CS224N: Investigating SMS Text Normalization using Statistical Machine Translation

Karthik Raghunathan, Stefan Krawczyk

Department of Computer Science, Stanford University
[rkarthik,stefank]@cs.stanford.edu

Abstract

In this project we explore two approaches to SMS text normalization. First we try a dictionary substitution approach used by most websites that provide such a service, and then modify it with our extension. This is followed by a statistical machine translation (MT) approach using off the shelf MT tools. We evaluate the performance of our system on three test sets from different sources and discuss the shortcomings of our system and results.

Index Terms: sms, english, machine translation, multiple domains

1. Statement

Stefan wrote tools to help with the data preparation and performed the dictionary substitution evaluations. We both more or less equally shared the tasks of annotating the data. Karthik took care of setting up the MT pipeline because he had access to a system and ran the machine translation scripts. We both analyzed the data and equally wrote up the report.

2. Introduction

There are many sites [1, 2, 3] on the internet that offer SMS English to English translation services. However the technology behind these sites is simple and uses straight dictionary subsitution, with no language model or any other approach to help them disambiguate between possible word substitutions.

A reasonable alternative to this would be taking a noisy channel approach to modeling the translation from SMS English to English. The English *signal* is sent across a noisy channel, as *an SMS*, which we then try to recover using a language and translation model. This should give us the ability to disambiguate between ambiguous expansions of an SMS message. E.g. in *"do u noe how 2?"* we need to disambiguate what each of the tokens means, specifically the *2*; it could map to "to", "two" or "too". With machine translation we hope to be able to produce a better alternative for SMS English to English translation, known as SMS text normalization in the natural language processing community (NLP). For this task we use Moses, an open source toolkit for statistical machine translation.

Some could argue that we could take a spelling correction approach to the problem, but such a system do not usually consider the context and cannot handle forms that are really two words, for example *"ru"* (are you).

The main contribution of this work is to present, evaluate and contrast the performance of two SMS normalization approaches on different sets of SMS messages from different countries, using off the shelf machine translation components.

The rest of the paper is organized as follows. Section 3 and 4 discusses prior work and corpus preparation. Section 5 explains our approaches and section 6 their evaluations. Section 7 discusses short comings of our results and section 8 concludes.

3. Prior Work

There is some prior work in using machine translation (MT) using SMS text. Aw et al [4] use MT in the context of normalizing SMS messages before they are translated into Chinese. The first employ a dictionary subsitution approach using frequencies, along with a bigram language model and compare that to using machine translation (phrase based machine translation). They show that using MT boosts BLEU scores for SMS English to English translation. Their data consisted of 5000 SMS messages from the NUS corpus[10], which we are also using so we should be able to get some comparable results. The corpus does not come with translated SMS messages, so they produced the parallel English text themselves. They did not mention how they handled smiley faces ":)" or punctuation in the messages when translating. They do not use any standard tools for the machine translation task and hand build these themselves, but they train their n-gram lanauge model on the English Gigaword corpus from LDC. They do not test on any SMS messages that are not from the NUS corpus and use the trigram BLEU score for their evaluations.

Choudhury et al.[5] investigate the structure of SMS langauge, using machine learning techniques to produce a Hidden Markov Model (HMM) of SMS words. They focus on word level decoding and not full message translation. E.g. trying to decode SMS words such as 2day (today), fne (phone) and dem (them). The HMM model they train allows them to decode unseen words in which they are relatively successfully.

Kobus et al [6] follows on from Aw and takes inspiration from Choudry by combining machine translation with an automated speech recoginition approach using HMMs. They reason that with the Aw approach *"lexical creatitvity attested in SMS messages can hardly be captured in a static phrase table ... normalized phrases are learned by rote, rather than modeled."* so they take the speech decoding inspired process of modelling SMS words as an alphabetic/syllabic approximation of the phonetic form and mapping it to a phone lattice which they then decode.

They use a French corpus of 25000 SMS messages to train on and another 11700 to tune their parameters, which were lowercased and stripped of punctuation signs. The machine translation system they uses is Giza++[8] along with Moses [9], which we will also be using in our system, and the ASR-like system that they built. They show that separately the two systems perform better in different respects, and show when combined they decrease the word error rate by 1.5 times, increasing BLEU scores, but show that they still get at least one error on 60% of messages.

4. Corpus Preparation

We decided to use the NUS corpus [10]as our main training set and the smaller HKU[12] and treasure my texts (TMT)[5] corpuses that we found as testing sets. All but the TMT corpus had parallel translated text, so we had to tokenize and annotate the data. We set about one weekend doing this, so we wrote some tools to help us with this task. We created dictionaries and parallel text of about 3000 NUS SMS messages, 408 HKU message and 427 TMT messages. We decided to strip smileys, keep punctuation, leave words that we thought were names, or did not know otherwise (for example the this was from Singapore so some abbreviations we did not know). The corpuses were already lowercased, so we did not have to do that.

To compare to the corpuses of the previous work, we are drawing from the same corpus that Aw et al. used. The parallel text they produced, replaced out of vocabulary words and non-standard SMS linoges, removed slang (and local colloquialisms) and inserted auxiliary or copula verbs and subject pronouns. In contrast, we produced translated parallel text that replaced slang and left local colloquialism in place. We feel that this is justifiable since people do not usually waste space in an SMS message, since it is character limited, so if they did want to convey something in local terminology it would not be fair to take this away in the English translation. For instance the SMS is a written conversation, so if it was a spoken conversation, that colloquialism would still have been used and if someone was transcribing it, it would have been left as is. But to be comparable with Aw et al.'s [4] results we also produced a test set where we deleted most of the top ten deletions that they showed in the translated text.

In Choudhury et al.[5] they used the IITKGP corpus, which we are also using. We are largely using their translations but where they placed unknown tokens for words they did not know how to translate ("??") and where they replaced names with "<x>", we decided to keep the original words in the translation to keep it in line with how we translated the other corpuses.

In Kobus et al.[6] they stripped punctuation completely. We feel that this would not be a realistic way to perform SMS normalization so we left in punctuation. But we did produce versions of the data where all punctuation was replaced by a "<PUNCT>" token to see how much leaving punctuation in affected our performance.

5. Two approaches to SMS Text Normalization

We employed two approaches for SMS text normalization. The first was dictionary substitution, the second a statistical machine translation approach. We evaluate their performance on three test sets, the first set contains 477 unseen SMSes from the NUS corpus. The second test set is 408 SMSes from a research project at the University of Hong Kong [12]. Lastly, we have a test set of 427 SMSes collected from http://www.treasuremytext.com (TMT) by Choudhury et al [5]. The first test set corresponds to held out data from the same corpus, the second, to data from a different corpus but of a similar domain and the third, to data from an entirely different domain. We feel that this breadth of testing will enable us to comprehensively examine our performance.

5.1. Search and Replace

We built three dictionaries for this task. One was built from our training data comprised of words that were abbreviated, shortened or were slang/SMS message colloquialisms. This comprised of 560 unique SMS token entries and we refer to it as the native dictionary. The other two were built from internet sources [1, 2, 3]. One was small (224 SMS entries) and comprised of only the first two sources, the other comprised of all the sources (1611 entries) and we refer to them as web-small and web-big.

For each dictionary, we ran it on each test set, substituting each word that matched in the dictionary. If there was more than one entry a random one was chosen. If there was no match the word was translated as is.

In addition to substituting randomly for each test set, we also made an empirically substituted set for the training data dictionary only, as we were able to get the frequencies of substitution from the training data.

5.2. Statistical Machine Translation

We use standard state-of-the-art open source tools to train a statistical machine translation system following the pipeline outlined in http://www.statmt.org/moses_steps.html. We use the SRI language modeling toolkit [7] for training a language model for normalized English and GIZA++ [8] for computing word alignments between SMS tokens and their normalized English counterparts. Moses [9] is then used for training a phrase-based statistical machine translation system capable of decoding from SMS-speak to English. We train on 2,000 SMSes from an SMS corpus collected by the National University of Singapore (NUS)[10] and evaluate on the test sets described above.

We also create two development sets, one containing 539 SMSes from the NUS corpus and another with 427 SMSes from Choudhury et al.'s TMT corpus and use these for parameter tuning using MERT [11]. The intuition behind having two different development sets was to have one tuning set for each test domain. As expected, the system performs best on the NUS and HKU test sets when the parameters are tuned only using the NUS development set. Similarly, it performs best on the TMT test set when MERT is run only on the TMT development set. We evaluate our system both using the commonly used BLEU metric [13] and Word Error Rate (WER).

6. Evaluation

6.1. Search and Replace Evaluation

Table 1: *Performance of Search and Replace*

System	NUS Test Set	HKU Test Set	TMT Test Set
	BLEU	BLEU	BLEU
Baseline	0.562	0.7025	0.4009
WEB-SMALL	0.6488	0.7218	0.4933
WEB-BIG	0.5573	0.8128	0.4311
NATIVE-EMP	0.8941	0.8770	0.5873
NATIVE-RANDOM	0.7945	0.7940	0.5335

We notice that in table 1 the bigger internet dictionary performs worse. This is due to the fact that there are more translations per entry, so we have a harder time choosing the right translation.

Interestingly the dictionary compiled from the training data performs better than any of the internet dictionaries, even with random subsitution. This suggests that currently available dictionaries do not have some core entries in their dicitionaries.

It is also interesting to note that the dicitonary with the empirical frequencies performs markedly better than any of the other dictionaries across all the sets. This suggests that even across varying geographical regions, there seems to be at least a core set of abbreviations that seem to be used more often than not. This suggests that our statistical MT system should be able to, given enough training data, be usable across varying SMS linguistic regions.

6.2. MT Performance Evaluation

We trained our SMT system using the standard Moses pipeline with all the default settings - a trigram language model, a maximum phrase length of 7 and a distortion limit of 6. These settings work well for conventional machine translation, where one translates between two entirely different languages. Because of the pecularity of our translation task we tried varying these settings and evaluating our systems on the same test sets to observe the change in performance. We also conducted experiments by varying corpus sizes to get a learning curve for our system.

6.2.1. Training corpus size

We first started out by measuring the performance of our system as a function of corpus size. As can be seen from the learning curve in figure 2, the BLEU scores computed for each of the test sets have been constantly increasing (and the WER decreasing accordingly) with increase in the amount of training data. This shows that our system is still data-hungry and we can still hope to get more improvements with additional data.

The out of vocabulary (OOV) rate (i.e. the percentage of tokens in the test set that are not seen during training) for the various experiments are tabulated in Table 2. As expected, the OOV rate is much higher for an out-of-domain test set like the TMT one as compared to NUS and HKU. In the absense of more training data, we decided to experiment by adding our SMS-lingo dictionary (which had been used for the SR-WEB approach) as additional parallel sentence pairs to the training corpus. This approach however ended up hurting our performance on the NUS and HKU test sets and only improved on TMT by 0.0085. Error analysis revealed that the dictionary contained a lot of noisy pairs like "*picnic* → problem in chair, not in computer", "*o* → opponent,", "*bak* → back at keyboard", etc. due to which the system was generating non-sensical translations like "are you man *opponent* woman ?" and "are we going for *problem in chair, not in computer* today ?". Other translations, though not corrupted to this extent, were still making mistakes by omitting necessary words or inserting unnecessary words because of the noisy variations introduced into the phrase table by the new alignments learned from the dictionary data.

Also, the reduction in the OOV rate is negligible for the NUS and HKU test sets even after adding a big dictionary of 1,795 SMS slangs which clearly shows the inefficacy of using such a standard dictionary. Most of the top OOV words turned out be either normal English words or non-standard abbreviations which were not present in the lingo dictionary either. The TMT set is the only set that got a small reduction of 2.7% in OOV rate after the dictionary addition. Overall, it can be concluded from these experiments that there is definitely scope for improvement using more data, but this has to come in the form of real-world SMS data rather than hand crafted dictionaries.

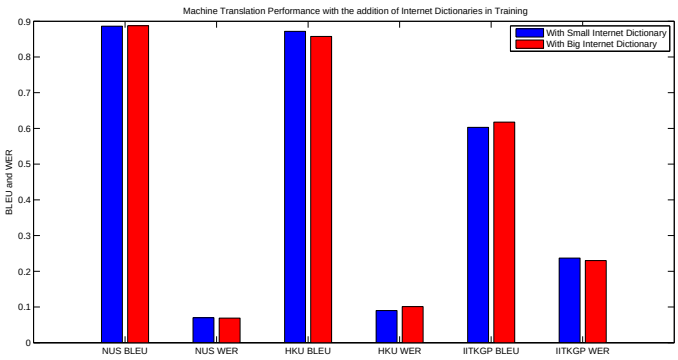

Figure 1: *MT performance with the addition of internet dictionaries*

Table 2: *OOV Rate (in %) on the different test sets for our experiments*

Training Corpus	NUS	HKU	TMT
500 NUS	19.01	24.25	35.49
1000 NUS	14.59	18.98	32.09
1500 NUS	10.63	14.06	26.84
1930 NUS	7.77	11.78	21.49
1930 NUS + 224 SMS-LINGO	7.68	11.71	20.64
1930 NUS + 1795 SMS-LINGO	7.55	11.08	18.79

6.2.2. Language model order

We tried varying the order of the n-grams being used in our Language model and running the same experiments again. As can be seen from Figure 3, there is hardly much variation (at most ≈ 0.015) among the systems with different LM orders. The BLEU scores peak while using the trigram model and gradually decrease on either side in the graph. The results produced by these systems are highly similar for most of the cases. The difference between them is mainly due to noise in the training corpus caused by human error during translation of the SMSes into their normalized references. Suppose the reference set

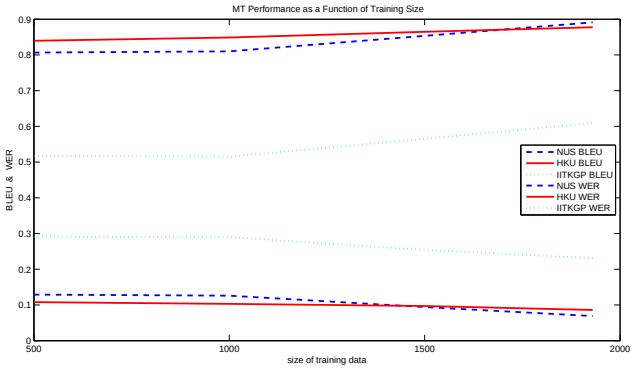

Figure 2: *MT performance varying the training set size*

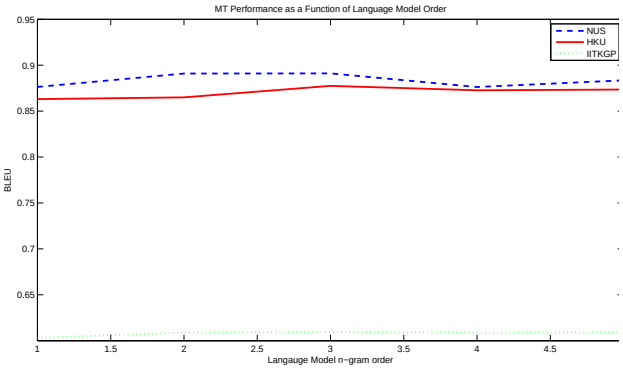

Figure 3: *MT performance varying language model order*

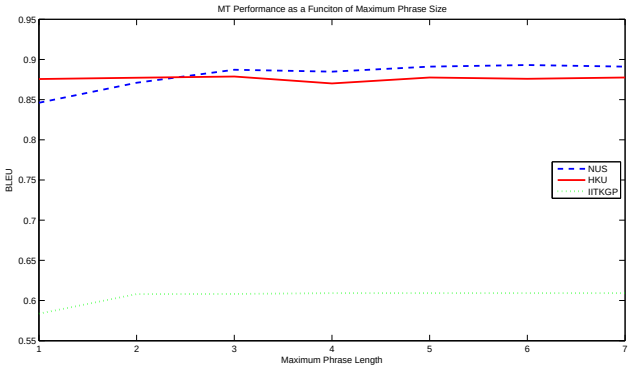

Figure 4: *MT performance varying phrase size*

in our training corpus had contained "want" and "*wanna*" (error made by human being who forgot to translate "*wanna*" to "want") equal number of times, then the unigram model would give equal probability to both where as a higher order LM might be able to overcome this by disambiguating using context (since the probability of higher n-grams containing "*wanna*" need not be the same as those containing "want"). On the other hand, if we had just one short SMS like "wanna go there" which had been retained as it is on the reference side, then the five-gram "<S>wanna go there </S>" gets a very high probability in the LM and biases the MT to choose this noisy phrase. Thus, going for both very high orders or very low orders seems to be less robust to noise in the training data. But as noted already, these are very rare cases and can be easily solved by training our LM on not just the target side of the MT parallel data, but a lot more English text from other reliable monolingual English corpora. This would have given us a more effective MT system as a whole and also a better comparison between the various LMs.

6.2.3. Maximum phrase length

We conducted another set of experiments by varying the maximum allowable phrase length in the phrase-table learned by Moses. Setting this length to 1 is the same as doing a word by word translation without using any context information. This works pretty well for the easy to translate HKU test set but not so well for the other two harder test sets. Our experiments show that context definitely helps and a maximum phrase length of 5 seems to be optimum for all the test sets. This is an interesting result to note since a phrase-table containing only phrases upto five tokens in length would be considerably smaller than the default phrase tables created by Moses (maximum length of seven). Though memory is not that big a concern at present with a corpus as small as ours, it would definitely be a criterion if we were to scale the system to handle much more traning data or to make it run on a hand-held device having much less memory than a PC.

6.2.4. Distortion limit

The distortion limit is a limit on the amount of reordering allowed by the decoder during translation. A limit of zero would mean that the decoder does a "monotone" translation, i.e. translating the whole sentence word by word in order. A limit of one means that reordering upto distance one is allowed and so on. Irrespective of the distortion limit set by us, the system always produced the same output because the system had learned distortion weights during MERT in a way such that it discouraged any kind of reordering. Thus, even on being giving an allowance to produce a reordered translation (e.g. with higher distortion limits), the system still chose to do a monotone translation. This is understable since we are not actually translating between two completely different languages with different word orders but just from English to English. Thus, the task of the system is only to translate each word in the SMS, one by one in order without caring about any possible reordering.

6.3. Comparison between approaches

The results of our best performing systems for each approach are presented in Table 3 The first row contains the baseline BLEU scores computed using the un-normalized test SMSes and their corresponding English references. This gives us an idea of the similarity between the SMSes and their translated normalized counterparts to begin with. For instance, the BLEU score of 0.7025 for the HKU test set shows that it is a relatively easy set to normalize since it shares a lot of similarity with plain English. On the other hand, the TMT data set with a BLEU of 0.4009 is a highly noisy one containing lot of abbreviations, mis-spellings and the kind of lingo representative of SMS-speak. The NUS test set lies inbetween these two extremes. The scores for our various systems are tabulated in the following rows.

Table 3: *Performance of our Text Normalization Systems*

System	NUS Test Set		HKU Test Set		TMT Test Set	
	BLEU	WER	BLEU	WER	BLEU	WER
Baseline	0.562	N/A	0.7025	N/A	0.4009	N/A
SR-WEB	0.6488	-	0.8128	-	0.4933	-
SR-NATIVE	0.8941	-	0.877	-	0.5873	-
SMT	0.8931	6.6%	0.8775	8.6%	0.6177	23%

Our SMT system is as good as the Search & Replace system using the native dictionary on the NUS and HKU test sets and does better by .0304 on the tougher TMT test set. Both these approaches do significantly better than just using a static dictionary of SMS lingo and using that in a substitutional approach like most websites (http://www.lingo2word.com/, http://www.transl8it.com/) seem to do. Building a native

dictionary for the given corpus was possible since we wrote a program to automatically generate the dictionary as we were manually translating the SMSes to create the references. But this is not possible everytime one acquires new parallel training data. On the other hand, it is more straight forward to add new parallel data to an SMT system to improve its performance. Also, it is our intuition that a system based on statistical principles would be able to generalize better to newer domains than a naive pattern match and replace algorithm. This is also supported by the difference in performance of the two systems on the TMT test set. We therefore mainly focus our discussion on analyzing our SMT system.

Here is an example sentence from the NUS test set:
Example 1:
yeah ... btw i'll b buyin e present on fri ... so if u wanna chip in den dun need to buy .... cya on sat den !

The reference translation for this SMS could be "yes ... by the way i will be buying the present on friday ... so if you want to chip in then do not need to buy ... see you on saturday then !". Or we could alternately have "yeah" and "don't" in place of "yes" and "do not" respectively if we want to be slightly informal. Thus, the correct set of tranformations required are

yeah → yes / yeah
btw → by the way
i'll → i will
b → be
buyin → buying
e → the
fri → friday
u → you
wanna → want to
den → then
dun → do not / don't
cya → see ya
sat → saturday

These are the translations produced for this SMS by the two websites and our system:

tranl8it: yeah ... by the way ill be buyin e present on fri ... so if you want to chip in then dont need to buy .... see ya on Saturday then !

lingo2word: yes sure ... between I will be buying electronic present on Friday ... so if you want to chip in den done need to buy .... see ya on sat den !

SMT: yeah ... by the way i will be buying the present on friday ... so if you want to chip in then do not need to buy .... see you on saturday then !

Both transl8it and lingo2word only manage to get few of the transformations right. transl8it fails to normalize the tokens *i'll*, *buying*, *e*, and *fri*. The lingo2word system on the other hand fails to normalize *den* and *sat*, and incorrectly translates *btw* → *between*, *e* → *electronic* and *dun* → *done*.

Here are a few other examples:
Example 2:
Original SMS: no lect ah ? i am free from 12 to 2pm .

tranl8it: know lect ah question question i am free from dozen to 2pm .

lingo2word: no lecturer ah ? I am free from dozen to two o'clock in the afternoon .

SMT: no lecture ah ? i am free from 12 to 2pm .

Example 3:
Original SMS: u all at serangoon garden oredi ?

tranl8it: you all at serangoon garden oredi question

lingo2word: you all at serangoon the farm oredi ?

SMT: you all at serangoon garden already ?

Example 4:
Original SMS: done w tt ages ago .

tranl8it: done with trailer trash ages ago .

lingo2word: done with trailer trash ages ago .

SMT: done with that ages ago .

These examples highlight quite a few significant ways in which our SMT system performs better than the other systems:

- One of the most successful ideas in machine translation has been to use a phrase-based approach as opposed to a word-based one, since it helps in disambiguating between the possible translation candidates for a given source language word using context information. However, in our particular problem almost all the SMS words can be mapped to unique English counterparts and the need to disambiguate using context seldom arises. But situations where disambiguation is required, though uncommon, arise nevertheless. Consider the case of the SMS token "*2*", which can be translated as "two", "too" or "to"; or "*fr*", which can be "for" or "from", depending on the context. In the first example above, "dun" could have been translated as "done" or "do not". lingo2word chooses "done" but our system, being a phrase-based one knows that "dun" is followed by the verb "need" and hence is more likely to be "do not" instead of "done". Similarly in example 2, it does not sound natural to translate "*12*" into "dozen" in that particular context. Had the text been "*12 eggs*" or "*12 bananas*", etc., the conversion to "dozen" would have justified but not in this case where the context is just a description of a time period. This is where having a context sensitive statistical system (like phrase-based SMT) can help instead of having one single rule "*12* → *dozen*" that fires all the time (the approach that seems to have been taken by the other two systems).
- SMS-speak is a language whose vocabulary keeps changing dynamically and is also highly dependent on the particular region or population using it. Each community seems to have its own "dialect" of SMS-speak, which explains why a standard SMS lingo dictionary fails to capture all possible variations of the language. For instance, these dictionaries (www.example1.com, www.example2.com) contain common transformations like "*u* → you", "*wanna* → want to", "*w* → with", etc.,

which are universal across all SMS-ing communities. But these may not be able to capture local community-specific usage of words like *oredi* for "already", *tt* for "that", *e* for "the", etc. On the other hand, a statistical system trained on a actual real-world corpus of SMSes learns all these mappings and hence performs better. In example 3, both of the dictionary based systems failed to translate *oredi* and in example 4, both mistranslated *tt* as "trailer trash". In example 1, tranl8it left *e* as it is, where as lingo2word translated it to "electronic" using the more common expansion of an "e-" prefix.

- Some other instances where a standard lingo dictionary fails are when the words in the SMS are mis-spelt or are intentional variations of standard abbreviations. For instance, the SMS "*hey pple ... happy new yr*" is translated by transl8it into "hey pple ... happy new year", because it would have only had the standard mapping of "*ppl* → people" in its dictionary. It is hence unable to handle the out of vocabulary word "*pple*", which is just a slight variation of "*ppl*". transl8it also faces similar problems with words like "*wannna*" (mis-spelling of *wanna* for "want to"), "*2morrow*" (alternately used instead of the more common *2mro* for "tomorrow"), etc. This is another scenario where our SMT system excels because it has been trained on an actual real-world corpus, enabling it to statistically learn such kinds of noisy variations.
- Yet another problem faced by dictionary based search and replace systems is when a proper English word itself can be an abbreviation in SMS-speak for some other word in normalized English. For instance, in few of the SMSes, the English word "*wear*" is at times used as a short-form for "where" in sentences like "*So wear do v meet ?*". *Wear*, being a proper English word does not feature as an SMS-lingo word in the dictionaries used by transl8it or lingo2word. These systems thus choose to keep "*wear*" intact without attempting to normalize it. A phrase-based MT system on the other hand can deduce from context that *wear* here means "where" and not the actual English verb that appears on the surface. Another example is that of "*sat*" which is an English word in itself (past tense of the verb "sit") but can also be used as a short form for "Saturday". As evident from example 1 above, lingo2word does not handle this case while our system does.
- Lastly, an advantage of using a system trained on parallel data which was manually translated by human beings is that such a system does not go into an overkill mode. The goal of our system is to just produce sensible and readable English, that can be used for other systems like Speech Synthesis, Information Retrieval, etc. and not necessarily expand every possible acronym or abbreviation that occurs in the text. Because of using hard-coded dictionary rules which are forced to fire in every situation, these websites (especially, lingo2word) at times end up normalizing more than necessary. Thus, even though our system chooses to retain "*e-mails*" as "e-mails" and "*sms*" as "sms", lingo2word expands them to "electronic mail" and "short message service". Especially when the message is of the form "*sms me once you reach home*", a translation like "short message service me once you reach home" sounds quite unnatural. It is thus beneficial to make the system learn from manually annotated corpora since human annotators only create translations that sound natural to them. Examples like the one just cited above, though weird to hear can eventually be understood. However, over-normalization often leads to totally non-sensical translations. For instance, transl8it converts the SMS, "*if u wan u have to send yr resume*" into "if you wide area network you have to send your resume", by mistaking *wan* (which is just "want" in SMS-speak) to be an abbreviation and expanding it incorrectly. Our SMT system is less prone to these kind of errors.

6.4. Comparison to Previous Work

To compare to Aw et al., we tested on a set with deletions similar to the ones they were deleting. We also made the dictionary map the deletions to the empty string so we could compare the performance with that too, as without it the performance would obviously be worse.

Table 4: *Comparison between Aw et al and our System*

System	NUS Test Set
	BLEU
Our Baseline	0.5465
Aw Baseline	0.5784
SR-NATIVE-EMP	0.8900
SR-NATIVE-RANDOM	0.7893
Aw Dictionary	0.6958
Our SMT	0.8611
Aw SMT	0.8070

We see we were able to get better results, note we used BLEU-4 scores, and not BLEU-3 as Aw et al. used. Interestingly with the deletions our MT system performs worse than the empirical native dictionary, suggesting that our langauge model cannot quite capture the deletions or the statistical frequencies of the data as well as it should.

To see how punctuation affects our MT system, as Kobus et al. did not have any punctuation in their system, we set punctuation (except exclamations and question marks) to be a single token. From our results the baselines all improved, but our MT system scores did not increase by as much as the baselines did. We got our best score on the NUS and HKU test sets getting .9029 and .9120 BLEU scores respectively.

7. Discussion

There are some short-comings of our work which can be fixed without a lot of effort. Firstly, we can try to use a better language model trained on a lot more English data, e.g. the Gigaword corpus distributed by the LDC. Secondly, we are right now using an SMS-aligned parallel corpora instead of a sentence-aligned one. It would be worth seeing how splitting multi-line SMSes into constituent sentences and having a parallel corpus aligned at the sentence level (as is the norm in MT) changes the performance of our system. Lastly, we could try to do the manual translation of SMSes into references (for the training data) in a more principled way with more than one annotator and using inter-annotator agreement as an indicator. This would ensure a more cleaner parallel training corpus to begin with and do away with a lot of noise that is introduced in the training set when different parts of the corpus have been prepared by different individuals.

Some of the other problems require a lot more contemplating and there is no one way to go about solving them. People mix and match shortenings, abbreviations and slang in their SMS messages. It is conceivable that this causes the phrase table to have rather sparse data. We feel this is part of the reason why disambiguation did not occur all the time correctly in our system. For instance, while trying to translate the "*2*'s" in *"a gud day 2 you 2"*, if the system has not seen enough examples in its training data where "*2*" is translated as "too", the high probability of "*2* → to" will cause the system to translate both the "*2's*" as "to". Even if we were to use context, it might turn out to be the case that the system saw examples of "*to u 2* → to you too" or "*2 u 2* → to you too", but never saw "*2 you 2* → to you too". One way of solving this problem is to have lots more data. Real MT systems have training data in the order of millions of sentence pairs which allows them to statistically capture such kinds of variation so as to help them during decoding time. But in the absence of such data for the SMS domain, one possibility is to articifically create more training data. E.g. one could take a sentence, keep the ambiguous words fixed and then just iterate over possibilites for the unambiguous tokens. So when we encounter a sentence like *"a gud day 2 you 2"*, we heuristically generate all possibilites like *"a gud day 2 u 2"*, *"a good day 2 u 2"*, *"a g'day 2 you 2"*, etc and add it to the training corpus, aligning each of these with the same reference translation.

Also, in our current work we tried sticking as much as possible to a standard MT pipeline and only varied those settings which could be provided as external configurable options to the tools that we used (e.g. order for the LM toolkit, maximum phrase length for Moses, etc.). We took this approach since we wanted to evaluate the efficacy of standard off-the-shelf open source MT tools for the SMT-to-English translation task. We could have been more flexible in being cleverer with features or coming up with other helpful hacks had we been building the actual MT system ourselves. One possibility would be to use an exhaustive English dictionary, check for words in the SMS that exist in this dictionary and just let them fall through to the target side without subjecting them to MT. Of course there will be exceptions (discussed in the evaluation section) which would have to be handled. We may also be able to get leverage by using a pronunciation dictionary like CMUDICT or CELEX and use that to do some grapheme-to-phoneme conversion and build a similar system to Kobus et al.[6]. Alternatively we could integrate Choudhry et al's [5] word model into the system for handling OOV words.

8. Conclusions

We show that using a dictionary with an appropriate coverage of entries and frequency usage produces SMS text normalization results better than current websites, however ambiguities are still a problem. For using such an approach in the real world, we argue about the infeasibility of acquiring such a dictionary every time and also show that its performance on an out of domain corpus is not quite good. This prompts us to use statistical machine translation as a good solution to the problem.

Using an SMT system built from off-the-shelf components, we experiment and evaluate language model, phrase length, training size and distortion limit settings, discussing the effects of each on performance. We show that on average, SMT outperforms search and replace, most notably when used on an out of domain test set. However the our MT performance is limited by the size of our training data and is also much better when the test data shares its domain with the training corpus.

We compare approaches with previous work, suggesting that our results are definitely competitive, but with the disclaimer that different approaches taken make direct comparison difficult. We finish with a discussion about the shortcomings of our system and possible improvements in the future to make it better.

9. References

[1] http://transl8it.com/

[2] http://www.lingo2word.com/translate.php

[3] http://www.dtxtrapp.com/

[4] Aw, AiTi and Zhang, Min and Xiao, Juan and Su, Jian, "A phrase-based statistical model for SMS text normalization", Proceedings of the COLING/ACL on Main conference poster sessions,2006, pages 33–40, Sydney, Australia.

[5] Choudhury, Monojit, Rahul Saraf, Vijit Jain, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. In Proceedings of the IJCAIWorkshop on "Analytics for Noisy Unstructured Text Data", pages 6370, Hyderabad, India.

[6] Kobus, Catherine and Yvon, Francios and Damnati, Geraldine, "Normalizing SMS: are two metaphors better than one?", Proceedings of the 22nd International Conference on Computational Linguistics, 2008, pages 441–448, Manchester, England.

[7] Stolcke, Andreas. 2002. Srilm an extensible language modeling toolkit. In Proceedings of the International Conference on Spoken Langage Processing (ICSLP), volume 2, pages 901904, Denver, CO.

[8] Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. Computational Linguistics, 29(1):1951.

[9] Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In Proc. Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session, Prague, Czech Republic.

[10] Yijue How and Min-Yen Kan (2005). Optimizing predictive text entry for short message service on mobile phones. In M. J. Smith & G. Salvendy (Eds.) Proc. of Human Computer Interfaces International (HCII 05). Lawrence Erlbaum Associates. Las Vegas, July 2005. ISBN 0805858075

[11] Och, Franz Josef. Minimum Error Rate Training in Statistical Machine Translation. Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, July 2003, pp. 160-167.

[12] Bodomo, Adams. Research Project on Linguistic Features of Mobile Communication. Department of Linguistics, HKU, September 2002.

[13] Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Technical Report RC22176

(W0109-022), IBM Research Division, Thomas J. Watson Research Center.