Social Text Normalization using Contextual Graph Random Walks

Hany Hassan
Microsoft Research
Redmond, WA
hanyh@microsoft.com

Arul Menezes
Microsoft Research
Redmond, WA
arulm@microsoft.com

Abstract

We introduce a social media text normalization system that can be deployed as a preprocessing step for Machine Translation and various NLP applications to handle social media text. The proposed system is based on unsupervised learning of the normalization equivalences from unlabeled text. The proposed approach uses Random Walks on a contextual similarity bipartite graph constructed from n-gram sequences on large unlabeled text corpus. We show that the proposed approach has a very high precision of (92.43) and a reasonable recall of (56.4). When used as a preprocessing step for a state-of-the-art machine translation system, the translation quality on social media text improved by 6%. The proposed approach is domain and language independent and can be deployed as a preprocessing step for any NLP application to handle social media text.

1 Introduction

Social Media text is usually very noisy and contains a lot of typos, ad-hoc abbreviations, phonetic substitutions, customized abbreviations and slang language. The social media text is evolving with new entities, words and expressions. Natural language processing and understanding systems such as Machine Translation, Information Extraction and Text-to-Speech are usually trained and optimized for clean data; therefore such systems would face a challenging problem with social media text.

Various social media genres developed distinct characteristics. For example, SMS developed a nature of shortening messages to avoid multiple keystrokes. On the other hand, Facebook and instant messaging developed another genre where more emotional expressions and different abbreviations are very common. Somewhere in between, Twitter's statuses come with some brevity similar to SMS along with the social aspect of Facebook. On the same time, various social media genres share many characteristics and typo styles. For example, repeating letters or punctuation for emphasizing and emotional expression such as *"'goooood morniiing"'*. Using phonetic spelling in a generalized way or to reflect a local accent; such as *"'wuz up bro"' (what is up brother)*. Eliminating vowels such as *"'cm to c my luv"'*. Substituting numbers for letters such as *"'4get"' (forget) , "'2morrow"' (tomorrow), and "'b4"' (before)*. Substituting phonetically similar letters such as *"'phone"' (fon)*. Slang abbreviations which usually abbreviates multi-word expression such as *"'LMS"' (like my status) , "'idk"' (i do not know), "'rofl"' (rolling on floor laughing)*.

While social media genres share many characteristics, they have significant differences as well. It is crucial to have a solution for text normalization that can adapt to such variations automatically. We propose a text normalization approach using an unsupervised method to induce normalization equivalences from noisy data which can adapt to any genre of social media.

In this paper, we focus on providing a solution for social media text normalization as a preprocessing step for NLP applications. However, this is a challenging problem for several reasons. First, it is not straightforward to define the Out-of-Vocabulary (OOV) words. Traditionally, an OOV word is defined as a word that does not exist in the vocabulary of a given system. However, this definition is not adequate for the social media text which has a very dynamic nature. Many words and named entities that do not exist in a given vocabulary should not be considered for normalization. Second, same OOV word may have many

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 1577–1586,
Sofia, Bulgaria, August 4-9 2013.

appropriate normalization depending on the context and on the domain. Third, text normalization as a preprocessing step should have very high precision; in other words, it should provide conservative and confident normalization and not overcorrect. Moreover, the text normalization should have high recall, as well, to have a good impact on the NLP applications.

In this paper, we introduce a social media text normalization system which addresses the challenges mentioned above. The proposed system is based on constructing a lattice from possible normalization candidates and finding the best normalization sequence according to an n-gram language model using a Viterbi decoder. We propose an unsupervised approach to learn the normalization candidates from unlabeled text data. The proposed approach uses Random Walks on a contextual similarity graph constructed form n-gram sequences on large unlabeled text corpus. The proposed approach is very scalable, accurate and adaptive to any domain and language. We evaluate the approach on the normalization task as well as machine translation task.

The rest of this paper is organized as follows: Section(2) discusses the related work, Section(3) introduces the text normalization system and the baseline candidate generators, Section(4) introduces the proposed graph-based lexicon induction approach, Section(5) discusses the experiments and output analysis, and finally Section(6) concludes and discusses future work.

2 Related Work

Early work handled the text normalization problem as a noisy channel model where the normalized words go through a noisy channel to produce the noisy text. (Brill and Moore, 2000) introduced an approach for modeling the spelling errors as a noisy channel model based on string to string edits. Using this model gives significant performance improvements compared to previously proposed models. (Toutanova and Moore, 2002) improved the string to string edits model by modeling pronunciation similarities between words. (Choudhury et al., 2007) introduced a supervised HMM channel model for text normalization which has been expanded by (Cook and Stevenson, 2009) to introduce unsupervised noisy channel model using probabilistic models for common abbreviation and various spelling errors types. Some researchers used Statistical Machine Translation approach for text normalization; formalizing the problem as a translation from the noisy forms to the normalized forms. (Aw et al., 2006) proposed an approach for normalizing Short Messaging Service (SMS) texts by translating it into normalized forms using Phrase-based SMT techniques on character level. The main drawback of these approaches is that the noisy channel model cannot accurately represent the errors types without contextual information.

More recent approaches tried to handle the text normalization problem using normalization lexicons which map the noisy form of the word to a normalized form. For example, (Han et al., 2011) proposed an approach using a classifier to identify the noisy words candidate for normalization; then using some rules to generate lexical variants and a small normalization lexicon. (Gouws et al., 2011) proposed an approach using an impoverished normalization lexicon based on string and distributional similarity along with a dictionary lookup approach to detect noisy words. More recently, (Han et al., 2012) introduced a similar approach by generating a normalization lexicon based on distributional similarity and string similarity. This approach uses pairwise similarity where any two words that share the same context are considered as normalization equivalences. The pairwise approach has a number of limitations. First, it does not take into account the relative frequencies of the normalization equivalences that might share different contexts. Therefore, the selection of the normalization equivalences is performed on pairwise basis only and is not optimized over the whole data. Secondly, the normalization equivalences must appear in the exact same context to be considered as a normalization candidate. These limitations affect the accuracy and the coverage of the produced lexicon.

Our approach also adopts a lexicon based approach for text normalization, we construct a lattice from possible normalization candidates and find the best normalization sequence according to an n-gram language model using a Viterbi decoder. The normalization lexicon is acquired from unlabeled data using random walks on a contextual similarity graph constructed form n-gram sequences on large unlabeled text corpus. Our approach has some similarities with (Han et al., 2012) since both approaches utilize a normaliza-

tion lexicon acquired form unlabeled data using distributional and string similarities. However, our approach is significantly different since we acquire the lexicon using random walks on a contextual similarity graph which has a number of advantages over the pairwise similarity approach used in (Han et al., 2012). Namely, the acquired normalization equivalence are optimized globally over the whole data, the rare equivalences are not considered as good candidates unless there is a strong statistical evidence across the data, and finally the normalization equivalences may not share the same context. Those are clear advantages over the pairwise similarity approach and result in a lexicon with higher accuracy as well as wider coverage. Those advantages will be clearer when we describe the proposed approach in details and during evaluation and comparison to the pairwise approach.

3 Text Normalization System

In this paper, we handle text normalization as a lattice scoring approach, where the translation is performed from noisy text as the source side to the normalized text as the target side. Unlike conventional MT systems, the translation table is not learned from parallel aligned data; instead it is modeled by the graph-based approach of lexicon generation as we will describe later. We construct a lattice from possible normalization candidates and find the best normalization sequence according to an n-gram language model using a Viterbi decoder.

In this paper, we restrict the normalization lexicon to one-to-one word mappings, we do not consider multi words mapping for the lexicon induction. To identify OOV candidates for normalization; we restrict proposing normalization candidates to the words that we have in our induced normalization lexicon only. This way, the system would provide more confident and conservative normalization. We move the problem of identifying OOV words to training time; at training time we use soft criteria to identify OOV words.

3.1 Baseline Normalization Candidates Generation

We experimented with two normalization candidate generators as baseline systems. The first is a dictionary based spelling correction similar to Aspell[1]. In this experiment we used the spell checker to generate all possible candidates for OOV words and then applied the Viterbi decoder on the constructed lattice to score the best correction candidates using a language model.

Our second candidates generator is based on a trie approximate string matching with K errors similar to the approach proposed in (Chang et al., 2010), where K errors can be caused by substitution, insertion, or deletion operations. In our implementation, we customized the errors operations to accommodate the nature of the social media text. Such as lengthening, letter substitution, letter-number substitution and phonetic substitution. This approach overcomes the main problem of the dictionary-based approach which is providing inappropriate normalization candidates to the errors styles in the social media text.

As we will show in the experiments in Section(5), dictionary-based normalization methods proved to be inadequate for social media domain normalization for many reasons. First, they provide generic corrections which are inappropriate for social media text. Second, they usually provide corrections with the minimal edit distance for any word or named entity regardless of the nature of the words. Finally, the previous approaches do not take into account the dynamics of the social media text where new terms can be introduced on a daily basis.

4 Normalization Lexicons using Graph-based Random Walks

4.1 Bipartite Graph Representation

The main motivation of this approach is that normalization equivalences share similar context; which we call contextual similarity. For instance, assume 5-gram sequences of words, two words may be normalization equivalences if their n-gram context shares the same two words on the left and the same two words on the right. In other words, they are sharing a wild card pattern such as *(word* $_1$ *word* $_2$ * *word* $_4$ *word* $_5$*)*.

This contextual similarity can be represented as a bipartite graph with the first partite representing the words and the second partite representing the n-gram contexts that may be shared by words. A word node can be either normalized word or noisy word. Identifying if a word is normalized or noisy (candidate for normalization) is crucial since this decision limits the candidate noisy words to be normalized. We adopted a soft criteria for iden-

[1] http://aspell.net/

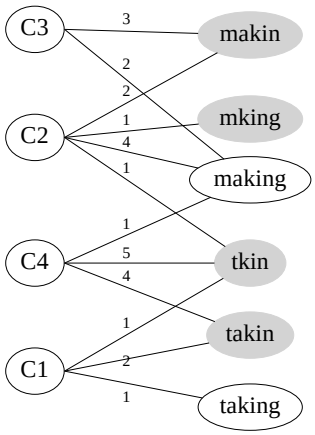

Figure 1: Bipartite Graph Representation, left nodes represent contexts, gray right nodes represent the noisy words and white right nodes represent the normalized words. Edge weight is the co-occurrence count of a word and its context.

tifying noisy words. A vocabulary is constructed from a large clean corpus. Any word that does not appear in this vocabulary more than a predefined threshold (i.e. 10 times) is considered as a candidate for normalization (noisy word). Figure(1) shows a sample of the bipartite graph $G(W, C, E)$, where noisy words are shown as gray nodes.

Algorithm 4.1: CONSTRUCTBIPARTITE($text$)

comment: Construct Bipartite Graph
output $(G(W, C, E))$
comment: Extract all *n-gram* sequences
$Ngrams \leftarrow$ EXTRACTNGRAMS($TextCorpus$)
for each $n \in Ngrams$
do
  comment: Check for center word
  if ISNOISY($CenterWord$)
  $W \leftarrow$ ADDSOURCENODE($CenterWord$)
  else
  $W \leftarrow$ ADDABSORBINGNODE($CenterWord$)
  comment: add the context pattern
  $C \leftarrow$ ADD($Context$)
  comment: edge weight
  $E \leftarrow$ ADD($Context, Word, count$)

The bipartite graph, $G(W, C, E)$, is composed of W which includes all nodes representing normalized words and noisy words, C which includes all nodes representing shared context, and finally E which represents the edges of the graph connecting word nodes and context nodes. The weight on the edge is simply the number of occurrences of a given word in a context. While constructing the graph, we identify if a node represents a noisy word (N) (called source node) or a normalized word (M) (called absorbing node). The bipartite graph is constructed using the procedure in Algorithm(4.1).

4.2 Lexicon generation using Random Walks

Our proposed approach uses Markov Random Walks on the bipartite graph in Figure(1) as defined in (Norris, 1997). The main objective is to identify pairs of noisy and normalized words that can be considered as normalization equivalences. In principal, this is similar to using random walks for semi-supervised label propagation which has been introduced in (Szummer and Jaakkola, 2002) and then used in many other applications. For example, (Hughes and Ramage, 2007) used random walks on Wordnet graph to measure lexical semantic relatedness between words. (Das and Petrov, 2011) used graph-based label propagation for cross-lingual knowledge transfers to induce POS tags between two languages. (Minkov and Cohen, 2012) introduced a path constrained graph walk algorithm given a small number of labeled examples to assess nodes relatedness in the graph. In this paper, we apply the label propagation approach to the text normalization problem.

Consider a random walk on the bipartite graph $G(W, C, E)$ starting at a noisy word (source node) and ending at a normalized word (absorbing node). The walker starts from any source node N_i belonging to the noisy words then move to any other connected node M_j with probability P_{ij}. The transition between each pair of nodes is defined by a transition probability P_{ij} which represents the normalized probability of the co-occurrence counts of the word and the corresponding context. Though the counts are symmetric, the probability is not symmetric. This is due to the probability normalization which is done according to the nodes connectivity. Therefore, the transition probability between any two nodes i, j is defined as:

$$P_{ij} = W_{ij} / \sum_{\forall k} W_{ik} \quad (1)$$

For any non-connected pair of nodes, $P_{ij} = 0$. It is worth noting that due to the bipartite graph representation; any word node, either noisy (source) or normalized (absorbing), is only connected to context nodes and not directly connected to any other word node.

The algorithm repeats independent random walks for *K* times where the walks traverse the graph randomly according to the transition probability distribution in Eqn(1); each walk starts from the source noisy node and ends at an absorbing normalized node, or consumes the maximum number of steps without hitting an absorbing node.

For any random walk the number of steps taken to traverse between any two nodes is called the hitting time (Norris, 1997). Therefore, the hitting time between a noisy and a normalized pair of nodes (n, m) with a walk r is $h_r(n, m)$. We define the cost between the two nodes as the average hitting time $H(n, m)$ of all walks that connect those two nodes:

$$H(n,m) = \sum_{\forall r} h_r(n,m)/R \quad (2)$$

Consider the bipartite graph in Figure(1), assume a random walk starting at the source node representing the noisy word *"tkin"* then moves to the context node *C1* then to the absorbing node representing the normalized word *"taking"*. This random walk will associate *"tkin"* with *"taking"* with a walk of two steps (hits). Another random walk that can connect the two words is [*"tkin"* $\rightarrow$ *C4* $\rightarrow$ *"takin"* $\rightarrow$ *C1* $\rightarrow$ *"taking"*], which has 4 steps (hits). In this case, the cost of this pair of nodes is the average number of hits connecting them which is 3.

It is worth noting that the random walks are selected according to the transition probability in Eqn(1); therefore, the more probable paths will be picked more frequently. The same pair of nodes can be connected with many walks of various steps (hits), and the same noisy word can be connected to many other normalized words.

We define the contextual similarity probability of a normalization equivalence pair n, m as $L(n, m)$. Which is the relative frequency of the average hitting of those two nodes, $H(n, m)$, and all other normalized nodes linked to that noisy word. Thus $L(n, m)$, is calculated as:

$$L(n,m) = H(n,m) / \sum_i H(n,m_i) \quad (3)$$

Furthermore, we add another similarity cost between a noisy word and a normalized word based on the lexical similarity cost, $SimCost(n, m)$, which we will describe in the next section. The final cost associated with a pair is:

$$Cost(n,m) = \lambda_1 L(n,m) + \lambda_2 SimCost(n,m) \quad (4)$$

Algorithm 4.2: INDUCELEXICON(G)

output $(Lexicon)$
INIT$((Lexicon))$
for each $n \in W \in G(W, C, E)$
do
  comment: for noisy nodes only
  if ISNOISY(n)
    INIT(Rn)
    comment: do K random walks
    for $i \leftarrow 0$ **to** K
    do
    $Rn \leftarrow$ RANDOMWALK(n)
    comment: Calculate Avg. hits and normalize
    $Ln \leftarrow$ NORMALIZE(Rn)
  comment: Calculate Lexical Sim Cost
  $Ln \leftarrow$ SIMCOST(Ln)
  $Ln \leftarrow$ PRUNE(Ln)
  $Lexicon \leftarrow$ ADD(Ln)

We used uniform interpolation, both λ_1 and λ_2 equals 1. The final Lexicon is constructed using those entries and if needed we prune the list to take top N according to the cost above. The algorithm is outlined in 4.2.

4.3 Lexical Similarity Cost

We use a similarity function proposed in (Contractor et al., 2010) which is based on Longest Common Subsequence Ratio (LCSR) (Melamed, 1999). This cost function is defined as the ratio of LCSR and Edit distance between two strings as follows:

$$SimCost(n,m) = LCSR(n,m)/ED(n,m) \quad (5)$$

$$LCSR(n,m) = LCS(n,m)/MaxLenght(n,m) \quad (6)$$

We have modified the Edit Distance calculation *ED(n,m)* to be more adequate for social media text. The edit distance is calculated between the consonant skeleton of the two words; by removing all vowels, we used Editex edit distance as proposed in (Zobel and Philip, 1996), repetition is reduced to a single letter before calculating the edit distance, and numbers in the middle of words are substituted by their equivalent letters.

5 Experiments

5.1 Training and Evaluation Data

We collected large amount of social media data to generate the normalization lexicon using the ran-

dom walk approach. The data consists of 73 million Twitter statuses. All tweets were collected from March/April 2012 using the Twitter Streaming APIs[2]. We augmented this data with 50 million sentences of clean data from English LDC Gigaword corpus [3]. We combined both data, noisy and clean, together to induce the normalization dictionary from them. While the Gigaword clean data was used to train the language model to score the normalized lattice.

We constructed a test set of 1000 sentences of social media which had been corrected by a native human annotator, the main guidelines were to normalize noisy words to its corresponding clean words in a consistent way according to the evidences in the context. We will refer to this test set as SM-Test. Furthermore, we developed a test set for evaluating the effect of the normalization system when used as a preprocessing step for Machine translation. The machine translation test set is composed of 500 sentences of social media English text translated to normalized Spanish text by a bi-lingual translator.

5.2 Evaluating Normalization Lexicon Generation

We extracted 5-gram sequences from the combined noisy and clean data; then we limited the space of noisy 5-gram sequences to those which contain only one noisy word as the center word and all other words, representing the context, are not noisy. As we mentioned before, we identify whether the word is noisy or not by looking up a vocabulary list constructed from clean data. In these experiments, the vocabulary is constructed from the Language Model data (50M sentences of the English Gigaword corpus). Any word that appears less than 10 times in this vocabulary is considered noisy and candidate for normalization during the lexicon induction process. It is worth noting that our notion of noisy word does not mean it is an OOV that has to be corrected; instead it indicates that it is candidate for correction but may be opted not to be normalized if there is no confident normalization for it. This helps to maintain the approach as a high precision text normalization system which is highly preferable as an NLP preprocessing step.

We constructed a lattice using normalization candidates and score the best Viterbi path with 5-gram language model. We experimented with two candidate generators as baseline systems, namely the dictionary-based spelling correction and the trie approximate match with K errors; where K=*3*. For both candidate generators the cost function for a given candidate is calculated using the lexical similarity cost in Eqn(5). We compared those approaches with our newly proposed unsupervised normalization lexicon induction; for this case the cost for a candidate is the combined cost of the contextual similarity probability and the lexical similarity cost as defined in Eqn(4). We examine the effect of data size and the steps of the random walks on the accuracy and the coverage of the induced dictionary.

We constructed the bipartite graph with the n-gram sequences as described in Algorithm 4.1. Then the Random Walks Algorithm in 4.2 is applied with 100 walks. The total number of word nodes is about 7M nodes and the total number of context nodes is about 480M nodes. We used MapReduce framework to implement the proposed technique to handle such large graph. We experimented with the maximum number of random walk steps of 2, 4 and 6; and with different portions of the data as well. Finally, we pruned the lexicon to keep the top 5 candidates per noisy word.

Table(1) shows the resulting lexicons from different experiments.

Lexicon	Lexicon	Data	Steps
Lex1	123K	20M	4
Lex2	281K	73 M	2
Lex3	327K	73M	4
Lex4	363K	73M	6

Table 1: Generated Lexicons, steps are the Random Walks maximum steps.

As shown in Table(1), we experimented with different data sizes and steps of the random walks. The more data we have the larger the lexicon we get. Also larger steps increase the induced lexicon size. A random walk step size of 2 means that the noisy/normalized pair shares the same context; while a step size of 4 or more means that they may not share the same context. Next, we will examine the effect of lexicon size on the normalization task.

[2]https://dev.twitter.com/docs/streaming-apis

[3]http://www.ldc.upenn.edu/Catalog/LDC2011T07

5.3 Text Normalization Evaluation

We experimented different candidate generators and compared it to the unsupervised lexicon approach. Table(2) shows the precision and recall on a the SM-Test set.

System	Candidates	Precision	Recall	F-Measure
Base1	Dict	33.9	15.1	20.98
Base2	Trie	26.64	27.65	27.13
RW1	Lex1	88.76	59.23	71.06
RW2	Lex2	90.66	54.06	67.73
RW3	**Lex3**	**92.43**	**56.4**	**70.05**
RW4	Lex4	90.87	60.73	72.8

Table 2: Text Normalization with different lexicons

In Table(2), the first baseline is using a dictionary based spell checker; which gets low precision and very low recall. Similarly the trie approximate string match is doing a similar job with better recall though the precision is worst. Both of the baseline approaches are inadequate for social media text since both will try to correct any word that is similar to a word in the dictionary. The Trie approximate match is doing better job on the recall since the approximate match is based on phonetic and lexical similarities.

On the other hand, the induced normalization lexicon approach is doing much better even with a small amount of data as we can see with system RW1 which uses Lex1 generated from 20M sentences and has 123K lexicon entry. Increasing the amount of training data does impact the performance positively especially the recall. On the other hand, increasing the number of steps has a good impact on the recall as well; but with a considerable impact on the precision. It is clear that increasing the amount of data and keeping the steps limit at "'4"' gives better precision and coverage as well. This is a preferred setting since the main objective of this approach is to have better precision to serve as a reliable preprocessing step for Machine Translation and other NLP applications.

5.4 Comparison with Pairwise Similarity

We present experimental results to compare our proposed approach with (Han et al., 2012) which used pairwise contextual similarity to induce a normalization lexicon of 40K entries, we will refer to this lexicon as HB-Dict. We compare the performance of HB-Dict and our induced dictionary (system RW3). We evaluate both system on SM-Test test set and on (Han et al., 2012) test set of 548 sentences which we call here HB-Test.

System	Precision	Recall	F-Measure
SM-Test			
HB-Dict	71.90	26.30	38.51
RW3	92.43	56.4	70.05
HB-Test			
HB-Dict	70.0	17.9	26.3
RW3	85.37	56.4	69.93

Table 3: Text Normalization Results

As shown in Table(3), RW3 system significantly outperforms HB-Dict system with the lexicon from (Han et al., 2012) on both test sets for both precision and recall. The contextual graph random walks approach helps in providing high precision lexicon since the sampling nature of the approach helps in filtering out unreliable normalization equivalences. The random walks will traverse more frequent paths; which would lead to more probable normalization equivalence. On the other hand, the proposed approach provides high recall as well which is hard to achieve with higher precision. Since the proposed approach deploys random walks to sample paths that can traverse many steps, this relaxes the constraints that the normalization equivalences have to share the same context. Instead a noisy word may share a context with another noisy word which in turn shares a context with a clean equivalent normalization word. Therefore, we end up with a lexicon that have much higher recall than the pairwise similarity approach since it explores equivalences beyond the pairwise relation. Moreover, the random walk sampling emphasis the more frequent paths and hence provides high precision lexicon.

5.5 Output Analysis

Table(4) shows some examples of the induced normalization equivalences, the first part shows good examples where vowels are restored and phonetic similar words are matched. Remarkably the correction "'viewablity"' to "'visibility"' is interesting since the system picked the more frequent form. Moreover, the lexicon contains some entries with foreign language words normalized to its English translation. On the other hand, the lexicon has some bad normalization such as "'unrecycled "' which should be normalized to "'non recycled"' but since the system is limited to one word correction it did not get it. Another interesting bad normalization is "'tutting"' which is new type of

dancing and should not be corrected to "'tweeting"'.

Noisy	Clean	Remarks
tnght	tonight	Vowels restored
darlin	darling	g restored
urung	orange	phonetic similarity
viewablity	visibility	good correction
unrecycled	recycled	negation ignored
tutting	tweeting	tutting is dancing type

Table 4: Lexicon Samples

Table 5 lists a number of examples and their normalization using both **Baseline1** and **RW3**. At the first example, **RW3** got the correct normalization as *"interesting"* which apparently is not the one with the shortest edit distance, though it is the most frequent candidate at the generated lexicon. The baseline system did not get it right; it got a wrong normalization with shorter edit distance. Example(2) shows the same effect by getting *"cuz"* normalized to *"because"*. At Example(3), both the baseline and RW3 did not get the correct normalization of *"yur" to "you are"* which is currently a limitation in our system since we only allow one-to-one word mapping in the generated lexicons not one-to-many or many-to-many. At Example(4), RW3 did not normalize *"dure"* to *"sure"* ; however the baseline normalized it by mistake to *"dare"*. This shows a characteristic of the proposed approach; it is very conservative in proposing normalization which is desirable as a preprocessing step for NLP applications. This limitation can be marginalized by providing more data for generating the lexicon. Finally, Example 4 shows also that the system normalize *"gr8"* which is mainly due to having a flexible similarity cost during the normalization lexicon construction.

1. Source: *Mad abt dt so mch intesting* **Baseline1:** *Mad at do so much ingesting* **RW3:** *Mad about that so much interesting*
2. Source: *i'l do cuz ma parnts r ma lyf* **Baseline1:** *I'll do cut ma parents r ma life* **RW3:** *I'll do because my parents are my life*
3. Source: *yur cuuuuute* **Baseline1:** *yur cuuuuute* **RW3:** *your cute*
4. Source: *I'm dure u will get a gr8 score* **Baseline1:** *I'm dare you will get a gr8 score* **RW3:** *I'm dure you will get a great score*

Table 5: Normalization Examples

5.6 Machine Translation Task Evaluation

The final evaluation of the text normalization system is an extrinsic evaluation where we evaluate the effect of the text normalization task on a social media text translating from English to Spanish using a large scale translation system trained on general domain data. The system is trained on English-Spanish parallel data from WMT 2012 evaluation [4]. The data consists of about 5M parallel sentences on news, europal and UN data. The system is a state of the art phrase based system similar to Moses (Hoang et al., 2007). We used The BLEU score (Papineni et al., 2002) to evaluate the translation accuracy with and without the normalization. Table(6) shows the translation evaluation with different systems. The translation with normalization was improved by about 6% from 29.02 to 30.87 using RW3 as a preprocessing step.

System	BLEU	Impreovemnet
No Normalization	29.02	0%
Baseline1	29.13	0.37%
HB-Dict	29.76	3.69%
RW3	30.87	6.37%

Table 6: Translation Results

6 Conclusion and Future Work

We introduced a social media text normalization system that can be deployed as a preprocessor for MT and various NLP applications to handle social media text. The proposed approach is very scalable, adaptive to any domain and language. We show that the proposed unsupervised approach provides a normalization system with very high precision and a reasonable recall. We compared the system with conventional correction approaches and with recent previous work; and we showed that it highly outperforms other systems. Finally, we have used the system as a preprocessing step for a machine translation system which improved the translation quality by 6%.

As an extension to this work, we will extend the approach to handle many-to-many normalization pairs; also we plan to apply the approach to more languages. Furthermore, the approach can be easily extended to handle similar problems such as accent restoration and generic entity normalization.

[4] http://www.statmt.or/wmt12

Acknowledgments

We would like to thank Lee Schwartz and Will Lewis for their help in constructing the test sets and in the error analysis. We would also like to thank the anonymous reviewers for their helpful and constructive comments.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 3340, Sydney, Australia.

Eric Brill and Robert C. Moore. 2000. *An improved error model for noisy channel spelling correction*, In ACL 2000: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, Englewood Cliffs, NJ, USA.

Ye-In Chang and Jiun-Rung Chen and Min-Tze Hsu 2010. A hash trie filter method for approximate string matching in genomic databases Applied Intelligence, 33:1, pages 21:38, Springer US.

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 of Document Analysis and Recognition, vol. 10, pp. 157:174.

Danish Contractor and Tanveer Faruquie and Venkata Subramaniam 2010. Unsupervised cleansing of noisy text. In COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics, pages 189:196.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization.. In CALC 09: Proceedings of the Workshop on Computational Approaches to Linguistic Creativity, pages 71:78, Boulder, USA.

Dipanjan Das and Slav Petrov 2011 Unsupervised part-of-speech tagging with bilingual graph-based projections Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 600:609, Portland, Oregon

Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In Proceedings of the First workshop on Unsupervised Learning in NLP, pages 82:90, Edinburgh, Scotland.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a twitter. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011), pages 368:378, Portland, Oregon, USA.

Bo Han and Paul Cook and Timothy Baldwin 2012. Automatically Constructing a Normalisation Dictionary for Microblogs. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012), pages 421:432, Jeju Island, Korea.

Hieu Hoang and Alexandra Birch and Chris Callisonburch and Richard Zens and Rwth Aachen and Alexandra Constantin and Marcello Federico and Nicola Bertoldi and Chris Dyer and Brooke Cowan and Wade Shen and Christine Moran and Ondrej Bojar 2007. Moses: Open source toolkit for statistical machine translation.

Thad Hughes and Daniel Ramage 2007. Lexical semantic relatedness with random graph walks Proceedings of Conference on Empirical Methods in Natural Language Processing EMNLP, pp. 581589, Prague

Fei Liu and Fuliang Weng and Bingqing Wang and Yang Liu 2011. Insertion, Deletion, or Substitution? Normalizing Text Messages without Precategorization nor Supervision Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 19:24, Portland, Oregon

Dan Melamed 1999. Bitext Maps and Alignment via Pattern Recognition. In Computational Linguistics, 25, pages 107:130.

Einat Minkov and William Cohen Graph Based Similarity Measures for Synonym Extraction from Parsed Text In Proceedings of the TextGraphs workshop 2012

J. Norris 1997. Markov Chains. Cambridge University Press.

Kishore Papineni and Salim Roukos and Todd Ward and Wei-jing Zhu 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. in Proceedings of ACL-2002: 40th Annual meeting of the Association for Computational Linguistics. , pages 311:318.

Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. Normalization of non-standard words. 2001.

Xu Sun and Jianfeng Gao and Daniel Micol and Chris Quirk 2010. Learning Phrase-Based Spelling Error Models from Clickthrough Data. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 266:274, Sweeden.

Martin Szummer and Tommi 2002. Partially labeled classification with markov random walks. In Advances in Neural Information Processing Systems, pages 945:952.

Kristina Toutanova and Robert C. Moore. Pronunciation modeling for improved spelling correction.. 2002. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL , pages 144151, Philadelphia, USA.

Justin Zobel and Philip Dart 1996. Phonetic string matching: Lessons from information retrieval. in Proceedings of the Eighteenth ACM SIGIR International Conference on Research and Development in Information Retrieval, pages 166:173, Zurich, Switzerland.