioc"*. All variants are produced for a given word. The ill-formed token *"cocuk" (kid)* contains four potential ASCII character set *(c,o,c,u)* that are intentionally replaced for Turkish accents. And variants of the words are *['cocuk', 'cocük', 'coçuk', 'coçük', 'cöcuk', 'cöcük', 'cöçuk', 'cöçük', 'çocuk', 'çocük', 'çoçuk', 'çoçük',*

'çöcuk', 'çöcük', 'çöçuk', 'çöçük']. It has four suspicious ASCII characters and, 2^4, 16 variants are produced. In this list only the term *"çocuk"* is in IV. If more candidates are in the list, we select them according to their term frequency scores. If any IV word is found, lexical similarity is used between each variant and IV list.

Another crucial problem is that tweets have both asciificaiton and letter repetition problem together. Problems are cascaded and multiplied then. We apply nested procedure to reach normalized form. Again, remaining phases are the similar to previous one.

3.5 Lexical Similarity

There are many well-known orthographic metrics: edit distance, longest common subsequence ratio and rank distance. In this study we used longest common subsequence metric as implemented in python library. If a token cannot be normalized in all the steps above, the system takes the best matches from the vocabulary. If the similarity score is lower than a specific threshold k, 0.6 (empirically selected), candidate is rejected. If more than one close matches share the same score, the most frequent one is proposed as in the entire phases. If there is no any candidate, it proceeds the next phase. Those tokens that have not been handled by the system so far are passed to next n-gram language model where semantic similarity measurements are used for selection as defined in the next section.

3.6 Simplified N-gram Language Model

N-grams language model is considered still the state-of-the-art language model to measure semantic similarity. The language model is using only the prior probability of the word sequence. To extract contextually similar *(OOV, IV)* pairs from a large collection of micro-blog site, tokens are represented in context word vector. The cosine similarity is widely used to compare two context vectors. For a given ill-formed word, the context words are indexed with their positions within a context window size. Standard way of construction of contextual similarity is visiting each occurrence of ill-formed words to produce a vector within a window size. The candidate whose vector is the closest to that of ill-formed token is selected. The weak point of this approach is that producing context vector needs some re-occurrence of ill-formed word. Ill-formed words are more likely to be unique or low frequent.

Moreover, we simplify the approach to reduce the complexity, when considering the huge size of corpus and text messages. Thus, simply taking neighbors of ill-formed words and looking their context vectors could be easier and less complex. Even a given ill-formed word is unique, the system exploits only those nearby words. For instance, for a given case *(left, ill-token, right)*, we build a vector of words that occur with the *pattern (left, *, right)* as in the formula if and only if either left or right must be IV word.

$$arg\ max_c\ LexicalSim(c, token),\ c \in matchPattern(left, *, right) \quad (1)$$

The words captured by the pattern are counted and sorted by their pattern frequencies and lexical similarities. The main defect of the solution is that both nearby words are ill-formed.

4 Results and Evaluation

4.1 Evaluation

In this experiment, we build a broader IV list by exploiting dictionary[2] and a reliable corpus. This is more than a standard dictionary in that we need to keep all formation of the words in order to reduce the time complexity of parsing a word.The number of unique tokens in Turkish language is very high when compared to English and some other languages.The size is about $\sim$ 2.2 million. For a given token, each step produces normalization candidates that are checked against that IV and if a match is found, the process is terminated.

Language usage in social media might vary due to culture, society, age, gender etc. When manually checking the system performance on Twitter data in Turkish Language with size of 1000 entries, we listed several problems and their distributions as shown in Table 1. In this table, two columns represents *model* and *model+n-gram*, where former refers to all steps until lexical similarity, latter is entire system including n-gram language model. The main problem originates from typing ASCII character instead of Turkish accents, with the ratio of 41%. Second problem is accent usage with the ratio of 16.4% due to socio-cultural, new generation etc. as discussed in previous sections. Third one is some typos such as dropping vowels, incorrectly typing some letters etc. Some suffixes -da -de must be separately added with a space. However, they are mostly typed adjacent so that the parser and the IV list cannot specify the word. Those are needed to be specified and solved before.

The system performance is listed in Table 1 for each subproblem where the baseline algorithm is simply choosing the lexically closest candidate from IV list. The table suggests that the model successfully solve asciification problem. Second, accent usage and repetition are also easily captured. While baseline function gets 40% precision, the proposed model gets about 77% precision and the n-gram supported model with gets 80%. Even though individual performance of n-gram language model is acceptable level, it improves system performance by only 3% increase due to that our initial model can capture many problems and remaining parts are limited and hard to be solved such that some unsolved cases could not be normalized by even human annotator.

Beside its contribution to system performance, it is worth testing how much n-gram model individually performs. We tested n-gram model against both the real twitter data set and artificially created data set. Success rate on real twitter set is 53 %. Artificial ill-formed words are automatically malformed by shifting, repeating, shuffling, deleting of original IV words in tweet data. Thus, we can measure the system performance against even huge amount of twitter data set,

[2] code.google.com/p/zemberek/

Table 1. Success rate of Models

	%	Baseline%	Model%	Model+N-gram%
Accent	**16.43**	38.04	75.00	75.00
ASCII	**41.25**	37.66	93.94	94.81
Slang	1.43	12.50	50.00	50.00
Proper	7.68	67.44	76.74	76.74
Vocatives	4.64	30.77	50.00	61.54
Type	**9.29**	51.92	61.54	67.31
Repetition	3.39	42.11	68.42	68.42
Unknown	6.07	0.00	0.00	0.00
No prob	3.75	28.57	52.38	66.67
Other	0.89	0.00	80.00	80.00
DeASCII	5.18	31.03	41.38	48.28
Average		0.40	**0.77**	**0.80**

as huge as possible. For artificially created over 16K ill-formed words, n-gram model showed a performance of 47% precision.

Artificial word formation is considered risky in other approaches, especially based on purely lexical based methods. Computer randomization might have some hidden rules even if randomization process is carefully and well prepared and that makes the model success invalid. Therefore our purely lexical-based models are checked against not the artificial data but the real data with real human mistakes. This is why we used two kinds of test set with different size. Therefore the size of real data set is less than that of artificial one. Running only n-gram model on artificial dataset does not have invalidity problem since that the model check neighbors that are not artificial and appears in real tweets. Only the target words are malformed. To clarify the situation, the average lexical similarity between artificially malformed words and their original ones is nearly 70% which is the similar average score in real data set. For example, the similarity between *"tmrrww"* and *"tomorrow"* is 71%, then it is acceptable level of artificial malformation whereas n-gram model applies lexical similarity only for sorting the candidates captured by distributional word space model.

The proposed model performs acceptable level of precision when compared to other studies. The results in [10] shows that 93% accuracy for English Tweets. Another proposed approach uses Random Walks on a contextual similarity graph [6] has 92.43% precision. The study [1] proposed a system that uses SMT with 79% accuracy. In Turkish [19], the proposed model with a 86% accuracy of ill formed word detection and 71% accuracy for candidate word generation.

In this paper, we propose an architecture of unsupervised text normalization Turkish language. The architecture orders the submodules by their degree of complexity. The important contribution is considered that lexical similarity functions, building a corpus-based IV dictionary and n-gram language modeling are integrated within an architecture for Turkish domain. For a better compar-

ison with other languages, all the experiments and results need to be checked with same input data and the comparison criteria must be well-designed in term of time complexity, data characteristics, circumferences and other factors. That would be in our future plan. Other future work is to create a more reliable look-up dictionary because automatically augmented dictionaries can contains some level of incorrect entries. Reducing the those errors improves the system performance both in accuracy and complexity. Comparing the model with other studies and using BLEU system as baseline will be in our other future tasks as well.

5 Conclusion

In this paper, we propose an architecture for Turkish text normalisation. Even though Turkish is one of the less-studied language, Internet usage of native Turkish speakers is more than expected. Therefore the studies on social media texting language are getting important. They necessitates some preprocessing phases such as text normalization, tokenization, boundry identification etc.

We exploit many resources to normalize a given ill-formed token; tweet data, a huge corpus and a reliable dictionary. In order to design an architecture, first we figured out texting language phenomena in social media short text. The problems can vary; letter repetition, missing vowels, asciification and others. The submodules are sorted in accordance with their time complexity. We utilized lexical similarity to compare the candidates and IV words to make a decision. The last phase of the model is semantic model that includes n-gram language modeling. We simplified n-gram language model to reduce time complexity. This module uses word space model to normalize an ill-formed token by looking at their neighbors. Neighbors are used as semantic space that is very similary to lexico-syntactic patterns. The candidates that are captured from this phase are again evaluated in terms of their lexical similarity. So this step actually employs both semantic and lexical similarity.

We divide our architecture into two models; model and model+n-gram. The first model is based on only lexical similarity, which refers to all phases except n-gram. Second is using both lexical and semantic similarity, entire architecture including n-gram model. First model achieves 77 % precision, second model gets 80 % precision. Both highly outperform baseline function that simply takes the lexically closest term in IV to ill-formed token and its success rate is about 40 %. We also evaluated n-gram model separately to check its success. When we run n-gram semantic model as a separate unit on the data, we get 53% precision. When running on an artificially created big data, we get about 47%. This shows us designed n-gram language model can have a sufficient capacity even though its contribution to entire architecture is limited.

This study might be considered first study that applies both lexical similarity and semantic similarity together to Turkish text normalization. When comparing the similar studies, the proposed architecture gets sufficient and promising results. For a better comparison with those studies, all the experiments might

be tested with same data and conditions. That would be in our future plan. Another future work is to extend look-up dictionary to reduce level of incorrect entries, which might improves the system performance. Using BLEU system for a better comparison will be in our other future tasks as well.

Acknowledgments

This paper was supported by Tübitak Program 1001 Scientific and Technological Research Council Projects Funding Program. Project (Code 113K175): Türkiye'de Kamuoyunun Bilime Yönelik Tutum Değişiminin Ölçülmesinde Metin Madenciliği Yolu ile Alternatif Bir Sistem

References

1. Max Kaufmann and Jugal Kalita. Syntactic normalization of Twitter messages. In *Proc. of the 8th International Conference on Natural Language Processing (ICON 2010), Kharagpur, India*, (2010)
2. Deana Pennell and Yang Liu. A character-level machine translation approach for normalization of sms abbreviations. In *Proc. of 5th International Joint Conference on Natural Language Processing*, pages 974–982 (2011)
3. Aiti Aw, Min Zhang, Juan Xiao and Jian Su. A phrase-based statistical model for sms text normalization. In *Proc. of the COLING/ACL on Main conference poster sessions (COLING-ACL '06). Association for Computational Linguistics, Stroudsburg, PA, USA*, pages 33–40 (2006)
4. Richard Beaufort, Sophie Roekhaut, Louise-Amelie Cougnon and Cedrick Fairon. 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). Association for Computational Linguistics, Stroudsburg, PA, USA*, pages 770–779 (2010)
5. Paul Cook and Suzanne Stevenson. Unsupervised model for text message normalization. In *Proc. of the Workshop on Computational Approaches to Linguistic Creativity , CALC '09, Association for Computational Linguistics, Stroudsburg, PA, USA*, pages 71–78 (2009)
6. Hany Hassan and Arul Menezes. Social text normalization using contextual graph random walks. In *Proc. of the 51st ACL, Association for Computational Linguistics, Sofia, Bulgaria*, pages 1577–1586 (2013)
7. Fei Liu, Fuliang Weng and Xiao Jiang. A broad-coverage normalization system for social media language. In *Proc. of the 50th ACL, Association for Computational Linguistics, Stroudsburg, PA, USA*, pages 1035–1044 (2012)
8. Clark and Kenji Araki. Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. *Procedia-Social and Behavioral Sciences*, 27:2–11 (2011)
9. Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld and Yunyao Li. Adaptive parser-centric text normalization. In *Proc. of the 51st ACL, Association for Computational Linguistics, Sofia, Bulgaria*, pages 1159–1168 (2013)
10. Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proc. of the 49th ACL HLT, Association for Computational Linguistics, Portland, Oregon, USA*, pages 368–378 (2011)

11. Bo Han, Paul Cook and Timothy Baldwin. Automatically Constructing a Normalisation Dictionary for Microblogs. In *EMNLP-CoNLL 2012, Jeju, Republic of Korea*, pages 421–432 (2012)
12. Bo Han, Paul Cook and Timothy Baldwin. Lexical Normalisation of Short Text Messages. In *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1) 5:1–5:27 (2013)
13. Bo Han, Paul Cook and Timothy Baldwin. Spanish Text Normalisation. In *Proc. of Tweet-norm: Tweet Normalization Workshop at SEPLN 2013, 67-71, Madrid, Spain*, pages 67–71 (2013)
14. Uladzimir Sidarenka, Tatjana Scheer and Manfred Stede. Rule-Based Normalization of German Twitter Messages. In *Proc. of the GSCL Workshop Verarbeitung und Annotation von Sprachdaten aus Genres internetbasierter Kommunikation*, (2013)
15. Catherine Kobus, François Yvon and Geraldine Damnati. Normalizing SMS: are two metaphors better than one?. In *Proc. of the 22nd International Conference on Computational Linguistics - Volume 1 (COLING '08), Vol. 1. Association for Computational Linguistics, Stroudsburg, PA, USA*, pages 441–448 (2008)
16. Pablo Ruiz, Montse Cuadros and Thierry Etchegoyhen. Lexical Normalization of Spanish Tweets with Rule-Based Components and Language Models. *Procesamiento del Lenguaje Natural*, 52:45–52 (2014)
17. Jordi Porta and Jose Luis Sancho. Word normalization in Twitter using finite-state transducers. In *Workshop on Tweet Normalization at SEPLN Tweet-Norm*, (2013)
18. Mohammad Arshi Saloot, Norisma Idris and Rohana Mahmud. An architecture for Malay Tweet normalization. *Information Processing and Management*, 50(5):621–633 (2014)
19. Dilara Torunoğlu and Gülşen Eryiğit. A Cascaded Approach for Social Media Text Normalization of Turkish. In *5th Workshop on Language Analysis for Social Media (LASM) at EACL, Gothenburg, Sweden*, (2014)
20. Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. ACM Trans. Speech Lang. Process., 5(1) 3:1–3:29 (2007)