Twitter Normalization via 1-to-N Recovering

Yafeng Ren(✉), Jiayuan Deng, and Donghong Ji

Computer School, Wuhan University, Wuhan, China
renyafeng@whu.edu.cn

Abstract. Twitter messages are written in an informal style, which hinders many information retrieval and natural language processing applications. Existing normalization systems have two major drawbacks. The first is that these methods largely require large-scale annotated training data. The second is that these systems assume that a nonstandard token is recovered to one standard word. However, there are many nonstandard tokens that should be recovered to two or more standard words, so the problem remains to be highly challenging. To address the above issues, we propose an unsupervised normalization system based on the context similarity. The proposed system does not require any annotated data. Meanwhile, a nonstandard token will be recovered to one or more standard words. Results show that the proposed approach achieves state-of-the-art performance.

Keywords: Twitter normalization · Forward search · Random walk · Spell checker

1 Introduction

User-generated contents have drastically increased in the past few years, driven by the development of microblogs. These user-generated contents, which contains rich information, become a heated research topic in natural language processing and text mining [1,25,26]. Twitter is one among these microblogging services that count about one billion of active users and 500 million of daily messages[1]. However, due to the nature of posts, twitter messages contain many nonstandard tokens, which are created both intentionally and unintentionally by people. For examples, substituting numbers for letters such as *2gether* (*together*) and *2morrow* (*tomorrow*), repeating letters for emphasizing the expression such as *coolllllll* (*cool*) and *birthdayyyyyy* (*birthday*), eliminating vowels such as *ppl* (*people*), and substituting phonetically similar letters such as *fon* (*phone*). Besides, there are another type of nonstandard tokens in reality, this type of nonstandard tokens mainly contains the omission of the spaces, punctuation and letters among successive multiple standard words, these nonstandard tokens also should be converted into their standard forms. e.g. *theres* (*there is*), *thatthe* (*that the*), *untiltheend* (*until the end*), and *ndyou* (*and you*). We randomly count 1000

[1] http://expandedrambling.com/.

W. Cellary et al. (Eds.): WISE 2016, Part I, LNCS 10041, pp. 19–34, 2016.
DOI: 10.1007/978-3-319-48740-3_2

tweets, and find that there are 1326 nonstandard tokens in which 294 tokens should be recovered to two or more standard words.

Twitter has become a very valuable information source for many natural language processing (NLP) applications, such as information extraction [29], summarization [17], sarcasm detection [34], sentiment analysis [27,28] and event discovery [3]. However, the nonstandard tokens limit the performance of standard NLP tools [10,29]. Previous work reported that the Stanford named entity recognizer (NER) experienced a performance drop from 90.8 % to 45.8 % on tweets [19], and the part-of-speech (POS) tagger and dependency parser degraded 12.2 % and 20.65 % on tweets, respectively. It is therefore of great importance to normalize twitter message before applying standard NLP techniques.

In recent years, some attempts have been made to normalize nonstandard tokens to their standard forms [10–12,15,18,33]. But these approaches all assume that a nonstandard token is normalized to one standard word. For example, *hiiiiii* is converted into *hi*. Note that *theyre* is converted into *they* and *itmay* is deployed as *it*. It is obvious that previous systems limit the performance of this task.

In this paper, we focus on the task of the normalization of English Twitter messages, which can be regarded as a pre-processing step for NLP applications. This is a challenging task based on two reasons. First, for a nonstandard token, it should be recovered to one or more standard words? Second, text normalization as a preprocessing step should have high precision and recall to have a good impact on various NLP applications.

In this paper, we propose an unsupervised normalization system to address the above challenges. Firstly, according to the characteristics of nonstandard tokens, a nonstandard token will be divided into the possible multiple words (standard word or noisy word) using forward search and backward search. Then the normalization candidates are generated for each noisy word by integrating random walk and spell checker, the essence of these two step is transform 1-to-N recovering into 1-to-1 recovering. Finally, the best normalization candidate is selected based on a n-gram language model.

The main contributions of this paper are as follows:

- In our system, a nonstandard token can be recovered to one or more standard words. To our knowledge, 1-to-N recovering in twitter normalization is proposed for the first time.
- Results show that our system achieves the best performance (a 10 % absolute increment compared to state-of-the-art). The proposed approach can be deployed as a preprocessing step for various NLP application to handle twitter message.

2 Related Work

With the rapid development of social media, text normalization system has drawn increasing attention in recent years. There are many systems proposed for tackling text normalization. Previous methods are mainly divided into two directions: NC-based model (Noisy Channel) and SMT-based model (Statistical Machine Translation).

Suppose the ill-formed text is T and its corresponding standard form is S. NC-based model [31] aims to find $argmax\ P(S|T)$ by computing $argmax\ P(T|S)P(S)$, in which $P(S)$ is usually a language model and $P(T|S)$ is an error model. Brill and Moore (2000) characterise the error model by computing the product of operation probabilities on slice-by-slice string edits [4]. Toutanova and Moore (2002) improve the model by incorporating the pronunciation information [32]. Choudhury et al. (2007) model the word-level text generation process for SMS messages [5], by considering graphemic/phonetic abbreviations and unintentional typos. Cook and Stevenson (2009) expand the error model by introducing the inferences from different erroneous formation processes [7]. However, these models make the strong assumption that a token $t_i \in T$ only depends on $s_i \in S$, ignoring the context around a token, which could be utilized to help in resolving ambiguity.

SMT-based model, which has been widely proposed as a means of context-sensitive text normalization, treats the ill-formed text as the source language, and the standard form as the target language. For example, Aw et al. (2006) propose a phrase-level SMT SMS normalization method with bootstrapped phrase alignments [2]. However, SMT method tends to suffer from a critical lack of training data. It is labor intensive to construct an annotated corpus to sufficiently cover ill-formed words and context-appropriate corrections.

Recent work focuses on normalizing the twitter message, which map a noisy form to a normalized form. Han and Baldwin (2011) develop a classifier for detecting the ill-formed words, and generate corrections based on the morphophonemic similarity [10]. Gouws et al. (2011) use a normalization lexicon based on string and distribution similarity to detect noisy words. Han et al. (2012) introduce a similar approach by generating a normalization lexicon based on distributional similarity and string similarity [11].

More recently, Hassan and Menezes (2013) first find the possible candidates based on bipartite graph [12], then they construct a lattice from possible normalization candidates, and determine the best normalization sequence according to a n-gram language model. Wang and Ng (2013) propose a beam-search decoder to effectively integrate various normalization operations [33], and apply their normalization to machine translation tasks for both Chinese and English. Li and Liu (2014) propose a re-ranking strategy to combine the results from different systems [15]. Besides, Li and Liu (2015) propose a joint Viterbi decoding process to determine each token's POS tag and non-standard token's correct form at the same time [16]. Cotelo et al. (2015) and Schulz et al. (2016) explore the text normalization on Spanish and Dutch, respectively [8,30].

However, the above methods have two major problems. The first is these methods largely require large-scale annotated training data, limiting their adaptability to new domains and languages. The second is these method assume that the relationship between nonstandard token and standard word is 1-to-1 recovering. But there are large amounts of nonstandard tokens (e.g. *howyou*, *havent*), which these nonstandard tokens should be converted into two or more standard words. Some key information may lost if we only use 1-to-1 recovering (e.g. *howyou* will be converted into *how*, and *havent* will be converted into *have*). In this paper, we propose an unsupervised normalization system, where a nonstandard token is converted into its best possible candidate with 1-to-N recovering so that the proposed system can satisfy various NLP applications in reality.

3 Text Normalization System

The main objective of this paper is convert the noisy tweet text as the source input into the normalized tweet as the target output. The overall framework of the proposed system is shown in Fig. 1. For a tweet, a standard dictionary first is used to determine whether it is need to be normalized. Then, a nonstandard token is considered 1-to-1 or 1-to-N recovering based on the characteristics of nonstandard tokens. For 1-to-N recovering, the nonstandard token will be

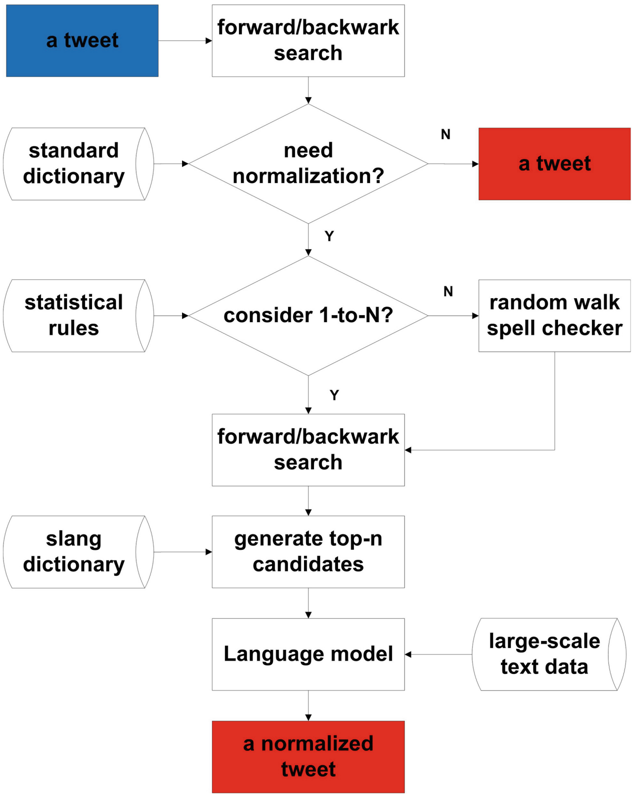

Fig. 1. The overall framework of the proposed normalization system. The blue area represents the input, and the red area represents the output. (Color figure online)

divided into multiple possible words using forward search and backward search. Then, noisy words among multiple possible words and the overall token (the nonstandard token itself) are generated some normalization candidates by integrating random walks and spell checker. Finally, we get the best normalized tweet by taking all candidates into consideration of n-gram language model. In the following section, we will introduce the proposed system in details.

3.1 Pre-processing and Standard Dictionary Generation

The first step of the proposed system is determine whether a tweet is needed to normalize. According to the nature of social media text, we need some pre-processing before a tweet is considered to normalize. The following tokens are excluded for normalization:

– The tokens begun with the symbol # or @, e.g. #kingJames, @michael;
– The tokens constructed completely by numbers, e.g. 2014;
– The emoticons or URLs;

In twitter, there are many named entities which should not be normalized. Our standard dictionary must enough broad so that it can include these named entities. To get a standard dictionary, we first calculate the frequency of all different tokens based on a large-scale clean corpus (LDC2011T07), and delete the tokens that its predefined threshold is lower than 5. Then, we filter out some tokens using the GUN Aspell dictionary (v0.60.6)[2]. The remaining tokens construct our standard vocabulary, which includes nearly 68,000 words. Naturally, a nonstandard token is defined as a token that does not exist in the standard vocabulary.

3.2 Statistical Rules

In the above section, we discuss which token need to be normalized. Next, the key challenge is that a nonstandard token should be normalized to one or more standard words. We conclude some characteristics by observing and analyzing large amounts of nonstandard tokens. For 1-to-1 recovering, there are following rules:

– The nonstandard token that its length is lower than 4 usually should be recovered to one standard word, e.g. *u* (*you*) and *r* (*are*);
– The nonstandard token that contains letters and numbers is usually should be recovered to one standard word, e.g. *b4* (*before*) and *2day* (*today*);

In our system, the nonstandard tokens that meets the one of these two rules are normalized one standard word. Conversely, the nonstandard token is considered to normalize one or more standard words. These rules can not capture all situations, for example, *ur* may be converted into *you are*, but *ur* may be converted into *your* or *our*, this type of nonstandard token is ambiguous, its best

[2] http://aspell.net/.

normalization depends on the specific context, but it belongs to 1-to-1 recovering normalization in most situations.

We also analyze the nonstandard tokens that should be recovered to two or more standard words. These tokens mainly include the following three classes:

– **Type 1:** The nonstandard tokens are constructed by two or more standard word, the space among these words are omitted by people, intentionally and unintentionally. e.g. *rememberwith* (*remember with*) and *Iloveyousomuch* (*I love you so much*);
– **Type 2:** The nonstandard tokens are constructed by one or more standard and a nonstandard word. e.g. *wasnt* (*was not*) and *ndyou* (*and you*);
– **Type 3:** The nonstandard tokens are constructed by the acronym of two or more standard words. e.g. *ur* (*you are*) and *sm* (*so much*);

Type 3 is very limited, and this type of nonstandard tokens are much ambiguity. In our system, for this type of nonstandard token, its possible normalization candidates are added into the slang dictionary[3] created by the web users. Finally, we find the best candidate according to its context by using language model.

3.3 Forward and Backward Search

For a nonstandard token, we must find the possible multiple words if this token need to be normalized two or more standard words. According to statistical nature of the nonstandard token. We propose to generate the possible multiple words by using forward search and backward search. Forward search can solve two types of nonstandard tokens, the first type is composed of one standard word and the nonstandard word from front to back, e.g. *theyre* (*they are*) and *wouldnt* (*would not*). The second type is composed of two or more standard word, e.g. *aboutthem* (*about them*) and *Iloveyou* (*I love you*). Forward search algorithm is shown as Algorithm 1:

In Algorithm 1, the trie is a tree that is constructed using the standard vocabulary. Based on the trie, we can quickly find the standard word for a nonstandard token.

Similar to forward search, backward search can also solve two types of nonstandard tokens, the first type is composed of one standard word and the nonstandard word from back to front, e.g. *anyou* (*and you*) and *looveyou* (*love you*). The second type is totally composed of two or more standard words. The specific algorithm of backward search is similar to forward search, we only change the search order for a nonstandard token.

After forward search and backward search, the nonstandard token is divided into multiple words, which include standard word and noisy word, e.g., *ndyou* is divided into a noisy word *nd* and a standard word *you*. Meanwhile, the nonstandard token itself (*ndyou*) will be considered as a noisy word, we must find the possible normalization candidates for all noisy words. In the following section, we will discuss how to find the possible candidates for a noisy word.

[3] http://www.noslang.com/dictionary.

Algorithm 1. ForwardSearch(token)

Input: a nonstandard token; trie;
Output: SegmentTable;

```
length=token.length();
for(i=1; i $\leq$ length;)
  {word1=token.substring(0, i);
    if (word1 $\in$ trie)
      {Add word1 to SegmentTable; word2=token.substring(i+1, length);
      if (word2 $\in$ trie || word2<4)
        {Add word2 to SegmentTable; i++; continue; }
      else
        ForwardSearch(word2);
      }
    else
      { if (i==length)
        Add token in SegmentTable;
        else
          i++;
      }
    }
```

3.4 Random Walk and Spell Checker

The noisy word includes many types, such as lengthening, letter substitution, letter-number substitution and phonetic substitution. The natural idea is spell checker to recover the noisy word. But spell checker mainly can recover the noisy words that are created unintentionally by users. In order to recover the noisy words created intentionally by user, we use random walk based on bipartite graph to find the proper candidate. In this paper, a noisy word generates some normalization candidates by integrating random walk and spell checker.

For a noisy word, to get its normalization candidates using random walk based on bipartite graph, there are two hypothesis for tweets as follows:

– **Hypothesis 1:** The words share the same or similar contexts should be the same word or semantically related words;
– **Hypothesis 2:** The users usually have different writing styles. For a twitter, some users may use all standard words to express, and other user may use one or more noisy words.

Hypothesis 1 tells us the normalization equivalence between a noisy word and a standard word by sharing the same or similar contexts. Hypothesis 2 tells us there are a large number of such contexts in twitter. Based on two hypothesis, we can generate the candidates for a noisy word by using contextual similarity. For instance, assume 5-gram sequences of words, two words may be the same word if their contexts share the same two words on the left and the same two words on the right, e.g. for the word *be4* and *before*, they share the same contexts *the day * the day* and *dress yourself * contract me*, so we think that these two words are the same or semantically related, and *before* is considered to the best normalization for the noisy word *be4* by using the string similarity and language model. To use the random walk, we firstly introduce the bipartite graph representation.

Bipartite Graph Representation. Contextual similarity can be represented as a bipartite graph where the first partite represents the words, and the second partite represents the contexts that are shared by words. A word node can be either noisy word or standard word. Figure 2 shows a sample of the bipartite graph $G(W, C, E)$, where standard words are shown as blue nodes.

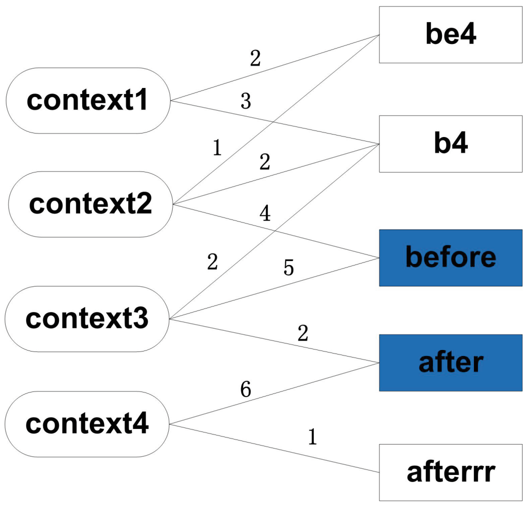

Fig. 2. Bipartite graph representation. Left nodes represent the contexts, right nodes with blue colour represent the standard words, and other nodes represent the noisy words. The weight of the edge is the co-occurrence of a word and its context. (Color figure online)

The bipartite graph, $G(W, C, E)$, is composed of W which includes the nodes representing noisy words and standard words, C includes the nodes representing shared context, and E represents the edges of the bipartite graph connecting word nodes and context nodes. The weight of the edge is the number of occurrences of a given word in a context. The bipartite graph is constructed using Algorithm 2.

Candidates Generation Using Random Walk. Random walk based on bipartite graph is defined [23] and then used in many NLP applications. For example, Hughes and Ramage (2007) used random walk on Wordnet graph to measure lexical semantic relatedness between words [13]. Das and Petrov (2011) used graph-based label propagation for cross-lingual knowledge to induce POS tags between two languages [9]. Minkov and Cohen (2012) introduced a path

Algorithm 2. ConstructBipartiteGraph(text)

```
Input: Twitter corpus;
Output: G(W,C,E);
Extract all 5-gram sequences from twitter corpus;
Store all sequences into NgramTable;
for each sequence ∈ NgramTable
{
  $W$ ← Add(CenterWord);
  $C$ ← Add(Context);
  $E$ ← Add(Context, Word, count);
}
```

constrained graph walk algorithm given a small number of labeled examples to assess nodes relatedness in the graph [22]. Hassan and Menezes (2013) used Markov random walk to consider normalization equivalences based on twitter and LDC data [12]. In this paper, we use graph-based random walk to find the normalization candidates for a noisy word by only using the twitter text.

For a noisy word, the random walk algorithm repeats independent random walk for 4 steps where the walks traverse the graph randomly according to roulette rule. Each walk starts from a noisy word node and ends at a standard word node, or consumes the maximum number of steps (4 steps in our algorithm) without hitting a standard word node. In our random walk, for a noisy word, the process of random walk algorithm will be iterated 100 times.

Consider a random walk on the bipartite graph $G(W, C, E)$ starting a noisy word and ending at a standard word. For the bipartite graph in Fig. 2, assume a random walk staring at the node representing the noisy word *be4* then moves to the context node **context2** then to the node representing the standard word *before*. This random walk will associate *be4* with *before*. Meanwhile, the noisy word *be4* can first find the noisy word *b4* by sharing **context1**, then noisy word *b4* can find the standard word *after* by sharing **context3**, this random walk will associate *be4* with *after*. For noisy word *be4*, which word is better candidate for normalization?

For a noisy word, we rank its all candidates based on the confidence $Conf(N, S)$ as the result of random walk, N represents the noisy word and S represents the standard word. $Conf(N, S)$ is calculated as:

$$Conf(N, S) = \alpha F(S) + \beta Sim(N, S) \qquad (1)$$

Where, α and β are weights, we use uniform interpolation, both $\alpha = \beta = 1$. $F(S)$ is calculated as follows:

$$F(S) = S\_frequency/total\_frequency \qquad (2)$$

$S\_frequency$ is the times of standard word S in all random walks, and $total\_frequency$ is total numbers of all standard words in 100 iteration. $Sim(N, S)$ is a similarity function based on Longest Common Subsequence Ratio

(LCSR) [6,20]. This function is defined as the ratio of LCSR and Edit distance between two string as follows:

$$Sim(N,S) = \frac{LCS(N,S)/MaxLength(N,S)}{ED(N,S)} \tag{3}$$

$LCS(N,S)$ represents the length of Longest Common Subsequence between word N and S. Edit distance calculation $ED(N,S)$ is modified to be more adequate for social media text according to previous work [12];

For a noisy word N, we can construct a lexicon that includes all normalization candidates by using random walk. We can prune the lexicon to take top-5 according to $ConValue(N,S)$. The advantage of random walk is that it can find the proper candidates for a nonstandard word. But random walk can not find candidates for all noisy words due to the limitation of our twitter data. So we must use another method to generate the normalization candidates for a noisy word. In the paper, we use spell checker to generate some candidates as a supplement for random walk.

Spell Checker. Spell checker is a simple and effective method in normalizing misspellings. In this paper, we use the Jazzy spell checker [14] that integrates the DoubleMetaphone phonetic matching algorithm and the Levenshtein distance using the near-miss strategy, which enables the interchange of two adjacent letters, and the replacing/deleting/adding of letters. In this paper, the edit distance is set to 2 for generating candidates.

Generating Top-N Candidates. For a nonstandard token, some candidates are generated by using random walk and spell checker. We need to rank for these candidates. In addition to considering the string similarity, we use the semantic similarity. The final score between a nonstandard token N and standard word S is:

$$S(N,S) = Sim(N,S) + C(vec(N), vec(S)) \tag{4}$$

$Sim(N,S)$ is defined in the above section, where the function $C(vec(N), vec(S))$ represents the cosine similarity between $vec(N)$ and $vec(S)$. In order to compute the semantic similarity of words, we use the tool word2vec[4] to implement it based on 8 million twitter message by using feed-forward neutral network language model [21], the vector dimension is set to 200. According to the value of $S(N,S)$, we select the top-n candidates for a nonstandard token N (n = 1, 3 and 5).

3.5 Normalizing Twitter Using Language Model

After generate top-n candidates for a nonstandard token, we need determine the best candidate to convert a noisy tweet into a normalized tweet. We must take consideration of the context information. Based on large amount of text data, we score the best Viterbi path with 3-gram and 4-gram language model. We use three different types of texts, which include twitter text, clean text (newspaper data) and mixed text (twitter text and newspaper data).

[4] http://code.google.com/p/word2vec/.

4 Experiments

4.1 Experimental Setup

The following datasets are used in our experiments. Dataset (1) and (2) are used for word-level evaluation, and dataset (3) is used for both word- and message-level evaluation. Note that dataset (1) and (2) only contain 1-to-1 recovering.

- **Dataset 1:** 3,802 nonstandard tokens along with their human-annotated normalized word forms. The nonstandard tokens are collected from a corpus with about 6,150 tweets between 2009 and 2010 [18].
- **Dataset 2:** 2,333 unique pairs of nonstandard tokens and standard words, collected from 2,577 Twitter messages [15,24].
- **Dataset 3:** 1,000 tweets are annotated by three native human. This dataset includes 850 noisy tweets and 150 clean tweets. The dataset contains 1,345 nonstandard tokens in which 297 tokens need to recovered to two or more standard words.
- **Dataset 4:** 8M tweets, which are collected from October to December 2014 using the Twitter Streaming APIs[5], these tweets are constructed to bipartite graph for random walk.
- **Dataset 5:** clean data from English LDC Gigaword corpus[6]. This dataset is used to construct a standard vocabulary.

The goal of word-level normalization is to convert the nonstandard tokens into standard words. For each nonstandard token, the system is considered correct if any of the corresponding standard words among the top-n candidates from the system. We adopt this word-level top-n accuracy to make our results comparable to the state-of-the-art systems. On the message-level, we evaluate the top-1, top-3 and top-5 system output using precision, recall, and F-score, calculated respectively to the nonstandard tokens based on language model.

4.2 Word-Level Results

The word-level results are presented in Table 1, evaluated on dataset (1) and (2) respectively. We present the top-n accuracy ($n = 1, 3, 5$) of the proposed approach.

The spell checker gives only 40 % to 60 % accuracy on dataset (1) and (2), indicating that the vast amount of intentionally created nonstandard tokens can hardly be tackled by a system relies solely on lexical similarity. The random walk performs surprisingly well, and shows robust performance across two datasets, random walk achieves 65 % to 80 % accuracy on all datasets. Compared to spell checker, random walk is effective for normalizing intentionally created tokens. Finally, the mixed system of spell checker and random walk achieves 90 % to 94 % accuracy in top-5 on dataset (1) and (2), showing the effectiveness of our proposed system.

[5] http://dev.twitter.com/docs/streaming-apis.
[6] http://www.ldc.upenn.edu/Catalog/LDC2011T07.

Table 1. Word-level results.

Method	Accuracy (%)					
	Dataset (1)			Dataset (2)		
	top-1	top-3	top-5	top-1	top-3	top-5
Spell checker	47.2	56.9	58.3	39.9	46.5	47.1
Random walk	67.2	73.7	79.6	64.7	71.4	76.4
Mixed system	76.8	87.3	**94.1**	75.2	85.1	**90.3**
Hassan and Menezes, 2013	74.4	85.3	92.8	74.1	83.0	88.2

Compared with previous work [12], our approach achieves better performance in accuracy. The main reason is that the normalization candidates by integrating random walk and spell checker have better coverage for nonstandard tokens created both unintentionally and intentionally.

4.3 Message-Level Results

The goal of message-level normalization is to replace a nonstandard token with the candidate that best fits the context. Our system is evaluated on dataset (3), and the results are shown in Table 2.

In Table 2, the "*w/o Context*" results are generated by replacing each nonstandard token using top-1 word-level candidate. Although the replacement process is static, it results in 84.3 % F-score due to the high performance of the word-level system. We explore three different data as language models (LM) for the Viterbi decoding process. For the twitter data as language model, it is clear that increasing the amount of candidate gives better precision and recall. When n is fixed to 5, we can get 86.4 % in F-score. As Table 2 shows, Mixed LM and

Table 2. Message-level results.

Dataset (3)	Language model	Message-level		
		Precision	Recall	F-score
Word-level (top-1)	*w/o Context*	78.2	91.3	84.3
w/Context (top-3)	Twitter LM	79.8	93.1	86.0
	LDC LM	77.6	90.6	83.6
	Mixed LM	79.6	92.9	85.7
w/Context (top-5)	Twitter LM	**80.2**	**93.6**	**86.4**
	LDC LM	77.2	90.1	83.1
	Mixed LM	79.7	93.0	85.8
Hassan and Menezes, 2013	Twitter LM	74.4	71.3	72.8
Li and Liu, 2014	Twitter LM	76.9	72.5	74.6

Twitter LM achieve better performance than the previous best results, demonstrating the effectiveness of language model. Results show that our proposed system have better performance in F-score to serve as a reliable preprocessing step for standard NLP applications.

For the LDC data as language model, the proposed system experiences a performance drop with the increasing of the amount of candidates. The main reason is twitter text is informal, there are some grammatical errors even if the nonstandard tokens are converted into the standard words. It is not difficult to explain the reason that twitter text is better than mixed text as language model for twitter normalization.

Compared with previous systems [12,15], our model outperforms the current systems, which reach a 10 % absolute increment. The main reason is our proposed method exploits 1-to-N recovering, but previous work limits 1-to-1 recovering in this task. The proposed system can be well applied in reality.

4.4 Output Analysis

Table 3 lists three examples from dataset (3) and their normalizations using previous methods and our proposed system. At the first example, the nonstandard tokens can both be recovered to their standard words. For example 2 and 3, previous methods can not get the proper normalization because they only use 1-to-1 recovering in which some information may be lost. Our proposed 1-to-N recovering can get the best normalization. Our proposed system is desirable as a preprocessing step for various NLP applications.

Table 3. Twitter normalization examples, **S** represents the source tweet, **B** represents the current approach (Li and Liu, 2014) and **O** is our proposed method.

S: *u* just follow the wrong *ppl* but that is ok !
B: *you* just follow the wrong *people* but that is ok !
O: *you* just follow the wrong *people* but that is ok !
S: *hiii*! I hope *youre* doing *welland* having a nice day
B: *hi*! I hope *you* doing *well* having a nice day
O: *hi*! I hope *you are* doing *well and* having a nice day
S: *youre* such *anamazing* human *beingwho* deserves so *muchlove*
B: *you* such *amazing* human *being* deserves so *much*
O: *you are* such *an amazing* human *being who* deserves so *much love*

5 Conclusion

In this paper, we proposed an unsupervised normalization system in which a noisy tweet can be converted into a normalized tweet. For a nonstandard token, our proposed system uses 1-to-N recovering to get its standard form, which contains one or more standard words. Experimental results show that the proposed system significantly outperforms the state-of-the-art systems in F-score on the dataset. The proposed approach can be deployed as a preprocessing step for various NLP applications to handle the tweet text.

Acknowledgments. This work is supported by the State Key Program of National Natural Science Foundation of China (Grant No. 61133012), the National Natural Science Foundation of China (Grant Nos. 61173062, 61373108) and the National Philosophy Social Science Major Bidding Project of China (Grant No. 11&ZD189).

References

1. Almeida, T.A., Silva, T.P., Santos, I., Hidalgo, J.M.G.: Text normalization and semantic indexing to enhance instant messaging and sms spam filtering. Knowl. Based Syst. **108**, 25–32 (2016)
2. Aw, A., Zhang, M., Xiao, J., Su, J.: A phrase-based statistical model for sms text normalization. In: Proceedings of the Joint Conference on Annual Meeting of the Association for Computational Linguistics and International Conference on Computational Linguistics, pp. 33–40 (2006)
3. Benson, E., Haghighi, A., Barzilay, R.: Event discovery in social media feeds. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pp. 389–398 (2011)
4. Brill, E., Moore, R.C.: An improved error model for noisy channel spelling correction. In: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pp. 286–293 (2000)
5. 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**(3–4), 157–174 (2007)
6. Contractor, D., Faruquie, T.A., Subramaniam, L.V.: Unsupervised cleansing of noisy text. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 189–196 (2010)
7. 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 (2009)
8. Cotelo, J.M., Cruz, F.L., Troyano, J., Ortega, F.J.: A modular approach for lexical normalization applied to spanish tweets. Expert Syst. Appl. **42**(10), 4743–4754 (2015)
9. Das, D., Petrov, S.: Unsupervised part-of-speech tagging with bilingual graph-based projections. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pp. 600–609 (2011)
10. 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, pp. 368–378 (2011)
11. Han, B., Cook, P., Baldwin, T.: Automatically constructing a normalisation dictionary for microblogs. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 421–432 (2012)
12. Hassan, H., Menezes, A.: Social text normalization using contextual graph random walks. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pp. 1577–1586 (2013)

13. Hughes, T., Ramage, D.: Lexical semantic relatedness with random graph walks. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 581–589 (2007)
14. Idzelis, M.: Jazzy: the java open source spell checker (2005)
15. Li, C., Liu, Y.: Improving text normalization via unsupervised model and discriminative reranking. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 86–93 (2014)
16. Li, C., Liu, Y.: Joint pos tagging and text normalization for informal text. In: Proceedings of the 24th International Conference on Artificial Intelligence, pp. 1263–1269 (2015)
17. Liu, F., Liu, Y., Weng, F.: Why is sxsw trending? exploring multiple text sources for twitter topic summarization. In: Proceedings of the Workshop on Languages in Social Media, pp. 66–75 (2011)
18. Liu, F., Weng, F., Wang, B., Liu, Y.: Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pp. 71–76 (2011)
19. Liu, X., Zhang, S., Wei, F., Zhou, M.: Recognizing named entities in tweets. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pp. 359–367 (2011)
20. Melamed, I.D.: Bitext maps and alignment via pattern recognition. Comput. Linguist. **25**(1), 107–130 (1999)
21. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
22. Minkov, E., Cohen, W.W.: Graph based similarity measures for synonym extraction from parsed text. In: Proceedings of the Workshop on Graph-based Methods for Natural Language Processing, pp. 20–24 (2012)
23. Norris, J.R.: Markov Chains. Cambridge University Press, New York (1998)
24. Pennell, D., Liu, Y.: A character-level machine translation approach for normalization of sms abbreviations. In: Proceedings of the 5th International Joint Conference on Natural Language Processing, pp. 974–982 (2011)
25. Ren, Y., Ji, D., Yin, L., Zhang, H.: Finding deceptive opinion spam by correcting the mislabeled instances. Chin. J. Electron. **24**(1), 52–57 (2015)
26. Ren, Y., Ji, D., Zhang, H.: Positive unlabeled learning for deceptive reviews detection. In: Proceedings of the 2014 Joint Conference on Empirical Methods in Natural Language Processing, pp. 488–498 (2014)
27. Ren, Y., Zhang, Y., Zhang, M., Ji, D.: Context-sensitive twitter sentiment classification using neural network. In: Proceedings of the 30th AAAI Conference on Artifical Intelligence, pp. 215–221 (2016)
28. Ren, Y., Zhang, Y., Zhang, M., Ji, D.: Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings. In: Proceedings of the 30th Conference on Artificial Intelligence, pp. 3038–3044 (2016)
29. Ritter, A., Clark, S., Etzioni, O., et al.: Named entity recognition in tweets: an experimental study. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 1524–1534 (2011)
30. Schulz, S., De Pauw, G., De Clercq, O., Desmet, B., Hoste, V., Daelemans, W., Macken, L.: Multi-modular text normalization of dutch user-generated content. ACM Trans. Intell. Syst. Technol. **7**(4), 1–22 (2016)
31. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mob. Comput. Commun. Rev. **5**(1), 3–55 (2001)

32. Toutanova, K., Moore, R.C.: Pronunciation modeling for improved spelling correction. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 144–151 (2002)
33. Wang, P., Ng, H.T.: A beam-search decoder for normalization of social media text with application to machine translation. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics, pp. 471–481 (2013)
34. Wang, Z., Wu, Z., Wang, R., Ren, Y.: Twitter sarcasm detection exploiting a context-based model. In: Proceedings of the International Conference on Web Information Systems Engineering, pp. 77–91 (2015)