Insertion, Deletion, or Substitution? Normalizing Text Messages without Pre-categorization nor Supervision

Fei Liu[1] **Fuliang Weng**[2] **Bingqing Wang**[3] **Yang Liu**[1]
[1]Computer Science Department, The University of Texas at Dallas
[2]Research and Technology Center, Robert Bosch LLC
[3]School of Computer Science, Fudan University
{feiliu, yangl}@hlt.utdallas.edu[1]
fuliang.weng@us.bosch.com[2], wbq@fudan.edu.cn[3]

Abstract

Most text message normalization approaches are based on supervised learning and rely on human labeled training data. In addition, the nonstandard words are often categorized into different types and specific models are designed to tackle each type. In this paper, we propose a unified letter transformation approach that requires neither pre-categorization nor human supervision. Our approach models the generation process from the dictionary words to nonstandard tokens under a sequence labeling framework, where each letter in the dictionary word can be retained, removed, or substituted by other letters/digits. To avoid the expensive and time consuming hand labeling process, we automatically collected a large set of noisy training pairs using a novel web-based approach and performed character-level alignment for model training. Experiments on both Twitter and SMS messages show that our system significantly outperformed the state-of-the-art deletion-based abbreviation system and the jazzy spell checker (absolute accuracy gain of 21.69% and 18.16% over jazzy spell checker on the two test sets respectively).

1 Introduction

Recent years have witnessed the explosive growth of text message usage, including the mobile phone text messages (SMS), chat logs, emails, and status updates from the social network websites such as Twitter and Facebook. These text message collections serve as valuable information sources, yet the nonstandard contents within them often degrade the existing language processing systems, calling the need of text normalization before applying the traditional information extraction, retrieval, sentiment analysis (Celikyilmaz et al., 2010), or summarization techniques. Text message normalization is also of crucial importance for building text-to-speech (TTS) systems, which need to determine pronunciation for nonstandard words.

2gether (6326)	togetha (919)	tgthr (250)	togeda (20)
2getha (1266)	togather (207)	t0gether (57)	toqethaa (10)
2gthr (178)	togehter (94)	togeter (49)	2getter (10)
2qetha (46)	togethor (29)	tagether (18)	2gtr (6)

Table 1: Nonstandard tokens originated from "**together**" and their frequencies in the Edinburgh Twitter corpus.

Text message normalization aims to replace the non-standard tokens that carry significant meanings with the context-appropriate standard English words. This is a very challenging task due to the vast amount and wide variety of existing nonstandard tokens. We found more than 4 million distinct out-of-vocabulary tokens in the English tweets of the Edinburgh Twitter corpus (see Section 2.2). Table 1 shows examples of nonstandard tokens originated from the word "together". We can see that some variants can be generated by dropping letters from the original word ("tgthr") or substituting letters with digit ("2gether"); however, many variants are generated by combining the letter insertion, deletion, and substitution operations ("toqethaa", "2gthr"). This shows that it is difficult to divide the nonstandard tokens into exclusive categories.

Among the literature of text normalization

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:shortpapers, pages 71–76,
Portland, Oregon, June 19-24, 2011.

(for text messages or other domains), Sproat et al. (2001), Cook and Stevenson (2009) employed the noisy channel model to find the most probable word sequence given the observed noisy message. Their approaches first classified the nonstandard tokens into various categories (e.g., abbreviation, stylistic variation, prefix-clipping), then calculated the posterior probability of the nonstandard tokens based on each category. Choudhury et al. (2007) developed a hidden Markov model using hand annotated training data. Yang et al. (2009), Pennell and Liu (2010) focused on modeling word abbreviations formed by dropping characters from the original word. Toutanova and Moore (2002) addressed the phonetic substitution problem by extending the initial letter-to-phone model. Aw et al. (2006), Kobus et al. (2008) viewed the text message normalization as a statistical machine translation process from the texting language to standard English. Beaufort et al. (2010) experimented with the weighted finite-state machines for normalizing French SMS messages. Most of the above approaches rely heavily on the hand annotated data and involve categorizing the nonstandard tokens in the first place, which gives rise to three problems: (1) the labeled data is very expensive and time consuming to obtain; (2) it is hard to establish a standard taxonomy for categorizing the tokens found in text messages; (3) the lack of optimized way to integrate various category-specific models often compromises the system performance, as confirmed by (Cook and Stevenson, 2009).

In this paper, we propose a general letter transformation approach that normalizes nonstandard tokens without categorizing them. A large set of noisy training word pairs were automatically collected via a novel web-based approach and aligned at the character level for model training. The system was tested on both Twitter and SMS messages. Results show that our system significantly outperformed the jazzy spell checker and the state-of-the-art deletion-based abbreviation system, and also demonstrated good cross-domain portability.

2 Letter Transformation Approach

2.1 General Framework

Given a noisy text message T, our goal is to normalize it into a standard English word sequence S.

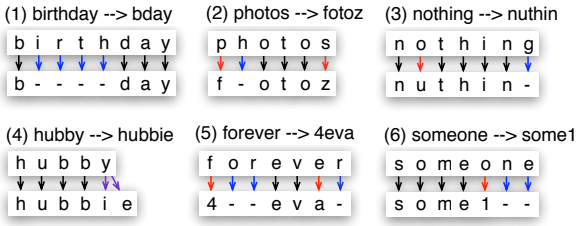

Figure 1: Examples of nonstandard tokens generated by performing letter transformation on the dictionary words.

Under the noisy channel model, this is equivalent to finding the sequence $\hat{S}$ that maximizes $p(S|T)$:

$$\hat{S} = \arg\max_S p(S|T) = \arg\max_S (\prod_i p(T_i|S_i))p(S)$$

where we assume that each non-standard token T_i is dependent on only one English word S_i, that is, we are not considering acronyms (e.g., "bbl" for "be back later") in this study. $p(S)$ can be calculated using a language model (LM). We formulate the process of generating a nonstandard token T_i from dictionary word S_i using a letter transformation model, and use the model confidence as the probability $p(T_i|S_i)$. Figure 1 shows several example (word, token) pairs[1]. To form a nonstandard token, each letter in the dictionary word can be labeled with: (a) one of the 0-9 digits; (b) one of the 26 characters including itself; (c) the null character "-"; (d) a letter combination. This transformation process from dictionary words to nonstandard tokens will be learned automatically through a sequence labeling framework that integrates character-, phonetic-, and syllable-level information.

In general, the letter transformation approach will handle the nonstandard tokens listed in Table 2 yet without explicitly categorizing them. Note for the tokens with letter repetition, we first generate a set of variants by varying the repetitive letters (e.g. C_i = {"pleas", "pleeas", "pleaas", "pleeaas", 'pleeeaas"} for T_i = {"pleeeaas"}), then select the maximum posterior probability among all the variants:

$$p(T_i|S_i) = \max_{\tilde{T}_i \in C_i} p(\tilde{T}_i|S_i)$$

[1]The ideal transform for example (5) would be "for" to "4". But in this study we are treating each letter in the English word separately and not considering the phrase-level transformation.

(1) abbreviation	tgthr, weeknd, shudnt
(2) phonetic sub w/- or w/o digit	4got, sumbody, kulture
(3) graphemic sub w/- or w/o digit	t0gether, h3r3, 5top, doinq
(4) typographic error	thimg, macam
(5) stylistic variation	betta, hubbie, cutie
(6) letter repetition	pleeeaas, togtherrr
(7) any combination of (1) to (6)	luvvvin, 2moro, m0rnin

Table 2: Nonstandard tokens that can be processed by the unified letter transformation approach.

2.2 Web based Data Collection w/o Supervision

We propose to automatically collect training data (annotate nonstandard words with the corresponding English forms) using a web-based approach, therefore avoiding the expensive human annotation. We use the Edinburgh Twitter corpus (Petrovic et al., 2010) for data collection, which contains 97 million Twitter messages. The English tweets were extracted using the TextCat language identification toolkit (Cavnar and Trenkle, 1994), and tokenized into a sequence of clean tokens consisting of letters, digits, and apostrophe.

For the out-of-vocabulary (OOV) tokens consisting of letters and apostrophe, we form n Google queries for each of them in the form of either "$w_1\ w_2\ w_3$" OOV or OOV "$w_1\ w_2\ w_3$", where w_1 to w_3 are consecutive context words extracted from the tweets that contain this OOV. n is set to 6 in this study. The first 32 returned snippets for each query are parsed and the words in boldface that are different from both the OOV and the context words are collected as candidate normalized words. Among them, we further select the words that have longer common character sequence with the OOV than with the context words, and pair each of them with the OOV to form the training pairs. For the OOV tokens consisting of both letters and digits, we use simple rules to recover possible original words. These rules include: 1 → "one", "won", "i"; 2 → "to", "two", "too"; 3 → "e"; 4 → "for", "fore", "four"; 5 → "s"; 6 → "b"; 8 → "ate", "ait", "eat", "eate", "ight", "aight". The OOV tokens and any resulting words from the above process are included in the noisy training pairs. In addition, we add 932 word pairs of chat slangs and their normalized word forms collected from InternetSlang.com that are not covered by the above training set.

These noisy training pairs were further expanded and purged. We apply the transitive rule on these initially collected training pairs. For example, if the two pairs "(cause, cauz)" and "(cauz, coz)" are in the data set, we will add "(cause, coz)" as another training pair. We remove the data pairs whose word candidate is not in the CMU dictionary. We also remove the pairs whose word candidate and OOV are simply inflections of each other, e.g., "(headed, heading)", using a set of rules. In total, this procedure generated 62,907 training word pairs including 20,880 unique candidate words and 46,356 unique OOVs.[2]

2.3 Automatic Letter-level Alignment

Given a training pair (S_i, T_i) consisting of a word S_i and its nonstandard variant T_i, we propose a procedure to align each letter in S_i with zero, one, or more letters/digits in T_i. First we align the letters of the longest common sequence between the dictionary word and the variant (which gives letter-to-letter correspondence in those common subsequences). Then for the letter chunks in between each of the obtained alignments, we process them based on the following three cases:

(a) (many-to-0): a chunk in the dictionary word needs to be aligned to zero letters in the variant. In this case, we map each letter in the chunk to "-" (e.g., "birthday" to "bday"), obtaining letter-level alignments.

(b) (0-to-many): zero letters in the dictionary word need to be aligned to a letter/digit chunk in the variant. In this case, if the first letter in the chunk can be combined with the previous letter to form a digraph (such as "wh" when aligning "sandwich" to "sandwhich"), we combine these two letters. The remaining letters, or the entire chunk when the first letter does not form a digraph with the previous letter, are put together with the following aligned letter in the variant.

(c) (many-to-many): non-zero letters in the dictionary word need to be aligned to a chunk in the variant. Similar to (b), the first letter in the variant chunk is merged with the previous alignment if they form a digraph. Then we map the chunk in the dictionary word to the chunk in the variant as one alignment, e.g., "someone" aligned to "some1".

[2]Please contact the first author for the collected word pairs.

The (b) and (c) cases above generate chunk-level (with more than one letter) alignments. To eliminate possible noisy training pairs, such as ("you", "haveu"), we keep all data pairs containing digits, but remove the data pairs with chunks involving three letters or more in either the dictionary word or the variant. For the chunk alignments in the remaining pairs, we sequentially align the letters (e.g., "ph" aligned to "f-"). Note that for those 1-to-2 alignments, we align the single letter in the dictionary word to a two-letter combination in the variant. We limit to the top 5 most frequent letter combinations, which are "ck", "ey", "ie", "ou", "wh", and the pairs involving other combinations are removed.

After applying the letter alignment to the collected noisy training word pairs, we obtained 298,160 letter-level alignments. Some example alignments and corresponding word pairs are:

e → '_' (have, hav) q → k (iraq, irak)
e → a (another, anotha) q → g (iraq, irag)
e → 3 (online, 0nlin3) w → wh (watch, whatch)

2.4 Sequence Labeling Model for $P(T_i|S_i)$

For a letter sequence S_i, we use the conditional random fields (CRF) model to perform sequence tagging to generate its variant T_i. To train the model, we first align the collected dictionary word and its variant at the letter level, then construct a feature vector for each letter in the dictionary word, using its mapped character as the reference label. This labeled data set is used to train a CRF model with L-BFGS (Lafferty et al., 2001; Kudo, 2005). We use the following features:

- Character-level features
  Character n-grams: c_{-1}, c_0, c_1, $(c_{-2}\ c_{-1})$, $(c_{-1}\ c_0)$, $(c_0\ c_1)$, $(c_1\ c_2)$, $(c_{-3}\ c_{-2}\ c_{-1})$, $(c_{-2}\ c_{-1}\ c_0)$, $(c_{-1}\ c_0\ c_1)$, $(c_0\ c_1\ c_2)$, $(c_1\ c_2\ c_3)$.
  The relative position of character in the word.
- Phonetic-level features
  Phoneme n-grams: p_{-1}, p_0, p_1, $(p_{-1}\ p_0)$, $(p_0\ p_1)$. We use the many-to-many letter-phoneme alignment algorithm (Jiampojamarn et al., 2007) to map each letter to multiple phonemes (1-to-2 alignment). We use three binary features to indicate whether the current, previous, or next character is a vowel.
- Syllable-level features
  Relative position of the current syllable in the word; two binary features indicating whether the character is at the beginning or the end of the current syllable. The English hyphenation dictionary (Hindson, 2006) is used to mark all the syllable information.

The trained CRF model can be applied to any English word to generate its variants with probabilities.

3 Experiments

We evaluate the system performance on both Twitter and SMS message test sets. The SMS data was used in previous work (Choudhury et al., 2007; Cook and Stevenson, 2009). It consists of 303 distinct nonstandard tokens and their corresponding dictionary words. We developed our own Twitter message test set consisting of 6,150 tweets manually annotated via the Amazon Mechanical Turk. 3 to 6 turkers were required to convert the nonstandard tokens in the tweets to the standard English words. We extract the nonstandard tokens whose most frequently normalized word consists of letters/digits/apostrophe, and is different from the token itself. This results in 3,802 distinct nonstandard tokens that we use as the test set. 147 (3.87%) of them have more than one corresponding standard English words. Similar to prior work, we use isolated nonstandard tokens without any context, that is, the LM probabilities $P(S)$ are based on unigrams.

We compare our system against three approaches. The first one is a comprehensive list of chat slangs, abbreviations, and acronyms collected by InternetSlang.com; it contains normalized word forms for 6,105 commonly used slangs. The second is the word-abbreviation lookup table generated by the supervised deletion-based abbreviation approach proposed in (Pennell and Liu, 2010). It contains 477,941 (word, abbreviation) pairs automatically generated for 54,594 CMU dictionary words. The third is the jazzy spell checker based on the Aspell algorithm (Idzelis, 2005). It integrates the phonetic matching algorithm (DoubleMetaphone) and Levenshtein distance that enables the interchanging of two adjacent letters, and changing/deleting/adding of letters. The system performance is measured using the n-best accuracy (n=1,3). For each nonstandard token, the system is considered correct if any of the corresponding standard words is among the n-best output from the system.

System Accuracy	**Twitter** (**3802** pairs)		**SMS** (**303** pairs)	
	1-best	3-best	1-best	3-best
InternetSlang	7.94	8.07	4.95	4.95
(Pennell et al. 2010)	20.02	27.09	21.12	28.05
Jazzy Spell Checker	47.19	56.92	43.89	55.45
LetterTran (Trim)	**57.44**	**64.89**	**58.09**	**70.63**
LetterTran (All)	**59.15**	**67.02**	**58.09**	**70.96**
LetterTran (All) + Jazzy	**68.88**	**78.27**	**62.05**	**75.91**
(Choudhury et al. 2007)	n/a	n/a	59.9	n/a
(Cook et al. 2009)	n/a	n/a	59.4	n/a

Table 3: N-best performance on Twitter and SMS data sets using different systems.

Results of system accuracies are shown in Table 3. For the system "LetterTran (All)", we first generate a lookup table by applying the trained CRF model to the CMU dictionary to generate up to 30 variants for each dictionary word.[3] To make the comparison more meaningful, we also trim our lookup table to the same size as the deletion table, namely "LetterTran (Trim)". The trimming was performed by selecting the most frequent dictionary words and their generated variants until the length limit is reached. Word frequency information was obtained from the entire Edinburgh corpus. For both the deletion and letter transformation lookup tables, we generate a ranked list of candidate words for each nonstandard token, by sorting the combined score $p(T_i|S_i) \times C(S_i)$, where $p(T_i|S_i)$ is the model confidence and $C(S_i)$ is the unigram count generated from the Edinburgh corpus (we used counts instead of unigram probability $P(S_i)$). Since the string similarity and letter switching algorithms implemented in jazzy can compensate the letter transformation model, we also investigate combining it with our approach, "LetterTran(All) + Jazzy". In this configuration, we combine the candidate words from both systems and rerank them according to the unigram frequency; since the "LetterTran" itself is very effective in ranking candidate words, we only use the jazzy output for tokens where "LetterTran" is not very confident about its best candidate ($(p(T_i|S_i) \times C(S_i)$ is less than a threshold $\theta = 100$).

We notice the accuracy using the InternetSlang list is very poor, indicating text message normalization is a very challenging task that can hardly be tackled by using a hand-crafted list. The deletion table has modest performance given the fact that it covers only deletion-based abbreviations and letter repetitions (see Section 2.1). The "LetterTran" approach significantly outperforms all baselines even after trimming. This is because it handles different ways of forming nonstandard tokens in an unified framework. Taking the Twitter test set for an example, the lookup table generated by "LetterTran" covered 69.94% of the total test tokens, and among them, 96% were correctly normalized in the 3-best output, resulting in 67.02% overall accuracy. The test tokens that were not covered by the "LetterTrans" model include those generated by accidentally switching and inserting letters (e.g., "absolotuely" for "absolutely") and slangs ("addy" or "address"). Adding the output from jazzy compensates these problems and boosts the 1-best accuracy, achieving 21.69% and 18.16% absolute performance gain respectively on the Twitter and SMS test sets, as compared to using jazzy only. We also observe that the "LetterTran" model can be easily ported to the SMS domain. When combined with the jazzy module, it achieved 62.05% 1-best accuracy, outperforming the domain-specific supervised system in (Choudhury et al., 2007) (59.9%) and the pre-categorized approach by (Cook and Stevenson, 2009) (59.4%). Regarding different feature categories, we found the character-level features are strong indicators, and using phonetic- and syllabic-level features also slightly benefits the performance.

4 Conclusion

In this paper, we proposed a generic letter transformation approach for text message normalization without pre-categorizing the nonstandard tokens into insertion, deletion, substitution, etc. We also avoided the expensive and time consuming hand labeling process by automatically collecting a large set of noisy training pairs. Results in the Twitter and SMS domains show that our system can significantly outperform the state-of-the-art systems and have good domain portability. In the future, we would like to compare our method with a statistical machine translation approach performed at the letter level, evaluate the system using sentences by incorporating context word information, and consider many-to-one letter transformation in the model.

[3] We heuristically choose this large number since the learned letter/digit insertion, substitution, and deletion patterns tend to generate many variants for each dictionary word.

5 Acknowledgments

The authors thank Deana Pennell for sharing the look-up table generated using the deletion-based abbreviation approach. Thank Sittichai Jiampojamarn for providing the many-to-many letter-phoneme alignment data sets and toolkit. Part of this work was done while Fei Liu was working as a research intern in Bosch Research and Technology Center.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL*, pages 33–40.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of the ACL*, pages 770–779.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175.

Asli Celikyilmaz, Dilek Hakkani-Tur, and Junlan Feng. 2010. Probabilistic model-based sentiment analysis of twitter messages. In *Proceedings of the IEEE Workshop on Spoken Language Technology*, pages 79–84.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10(3):157–174.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text messages normalization. In *Proceedings of the NAACL HLT Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78.

Matthew Hindson. 2006. English language hyphenation dictionary. *http://www.hindson.com.au/wordpress/2006/11/11/english-language-hyphenation-dictionary/*.

Mindaugas Idzelis. 2005. Jazzy: The java open source spell checker. *http://jazzy.sourceforge.net/*.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Proceedings of the HLT/NAACL*, pages 372–379.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing sms: Are two metaphors better than one? In *Proceedings of the COLING*, pages 441–448.

Taku Kudo. 2005. CRF++: Yet another CRF took kit. *http://crfpp.sourceforge.net/*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the ICML*, pages 282–289.

Deana L. Pennell and Yang Liu. 2010. Normalization of text messages for text-to-speech. In *Proceedings of the ICASSP*, pages 4842–4845.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. The edinburgh twitter corpus. In *Proceedings of the NAACL HLT Workshop on Computational Linguistics in a World of Social Media*, pages 25–26.

Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech and Language*, 15(3):287–333.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the ACL*, pages 144–151.

Dong Yang, Yi cheng Pan, and Sadaoki Furui. 2009. Automatic chinese abbreviation generation using conditional random field. In *Proceedings of the NAACL HLT*, pages 273–276.