Improving Text Normalization Using Character-blocks based Models and System Combination

ChenLi YangLiu
Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Road, Richardson, Texas
{chenli,yangl}@hlt.utdallas.edu

ABSTRACT
There are many abbreviation and non-standard tokens in SMS and Twitter messages. Normalizing these non-standard tokens will ease natural language processing modules for these domains. Recently, character-level machine translation (MT) and sequence labeling methods have been used for this normalization task, and demonstrated competitive performance. In this paper, we propose an approach to segment words into blocks of characters according to their phonetic symbols, and apply MT and sequence labeling models on such block-level. We also propose to combine these methods, as well as with other existing methods, in order to leverage their different strengths. Our experiments show our proposed approach achieved high precision and broad coverage.

TITLE AND ABSTRACT IN CHINESE

使用字符块模型和系统合成提高文字纠错率

在手机短信和Twitter消息中存在许多缩写和非规范的单词。纠正这些非规范的单词会给后续的自然语言处理模型带来便利。近年来，基于字符层面的机器翻译方法和序列标注方法被广泛应用到这个任务中，并且大大提高了纠错率。本文介绍了一种根据单词发音来切割单词的方法，然后把上述的两种模型利用在切割后的字符块层面进行纠错尝试。另外，我们还尝试了将现有的方法和提出的方法进行各种合成以期利用各自的优点来提高最终的效果。最后我们的实验表明我们的方法在对非规范词的纠错召回率和准确率上都有了明显的提高。

KEYWORDS: text normalization, machine translation, CRF, NLP application.

KEYWORDS IN CHINESE: 文本纠错，机器翻译，条件随机场，自然语言处理应用.

Proceedings of COLING 2012: Technical Papers, pages 1587–1602,
COLING 2012, Mumbai, December 2012.

1 Introduction

There has been a rapid increase in social text in the last few years, including the mobile phone text messages (SMS), comments from the social media websites such as Facebook and Twitter, and real-time communication platforms like MSN and Gtalk. For example, by July 2012[1], Twitter has more than 500 million users, and more than 340 million new tweets are sent everyday. This trend attracts a lot of research in order to mine valuable information and knowledge from this data. Unfortunately, traditional NLP tools sometimes perform poorly when processing this kind of text. One of reasons is that social text is very informal, and contains many misspelled words, abbreviations and many other non-standard tokens. There are different efforts to develop robust language processing techniques for this domain. One is to develop NLP tools that are designed specially for tweets or SMS. For example, the system in (Ritter et al., 2011) reduced the errors of POS tagging by 41% compared with Stanford POS Tagger, and by 22% in parsing tweets compared with OpenNLP chunker tool. Another way is to perform some preprocessing on the original informal text, such as normalization, such that the resulting text is easier for the subsequent modules. The task of normalization is to automatically convert the non-standard tokens into corresponding standard words. Intuitively this will ease subsequent language processing modules in this domain. For example, if '2mr' is converted to 'tomorrow', a text-to-speech system will know how to pronounce it, a POS tagger can label it correctly, and an information extraction system can identify it as a time expression.

Non-standard tokens are formed in different ways in social text, and because of different reasons, including factors such as length limitation, individual's writing style, need to convey emotional or other information. A lot of research has been conducted for text normalization. Some are designed to handle one type of non-standard words, such as deletion (e.g., 'abt' for 'about') (Pennell and Liu, 2010); some can deal with different kinds of non-standard tokens (Liu et al., 2011). Various modeling approaches have been explored for this problem. In particular, a character-level MT method (Pennell and Liu, 2011) and sequence labeling approach (Pennell and Liu, 2010) were used recently and showed good performance. Their essential thought is to make alignment between characters in non-standard tokens and standard words, and predict the probability of the candidate standard words given the non-standard tokens.

In this paper, we propose to use character blocks for alignment, rather than single characters, but adopt the same framework as the above character based methods. Since some characters appear together frequently (e.g., 'ght', 'tion', and 'ure'), we expect that keeping such combinations in the alignment is better than splitting into individual characters in improving system precision (yielding fewer candidates but more accurate). To generate the character blocks, we resort to the word's phonetic symbol and some common pronunciation rules. This results in much better alignment for the MT and sequence labeling methods. In addition, we use two other normalization methods: spell checker (Damerau, 1964) and character-level two-step MT model (Li and Liu, 2012). We further propose to combine these different systems to take advantage of each individual system's strength. For example, the spell checking model is good at correcting simple errors such as 'talkin' (meaning 'talking'), 'freind' (meaning 'friend'), and 'tgether' (meaning 'together'); character-level two-step MT is designed to process the non-standard tokens generated by phoneme similarity like 'ate' (meaning 'eight' in certain context); character blocks level MT and sequence labeling complement in improving the precision on transferring non-standard tokens that have high grapheme similarity with possible standard

[1] http://en.wikipedia.org/wiki/Twitter

words. Results on one public test set show that our proposed individual systems and combined systems perform better than prior work.

2 Related Work

Text normalization has been well studied in text-to-speech field. See (Sproat et al., 2001) for a good report of this problem. Recently, much research has been done on this problem for social text domain, which has many abbreviations or non-standard tokens. A simple approach for normalization would be applying traditional spell checking model, which is usually based on edit distance (Damerau, 1964; Levenshtein, 1966). This method works well for non-standard tokens that are generated by a few simple operations such as substitution, insertion, and deletion of letters of the standard words. However, it cannot handle words such as 'ate' (meaning 'eight'), where non-standard tokens are created based on the phoneme similarity with the standard words, or 'bday' (meaning 'birthday'), which involves too many operations.

Another popular and successful line of work in normalization adopts a noisy channel model. For a non-standard token A, this method finds the most possible standard word S based on Bayes rule: $argmaxP(S|A) = argmaxp(A|S)*p(S)$. Different methods have been used to compute p(A|S). (Pennell and Liu, 2010) used a CRF sequence modeling approach for deletion-based abbreviation. (Liu et al., 2011) further extended this work – they also used CRF framework, but considered more types of nonstandard words without explicit pre-categorization for non-standard tokens. They also used a set of automatically collected training data. (Liu et al., 2012) continued their work by adopting visual prime approach to improve the coverage of candidate list. (Xue et al., 2011) modeled p(A|S) by computing the grapheme and phoneme similarity and then combined those results with context channel and acronym channel.

Some research also utilized the noisy channel model at sentence level. For informal text T and its possible standard form S, using a noisy channel model: $argmaxP(S|T) = argmaxP(T|S)P(S)$, where P(S) is a normal word-based language model, and P(T|S) is an error model. (Choudhury et al., 2007) used hidden Markov model to simulate SMS messages generation, considering the non-standard tokens in input sentence as emission state in HMM and labeling results are possible candidates. (Cook and Stevenson, 2009) extended the work by adding several more subsystems in the error model according to the most common non-standard tokens' formation process. (Wong and Xia, 2008) used the noisy channel model to handle the normalization of Chinese chat language based on a specific property of Chinese language – the similarity of two Chinese characters is measured based on the initial (shengmu) and final (yunmu) pinyin.

Machine translation is another commonly chosen method for text normalization. (Aw et al., 2006) treated SMS as another language, then machine translation techniques are used to translate this 'foreign language' to regular English. (Contractor et al., 2010) used an MT model as well but the focus of his work is to generate an unsupervised method to clean noisy text in this domain. (Pennell and Liu, 2011) firstly introduced an MT method at the character-level for normalization. In fact, this is another way to compute the word-level $p(A|S)$ mentioned before. (Li and Liu, 2012) extended this character-level MT by leveraging phonetic symbol, translating nonstandard words to phones first, and then phones to standard words. They called it character-level two-step MT and showed better overall coverage of candidates than the one-step MT. (Kobus et al., 2008) tackled normalization through an ASR-like system based on the fact that text messages can be represented by phonetic symbols. They decoded SMS message through a nondeterministic phonemic transduction.

For the normalization task for sentences (or messages), a system needs to first identify words that need to be normalized. This is typically done by simply checking whether a word is in a given dictionary. (Han and Baldwin, 2011) conducted a pilot study on determining whether an out-of-vocabulary (OOV) word is a non-standard token that needs normalization or it is just a well formed OOV word. Then for those ill-formed OOV words, they used grapheme and phoneme level similarity to generate candidate words.

Most of the above systems are specially designed to tackle certain aspects of the normalization task. Some focus on improving precision and some aim for a broad coverage. For example, character-level MT from (Pennell and Liu, 2011) achieved top-1 precision of 60.39% on the SMS test set from (Choudhury et al., 2007), and the sequence labeling model from (Liu et al., 2011) improved it to 62.05%, but they have limitations on the candidate coverage (the top-10 precision in (Pennell and Liu, 2011) is around 75%). In addition, (Pennell and Liu, 2010) can only handle the abbreviations. The character-level two-step MT model from (Li and Liu, 2012) improved the top-10 coverage to over 80% on the same data, however, its top-1 precision is not much better than previous work. The visual prime approach from (Liu et al., 2012) improves the overall coverage to around 94%, but the online computation is very time consuming since for a non-standard token T, the system needs to compute the visual prime value between T and all the words S that start with the same character as T. In this paper, we propose an enhanced MT and sequence labeling model, which are built at the character-block level. Furthermore, we combine multiple systems. The performance on a public test set shows our system yields better precision and recall/coverage. Compared to the visual prime approach (Liu et al., 2012), our system also has lower computational cost. To the best of our knowledge, we are the first to use word segmentation to obtain phonetically meaningful character blocks in the normalization task, and also first to combine MT and sequence labeling models.

3 Normalization System

For a non-standard token, we use four different subsystems to normalize it. Each system generates a list of candidate standard words. We then use heuristic rules to re-rank the candidates to form a final candidate list. The four subsystems are developed based on previous work, however, one key contribution of ours in this study is that we propose to segment words into several blocks based on the word's pronunciation and use these blocks as the units in the machine translation or sequence labeling approaches. This helps yield better alignment for model training. In this section, we first describe our four normalization subsystems in details, and then the combination rules.

3.1 Character-Block Level MT

3.1.1 Character-level Machine Translation

This method was first used in (Pennell and Liu, 2011). Similar to machine translation for a word sequence (i.e., a sentence), character-level machine translation translates the character sequence as seen in a non-standard token to another character sequence of a proper word. Formally, for a non-standard token $A = a_1a_2a_3...a_n$, the task is to find the most likely standard word $S = s_1s_2s_3...s_m$, where a_i and s_i are the characters in the words. A statistical machine translation method is used for this task:

$$\hat{S} = argmaxP(S|A) = argmaxP(A|S)P(S) = argmaxP(a_1a_2a_3...a_n|s_1s_2s_3...s_m)P(s_1s_2s_3...s_m) \quad (1)$$

where $P(a_1a_2a_3...a_n|s_1s_2s_3...s_m)$ is from a character-level translation model, and $P(s_1s_2s_3...s_m)$ is from a character-level language model. The translation model is trained using a parallel corpus containing pairs of non-standard tokens and standard words, and the character n-gram language model can be trained using an English dictionary. During testing, the translation module generates hypotheses of character sequences for a given non-standard token. An English dictionary can be used to remove candidates that are not in the dictionary and preserve N-best candidates.

3.1.2 Character-block Level Machine Translation

When using the character-level MT normalization method as described above, we find some character alignments between the non-standard tokens and standard words are not correct. Fig 1(a) shows such an incorrect example. The second 'e' in 'yesterday' is aligned to 's' in the non-standard token, rather than to 'null' (i.e., character deletion). This wrong alignment will affect translation model training and subsequently the decoding process. For this example, if 'er' in 'yesterday' can be viewed as a block or segment, it can be easily aligned to 'r' in the non-standard token 'ystrdy'. This can be achieved using pronunciation information – 'er' together is pronounced as 'ɚ' in the above example. Therefore, we propose to segment an English word according to its phonetic symbol or pronunciation, and then use these segments as units in the MT system. For the example of 'yesterday', when considering its pronunciation, it is segmented as 'y e s t er d ay'.

In our experiment, we collected about 60 most common pairs of character-blocks and their corresponding phonetic symbols, and use them for segmentation. Some example pairs are shown in Table 1. Once we perform this new segmentation, the normalization procedure is similar to the character-based one described earlier, except that now the units in the MT system are characters or character blocks. Fig 1 (b) shows the translation alignment when using the character blocks. During testing, the MT system generates sequences of characters or character blocks. Again, we remove the spaces and only keep a word candidate if it is in the dictionary.

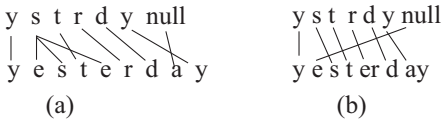


Figure 1: Alignment of 'ystrdy' and 'yesterday' using Character-level MT (a) and Character-block level MT (b)

3.2 Character-Block Level Sequence Labeling

3.2.1 Character-level Sequence Labeling

In (Pennell and Liu, 2010), a CRF sequence modeling approach was used for normalizing deletion-based abbreviation. Given a pair of standard word *W* and its abbreviation *A*, every character in *W* is labeled as 'Y' if it appears in *A*, otherwise 'N'. Fig 2 (a) presents a training instance. A sequence labeling model can be trained using characters as instances, each of which is represented by some features (e.g., character n-grams, position, phonetic information). Next, this model is applied to all the dictionary words. N-best labeling results (i.e., possible abbrevi-

Size of block	Block	Phonetic symbol
2	ea	eɪ,iə,iæ,i
	oo	u,ʊ,ʌ,oʊɑ,oʊə
	ch	k,ʧ
3	ear	ɛər,ɪər,ɜr
	oar	ɔr
	ght	t
4	tion	ʃən
	cial	ʃəl
	sure	ʃər, ʃɜr, ʒər

Table 1: Example pairs of character blocks and corresponding phonetic symbols.

ations) are generated for each word. This forms a lookup table – an abbreviation is associated with all the words that can possibly generate this abbreviation. During testing, for a given abbreviation A, if it is an entry in the lookup table, all of its candidate words will be returned, each with a score $P(A|W) * P(W)$, where $P(A|W)$ is based on the sequence labeling model, and $P(W)$ is from a word-based unigram LM. The best candidate can be obtained from this ranked list. Note if the test abbreviation is not an entry of the lookup table, it means this model fails to normalize this abbreviation token.

(Liu et al., 2011) extended the above model by relaxing the limitation that it only tackles the abbreviation tokens. Rather than labeling 'Y' or 'N' for every character in standard words, they give each character the following 4 types of label: (1) one of the 0-9 digits; (2) one of the 26 characters; (3) null character '_', and (4) letter combination (they used 5 common combinations, such as 'wh', 'ck', and 'ey'). Therefore this system can handle other types of non-standard tokens, such as substitution. An automatic letter level alignment method was used to assign every character in the standard word with a proper label in the non-standard token. Fig 2 (b) presents such a training instance.

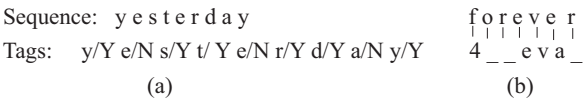


Figure 2: Training examples in sequence labeling models: (a) used in (Pennell and Liu, 2010) and (b) used in (Liu et al., 2011).

3.2.2 Character-Block Level Sequence Labeling

In the above sequence labeling approach, characters are used as instances in the CRF model (although the labels include other symbols or a few letter combination in (Liu et al., 2011)). Similar to the character-based MT method, this can introduce some problems in the alignment. As Fig 2 (b) shows, in the pair of 'forever' and '4eva', letter 'o' and 'r' are aligned to the empty character. Intuitively it is better to align the whole letter chunk 'for' with '4' (actually this is similar to the phrase table in the MT method). Therefore in the sequence labeling model, we

also propose to segment a word into character-blocks rather than single characters, in order to avoid this kind of improper alignments.

Given a training pair (S,T), where T is a non-standard token, and S is the corresponding standard word. First, S is segmented into a sequence of character-blocks and characters, in the same way as in Section 3.1. Then we align it with the characters in non-standard tokens based on the longest common sequence algorithm. At last, we use similar rules as in (Liu et al., 2011) to process the remaining characters between the aligned characters. The only difference is that in non-standard tokens we do not have predefined frequent letter combinations to help decide whether to group a character or character-block with the previous or the following chunk. We put them together with the previous chunk if they have no characters or character-blocks to align in the standard word.

After assigning proper labels for every character and character-block in the standard word, we extract the following features:

- Character-blocks n-gram: c_{-1}, c_0, c_1, $c_{-2}c_{-1}$, $c_{-1}c_0$, c_0c_1, c_1c_2, $c_{-3}c_{-2}c_{-1}$, $c_{-2}c_{-1}c_{-0}$, $c_{-1}c_0c_1$, $c_0c_1c_2$, $c_1c_2c_3$. These are the same as those used (Liu et al., 2011), except that c_i can be character blocks in our work.

- Phonetic features: p_{-1}, p_0, p_1, $p_{-2}p_{-1}$, $p_{-1}p_0$, p_0p_1, p_1p_2, where p_i is the phone corresponding to character or character block c_i. In segmenting words into character-blocks by using phonetic symbols, we notice that for some words the resulting number of blocks is the same as the number of phones. In this case, it is straightforward to align the two sequences (one to one mapping). Otherwise, for the words in the training set, we manually align the two sequences in order to derive the phonetic featuers.

- Syllable features: The website http://www.dict.cn provides syllabification information. We use their syllable segments to assign feature 'B', 'I', and 'L' label for every character-block or character. 'B' means this character or character-block is at the beginning of a syllable, 'I' means inside of a syllable, and 'L' means at the end position of a syllable.

- Word boundary feature: We distinguish the first, last, and inside characters and character-blocks by this feature.

Table 2 shows an example training instance with its features.

Character-block level word	for	e	v	er
Phonetic symbols	fɔr	e	v	ər
Syllable boundary	B	B	I	L
Word boundary	B	I	I	L
Label	4	e	v	a

Table 2: Example of an instance with features in character-block level sequence labeling system.

After training the character-block level CRF model, similar to the previous character-level sequence labeling methods, we apply it to the standard words in the dictionary to create the reverse look-up table, and use it to retrieve candidate words for non-standard tokens during

testing. One thing we need to point it out is that for the words that have different number of character segments and phonetic symbols, we do not include them when building the look up table (since generating high quality phonetic features needs human intervention, which is time consuming).

(Pennell and Liu, 2011),

3.3 Character-level Two-step MT

Li and Liu (2012) introduced a character-level two-step MT method for normalization. Instead of directly translating the character sequence, it first translates the non-standard tokens to possible phonetic sequences, and then to proper words. The purpose of this design is to tackle some difficult cases. For example, for the non-standard word 'rit', it is not easy for the single translation model to convert it to 'right' if there are not enough samples containing 'ght' and 't' in the training pairs. In this case, we can first translate 'rit' to possible phonetic sequences such as 'rait' or 'rIt', and then convert them back to in-vocabulary words. The two steps can both be implemented using machine translation.

In re-implementing this system, we use a different rule to rerank the the candidates compared to that used in (Li and Liu, 2012). For the phonetic sequence generated in the first stage using this method, some can be converted to proper words directly by a simple lookup in the dictionary that consists of word along with their pronunciation (we call these candidates List 1); others have to go through the second-stage translation module to generate possible word candidates (we call these List 2). Our reranking procedure uses a set of heuristic rules to update the probabilities for the candidates in List 2 depending on a candidate's positions in the two lists. For example, one rule is that, if a candidate ranks between 3 and 10 in List 2, and ranks above position 5 in List 1, we change its posterior probability to the same as that of the first candidate's in List 2. Another rule is that, if a candidate ranks between 20 and 40 in List 2 and ranks above position 5 in List 1, we change its posterior probability to the same as that of the 20th candidate in List 2. The essence of this set of rules is to change some candidates' positions to 1, 2, 3, 10 or 20 (we measure the top-N performance for these numbers of candidates). Finally for those candidates with the same posterior probabilities because of the above re-ranking process, we re-rank them again by their longest common sequence with the non-standard tokens.

3.4 Spell Checker

The last normalization subsystem we use is the Jazzy Spell Checker,[2] which is based on edit distance and integrates a phonetic matching algorithm as well.

3.5 System Combination

The above four methods have different strength: Spell checker is very good at correcting simple misspelled words that are formed by a few operations of deletion, insertion or substitution. MT and sequence labeling models have advantages in dealing with more complicated non-standard tokens, regardless of the types. The two-step MT model is designed to tackle the cases when the non-standard tokens have similar pronunciation with the correct answer, but are rather

[2]Jazzy: The java open source spell checker. http://jazzy.sourceforge.net

different on character surface (e.g., 'lrg' and 'large'). In general, the character-block level MT normalization system performs well (as shown later in the experiments).

In order to utilize the advantages of different systems, we propose to combine hypotheses from different systems to obtain the final result. To better evaluate the contributions of these subsystems in the combined system, we examine the combination of different systems:

- Comb1 (two systems): Character-block level MT system + Character-level two-step MT system
- Comb2 (two systems): Character-block level MT system + Character-block level sequence labeling system
- Comb3 (three systems): Character-block level MT system + Character-level two-step MT system + Jazzy Spell Checker
- Comb4 (three systems): Character-block level MT system + Character-block level sequence labeling system + Jazzy Spell Checker
- Comb5 (three systems): Character-block level MT system + Character-level two-step MT system + Character-block level sequence labeling system
- Comb6 (four systems): Character-block level MT system + Character-level two-step MT system + Character-block level sequence labeling system + Jazzy Spell Checker

Since the scores from different systems are in different range and vary greatly, we develop rules to combine them. The combination rules used for these systems are:

- Comb1 and Comb2: These involve two subsystems. The essence of combination rules is to re-rank. The steps we used are:
  (i) create an empty final candidate list;
  (ii) find the common candidates from the two candidate lists (i.e., intersection of the two sets). Since we notice that candidates below position 20 are often not very accurate or not useful for future use (e.g., sentence level normalization/decoding), we decide to obtain the intersection from the top 10 candidates of the two systems' result lists.
  (iii) put elements from the intersection set into the final candidate list, reversely rank them according to the sum of their positions in the two original lists (the smaller, the better). If there is a tie, we re-rank again according to their longest common sequence with the non-standard tokens (the higher, the better).
  (iv) after dealing with the elements in the intersection set, we add the remaining candidates from the two systems into the final list – the first remaining candidate from the character-block level MT system, then the first one from the other system's list, then the second ones from both lists, and so on, until one list is empty and we put all the rest candidates into the final candidate list.
- Comb3 and Comb4: These involve three subsystems and one is Jazzy Spell Checker. We only use the first candidate of the Jazzy spell checker. The combination rule is, if these three subsystems or any of the two subsystems have the same first candidate, select it as the first candidate in the final list, then ignore the Jazzy Spell Checker's result and combine the rest two candidate lists as above. Otherwise, ignore Jazzy Spell Checker's results and just combine the two subsystems as described above for Comb1 and Comb2.

- Comb5: This has three complicated subsystems (not including Jazzy spell checker). The essence of the combination rule is similar to the one used for two subsystems, but here we will consider the intersection of the three systems' top 10 results first, and then three intersection sets from any two subsystems' results. We use similar position information to rank candidates in these common sets. After this step, we first add the remaining candidates from the top 30 results from the sequence labeling system, then the top 30 from character-block lever MT, and at last those from character-level two-step MT.
- Comb6: This includes all the 4 subsystems. If the first candidate of all the systems are the same, or the first candidate of any three sub-systems are the same, choose it as the first candidate in the final list, then ignore the Jazzy Spell Checker's result and combine the rest three candidate lists as above. Otherwise, ignore Jazzy Spell Checker's results and combine the rest three system as Comb5.

Note if any one individual system's result list is empty, the condition is reduced to the combination of fewer individual systems. In addition, no duplicate candidates are included in the final candidate list.

4 Experiments

4.1 Experiment Setup

We use the following data sets. Data set 4 is first used as the training set and the other three as test sets. To evaluate the impact of training size, we use data sets 3 and 4 for training, and evaluate the models on data sets 1 and 2.

- Data 1: 303 pairs of non-standard tokens and standard words collected from SMS in 2007, used in (Choudhury et al., 2007).
- Data 2: 558 pairs of non-standard tokens and standard words collected from 549 tweets in 2010 by (Han and Baldwin, 2011).
- Data 3: 3,998 pairs of non-standard tokens and standard words collected from 6160 tweets between 2009 and 2010 by (Liu et al., 2011). We made another pass of this data set and corrected some annotation.
- Data 4: 2,028 unique pairs of non-standard tokens and standard words, collected from 4,660 Twitter messages, selected from the Edinburgh Twitter Data set collected in Aug 2009 by (Pennell and Liu, 2011).

The dictionary we used is obtained from http://ciba.iciba.com/, which includes 75,262 English word entries and corresponding phonetic symbols (IPA symbols). This is used in various modules in the normalization systems. We segment the English words into character-blocks and use them in both the character-block level MT and character-block sequence labeling model. As described earlier, in order to extract phonetic features for character-block sequence labeling model, we only use the words that have the same number of character-block segments and phones. Therefore, the number of the final standard words used to create the lookup table is 10,105. We choose 10,000-best output in CRF sequence labeling testing. This resulted in about 48,995K non-standard token entries in the lookup table, and the average number of standard words for an entry is about 8. We use CRF++ toolkit for our sequence labeling model.

For the two-step MT system, the training data for the first step is the parallel pairs containing non-standard tokens and their pronunciations (this is the pronunciation for the corresponding normalized words). We obtain this by replacing the standard words in the training set with their phonetic symbols. For the second stage, the training pairs are all the dictionary words and their phonetic symbols.

We train a 5-gram character language model from the Edinburgh Twitter corpus for both character-block level MT model and the second step of the two-step MT method. A phone-based language model is built from the IPAs in the dictionary, and used for the first step translation model in the two-step MT method. SRILM toolkit (Stolcke, 2002) is used to build these language models. We use Moses (Koehn et al., 2007) for all of our experiments. Giza++ (Och and Ney, 2003) is used for automatic word alignment for the three MT systems. We use 8 iterations by IBM model 1, 6 iterations by HMM model, and 6 iterations by IBM model 3 in the alignment. The final score from the translation model is a combination of the four sub-models: phrase translation, language model, reordering model, and the word length penalty weight.

In this study, we only focus on the word-level normalization in all of our experiments. A word-based unigram language model is combined with the normalization score when calculating the final score for a candidate for every individual system except the Jazzy Spell Checker system.

4.2 Experiment Result and Discussion

Tables 3, 4, and 5 present results on test data set 1, 2 and 3, using data set 4 as the training set. Similar to previous work, we evaluate n-best performance (n equals to 1, 3, 10 and 20), and also the candidate coverage (how many test cases' correct answers can be obtained in the final list regardless of its positions). We also include in the table the average number of candidates in the final list to make the coverage result more meaningful. Results in the tables are shown for individual systems and the combined systems, as well as those from previous work.

Among the individual systems, we notice that for data set 1 and 2, character-block level MT system's top one precision outperforms other individual systems, and character-level two-step MT has advantage on the top-n coverage (n from 3) compared with other systems. Character-block level sequence labeling usually has a lower top-1 precision than the two MT models, but its top-20 precision is not much poorer than others (sometimes it is even better). Character-block level sequence labeling can successfully normalize non-standard tokens such as 'gf', 'prof', and 'pro' (to 'girlfriend', 'professor', and 'problem' or 'probably'). However, these cases are very hard for the MT systems to tackle with. MT systems often fail to find correct candidates or sometimes offer no results for these cases. Jazzy spell checker has the worst performance on these two test sets compared to other three systems. In addition, because of its limited number of candidates, its top-n coverage is much lower than others.

The patterns on data set 3 are different. It indicates this data set has different characteristics than the other two. On data set 3, the sequence labeling method performs the best, although its absolute performance is similar to that on the other two data sets. Test data set 3 includes many test cases that the MT system is not good at tackling, such as those examples given above. Because of the poor performance of the MT systems on this data set, there is not much performance difference between them and the Jazzy spell checker.

For all the data sets, we can see that generally the sequence labeling method generates more candidates than others. This is partly because when creating the lookup table, we generated

a large number (10,000) of possible variations for a standard word in the dictionary in order achieve high coverage. Jazzy spell checker has limited number of candidate words, which explains its lower coverage than other methods, even though sometimes it can achieve competitive performance in the top-1 accuracy (e.g., on data set 3). Comparing the character-level MT and character-block level MT, we can see that our proposed method generates fewer number of candidates, but with better accuracy. This will be very beneficial when rescoring these candidates in sentence-level decoding/normalization.

Regarding the combination results, we can see that in general the combined system performs better than the individual systems. Comparing results for comb1 and comb3, and comb2 and comb4, we can see that after adding Jazzy Spell Checker's candidate, the top1 precision has some improvement, but the top-n coverage has no change. This is because we only use the first candidate from the spell checker. It also shows that Jazzy spell checker has some advantages in dealing with some cases and that adding it can yield performance gain. Comparing comb4 and comb6 (the latter includes Jazzy spell checker), we can see that for data set 2, comb4 has better top-1 performance, but not the top-n coverage, whereas on data set 1 and 3, comb6 performs better. We think this might be explained by the relative performance of the character-level two-step MT system. On data set 1 and 3, the character-level two-step MT has better top-n performance (n from 3) than the character-block MT. However, it is different for data set 2. Another important reason for the mixed results for these combined systems is the heuristic rules we use in system combination. We believe better ways of combination can improve performance and make the systems more robust.

Comparing our system performance with other previously reported results, we can see that on data set 1 (which is the most widely used data in previous studies), our system performs the best. On data set 2 and 3, out results are slightly worse than (Liu et al., 2012). One important reason is the training data size – ours is much smaller. However, our system uses fewer candidates and is more efficient than (Liu et al., 2012), which considers all the English words with the same initial character. For the size of the English dictionary we use, which consists of 75,262 words, the average number of candidates would be about 2,894, much higher than the candidates we have in our systems.

For the character-block sequence labeling system, we also performed some manual correction to fix some bad alignment between the character blocks and their phonetic symbol. Results for this modified sequence labeling system and using it in the combination with others are shown at the end of the tables (marked up **). We can see that there is consistent improvement after doing manual correction for the alignment, suggesting the alignment quality is important for model training. This is consistent with the finding that using character blocks outperforms character-based models, since the former has better alignment quality.

Since three of the normalization systems are statistical models, we evaluate the effect of the training data size on the system performance. In this experiment, we also include data set 3 in the labeled training data. We randomly chose some training pairs from it and combine with data set 4, creating training data with 3,000, 4,000 and 5,480 unique pairs respectively (5,480 pairs is the union of data set 3 and 4). Fig 3(a) shows the learning curve evaluated on data set 1 and 2 using these 4 training sizes. This is the result using the combined system 6 involving manual correction. As expected, the performance improves as the training data becomes larger. The pattern suggests that adding more training data may yield further gain. Note that for data set 2, when the training data is 5,480 pairs, it can achieve top one precision of 75.8%, which is

SMS Dataset (303 pairs)	Accuracy					# Cand
	Top1	Top3	Top10	Top20	Coverage	
Character-level MT	65.7	73.7	77	77.3	77.7	12.5
Character-block level MT	67.3	74.7	77.3	78	78	9.2
Character-block level sequence labeling	55	65.3	70.7	72.7	76	46
Character-level two-step MT	64.7	76.3	80.7	81.7	82.4	21.9
Jazzy Spell Checker	43.3	46.7	46.7	46.7	46.7	1.3
Comb6	72.6	83	88.3	90.3	93.7	45.2
Liu et al., 2012	64.36	80.2	89.8	91.8	94.1	n/a
Pennell et al., 2011	60.4	74.6	75.6	75.6	n/a	n/a
Cook et al., 2009	59.4	n/a	83.8	87.8	n/a	n/a
Choundhury et al., 2007	59.9	n/a	84.3	88.7	n/a	n/a
Comb1	68.7	77	82.7	83.3	85	25.7
Comb2	70.7	84	88	91	94	52.7
Comb3	69.7	77.3	82.7	83.3	85	25.7
Comb4	71.3	84	88.3	91	94	52.7
Comb5	73	83	88.3	90.3	93.7	45.2
Character-block level sequence labeling**	58.6	66.7	72.3	74	76.4	59.3
Comb6**	**74.6**	**84.6**	**90.3**	**92**	**94.3**	59.3

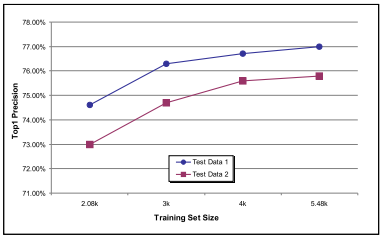
Table 3: Results on Data Set 1. ** means there is manual correction in alignment in the sequence labeling method.

Twitter Dataset (558 pairs)	Accuracy					# Cand
	Top1	Top3	Top10	Top20	Coverage	
Character-level MT	62.2	71.3	75	76.5	77	14.2
Character-block level MT	63.1	72.6	76.2	76.9	77.1	10.7
Character-block level sequence labeling	57.1	67	72	75.6	80.4	62.8
Character-level two-step MT	61.1	71.8	78.3	79.8	81.2	27.5
Jazzy Spell Checker	44.8	50.9	51.4	51.6	51.6	1.5
Comb6	71.7	81.5	86	**89.5**	94.2	38
Liu et al., 2012	**75.69**	n/a	n/a	n/a	n/a	n/a
Han and Baldwin et al., 2011	75.3	n/a	n/a	n/a	n/a	n/a
Comb1	66.2	75	82.4	84.7	86.1	28.9
Comb2	70.6	80.4	84.5	87	92.2	63.9
Comb3	68	75.8	82.4	84.7	86.1	28.9
Comb4	73.1	81.7	84.5	87	92.2	63.9
Comb5	71.1	81.5	86	89.5	94.2	38
Character-block level sequence labeling**	57.4	67	72.5	75.1	80	80.7
Comb6**	73.0	**81.9**	**86.7**	89.2	**94.2**	38.9

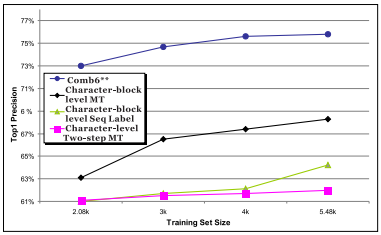
Table 4: Results on Data Set 2. ** means there is manual correction in alignment in the sequence labeling method.

Twitter Dataset (3,998 pairs)	Accuracy					# Cand
	Top1	Top3	Top10	Top20	Coverage	
Character-level MT	49.6	56.8	60.3	63.4	63.7	14.5
Character-block level MT	49.8	57.8	61.2	61.8	62	10.7
Character-block level sequence labeling	54	65.5	74.3	78.3	86.9	52.7
Character-level two-step MT	47.3	60.4	66.3	68.8	69.3	25.8
Jazzy Spell Checker	48.7	54.2	54.7	54.7	54.7	1.6
Comb6	61.4	73.5	82.5	86.3	89.5	40.2
Liu et al., 2012	**69.81**	**82.51**	**92.24**	**93.79**	**95.71**	n/a
Comb1	51	61.7	70	72	73.5	44.3
Comb2	57	70.8	78.9	81.9	86.9	77
Comb3	54	62.2	70	72	73.5	44.3
Comb4	61.4	72	78.9	81.9	86.9	77
Comb5	58.9	73.5	82.5	86.3	89.5	40.2
Character-block level sequence labeling**	58.6	70.2	78.4	81.6	85.5	71.1
Comb6**	62.6	75.1	84	87.5	90.7	45.4

Table 5: Results on Data Set 3. ** means there is manual correction of alignment in the sequence labeling method.



(a) Learning curve of the combination system Comb6** on Data set 1 and 2



(b) Learning curve of individual system and Comb6** on Data set 2

Figure 3: Learning curve for different data sets and differen systems

higher than the state-of-the-art result. We also plot the learning curves for individual systems along with the combined system in Fig 3(b). This is for data set 2. Similar trends are observed for data set 1. It is clear that the combined system outperforms all the individual systems. We can see that the performance of the character-block level MT improves steadily as the training data grows. In contrast, the training size effect on the character-level two-step MT is rather small. This suggests that the two-step MT may not need a large training set, and at the same time this method also has limits and it is hard to make further improvement. Finally for the character-block level sequence labeling, we notice a jump in its performance when training data grows from 4k to 5.48k, which may imply that the added new instances in the last trail are a better match with the test set, and there is still potential for performance gain. More annotated data is needed for a better understanding of the properties of these statistical models.

5 Conclusion and Future Work

In this paper we presented a character-block level MT and sequence labeling method for normalization in social text, and proposed various ways to combine these two novel methods with two other approaches: character-level two-step MT and Jazzy Spell checker. Experiments on several data sets show our methods perform competitively. In particular, on a widely used public test set containing 303 non-standard tokens (Choudhury et al., 2007), our system yields higher 1-best accuracy (74.6%) and better coverage (94.6%) than previous work. Results demonstrate that our proposed character blocks help generate better alignment in the MT and sequence labeling methods, which further improves normalization performance. We believe that our approach is general and applicable to many languages. The idea of aligning standard and non-standard word tokens is not language specific, and we expect it works for other languages, especially for Western European languages that are similar to English in terms of the way of forming non-standard words. Even for languages such as Chinese, there is previous work (Yang et al., 2009) that uses the CRF sequence labeling method for abbreviation detection. In our future work, we plan to first design more robust combination rules to combine different systems (e.g., weighted combination or reranking). Second we will perform sentence level normalization where word n-gram language models will be incorporated to re-rank the candidates.

Acknowledgments

This work is partly supported by DARPA under Contract No. HR0011-12-C-0016. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Aw, A., Zhang, M., Xiao, J., Su, J., and Su, J. (2006). A phrase-based statistical model for sms text normalization. In *Processing of COLING/ACL*, pages 33–40.

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., and Basu, A. (2007). Investigation and modeling of the structure of texting language. *IJDAR*, 10(3-4):157–174.

Contractor, D., Faruquie, T. A., Subramaniam, L. V., and Subramaniam, L. V. (2010). Unsupervised cleansing of noisy text. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters, Association for Computational Linguistics*, pages 189–196, Stroudsburg, PA, USA.

Cook, P. and Stevenson, S. (2009). An unsupervised model for text message normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78, Boulder, Colorado. Association for Computational Linguistics.

Damerau, F. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

Han, B. and Baldwin, T. (2011). Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceeding of 49th ACL*, pages 368–378, Portland, Oregon, USA.

Kobus, C., Yvon, F., and Damnati, G. (2008). Normalizing sms: are two metaphors better than one. In *Proceedings of 22nd International Conference on Computational Linguistics*, pages 441–448.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstrartion Sessions*, pages 177–180.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707.

Li, C. and Liu, Y. (2012). Normalization of text messages using character- and phone-based machine translation approaches. In *Proceedings of 13th Interspeech*.

Liu, F., Weng, F., and Jiang, X. (2012). A broad-coverage normalization system for social media language. In *Proceedings of the 50th ACL*, pages 1035–1044, Jeju Island, Korea.

Liu, F., Weng, F., Wang, B., and Liu, Y. (2011). Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th ACL*, pages 71–76, Portland, Oregon, USA.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Pennell, D. and Liu, Y. (2010). Normalization of text messages for text-to-speech. In *ICASSP*, pages 4842–4845.

Pennell, D. and Liu, Y. (2011). A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 974–982, Chiang Mai, Thailand.

Ritter, A., Clark, S., Mausam, and Etzioni, O. (2011). Named entity recognition in tweets: An experimental study. In *Proceedings of 2011 EMNLP*, pages 1524–1534.

Sproat, R., Black, A. W., Chen, S. F., Kumar, S., Ostendorf, M., and Richards, C. (2001). Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Stolcke, A. (2002). SRILM-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286.

Toutanova, K. and Moore, R. (2002). Pronunciation modeling for improved spelling correction. In *Proceedings of 40th ACL*, pages 144–151, Philadelphia, Pennsylvania, USA.

Wong, K.-F. and Xia, Y. (2008). Normalization of chinese chat language. *Language Resources and Evaluation*, 42(2):219–242.

Xue, Z., Yin, D., Davison, B. D., and Davison, B. D. (2011). Normalizing microtext. In *Proceedings of 25th AAAI*.

Yang, D., Pan, Y.-C., and Furui, S. (2009). Automatic chinese abbreviation generation using conditional random field. In *HLT-NAACL (Short Papers)*.