A Modular Approach for Social Media Text Normalization

Palak Rehan, Mukesh Kumar and Sarbjeet Singh

Abstract The normalized data is the backbone of various Natural Language Processing (NLP), Information Retrieval (IR), data mining, and Machine Translation (MT) applications. Thus, we propose an approach to normalize the colloquial and breviate text being posted on the social media like Twitter, Facebook, etc. The proposed approach for text normalization is based upon Levenshtein distance, demetaphone algorithm, and dictionary mappings. The standard dataset named lexnorm 1.2, containing English tweets is used to validate the proposed modular approach. Experimental results are compared with existing unsupervised approaches. It has been found that modular approach outperforms other exploited normalization techniques by achieving 83.6% of precision, recall, and F-scores. Also 91.1% of BLUE scores have been achieved.

1 Introduction

Social media networks like Twitter, Facebook, WhatsApp, etc., are most commonly used medium for sharing news, opinions, and to stay in touch with peers. Messages on Twitter are limited to 140 characters. This led users to create their own novel syntax in tweets to express more in lesser words. Free writing style, use of URLs, markup syntax, inappropriate punctuations, ungrammatical structures, abbreviations, etc., make it harder to mine useful information from them. There is no standard way of posting tweets. This lack of standardization hampers NLP and MT tasks and renders huge volume of social media data useless. Therefore, there is a

P. Rehan (✉) · M. Kumar · S. Singh
Computer Science & Engineering Department, University Institute of Engineering and Technology, Panjab University, Chandigarh, India
e-mail: palakrehan@gmail.com

M. Kumar
e-mail: mukesh_rai9@yahoo.com

S. Singh
e-mail: sarbjeet@pu.ac.in

S. C. Satapathy et al. (eds.), *Information and Decision Sciences*,
Advances in Intelligent Systems and Computing 701,
https://doi.org/10.1007/978-981-10-7563-6_20

need to reform such text forms into standard forms. This can be achieved by normalization which is a preprocessing step for any application that handles social media text. Process of converting ill-formed words into their canonical form is known as normalization. In this paper, we propose a modular approach for the lexical text normalization which is applied on English tweets. First, it is necessary to categorize text into two classes: Out-Of-Vocabulary (OOV) and In-Vocabulary (IV). Words that are not in its standard form are considered as OOV while words which are part of standard orthography fall under IV category. For example, "talkin" is an OOV word (nonstandard word) having "talking" as its IV form. Some words, although may be correct, coincide with other IV words, like "wit" is an IV word which may indicate "with" as its correct form. Such words are neglected for normalization task.

Due to the presence of unsurprisingly long tail of OOV words, a method that does not require annotated training data is preferred. Thus, an unsupervised approach has been proposed with different modular stages for preprocessing, candidate generation, and candidate selection steps. The proposed cascaded method achieves state-of-the-art results on English twitter dataset and can be applied to any other social media dataset. In addition, a pilot study is conducted on peer methodologies employed inside the proposed approach.

2 Related Work

Previous work attempted noisy channel model as one of the text normalization techniques. Brill and Moore characterized the noisy channel model based on string edits for handling the spelling errors. Toutanova and Moore [1] improved the above model by embedding information regarding pronunciation. Choudhury et al. [2] proposed a supervised approach based on Hidden Markov Model (HMM) for SMS text by considering graphemic/phonetic abbreviations and unintentional typos. Cook and Stevenson [3] expanded error model by introducing probabilistic models for different erroneous forms according to sampled error distribution. This work tackled three common types: stylistic variation, prefix clipping, and subsequence abbreviations. Yi yang et al. [4] presented a unified log-linear unsupervised statistical model for text normalization using maximum likelihood framework and novel sequential Monte Carlo training algorithm.

Some of the previous work was based on Statistical Machine Translation (SMT) approach for normalization. SMT deals with context-sensitive text by treating noisy forms as the source language and the standard form as the target language. Aw et al. proposed an approach for Short Messaging Service (SMS) text normalization using phrase-level SMT and bootstrapped phrase alignment techniques. The main drawback of SMT approach is that it needs a lot of training data and it cannot accurately represent error types without contextual information.

Gouws et al. [5] developed an approach based on string and distributional similarity along with dictionary lookup method to deal with ill-formed words. Han et al. introduced similar technique based on distributional similarity and string similarity. Selection of correct forms was performed on pairwise basis. Mohammad Arshi et al. [6] proposed a tweet normalization approach. First, candidates were generated by targeting lexical, phonemic, and morphophonemic similarities. More recent approaches handle the text normalization using CRFs and neural networks. Min et al. [7] proposed a system where Long Short-Term Memory (LSTM) recurrent neural networks using character sequences and Part-Of-Speech (POS) tags had been used for predicting word-level edits. Yang and Kim used an CRF-based approach. CRF using both brown clusters and word embeddings that were trained using canonical correlation analysis as features.

Abiodun Modupe [8] developed a semi-supervised probabilistic approach for normalizing informal short text messages. Language model probability had been used to enhance the relationships between formal and informal word. Then, string similarity was employed with a linear model to include features for both word-level transformations and local context similarity.

Proposed approach in this paper also adopts lexical based text normalization using unsupervised methods in order to handle wide variety of ill-formed words. This approach can be applied to different social media text messages.

3 Proposed Work

Inspired by earlier work done on text normalization, a modular approach for lexical text normalization is proposed, which has been applied to English tweets. Pre-processing, Candidate Generation, and Candidate Selection are the three main stages of the proposed system.

Text refining is applied on extracted strings and thereafter categorization into IV and OOV lexicons is performed. Candidate Generation stage generates list of possible correct words for an input OOV word. In the end, Candidate Selection stage selects a best possible candidate from all generated candidates. Raw tweets are fed into preprocessor whose output will act as input for the token qualifier which will segregate input into two heaps: OOV and IV words. Out of total OOV tokens, filtration is performed with the help of available python packages as punctuation symbols, hashtags, URLS, @mentions, and proper nouns. *(Name of persons, locations, brands, movies, etc., comes under proper noun).

OOV tokens detected by token qualifier will be processed by the candidate generator which will generate possible normalized candidates via three different techniques: Levenshtein distance, demetaphone algorithm, and dictionary mappings. Candidate selector module will work on candidate list generated by the candidate generator and will generate best possible candidate for each OOV.

Preprocessing module takes raw tweets as input. Tokenization in the form of strings is performed and then unwanted strings containing @, hashtags, URLS,

punctuations, emoticons, and any foreign language are filtered out. Output strings generated after filtration are considered as lexicons. This is initial step required to carry any NLP task. Output generated by preprocessing module is fed as input to the Token Qualifier. It performs classification of lexicon into two categories: OOV and IV. To predict whether a given lexicon is OOV, many Standard English spell checker like GNU Aspell, Hanspell, and dictionaries (Pyenchant corpus) are available.

According to research, correct formed English words having repeating characters are found to have maximum of two character repetitions. Thus, repetition of more than two characters in a string is trimmed off to two characters (helloooo → hello, gooood → good). Regular expressions are applied to OOV strings with alphanumeric text. Some of the transformations with examples are given as: 4 → fore (B4 → bfore), 2 → to (2night → tonight), 9 → ine (F9 → fine), etc. After applying trimming and regular expressions, OOV words that are going to be processed further are obtained. First technique to generate candidates for OOV word is through Levenshtein distance (also known as edit distance). Edit distance is number of applied insertions, deletions, and alterations in order to transform one string into another. It is used to handle spelling errors. Edit distance >2 results in generation of large number of candidates most of which are inappropriate and at same time are complex to process. So, we prefer edit distance with <=2. Algorithm 4 takes input of Algorithm 3 and generates strings having edit distance <=2 with respect to input OOV. In order to have precise and limited generated candidate list, string similarity measures are applied on candidate list generated via edit distance (<=2).

Algorithm 1: Levenshtein_candidates (Modified_OOV)
{

1. *Levenshtein_set* = []
   near_by_editcandidates = []
2. *Input the Modified_OOV*
3. *Generate strings having edit distance* ≤ 2 *from input OOV.*
4. *If generated string* (*from step* 3) *english vocabulary*:

   Add generated strings to Levenshtein_set only

5. *Apply string similarity measures* (*fuzzy_ratio*) *between input and each of corresponding strings in Levenshtein_set*:

   5.1 *Select those pairs having maximum similarity ratio.*
   5.2 *Add above pairs in near_by_editcandidates*

6. *Return near_by_editcandidates*

   }

Nowadays, Internet slangs like lol → laughing out loud and abbreviations (Cuz → because) are common in social media text. So that we generate candidates using dictionary mapping by applying algorithm 2.

Algorithm 2: Slang_candidates (Modified_OOV)
{

1. *Slang_output* = []
2. *Input the Modified_OOV*
3. *Check input in Slang dictionay*
   // Slang dictionary is prepared by collecting internet abbreviations
   // from www.noslang.com *on* 4 *nov*, 2016

   3.1 *If input is found in dictionary*:

   output corresponding mapping to Slang_output

4. *Return Slang_output* }

In order to handle errors due to phonemes (words that sound same), demetaphone algorithm is used. Words like nite and night are phonemes of each other. In order to have limited, precise candidate is set and to reduce processing complexity, string similarity measures are applied on phonemes generated by the demetaphone Algorithm.

Algorithm 3: Demetaphone_candidates (Modified_OOV)
{

1. *Demetaphone_set* = []
   Relevant_demetaphone = []
2. *Input the* $Modified_{OOV}$
3. *Generate demetaphone code for each input and english vocabulary word pair.*
4. *Add pairs having same code to Demetaphone_set.*
5. *In order to have only relevant and limited pairs, apply string similarity measures (fuzzy_ratio) to each pair in Demetaphone_set.*

   5.1 *Select only those pairs that have maximum similarity ratio.*
   5.2 *Add above pairs to Relevant_demetaphone set.*

6. *Return Relevant_demetaphone.*
   }

Candidate list generated by all three techniques (output of Algorithm 1–3) acts as input to candidate scorer (Algorithm 4). Equal probability to each candidate in list corresponding to a OOV lexicon is assigned. Aggregate probabilities of all those candidates which are present in more than one list are calculated by performing summation on their probabilities. This will act as score. Prepare an aggregate list by combining candidate lists of all three candidate generation techniques.

Algorithm 4: Candidate_Scorer (near_by_editcandidates, Relevant_demetaphone, Slang_output)
{

1. $Edit_{score}$ = [], $demetaphone_{score}$ = [], $slang_{score}$ = [], $combine_{score}$ = [], $aggregate_{candidates}$ = []
2. *Assign equal probability to each candidates in near_by_editcandidates and store probabilities in Edit_score set.*
3. *Assign equal probability to each candidates in Relevant_demetaphone and in Slang_output and store them in demetaphone_score and slang_score respectively.*
4. *Aggregate probabilities for common candidates that are present in all above three sets. Combine score corresponding to a Modified_OOV token is computed as*:

$$Edit_{score[Modified_{OOV}]} + demetaphone_{score[Modified_{OOV}]} + slang_{score[Modified_{OOV}]}$$

5. *Prepare* $aggregate_{candidates}$ *set by combining candidates from near_by_editcandidates, Relevant_demetaphone, Slang_output*
6. *Return* $combine_{score}$, $aggregate_{candidates}$ }

Aggregate candidate list and score list prepared by Algorithm 4 will act as input to Algorithm 5. Select that candidate from aggregate candidate list (for an OOV lexicon) corresponding to which maximum scores are present in score list. In case there are more than one candidate with same scores, then apply POS tagging. During POS tagging, assign scores according to the importance of context like nouns will be given highest weight followed by the verb and then the adjective. This will return a single best candidate for each incorrect word.

Proposed modular approach works on raw tweets. Preprocessing is done by removing unwanted strings (punctuations, hashtags, etc.). Token qualifier is then called to detect OOV and IV words. Rules are applied to the output of the token qualifier to generate OOV tokens which will be used for further processing. Candidates are generated via Levenshtein distance (Algorithm 1), demetaphone algorithm (Algorithm 2), and dictionary approach (Algorithm 3). In order to select best possible normalized word corresponding to an OOV word, candidate scorer (Algorithm 4) and candidate selector (Algorithm 5) are employed.

Algorithm 5: Candidate_selector (aggregate$_{candidates}$, combine$_{score}$)
{

1. *Best_candidate* = []
2. *Read candidates from aggregate_candidates.*
3. *Select those candidates from* $aggregate_{candidates}$ *corresponding to which maximum probability is present in combine_score set.*
4. *If* (*only one candidate is outputed from step* 3): *Store it in best_candidate set.*
5. *Else*:

5.1 *Apply POS* (*Part of speech*) *tag.*

5.2 *Assign maximum score* (*say* 1) *to noun, followed by verb* (0.5) *and then adjective* (0.25)
5.3 *Select candidate with maximum score // obtained after scoring of* 5.2
5.4 *Store result in Best_candidate*

6. *Return Best_candidate*

}

4 Experimental Setup and Results

Proposed modular approach has been implemented on LexNorm 1.2 dataset which was an updated version of dataset for lexical normalization described in [9]. This dataset contains English messages sampled from Twitter API (from August to October, 2010). Results are evaluated on the basis of precision, recall, F-score, and BLEU score. The proposed work is performed on Python 2.7 version for windows and natural language processing inbuilt python packages are utilized to execute modules. Let $T_{dataset}$ be all tokens from dataset and let OOV_t be the list of all detected OOV in dataset $\in T_{dataset} \bullet gen^t_{oov}$ be the generated candidates for an oov $\in OOV_t \cdot sel^t_{oov}$ be the best normalized candidate selected by system for an oov token, oov $\in OOV_t \cdot cor^t_{oov}$ be the tagged correction for an oov $\in OOV_t \cdot norm^t_{oov}$ be the set of normalized oov tokens $\in OOV_t$ normalized by system.

$$Precision(P) = \frac{\sum_{t \in T_{dataset}} \left| \left\{ sel^t_{oov} : sel^t_{oov} = cor^t_{oov}, sel^t_{oov} \in gen^t_{oov}, oov \in OOV_t \right\} \right|}{\sum_{t \in T_{dataset}} \left| \left\{ norm^t_{oov} : norm^t_{oov}, oov \in OOV_t \right\} \right|} \tag{1}$$

$$Recall(R) = \frac{\sum_{t \in T_{dataset}} \left| \left\{ sel^t_{oov} : sel^t_{oov} = cor^t_{oov}, sel^t_{oov} \in gen^t_{oov}, oov \in OOV_t \right\} \right|}{\sum_{t \in T_{dataset}} \left| \left\{ oov : oov \in OOV_t \right\} \right|} \tag{2}$$

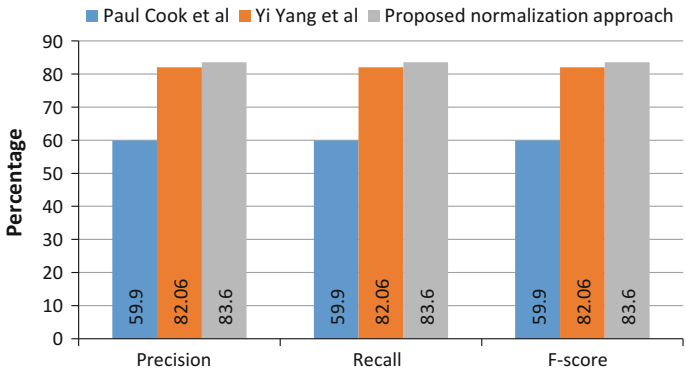

Fig. 1 Comparative results with unsupervised methods

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

Figure 1 shows that the proposed modular approach yields better accuracy as compared to existing unsupervised methods. Modular approach has 1.54% better results than log linear model for unsupervised text normalization [4]. Moreover, an unsupervised model for text normalization proposed by Paul cook et al. [3] also has low performance (57.9% accuracy) than the proposed approach (having 83.6% performance).

5 Conclusion

Text normalization addresses all forms of OOV words and aimed at standardization of these words. Proposed approach is based on three methods: Levenshtein distance, demetaphone algorithm, and dictionary approach. Experimental results are calculated for Lexnorm 1.2, standard dataset for twitter messages. The proposed system is compared with existing unsupervised text normalization methods. It has been found that modular approach outperforms other exploited normalization techniques by achieving 83.6% of precision, recall, and F-scores. Also 91.1% of BLUE scores have been achieved.

References

1. Toutanova, K., Moore, R.C.: Pronunciation modeling for improved spelling correction. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL 02, pp. 144–151, Philadelphia, USA (2002)
2. Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., Basu, A.: Investigation and modeling of the structure of texting language. Int. J. Doc. Anal. Recogn. **10**, 157–174 (2007)
3. Cook, P., Stevenson, S.: An unsupervised model for text message normalization. In: Proceedings of the Workshop on Computational Approaches to Linguistic Creativity, pp. 71–78. Association for Computational Linguistics, Boulder, USA, June (2009)
4. Yang, Y., Eisenstein, J.: A log-linear model for unsupervised text normalization. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), pp. 61–72, Seattle, USA, Oct 2013
5. Gouws, S., Hovy, D., Metzler, D.: Unsupervised mining of lexical variants from noisy text. In: Proceedings of the First workshop on Unsupervised Learning in NLP, pp. 82–90, Edinburgh, Scotland (2011)
6. Saloot, M.A., Idris, N., Shuib, L., Raj, R.G., Aw, A.: Toward tweets normalization using maximum entropy. In Proceedings of the ACL 2015 Workshop on Noisy User-generated Text, pp. 19–27. Association for Computational Linguistics, Beijing, China, 31 July 2015 (2015)
7. Min, W., Mott, B., Lester, J., Cox, J.: Ncsu_sas_wookhee: a deep contextual long-short term memory model for text normalization. In: proceedings of WNUT, Beijing, China (2015)

8. Modupe, A., Celik, T., Marivate, V., Diale, M.: Semi-supervised probabilistics approach for normalising informal short text messages. In: Conference on Information Communication Technology and Society (ICTAS). IEEE (2017)
9. Han, B., Baldwin, T.: Lexical normalisation of short text messages: makn sens a# twitter. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 368–378. Association for Computational Linguistics, Portland, Oregon, June (2011)