A Broad-Coverage Normalization System for Social Media Language

Fei Liu, Fuliang Weng, Xiao Jiang
Research and Technology Center
Robert Bosch LLC
{fei.liu, fuliang.weng}@us.bosch.com
{fixed-term.xiao.jiang}@us.bosch.com

Abstract

Social media language contains huge amount and wide variety of nonstandard tokens, created both intentionally and unintentionally by the users. It is of crucial importance to normalize the noisy nonstandard tokens before applying other NLP techniques. A major challenge facing this task is the system coverage, i.e., for *any* user-created nonstandard term, the system should be able to restore the correct word within its top n output candidates. In this paper, we propose a cognitively-driven normalization system that integrates different *human* perspectives in normalizing the nonstandard tokens, including the enhanced letter transformation, visual priming, and string/phonetic similarity. The system was evaluated on both word- and message-level using four SMS and Twitter data sets. Results show that our system achieves over 90% word-coverage across all data sets (a 10% absolute increase compared to state-of-the-art); the broad word-coverage can also successfully translate into message-level performance gain, yielding 6% absolute increase compared to the best prior approach.

1 Introduction

The amount of user generated content has increased drastically in the past few years, driven by the prosperous development of the social media websites such as Twitter, Facebook, and Google+. As of June 2011, Twitter has attracted over 300 million users and produces more than 2 billion tweets per week (Twitter, 2011). In a broader sense, Twitter messages, SMS messages, Facebook updates, chat logs, Emails, etc. can all be considered as "social text", which is significantly different from the traditional news text due to the informal writing style and the conversational nature. The social text serves as a very valuable information source for many NLP applications, such as the information extraction (Ritter et al., 2011), retrieval (Subramaniam et al., 2009), summarization (Liu et al., 2011a), sentiment analysis (Celikyilmaz et al., 2010), etc. Yet existing systems often perform poorly in this domain due the to extensive use of the nonstandard tokens, emoticons, incomplete and ungrammatical sentences, etc. It is reported that the Stanford named entity recognizer (NER) experienced a performance drop from 90.8% to 45.8% on tweets (Liu et al., 2011c); the part-of-speech (POS) tagger and dependency parser degraded 12.2% and 20.65% respectively on tweets (Foster et al., 2011). It is therefore of great importance to normalize the social text before applying the standard NLP techniques. Text normalization is also crucial for building robust text-to-speech (TTS) systems, which need to determine the pronunciations for nonstandard words in the social text.

The goal of this work is to automatically convert the noisy nonstandard tokens observed in the social text into standard English words. We aim for a robust text normalization system with "broad coverage", i.e., for *any* user-created nonstandard token, the system should be able to restore the correct word within its top n candidates ($n = 1, 3, 10...$). This is a very challenging task due to two facts: first, there exists huge amount and a wide variety of nonstandard tokens. (Liu et al., 2011b) found more than 4 million distinct out-of-vocabulary tokens in the Edinburgh Twitter corpus (Petrovic et al., 2010); second, the nonstandard tokens consist

2gether (6326)	togetha (919)	tgthr (250)	togeda (20)
2getha (1266)	togather (207)	t0gether (57)	toqethaa (10)
2gthr (178)	togehter (94)	togeter (49)	2getter (10)
u (3240535)	ya (460963)	yo (252274)	yaa (17015)
yaaa (7740)	yew (7591)	yuo (467)	youz (426)
yooooooou (186)	youy (105)	yoiu (128)	yooooouuuu (82)

Table 1: Nonstandard tokens and their frequencies in the Edinburgh Twitter corpus. The corresponding standard words are "**together**" and "**you**", respectively.

of a mixture of both unintentional misspellings and intentionally-created tokens for various reasons[1], including the needs for speed, ease of typing (Crystal, 2009), sentiment expressing (e.g., "coooool" (Brody and Diakopoulos, 2011)), intimacy and social purpose (Thurlow, 2003), etc., making it even harder to decipher the social messages. Table 1 shows some example nonstandard tokens.

Existing spell checkers and normalization systems rely heavily on lexical/phonetic similarity to select the correct candidate words. This may not work well since a good portion of the correct words lie outside the specified similarity threshold (e.g., (tomorrow, "tmrw")[2]), yet the number of candidates increases dramatically as the system strives to increase the coverage by enlarging the threshold. (Han and Baldwin, 2011) reported an average of 127 candidates per nonstandard token with the correct-word coverage of 84%. The low coverage score also enforces an undesirable performance ceiling for the candidate reranking approaches. Different from previous work, we tackle the text normalization problem from a cognitive-sensitive perspective and investigate the human rationales for normalizing the nonstandard tokens. We argue that there exists a set of letter transformation patterns that humans use to decipher the nonstandard tokens. Moreover, the "visual priming" effect may play an important role in human comprehension of the noisy tokens. "Priming" represents an implicit memory effect. For example, if a person reads a list of words including the word *table*, and is later asked to complete a word starting with *tab-,* it is very likely that he answers *table* since the person is *primed*.

In this paper, we propose a broad-coverage normalization system by integrating three human perspectives, including the enhanced letter transformation, visual priming, and the string and phonetic similarity. For an arbitrary nonstandard token, the three subnormalizers each suggest their most confident candidates from a different perspective. The candidates can then be heuristically combined or reranked using a message-level decoding process. We evaluate the system on both word- and message-level using four SMS and Twitter data sets. Results show that our system can achieve over 90% word-coverage with limited number of candidates and the broad word-coverage can be successfully translated into message-level performance gain. In addition, our system requires no human annotations, therefore can be easily adapted to different domains.

2 Related work

Text normalization, in its traditional sense, is the first step of a speech synthesis system, where the numbers, dates, acronyms, etc. found in the real-world text were converted into standard dictionary words, so that the system can pronounce them correctly. Spell checking plays an important role in this process. (Church and Gale, 1991; Mays et al., 1991; Brill and Moore, 2000) proposed to use the noisy channel framework to generate a list of corrections for any misspelled word, ranked by the corresponding posterior probabilities. (Sproat et al., 2001) enhanced this framework by calculating the likelihood probability as the chance of a noisy token and its associated tag being generated by a specific word.

With the rapid growth of SMS and social media content, text normalization system has drawn increasing attention in the recent decade, where the focus is on converting the noisy nonstandard tokens in the informal text into standard dictionary words. (Choudhury et al., 2007) modeled each standard English word as a hidden Markov model (HMM) and calculated the probability of observing the noisy-token under each of the HMM models; (Cook and Stevenson, 2009) calculated the sum of the probabilities of a noisy token being generated by a specific word and a word formation process; (Beaufort et al., 2010) employed the weighted finite-state machines (FSMs) and rewriting rules for normalizing French SMS; (Pennell and Liu, 2010) focused on tweets created by handsets and developed a CRF tagger for deletion-based abbreviation. The text normalization problem was also tackled under the machine transla-

[1]For this reason, we will use the term "nonstandard tokens" instead of "ill-formed tokens" throughout the paper.

[2]We use the form (standard word, "nonstandard token") to denote an example nonstandard token and its corresponding standard word.

tion (MT) or speech recognition (ASR) framework. (Aw et al., 2006) adapted a phrase-based MT model for normalizing SMS and achieved satisfying performance. (Kobus et al., 2008) showed that using a statistical MT system in combination with an analogy of the ASR system improved performance in French SMS normalization. (Pennell and Liu, 2011) proposed a two-phase character-level MT system for expanding the abbreviations into standard text.

Recent work also focuses on normalizing the Twitter messages, which is generally considered a more challenging task. (Han and Baldwin, 2011) developed classifiers for detecting the ill-formed words and generated corrections based on the morpho-phonemic similarity. (Liu et al., 2011b) proposed to normalize the nonstandard tokens without explicitly categorizing them. (Xue et al., 2011) adopted the noisy-channel framework and incorporated orthographic, phonetic, contextual, and acronym expansion factors in calculating the likelihood probabilities. (Gouws et al., 2011) revealed that different populations exhibit different shortening styles.

Most of the above systems limit their processing scope to certain categories (e.g., deletion-based abbreviations, misspellings) or require large-scale human annotated corpus for training, which greatly hinders the scalability of the system. In this paper, we propose a novel cognitively-driven text normalization system that robustly tackle both the unintentional misspellings and the intentionally-created noisy tokens. We propose a global context-based approach to purify the automatically collected training data and learn the letter transformation patterns without human supervision. We also propose a cognitively-grounded "visual priming" approach that leverages the "priming" effect to suggest the candidate words. By integrating different perspectives, our system can successfully mimic the human rationales and yield broad word-coverage on both SMS and Twitter messages. To the best of our knowledge, we are the first to integrate these human perspectives in the text normalization system.

3 Broad-Coverage Normalization System

In this section, we describe our broad-coverage normalization system, which consists of four key components. For a standard/nonstandard token, three subnormalizers each suggest their most confident candidates from a different perspective[3]: "Enhanced Letter Transformation" automatically learns a set of letter transformation patterns and is most effective in normalizing the intentionally created nonstandard tokens through letter insertion, repetition, deletion, and substitution (Section 3.1); "Visual Priming" proposes candidates based on the visual cues and a *primed* perspective (Section 3.2); "Spell Checker" corrects the misspellings (Section 3.3). The fourth component, "Candidate Combination" introduces various strategies to combine the candidates with or without the local context (Section 3.4). Note that it is crucial to integrate different human perspectives so that the system is flexible in processing both unintentional misspellings and various intentionally-created noisy tokens.

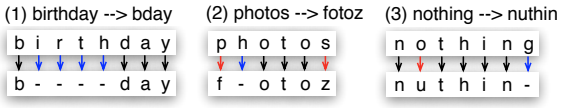

Figure 1: Examples of nonstandard tokens generated by performing letter transformation on the dictionary words.

3.1 Enhanced Letter Transformation

Given a noisy token t_i seen in the text, the letter transformation subnormalizer produces a list of correction candidates s_i under the noisy channel model:

$$\hat{s} = \arg\max_{s_i} p(s_i|t_i) = \arg\max_{s_i} p(t_i|s_i)p(s_i)$$

where we assume each nonstandard 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. $p(s_i)$ can be calculated as the unigram count from a background corpus. We formulate the process of generating a nonstandard token t_i from the 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.

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[4]. This transformation process from dictio-

[3]For the dictionary word, we allow the subnormalizers to either return the word itself or candidates that are the possibly intended words in the given context (e.g., (with, "wit")).

[4]The set of letter combinations used in this work are {ah, ai, aw, ay, ck, ea, ey, ie, ou, te, wh}

nary words to nonstandard tokens will be learned by a character-level sequence labeling system using the automatically collected (word, token) pairs. Next, we create a large lookup table by applying the character-level labeling system to the standard dictionary words and generate multiple variations for each word using the n-best labeling output, the labeling confidence is used as $p(t_i|s_i)$. During testing, we search this lookup table to find the best candidate words for the nonstandard tokens. For 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:

$$p(t_i|s_i) = \max_{\tilde{t}_i \in C_i} p(\tilde{t}_i|s_i)$$

Different from the work in (Liu et al., 2011b), we enhanced the letter transformation process with two novel aspects: first, we devise a set of phoneme-, syllable-, morpheme- and word-boundary based features that effectively characterize the formation process of the nonstandard tokens; second, we propose a global context-aware approach to purify the automatically collected training (word, token) pairs, resulting system yielded similar performance but with only one ninth of the original data. We name this subnormalizer "Enhanced Letter Transformation".

3.1.1 Context-Aware Training Pair Selection

Manual annotation of the noisy nonstandard tokens takes a lot of time and effort. (Liu et al., 2011b) proposed to use Google search engine to automatically collect large amount of training pairs. Yet the resulting (work, token) pairs are often noisy, containing pairs such as (events, "ents"), (downtown, "downto"), etc. The ideal training data should consist of the most frequent nonstandard tokens paired with the corresponding corrections, so that the system can learn from the most representative letter transformation patterns.

Motivated by research on word sense disambiguation (WSD) (Mihalcea, 2007), we hypothesize the nonstandard token and the standard word share a lot of common terms in their global context. For example, "luv" and "love" share "i", "you", "u", "it", etc. among their top context words. Based on this finding, we propose to filter out the low-quality training pairs by evaluating the global contextual similarity between the word and token. To the best of our knowledge, we are the first to explore this global contextual similarity for the text normalization task.

Given a noisy (word, token) pair, we construct two context vectors v_i and v_j by collecting the most frequent terms appearing before or after the work/token. We consider two terms on each side of the word/token as context and restrict the vector length to the top 100 terms. The frequency information were calculated using a large background corpus; stopwords were not excluded from the context vector. The contextual similarity of the (word, token) pair is defined as the cosine similarity between the context vectors v_i and v_j:

$$ContextSim(v_i, v_j) = \frac{\sum_{k=1}^{n} w_{i,k} \times w_{j,k}}{\sqrt{\sum_{k=1}^{n} w_{i,k}^2} \times \sqrt{\sum_{k=1}^{n} w_{j,k}^2}}$$

where $w_{i,k}$ is the weight of term t_k within the context of term t_i. The term weights are defined using a normalized TF-IDF method:

$$w_{i,k} = \frac{TF_{i,k}}{TF_i} \times \log(\frac{N}{DF_k})$$

where $TF_{i,k}$ is the count of term t_k appearing within the context of term t_i; TF_i is the total count of t_i in the corpus. $\frac{TF_{i,k}}{TF_i}$ is therefore the relative frequency of t_k appearing in the context of t_i; $\log(\frac{N}{DF_k})$ denotes the inverse document frequency of t_k, calculated as the logarithm of total tweets (N) divided by the number of tweets containing t_k.

To select the most representative (word, token) pairs for training, we rank the automatically collected 46,288 pairs by the token frequency, filter out pairs whose contextual similarity lower than a threshold θ (set empirically at 0.0003), and retain only the top portion (5,000 pairs) for experiments.

3.1.2 Character-level Sequence Labeling

For a dictionary word s_i, we use the conditional random fields (CRF) model to perform character-level labeling to generate its variant t_i. In the training stage, we align the collected (word, token) pairs at the character level (Liu et al., 2011b), then construct a feature vector for each letter of the dictionary word, using its mapped character as the reference label. This aligned data set is used to train a CRF model (Lafferty et al., 2001; Kudo, 2005)

Character	a	d	v	e	r	t	i	s	e	m	e	n	t	s
Phoneme	AE	D	V	ER	ER	T	AY	Z	_	M	AH	N	T	S
Phoneme boundary	O	O	O	B1	L1	O	O	O	O	O	O	O	O	O
Syllable boundary	B	L	B	I	L	B	I	I	L	B	I	I	I	L
Morpheme boundary	B	I	I	I	I	I	I	I	L	B	I	I	L	U
Word boundary	B	I	I	I	I	I	I	I	I	I	I	I	I	L

Table 2: Example boundary tags for word "**advertisements**" on the phoneme-, syllable-, morpheme-, and word-level, labeled with the "BILOU" encoding scheme.

with L-BFGS optimization. We use the character/phoneme n-gram and binary vowel features as in (Liu et al., 2011b), but develop a set of boundary features to effectively characterize the letter transformation process.

We notice that in creating the nonstandard tokens, humans tend to drop certain letter units from the word or replace them with other letters. For example, in abbreviating "advertisements" to "ads", humans may first break the word into smaller units "ad-ver-tise-ment-s", then drop the middle parts. This also conforms with the word construction theory where a word is composed of smaller units and construction rules. Based on this assumption, we decompose the dictionary words on the phoneme-, syllable-, morpheme-, and word-level[5] and use the "BILOU" tagging scheme (Ratinov and Roth, 2009) to represent the unit boundary, where "BILOU" stands for B(egin), I(nside), L(ast), O(utside), and U(nit-length) of the corresponding unit[6]. Example "BILOU" boundary tags were shown in Table 2.

On top of the boundary tags, we develop a set of conjunction features to accurately pinpoint the current character position. We consider conjunction features formed by concatenating character position in syllable and current syllable position in the word (e.g., conjunction feature "L_B" for the letter "d" in Table 2). A similar set of features are also developed on morpheme level. We consider conjunction of character/vowel feature and their boundary tags on the syllable/morpheme/word level; conjunction of phoneme and phoneme boundary tags, and absolute position of current character within the corresponding syllable/morpheme/word.

We use the aforementioned features to train the CRF model, then apply the model on dictionary words s_i to generate multiple variations t_i for each word. When a nonstandard token is seen during testing, we apply the noisy channel to generate a list of best candidate words: $\hat{s} = \arg\max_{s_i} p(t_i|s_i)p(s_i)$.

3.2 Visual Priming Approach

A second key component of the broad-coverage normalization system is a novel "Visual Priming" subnormalizer. It is built on a cognitively-driven "priming" effect, which has not been explored by other studies yet was shown to be effective across all our data sets.

"Priming"[7] is an implicit memory effect caused by spreading neural networks (Tulving and Stark, 1982). As an example, in the word-stem completion task, participants are given a list of study words, and then asked to complete word "stems" consisting of first 3 letters. A priming effect is observed when participants complete stems with words on the study list more often than with the novel words. The study list activates parts of the human brain right before the stem completion task, later when a word stem is seen, less additional activation is needed for one to choose a word from the study list.

We argue that the "priming" effect may play an important role in human comprehension of the noisy tokens. A person familiarized with the "social talk" is highly *primed* with the most commonly used words; later when a nonstandard token shows only minor visual cues or visual stimulus, it can still be quickly recognized by the person. In this process, the first letter or first few letters of the word serve as a very important visual stimulus. Based on this assumption, we introduce the "priming" subnormalizer based only on the word frequency and the minor visual stimulus. Concretely, this approach proposes candidate words based on the following equation:

$$VisualPrim(s_i|t_i) = \frac{len(LCS(t_i, s_i))}{len(t_i)} \times \log(TF(s_i))$$

Where $TF(s_i)$ is the term frequency of s_i as in the background social text corpus; $\log(TF(s_i))$ *primes* the system with the most common words in the social text; $LCS(\cdot)$ means the longest common character subsequence; $len(\cdot)$ denotes the length of the

[5]Phoneme decomposition is generated using the (Jiampojamarn et al., 2007) algorithm to map up to two letters to phonemes (2-to-2 alignment); syllable boundary acquired by the hyphenation algorithm (Liang, 1983); morpheme boundary determined by toolkit Morfessor 1.0 (Creutz and Lagus, 2005).

[6]For phoneme boundary, we use "B1" and "L1" to represent two different characters aligned to one phoneme and "B2", "L2" represent same characters aligned to one phoneme.

[7]http://en.wikipedia.org/wiki/Priming_(psychology)

character sequence. Together $\frac{len(LCS(t_i,s_i))}{len(t_i)}$ provides the minor visual stimulus from t_i. Note that the first character has been shown to be a crucial visual cue for the brain to understand jumbled words (Davis,), we therefore consider as candidates only those words s_i that start with the same character as t_i. In the case that the nonstandard token t_i starts with a digit (e.g., "2moro"), we use the mostly likely corresponding letter to search the candidates (those starting with letter "t"). This setting also effectively reduces the candidate search space.

The "visual priming" subnormalizer promotes the candidate words that are frequently used in the social talk and also bear visual similarity with the given noisy token. It slightly deviates from the traditional "priming" notion in that the frequency information were acquired from the global corpus rather than from the prior context. This approach also inherently follows the noisy channel framework, with $p(t_i|s_i)$ represents the visual stimulus and $p(s_i)$ being the logarithm of frequency. The candidate words are ranked by $\hat{s} = \arg\max_{s_i} VisualPrim(s_i|t_i)$. We show that the "priming" subnormalizer is robust across data sets abide its simplistic representation.

3.3 Spell Checker

The third subnormalizer is the spell checker, which combines the string and phonetic similarity algorithms and is most effective in normalizing the misspellings. We use the Jazzy spell checker (Idzelis, 2005) 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.

3.4 Candidate Combination

Each of the three subnormalizers is a stand-alone system and can suggest corrections for the nonstandard tokens. Yet we show that each subnormalizer mimics a different perspective that humans use to decode the nonstandard tokens, as a result, our broad-coverage normalization system is built by integrating candidates from the three subnormalizers using various strategies.

For a noisy token seen in the informal text, the most convenient way of system combination is to harvest up to n candidates from each of the subnormalizers, and use the pool of candidates (up to $3n$) as the system output. This sets an upper bound for other candidate combination strategies, and we name this approach "**Oracle**".

A second combination strategy is to give higher priority to candidates from high-precision subsystems. Both "Letter Transformation" and "Spell Checker" have been shown to have high precision in suggesting corrections (Liu et al., 2011b), while "Visual Priming" may not yield high precision due to its definition. We therefore take the top-3 candidates from each of the "Letter Tran." and "Spell Checker" subsystems, but put candidates from "Letter Tran." ahead of "Spell Checker" if the confidence of the best candidate is greater than a threshold λ and vice versa. The list of candidates is then compensated using the "Visual Priming" output until the total number reaches n. We name this approach "**Word-level**" combination since no message-level context information is involved.

Based on the "Word-level" combination output, we can further rerank all the candidates using a message-level Viterbi decoding process (Pennell and Liu, 2011) where the local context information is used to select the best candidate. This approach is named "**Message-level**" combination.

4 Experiments

4.1 Experimental Setup

We use four SMS and Twitter data sets to evaluate the system effectiveness. Statistics of these data sets are summarized in Table 3. Data set (1) to (3) are used for word-level evaluation; data set (4) for both word- and message-level evaluation. In Table 3, we also present the number of distinct nonstandard tokens found in each data set, and notice that only a small portion of the nonstandard tokens correspond to multiple standard words. We calculate the dictionary coverage of the manually annotated words since this sets an upper bound for any normalization system. We use the Edinburgh Twitter corpus (Petrovic et al., 2010) as the background corpus for frequency calculation, and a dictionary containing 82,324 words.[8] The nonstandard tokens may consist of both numbers/characters and apostrophe.

[8]The dictionary is created by combining the CMU (CMU, 2007) and Aspell (Atkinson, 2006) dictionaries and dropping words with frequency < 20 in the background corpus. "rt" and all single characters except "a" and "i" are excluded.

Index	Domain	Time Period	#Msgs	#Uniq Nonstan. Tokens	%Nonstan. Tkns w/ Multi-cands	%Dict cov. of cands	Reference
(1)	SMS	Around 2007	n/a	303	1.32%	100%	(Choudhury et al., 2007)
(2)	Twitter	Nov 2009 – Feb 2010	6150	3802	3.87%	99.34%	(Liu et al., 2011)
(3)	SMS/Twitter	Aug 2009	4660	2040	2.41%	96.84%	(Pennell and Liu, 2011)
(4)	Twitter	Aug 2010 – Oct 2010	549	558	2.87%	99.10%	(Han and Baldwin, 2011)

Table 3: Statistics of different SMS and Twitter data sets.

The goal of word-level normalization is to convert the list of distinct nonstandard tokens into standard words. 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. We adopt this word-level n-best accuracy to make our results comparable to other state-of-the-art systems. On message-level, we evaluate the 1-best system output using precision, recall, and f-score, calculated respective to the nonstandard tokens.

4.2 Word-level Results

The word-level results are presented in Table 4, 5, and 6, evaluated on data set (1), (2), (3) respectively. We present the n-best accuracy ($n = 1, 3, 10, 20$) of the system as well as the "Oracle" results generated by pooling the top-20 candidates from each of the three subnormalizers. The best prior results on the data sets are also included in the tables.

We notice that the broad-coverage system outperforms all other systems on the reported data sets. It achieves about 90% word-level accuracy on data set (1) and (2) with the top-10 candidates (an average 10% performance gain compared to (Liu et al., 2011b)). This is of crucial importance to a normalization system, since the high accuracy and limited number of candidates will enable more sophisticated reranking or supervised learning techniques to select the best candidate. We also observe the "Oracle" system has averagely only 5% gap to the dictionary coverage. A detailed analysis shows that the human annotators perform many semantic/grammar corrections as well as inconsistent annotations, e.g., (sleepy, "zzz"), (disliked, "unliked"). These are out of the capabilities of the current text normalization system and partly explains the remaining 5% gap.

Regarding the subnormalizer performance, the spell checker yields only 50% to 60% accuracy on all data sets, indicating that the vast amount of the intentionally created nonstandard tokens can hardly be tackled by a system relies solely on the lexical/phonetic similarity. The "Visual Priming" subnormalizer performs surprisingly well and shows robust performance across all data sets. A minor side-effect is that the candidates were restricted to have the same first letter with the noisy token, this sets the upper bound of the approach to 89.77%, 92.45%, and 93.51%, respectively on data set (1), (2), and (3). Compared to other subnormalizers, the "Enhanced Letter Tran." is effective at normalizing intentionally created tokens and has better precision regarding its top candidate ($n = 1$). We demonstrate the context-aware training pair selection results in Figure 2, by plotting the learning curve using different amounts of training data, ranging from 1,000 (word, token) pairs to the total 46,288 pairs. We notice that the system can effectively learn the letter transformation patterns from a small number of high quality training pairs. The final system was trained using the top 5,000 pairs and the lookup table was created by generating 50 variations for each dictionary word.

SMS Dataset (303 pairs)	Word Level Accuracy (%)				
	1-best	3-best	10-best	20-best	Oracle
Jazzy Spell Checker	43.89	55.45	56.77	56.77	n/a
Visual Priming	54.13	74.92	84.82	87.13	n/a
Enhanced Letter Tran.	61.06	74.92	80.86	82.51	n/a
Broad-Cov. System	**64.36**	**80.20**	**89.77**	**91.75**	**94.06**
(Pennell et al., 2011)⋆	60.39	74.58	75.57	75.57	n/a
(Liu et al., 2011)	62.05	75.91	81.19	81.19	n/a
(Cook et al., 2009)	59.4	n/a	83.8	87.8	n/a
(Choudhury et al., 2007)⋆	59.9	n/a	84.3	88.7	n/a

Table 4: Word-level results on data set (1). ⋆ denotes system requires human annotations for training.

Twitter Dataset (3802 pairs)	Word Level Accuracy (%)				
	1-best	3-best	10-best	20-best	Oracle
Jazzy Spell Checker	47.19	56.92	59.13	59.18	n/a
Visual Priming	54.34	70.59	80.83	84.74	n/a
Enhanced Letter Tran.	61.05	70.07	74.04	74.75	n/a
Broad-Cov. System	**69.81**	**82.51**	**92.24**	**93.79**	**95.71**
(Liu et al., 2011)	68.88	78.27	80.93	81.17	n/a

Table 5: Word-level results on data set (2).

4.3 Message-level Results

The goal of message-level normalization is to replace each occurrence of the nonstandard token with the candidate word that best fits the local context.

SMS/Twitter Dataset (2404 pairs)	Word Level Accuracy (%)				
	1-best	3-best	10-best	20-best	Oracle
Jazzy Spell Checker	39.89	46.51	48.54	48.67	n/a
Visual Priming	54.12	68.59	78.83	83.11	n/a
Enhanced Letter Tran.	57.65	67.18	71.01	71.88	n/a
Broad-Cov. System	**64.39**	**78.29**	**86.56**	**88.69**	**91.60**
(Pennell et al., 2011)⋆	37.40	n/a	n/a	72.38	n/a

Table 6: Word-level results on data set (3). ⋆ denotes system requires human annotations for training.

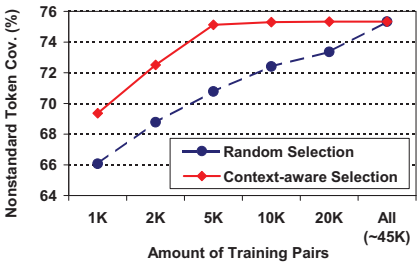

Figure 2: Learning curve of the enhanced letter transformation system using random training pair selection or the context-aware approach. Evaluated on data set (2).

We use the word-level "Broad-Cov. System" for candidate suggestion and the Viterbi algorithm for message-level decoding. The system is evaluated on data set (4) and results shown in Table 7. Following research in (Han and Baldwin, 2011), we focus on the the normalization task and assume perfect non-standard token detection.

The "Word-level w/o Context" results are generated by replacing each nonstandard token using the 1-best word-level candidate. Although the replacement process is static, it results in 70.97% f-score due to the high performance of the word-level system. We explore two language models (LM) for the Viterbi decoding process. First, a bigram LM is trained using the Edinburgh Twitter corpus (53,794,549 English tweets) with the SRILM toolkit (Stolcke, 2002) and Kneser-Ney smoothing; second, we retrieve the bigram probabilities from the Microsoft Web N-gram API (Wang et al., 2010) since this represents a more comprehensive web-based corpus. During decoding, we use the "VisualPrim" score as the emission probability, since this score best fits the log scale and applies to all candidates. For the Twitter LM, we apply a scaling factor of 0.5 to the "VisualPrim" score to make it comparable in scale to the LM probabilities. We use the 3-best word-level candidates for Viterbi decoding. In addition, we add the commonly used corrections for

Twitter Dataset (549 Tweets)		Message-level P/R/F		
		Precision (%)	Recall (%)	F-score (%)
Word-level w/o Context		75.69	66.81	70.97
w/ Context	Web LM	**79.12**	**77.11**	**78.10**
	Twitter LM	**84.13**	**78.38**	**81.15**
(Han and Baldwin, 2011)⋆		75.30	75.30	75.30

Table 7: Message-level results on data set (4). ⋆ denotes system requires human annotations for training.

16 single-characters, e.g., for "r", "c", we add "are", "see" to the candidate list if they are not already presented. A default "VisualPrim" score ($\eta = 25$) is used for these candidates. As seen from Table 7, both Web LM and Twitter LM achieve better performance than the best prior results, with Twitter LM outperforms the Web LM, yielding a f-score of 81%. This shows that a vanilla Viterbi decoding process is able to outperform the fine-tuned supervised system given competitive word-level candidates. In future, we will investigate other comprehensive message-level candidate reranking process.

5 Conclusion

In this paper, we propose a broad-coverage normalization system for the social media language without using the human annotations. It integrates three key components: the enhanced letter transformation, visual priming, and string/phonetic similarity. The system was evaluated on both word- and message-level using four SMS and Twitter data sets. We show that our system achieves over 90% word-coverage across all data sets and the broad word-coverage can be successfully translated into message-level performance gain. We observe that the social media is an emotion-rich language, therefore future normalization system will need to address various sentiment-related expressions, such as emoticons (":d", "X-8"), interjections ("bwahaha", "brrrr"), acronyms ("lol", "lmao"), etc., whether and how these expressions should be normalized is an unaddressed issue and worths future investigation.

Acknowledgments

We thank the three anonymous reviewers for their insightful comments and valuable input. We thank Prof. Yang Liu, Deana Pennell, Bo Han, and Prof. Tim Baldwin for sharing the annotated data and the useful discussions. Part of this work was done while Xiao Jiang was a research intern in Bosch Research.

References

Kevin Atkinson. 2006. Gnu aspell. *http://aspell.net/*.

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for sms text normalization. In *Proceedings of 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 ACL*, pages 770–779.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of ACL*.

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! Using word lengthening to detect sentiment in microblogs. In *Proceedings of EMNLP*, pages 562–570.

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.

Kenneth W. Church and William A. Gale. 1991. Probability scoring for spelling correction. *Statistics and Computing*, 1:93–103.

CMU. 2007. The cmu pronouncing dictionary. *http://www.speech.cs.cmu.edu/cgi-bin/cmudict*.

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.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0. In *Computer and Information Science, Report A81, Helsinki University of Technology*.

David Crystal. 2009. Txtng: The gr8 db8. *Oxford University Press*.

Matt Davis. Reading jumbled texts. *http://www.mrc-cbu.cam.ac.uk/personal/matt.davis/Cmabrigde/*.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. #hardtoparse: POS tagging and parsing the twitterverse. In *Proceedings of the AAAI Workshop on Analyzing Microtext*, pages 20–25.

Stephan Gouws, Donald Metzler, Congxing Cai, and Eduard Hovy. 2011. Contextual bearing on linguistic variation in social media. In *Proceedings of the ACL Workshop on Language in Social Media*, pages 20–29.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of ACL*, pages 368–378.

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 HLT/NAACL*, pages 372–379.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing sms: Are two metaphors better than one? In *Proceedings of 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 ICML*, pages 282–289.

Franklin Mark Liang. 1983. Word hy-phen-a-tion by com-put-er. In *PhD Dissertation, Stanford University*.

Fei Liu, Yang Liu, and Fuliang Weng. 2011a. Why is "sxsw" trending? Exploring multiple text sources for twitter topic summarization. In *Proceedings of the ACL Workshop on Language in Social Media (LSM)*, pages 66–75.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011b. Insertion, deletion, or substitution? Normalizing text messages without pre-categorization nor supervision. In *Proceedings of ACL*, pages 71–76.

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011c. Recognizing named entities in tweets. In *Proceedings of ACL*, pages 359–367.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management: An International Journal*, 27(5):517–522.

Rada Mihalcea. 2007. Using wikipedia for automatic word sense disambiguation. In *Proceedings of NAACL*, pages 196–203.

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

Deana L. Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 974–982.

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.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL*, pages 147–155.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of EMNLP*.

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.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proceedings of ICSLP*, pages 901–904.

L. Venkata Subramaniam, Shourya Roy, Tanveer A. Faruquie, and Sumit Negi. 2009. A survey of types of text noise and techniques to handle noisy text. In *Proceedings of AND*.

Crispin Thurlow. 2003. Generation txt? the sociolinguistics of young people's text-messaging. *Discourse Analysis Online*.

Endel Tulving and Daniel L. Schacter; Heather A. Stark. 1982. Priming effects in word fragment completion are independent of recognition memory. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 8(4).

Twitter. 2011. *http://en.wikipedia.org/wiki/Twitter*.

Kuansan Wang, Christopher Thrasher, Evelyne Viegas, Xiaolong Li, and Bo june (Paul) Hsu. 2010. An overview of microsoft web n-gram corpus and applications. In *Proceedings of NAACL-HLT*, pages 45–48.

Zhenzhen Xue, Dawei Yin, and Brian D. Davison. 2011. Normalizing microtext. In *Proceedings of the AAAI Workshop on Analyzing Microtext*, pages 74–79.