A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations

Deana L. Pennell
The University of Texas at Dallas
800 W. Campbell Road
Richardson, TX 75080
deana@hlt.utdallas.edu

Yang Liu
The University of Texas at Dallas
800 W. Campbell Road
Richardson, TX 75080
yangl@hlt.utdallas.edu

Abstract

This paper describes a two-phase method for expanding abbreviations found in informal text (*e.g.*, email, text messages, chat room conversations) using a machine translation system trained at the character level during the first phase. In this way, the system learns mappings between character-level "phrases" and is much more robust to new abbreviations than a word-level system. We generate translation models that are independent of the way in which the abbreviations are formed and show that the results show little degradation compared to when type-dependent models are trained. Our experiments on a large data set show our proposed system performs well when tested both on isolated abbreviations and, with the incorporation of a second phase utilizing an in-domain language model, in the context of neighboring words.

1 Introduction

Text messaging (SMS) is a rapidly growing form of alternative communication for cell phones. This popularity has caused safety concerns leading many US states to pass laws prohibiting texting while driving. The technology is also difficult for users with visual impairments or physical handicaps to use. We believe a text-to-speech (TTS) system for cell phones can decrease these problems to promote safe travel and ease of use for all.

Text normalization is the usual first step for TTS. Text message lingo is also similar to the chatspeak that is prolific on forums, blogs and chatrooms. Screen readers will thus benefit from such technology, enabling visually impaired users to take part in internet culture. In addition, normalizing informal text is important for various language processing tasks, such as information retrieval, summarization, and keyword, topic, sentiment and emotion detection, which are currently receiving a lot of attention for informal domains.

Normalization of informal text is complicated by the large number of abbreviations used. Some previous work on this problem used phrase-based machine translation (MT) for SMS normalization; however, a large annotated corpus is required for such a supervised learning method since the learning is performed at the word level. By definition, this method cannot make a hypothesis for an abbreviation it did not see in training. This is a serious limitation in a domain where new words are created frequently and irregularly.

We propose a two-phase approach. In the first phase, an MT model is trained at the character-level rather than the word- or phrase-level, allowing recognition of common abbreviation patterns regardless of the words in which they appear. We decode the hypotheses in the second phase using a language model for the final prediction.

2 Related Work

Text normalization is an important first step for any text-to-speech (TTS) system and has been widely studied in many formal domains. Sproat et al. (2001) provides a good resource for text normalization and its associated problems. Spell-checking algorithms are mostly ineffective on this type of data because they do not account for the phenomena in informal text. Some prior work instead focused on single typographic errors using edit distance (Kukich, 1992), perhaps combined with pronunciation modeling, such as (Toutanova and Moore, 2002).

One line of research views normalization as a noisy channel problem. Choudhury et al. (2007) describe a supervised noisy channel model using HMMs for SMS normalization. Cook and Stevenson (2009) extend this work to create an unsuper-

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 974–982,
Chiang Mai, Thailand, November 8 – 13, 2011. ©2011 AFNLP

vised noisy channel approach using probabilistic models for common abbreviation types and choosing the English word with the highest probability after combining the models. Deletion-based abbreviations were addressed in our past work using statistical models (maximum entropy and conditional random fields) combined with an in-domain language model (LM) (Pennell and Liu, 2010; Pennell and Liu, 2011). Liu et al. (2011) extend the statistical model to be independent of abbreviation type with good results.

Whitelaw et al. (2009) used a noisy channel model based on orthographic edit distance using the web to generate a large set of automatically generated (noisy) pairs to be used for training and for spelling suggestions. Although they use the web for collection, they do not focus on informal text but rather on unintentional spelling mistakes. Beaufort et al. (2010) combine a noisy channel model with a rule-based finite-state transducer and got reasonable results on French SMS, but have not tried their method on English text. Han and Baldwin (2011) first determine whether a given out-of-vocabulary (OOV) word needs to be expanded or is some other type of properly-formed OOV. For those predicted to be ill-formed, a confusion set of possible candidate words is generated based on a combination of lexical and phonetic edit distance and the top word is chosen in context using LM and dependency parse information.

Machine translation (MT) techniques trained at the word- or phrase-level are also common. Translation of SMS from one language to another led Bangalore et al. (2002) to use consensus translations to bootstrap a translation system for instant messages and chat rooms where abbreviations are common. Aw et al. (2006) view SMS lingo as if it were another language with its own words and grammar to produce grammatically correct English sentences using MT. Q. and H. (2009) trained an MT system using three on-line SMS dictionaries for normalizing chat-like messages on Twitter. Kobus et al. (2008) incorporate a second phase in the translation model that maps characters in the texting abbreviation to phonemes, which are viewed as the output of an automatic speech recognition (ASR) system. They use a non-deterministic phonemic transducer to decode the phonemes into English words. The technical paper of Raghunathan and Krawczyk (2009) details an explicit study varying language model orders, distortion limit and maximum phrase length allowed by the MT system during decoding. Contractor et al. (2010) also uses an SMT model; however, in an attempt to get around the problem of collecting and annotating a large parallel corpus, they automatically create a noisy list of word-abbreviation pairs for training using some heuristics. As far as we know, we are the first to use an MT system at the character-level for this task.

3 Data

Due to the lack of a large parallel corpus suitable for our study, we are building a corpus using status updates from twitter.com. Twitter allows users to update status messages by sending an SMS message to a number designated for the Twitter service. To ensure that our corpus is representative of the domain we are modeling, we use Twitter's meta-data to collect only messages sent via SMS. Some examples of highly abbreviated messages from our corpus are shown below.

(a) `Aye.oops,dat was spose to be a txt`

(b) `Rndm fct bout wife:  n the past 10 yrs I can cnt on one hand the num Xs she's 4gotn to unlock my car door`

(c) `OMG I LOVE YOU GUYS. You pwn :)  !!!`

(d) `i need to qo to bedd qotta wakee up at 7am for school....`

(e) `heard it again!  xD 3 TIMES.a sng i nvr hear!`

3.1 Choosing Messages for Annotation

An annotator's time is wasted if he is presented with many messages containing no abbreviations, or with sentences all containing the same, very common abbreviations. A scoring system was thus devised using the following metrics:

1. **Word Count Index.** A low word count index indicates that a message is close to the mean message length. Messages with fewer than five words are removed from consideration. We calculate the index as $|N-E(N)|/\sigma(N)$, where N is the number of words in the message and mean and standard deviation are calculated over the entire corpus.

2. **Perplexity Scores.** Two perplexity scores are calculated against character-level language

models. A low perplexity of the message compared to standard English text indicates that the message is less likely to be in a foreign language or jibberish. Similarly, a low perplexity of the message compared to our corpus indicates that the message is more representative of the domain. A sentence is removed if in either case the perplexity value is greater than a threshold (1000 in our study).

3. **OOV Count.** This is a simple count of the number of out of vocabulary (OOV) words in the message compared to an English dictionary, which we denote N_{OOV}. This metric helps guarantee that we select messages containing many OOV words. We remove the sentence completely when $N_{OOV} = 0$.
4. **OOV Percentages.** This metric consists of two scores: the first is N_{OOV}/N; the second is a non-duplicate OOV percentage, where we remove all repeated words and then recalculate the percentage. If the first score is greater than 0.5 but the second is not, we remove the message from consideration.
5. **OOV Frequency Scores.** For each OOV token (including emoticons) we find the frequency of the token across the entire corpus. This ensures that we annotate those abbreviations that are commonly used.

A sorted list is generated for each metric. The final score for each sentence is a weighted average of its position in each list, with more weight given to the non-duplicate OOV percentage and less weight given to the OOV frequency scores. The sentences are ranked in a penultimate list based on this final score. Finally, a post processing step iterates through the list to remove sentences introducing no new OOV words compared to higher-ranked sentences. Messages were annotated in the order of rank.

3.2 Annotation

Five undergraduate students were hired for the annotation task. In total, 4661 twitter status messages were annotated. Of these, 74 were given to multiple annotators so that we can calculate inter-annotator agreement. All 74 of these messages were also annotated by the first author as a standard for comparison. There were 7769 tokens annotated as an abbreviation by at least one annotator with 3761 (48%) unique to a single annotator.

We calculate pairwise agreement for whether or not a token is an abbreviation for all tokens given to each pair, including those that both agreed were not abbreviations. The Fleiss' Kappa $\kappa = 0.891$ for these pairwise values is quite high so the number of annotators can be reduced without negative effects to the project as long as they are familiar with the domain.

4 Normalization Approach

We describe a two-phase approach for SMS normalization. The first phase uses a character-level MT system to generate possible hypotheses for each abbreviation. The second uses a language model (LM) to choose a hypothesis in context.

Typically, an SMT system translates a sentence from one language to another. An alignment step learns a mapping of words and phrases between the two languages using a training corpus of parallel sentences. During testing, this mapping is used along with LMs to translate a sentence from one language to another. While researchers have used this method to normalize abbreviations (Bangalore et al., 2002; Aw et al., 2006; Kobus et al., 2008; Q. and H., 2009; Contractor et al., 2010), it is not robust to new words and leads to poor accuracy in this domain where new terms are used frequently and inconsistently.

Our system differs in that we train the MT model at the character-level during the first phase; that is, rather than learning mappings between words and phrases we instead map characters to other characters in what can be a many-to-many mapping. For example, the ending "-er" (as in "teacher" or "brother") is often abbreviated with the single character 'a'. Characters may also be mapped to symbols ("@" for "at"), numbers ("8" for "ate") or nothing at all (deletions).

Formally, for an abbreviation a : $c_1(a), c_2(a), ..., c_m(a)$ (where $c_i(a)$ is the i^{th} character in the abbreviation), we use an MT system to find the proper word hypothesis:

$$\begin{aligned} \hat{w} &= \arg\max p(w|a) \\ &= \arg\max p(w)p(a|w) \\ &= \arg\max p(c_1(w), ...c_n(w)) \\ &\quad \times p(c_1(a), ..., c_m(a)|c_1(w), ...c_n(w)) \end{aligned} \tag{1}$$

where $c_i(w)$ is a character in the English word w, $p(c_1(w), ...c_n(w))$ is obtained using a character LM, and $p(c_1(a), ..., c_m(a)|c_1(w), ...c_n(w))$ is based on the learned phrase translation table.

Pairs of words from our annotated data are used for training: the original token (which may or may not be an abbreviation) and its corresponding English word. We removed tokens which the annotators were unable to translate and those marked as sound effects (*e.g.*, ahhhhh, zzzzz, blech, hrmpf, etc.). Punctuation was removed from the remaining data, excluding that found in emoticons (which we treat as words) and apostrophes in common contractions and possessive forms. To facilitate character-level training, we insert a space between each character and replace any spaces with the underscore character.

Because training is done at the character-level, each hypothesis (w) for an abbreviation (a) is a sequence of characters, which may or may not be a valid word. To see why, examine the partial phrase-table shown in Figure 1[1]; using this table, "hab" could be translated to "hib", "habulary" or "habou" as well as the word "have". It is also very likely that a hypothesis will appear many times in the hypothesis list due to different segmentation (two characters may be generated by a single phrase mapping or by two different mappings, one for each character). We generate 20 *distinct* hypotheses for each token and then eliminate hypotheses that do not occur in the CMU Lexicon.

Until this point, we have only normalized a single abbreviation without context. In a realistic setting contextual information is available to help with translation. We thus introduce a second phase using a word-level LM to disambiguate hypotheses when context is available. This is the typical noisy channel model used for speech recognition or MT decoding and is comparable to equation 1.

Determining the standard English sentence, $W = w_1 w_2 ... w_n$, from a given informal sentence, $A = a_1 a_2 ... a_n$, can be formally described as:

$$\begin{aligned} \widehat{W} &= \arg\max P(W|A) \\ &= \arg\max P(W)P(A|W) \\ &\approx \arg\max \prod P(w_i|w_{i-n+1}...w_{i-1}) \\ &\quad \times \prod P(a_i|w_i) \\ &= \arg\max(\sum \log P(w_i|w_{i-n+1}...w_{i-1}) \\ &\quad + \sum \log P(a_i|w_i)) \end{aligned} \tag{2}$$

where the approximation is based on the assumption that each abbreviation depends only on the corresponding word (we are not considering one-to-many mappings in this study), and a word is dependent on its previous $(n-1)$ words. In other words, this probability is represented by a traditional *n*-gram LM.

The abbreviation score in Equation 2, $P(a_i|w_i)$, represents the likelihood that abbreviation a_i is derived from word w_i, and can be obtained from:

$$p(a_i|w_i) \propto \frac{p(w_i|a_i)}{p(w_i)} \tag{3}$$

where $p(w_i|a_i)$ is the abbreviation model (AM) score from the character-level MT system, and $p(w_i)$ is from the character LM we used in MT decoding. In this study, we just use the score from the MT system as the likelihood score, without dividing by the character-level LM contribution. This is equivalent to using both character- and a word-level LMs during decoding.

Equation 2 assumes that the AM and LM should be weighted equally, but in actuality one model may prove to be more helpful than the other. For this reason, we allow the terms from equation 2 to be weighted differently, yielding the final equation

$$\begin{aligned} \widehat{W} &= \arg\max(\alpha \sum \log P(w_i|w_{i-n+1}...w_{i-1}) \\ &\quad + \beta \sum \log P(a_i|w_i)) \end{aligned} \tag{4}$$

where α and β are determined empirically.

5 Experiment 1: Evaluation on Isolated Abbreviations

We use the Moses MT system (Koehn et al., 2007) in all our experiments, including Giza++ for alignment. The score assigned to a translation is derived from four models: the phrase translation table, the language model, the reordering model,

[1]Aside from the possible translations, the phrase table also shows five values for each word. They correspond to the inverse phrase translation probability $\phi(f|e)$, the inverse lexical weighting $lex(f|e)$, the direct phrase translation probability $\phi(e|f)$, the direct lexical weighting $lex(e|f)$ and the phrase penalty (always $exp(1) = 2.718$), where e and f are the English and foreign phrases, respectively. We do not currently make use of these values.

```
a b ||| _ a b ||| 0.714286 0.880834 0.018315 0.164514 2.718
a b ||| a _ b ||| 0.5 0.880834 0.010989 0.164514 2.718
a b ||| a b b ||| 0.181818 0.880834 0.00732601 0.905748 2.718
a b ||| a b o u ||| 0.00714286 0.880834 0.003663 0.00346958 2.718
a b ||| a b o ||| 0.00662252 0.880834 0.003663 0.0690538 2.718
a b ||| a b u l a r y ||| 1 0.440819 0.003663 4.55505e-08 2.718
a b ||| a b ||| 0.894737 0.880834 0.934066 0.934993 2.718
a b ||| a p ||| 0.00956938 0.000874125 0.00732601 0.000887968 2.718
a b ||| a v e ||| 0.00249377 0.000970303 0.003663 3.35128e-05 2.718
a b ||| b ||| 0.000325839 0.308415 0.003663 0.968721 2.718
a b ||| i b ||| 0.0238095 0.308415 0.003663 0.077216 2.718
```

Figure 1: An excerpt from a phrase table showing possible translations when the character sequence "ab" is found in a message.

and a word length penalty. We use Moses' default weights for these models, but plan to adjust the weights in the future to determine each model's contribution to a good translation. For each abbreviation we generate 20 distinct hypotheses. We finally remove non-word hypotheses using the CMU lexicon. Note that occasionally Moses generates a positive score (impossible for a real log-probability). When this occurs, we use the value -0.1 instead, indicating that this pair is very likely.

Note that our accuracy is bounded from above by the percentage of pairs in testing that appear in the 20-best hypotheses generated by Moses. For this reason, we first wish to find a setup to maximize the upper bound on our performance.

5.1 Experimental Variations

There are different factors that may affect the performance of our system for individual words: the character-level LM used for decoding, the way the models are trained, and the data used to train those models. We thus evaluate different experimental variations, described below.

Training/Testing Configuration

To test the effect of using more specific versus more general data when training the MT system, we used the following training sets:

1. **General Setup:** Training and testing are both done using all of the annotated data, including words that are already in standard form in the message.
2. **Single-word Abbreviation Setup:** In training, we remove those abbreviations corresponding to multiple words (for example, "brb" representing "be right back") and those words already in standard form. Again we test the system using all of the annotated data.
3. **Type-dependent Setup:** We train a separate model for each major abbreviation type (see Table 1) except insertions. The type of each abbreviation in the test set has been annotated so we use its respective model for testing. We assumed that knowledge about the abbreviation formation method would increase the model's ability to translate the abbreviation. Currently there is no known procedure for automatically determining the abbreviation type without having prior knowledge of its translation, so this setup is just for comparison purpose to evaluate how well the more general methods are performing.

Formation Method	Example
Deletions	ppl (people)
Substitutions	2nite (tonight)
Repetitions	yeeeeesssss (yes)
Swaps	tounge (tongue)
Insertions	borded (bored)

Table 1: Examples of major abbreviation types.

Language Model (LM) Order

We wanted to determine how the order of the character-level LM used by Moses affected performance. A smaller order model may not capture larger phrases or long distance dependencies, but the smaller model may generalize better due to the sparseness of many high-order phrases.

We explore different character LMs used in decoding to generate word hypotheses, including character-based 3-, 5-, and 7-gram LMs. These are trained on a subset of 16,543,813 messages from the Edinburgh Twitter corpus (Petrovic et al., 2010) containing no OOV words compared to an English dictionary. Each model is used in the above training/testing configurations.

	Order 3		Order 5		Order 7	
	Top-1	**Top-20**	**Top-1**	**Top-20**	**Top-1**	**Top-20**
All (14611)	59.30	81.32	62.62	81.49	63.30	81.95
Abbreviations (2842)	11.75	53.24	30.33	62.95	33.04	63.83
Standard Form (10629)	76.89	94.39	75.86	91.55	75.91	91.81
Deletions (1406)	8.68	52.63	31.58	62.87	34.71	63.51
Substitutions (620)	9.35	37.91	22.58	51.94	25.32	52.58
Repetitions (510)	23.14	81.37	32.55	79.02	34.90	79.80
Swaps (106)	33.02	70.75	69.81	86.79	66.04	90.57
Insertions (55)	0.00	27.27	14.55	49.09	20.00	56.36
Combination (140)	0.71	23.57	21.43	43.57	25.00	43.57

Table 2: Accuracy using the General setup (%).

	Order 3		Order 5		Order 7	
	Top-1	**Top-20**	**Top-1**	**Top-20**	**Top-1**	**Top-20**
All (14611)	75.97	91.23	75.75	92.50	75.57	92.53
Abbreviations (2842)	14.53	63.16	36.45	72.31	37.40	72.38
Standard Form (10629)	92.47	98.79	86.31	97.93	85.84	97.95
Deletions (1406)	11.38	64.15	39.90	74.04	40.61	74.11
Substitutions (620)	11.13	48.23	25.48	60.80	27.42	61.45
Repetitions (510)	28.24	84.31	37.45	82.75	38.24	82.55
Swaps (106)	35.85	91.51	76.42	96.23	73.58	95.28
Insertions (55)	0.00	36.36	18.18	54.55	20.00	58.18
Combination (140)	1.43	33.57	25.00	59.29	27.14	57.14

Table 3: Accuracy using the Single-word setup (%).

5.2 Results

The results from different training/testing configurations and LM orders are shown in Tables 2, 3 and 4. We use a cross-validation setup where we train the MT system using the annotations from four annotators and test on the data from the fifth. The mean score from the five runs is reported here. In all three tables, we show top-1 and top-20 accuracy. For top-1 accuracy, the system is considered correct if and only if the top hypothesis given by Moses exactly matches the standard English form. If the correct standard English form is listed anywhere in the 20 hypotheses generated by Moses, we count it correct in top-20 accuracy. The top 20-accuracy represents the possible gain we can achieve by re-ranking the lists using context.

We break down the results by abbreviation type to determine any performance difference. The top portion of Tables 2 and 3 shows results for all 14611 unique pairs in the annotated data, including words already in standard form and abbreviations corresponding to multiple words. The following two rows show results on abbreviations of single words and words already in standard form, respectively. The lower portion breaks down the results by abbreviation type. The rows listing specific types include those abbreviations using *only* that abbreviation method. The final row lists results for those abbreviations formed using multiple methods. The type-dependent setup (Table 4) reports results for each type when using the model trained for that specific type.[2]

Removing multi-word abbreviations and words already in standard form from training generally increases accuracy. The type-dependent model usually outperforms the single-word model, as expected. Comparing Tables 3 and 4 shows that performance decrease using the type-independent single-word model is usually small, helping avoid costly human annotation of types in the future. Contrary to expectations, the type-independent

[2]We do not train a separate model for the insertions due to the small number of abbreviations of that type in our data. Looking at those marked as insertions, most of them appear to be typographical errors caused by "fat fingering" a keypad (*e.g.* "jusyt" for "just"), although a few appear stylistic in nature, *e.g.*,"confrused" for "confused".

	Order 3		Order 5		Order 7	
	Top-1	**Top-20**	**Top-1**	**Top-20**	**Top-1**	**Top-20**
Deletions (1406)	11.45	64.08	38.98	73.33	40.11	73.47
Substitutions (620)	12.26	55.48	27.58	62.58	27.74	63.06
Repetitions (510)	28.63	84.51	39.02	83.33	39.61	83.53
Swaps (106)	35.85	97.17	77.36	97.17	77.36	97.17

Table 4: Accuracy using the type-dependent training and testing (%).

model outperformed the type-dependent model for the deletion type. Examining the abbreviations formed by combining at least two methods helps explain why this occurs. Of the 140 abbreviations combining abbreviation types, 102 contain at least one deletion. Adding these abbreviations in training yields additional examples for the model to learn from.

There is a large increase in accuracy between using a 3-gram versus a 5-gram model. However, the jump from 5-gram to 7-gram is much smaller, and in some cases it even decreases performance. We thus recommend using a 5-gram character model and training using only single-word abbreviations.

6 Experiment 2: Incorporation of Contextual Information

In many cases, there are many word hypotheses for a given abbreviation. Without contextual information the top hypothesis is static. In some cases the system can use contextual information to suggest the correct word even when it may not be the most likely choice using the AM alone. To evaluate the effect of context on normalization, we create a test set containing trigrams using one word of context on either side of each abbreviation. If an abbreviation falls at the beginning or end of a message we use <s> and </s> for the left or right context, respectively. We replace OOV context words with their annotated standard forms.

The top performing setup from Section 5 is used to build the AM, which is then combined with a bigram LM for decoding using equation 2. The LM uses Kneser-Ney smoothing and was trained on the same portion of the Edinburgh corpus as used for the character-level models. We do not include the <unk> token in the LM because its probability is quite high and preliminary tests (not included here) show a degradation in performance when it is utilized. Instead, we use a log probability of -99 for all unknown terms seen in testing.

LM-only	**Full**	**Partial**	**Section 5.2**[3]
13.5	69.7	69.3	36.45

Table 5: Accuracy on tests using context (%).

We performed three types of tests in order to determine the best setup for decoding using context.

1. **LM-only.** This system serves as a baseline for comparison. We eliminate the abbreviation model completely and use only the LM for decoding. In this case, all unigrams in the LM are considered to be possible candidates for each abbreviation.

2. **Full.** We use the LM in combination with the abbreviation model using equation 2 with $\alpha = 1.0$ and $\beta = 0.1$ (found empirically).

3. **Partial.** Due to the small weight found for the abbreviation model in the previous setup, we wish to show that the abbreviation score is helpful for decoding. We use the abbreviation model to generate the candidate words for each abbreviation, but we do not use the abbreviation score in decoding.[4]

We again performed cross validation by first training an AM using the data from four of the annotators and then testing on the trigrams formed from the final annotator's data. In total there were 4415 context trigrams used in testing. This is larger than the 2842 abbreviations shown in the previous section because the previous tests used unique abbreviations, which may in reality correspond to multiple words and unique contexts. Table 5 shows the normalization results. As a basis of comparison, the state-of-the-art Jazzy spell-

[3]Using single-word training and 5-gram character LM.

[4]We eliminate the AM term from equation 4, yielding $\widehat{W} = \arg\max_{w_i \in S}(\sum \log P(w_i|w_{i-n+1}...w_{i-1})$ where the candidate set S is generated by the MT system. Equivalently, we set $\beta = 0$.

	Top-1	Top-3	Top-10	Top-20
Jazzy (Idzelis, 2005)	49.86	53.13	54.78	55.44
(Choudhury et al., 2007)	59.9	–	84.3	88.7
(Cook and Stevenson, 2009)	59.4	–	83.8	87.8
(Liu et al., 2011)[a]	58.09	70.96	–	–
(Liu et al., 2011)[b]	62.05	75.91	–	–
This Work	60.39	74.58	75.57	75.57

Table 6: System comparison on the 303-term test set from (Choudhury et al., 2007).

[a] LetterTran.
[b] LetterTran + Jazzy.

checker (Idzelis, 2005) achieves 38% accuracy on this dataset.

It is clear that the LM-only setup is not sufficient for this task and that our AM is necessary. The **Full** and **Partial** setups perform similarly; while the **Full** model consistently performs slightly better for all annotators, it is probably not a significant difference. We hope that by optimizing Moses' parameters we can obtain higher performance from the AM.

Compared to the 1-best results from Section 5.2 we see that incorporating contextual information yields significant gain. Our performance is bounded from above by the percentage of correct solutions that appear in our candidate lists. For the contextual task, 77.2% of the correct words appear in the lookup lists after the AM is applied, showing that we can still improve the decoding process. Fully 91.7% of the correct words appear in the LM, meaning that the AM is still eliminating many correct answers from consideration. We are investigating methods to address these shortcomings for our future work.

7 Comparison to Past Work

Although there has been very little work on this task until quite recently, multiple studies have been done using the small 303-term dataset first used by (Choudhury et al., 2007). For this reason, we run our system on this small data set. We also use the state-of-the-art Jazzy spell-checker (Idzelis, 2005) as a baseline.

System comparisons are shown in Table 6. The results shown for other systems (except Jazzy) are those reported in their respective papers; we did not re-implement their systems. Our system performs comparably to the work of (Liu et al., 2011) on this dataset. Although we outperform both (Choudhury et al., 2007) and (Cook and Stevenson, 2009) in top-1, they outperform our system at top-10 and top-20. With better re-ranking, their systems have the potential to outperform ours. One of our goals is thus to obtain better coverage.

8 Conclusions and Future Work

In this paper we presented a two-phase approach using character-level machine translation for informal text normalization. When combined with contextual information at the word level, our results are state-of-the art, but there is still some room for improvement.

Although they are far more common, abbreviations formed by deletion and substitution method have worse performance compared to those formed by repetition or swap. It may be useful to create systems specific to those formation types, such as the deletion system from (Pennell and Liu, 2010). Provided that these systems are not trained using the MT approach, it is likely that they contain complementary (rather than redundant) information. Combining hypotheses from multiple sources may increase performance on these abbreviation types.

Although we incorporate contextual information in Section 6, the setup is not entirely realistic. We test only on abbreviations, assuming that we know which words are abbreviations and which are not. A simple heuristic of only applying our system to OOV words (compared to a dictionary) may not perform well. Proper nouns are often OOV and should not be expanded, but we *would* like to expand "off" ("office") or "cat" ("category"), which are also dictionary words. We also assume there are not other abbreviations to be expanded in the context. In reality, many users write highly abbreviated messages, posing greater difficulties for sentence-level decoding. We will address sentence-level decoding in future work.

References

AiTi Aw, Min Zhang, Juan Xian, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *COLING/ACL*, pages 33–40, Sydney, Australia.

Srinivas Bangalore, Vanessa Murdock, and Giuseppe Riccardi. 2002. Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system. In *19th International Conference on Computational Linguistics*, pages 1–7, Taipei, Taiwan.

R. Beaufort, S. Roekhaut, L.A. Cougnon, and C. Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779, Uppsala, Sweden. Association for Computational Linguistics.

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*, 10:157–174.

Danish Contractor, Tanveer A. Faruquie, and L. Venkata Subramaniam. 2010. Unsupervised cleansing of noisy text. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 189–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

B. Han and T. 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. Association for Computational Linguistics*.

Mindaugas Idzelis. 2005. Jazzy: The java open source spell checker.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing SMS: Are two metaphors better than one? In *22nd International Conference on Computational Linguistics*, pages 441–448, Manchester, UK.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karen Kukich. 1992. Technique for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

F. Liu, F. Weng, B. Wang, and Y. Liu. 2011. Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. *Proc. of ACL-HLT*.

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

Deana Pennell and Yang Liu. 2011. Toward text message normalization: Modeling abbreviation generation. In *ICASSP*, Prague, Czech Republic, May.

Saša Petrovic, Miles Osborne, and Victor Lavrenko. 2010. The edinburgh twitter corpus. In *NAACL-HLT Workshop on Computational Linguistics in a World of Social Media*, pages 25–26, Los Angeles, California, USA.

Carlos A. Henríquez Q. and Adolfo Hernández H. 2009. A ngram-based statistical machine translation approach for text normalization on chat-speak style communications. In *Content Analysis for the Web 2.0 (CAW2.0 2009)*, Madrid, Spain.

K. Raghunathan and S. Krawczyk. 2009. Cs224n: Investigating sms text normalization using statistical machine translation.

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

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

C. Whitelaw, B. Hutchinson, G.Y. Chung, and G. Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 890–899. Association for Computational Linguistics.