Lexical Normalisation of Short Text Messages: Makn Sens a #twitter

Bo Han and **Timothy Baldwin**
NICTA Victoria Research Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne
hanb@student.unimelb.edu.au tb@ldwin.net

Abstract

Twitter provides access to large volumes of data in real time, but is notoriously noisy, hampering its utility for NLP. In this paper, we target out-of-vocabulary words in short text messages and propose a method for identifying and normalising ill-formed words. Our method uses a classifier to detect ill-formed words, and generates correction candidates based on morphophonemic similarity. Both word similarity and context are then exploited to select the most probable correction candidate for the word. The proposed method doesn't require any annotations, and achieves state-of-the-art performance over an SMS corpus and a novel dataset based on Twitter.

1 Introduction

Twitter and other micro-blogging services are highly attractive for information extraction and text mining purposes, as they offer large volumes of real-time data, with around 65 millions tweets posted on Twitter per day in June 2010 (Twitter, 2010). The quality of messages varies significantly, however, ranging from high quality newswire-like text to meaningless strings. Typos, ad hoc abbreviations, phonetic substitutions, ungrammatical structures and emoticons abound in short text messages, causing grief for text processing tools (Sproat et al., 2001; Ritter et al., 2010). For instance, presented with the input *u must be talkin bout the paper but I was thinkin movies* ("You must be talking about the paper but I was thinking movies"),[1] the Stanford parser (Klein and Manning, 2003; de Marneffe et al., 2006) analyses *bout the paper* and *thinkin movies* as a clause and noun phrase, respectively, rather than a prepositional phrase and verb phrase. If there were some way of preprocessing the message to produce a more canonical lexical rendering, we would expect the quality of the parser to improve appreciably. Our aim in this paper is this task of lexical normalisation of noisy English text, with a particular focus on Twitter and SMS messages. In this paper, we will collectively refer to individual instances of typos, ad hoc abbreviations, unconventional spellings, phonetic substitutions and other causes of lexical deviation as "ill-formed words".

The message normalisation task is challenging. It has similarities with spell checking (Peterson, 1980), but differs in that ill-formedness in text messages is often intentional, whether due to the desire to save characters/keystrokes, for social identity, or due to convention in this text sub-genre. We propose to go beyond spell checkers, in performing deabbreviation when appropriate, and recovering the canonical word form of commonplace shorthands like *b4* "before", which tend to be considered beyond the remit of spell checking (Aw et al., 2006). The free writing style of text messages makes the task even more complex, e.g. with word lengthening such as *goooood* being commonplace for emphasis. In addition, the detection of ill-formed words is difficult due to noisy context.

Our objective is to restore ill-formed words to their canonical lexical forms in standard English. Through a pilot study, we compared OOV words in Twitter and SMS data with other domain corpora,

[1] Throughout the paper, we will provide a normalised version of examples as a gloss in double quotes.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 368–378,
Portland, Oregon, June 19-24, 2011.

revealing their characteristics in OOV word distribution. We found Twitter data to have an unsurprisingly long tail of OOV words, suggesting that conventional supervised learning will not perform well due to data sparsity. Additionally, many ill-formed words are ambiguous, and require context to disambiguate. For example, *Gooood* may refer to *Good* or *God* depending on context. This provides the motivation to develop a method which does not require annotated training data, but is able to leverage context for lexical normalisation. Our approach first generates a list of candidate canonical lexical forms, based on morphological and phonetic variation. Then, all candidates are ranked according to a list of features generated from noisy context and similarity between ill-formed words and candidates. Our proposed cascaded method is shown to achieve state-of-the-art results on both SMS and Twitter data.

Our contributions in this paper are as follows: (1) we conduct a pilot study on the OOV word distribution of Twitter and other text genres, and analyse different sources of non-standard orthography in Twitter; (2) we generate a text normalisation dataset based on Twitter data; (3) we propose a novel normalisation approach that exploits dictionary lookup, word similarity and word context, without requiring annotated data; and (4) we demonstrate that our method achieves state-of-the-art accuracy over both SMS and Twitter data.

2 Related work

The noisy channel model (Shannon, 1948) has traditionally been the primary approach to tackling text normalisation. Suppose the ill-formed text is T and its corresponding standard form is S, the approach aims to find $\arg\max P(S|T)$ by computing $\arg\max P(T|S)P(S)$, in which $P(S)$ is usually a language model and $P(T|S)$ is an error model. Brill and Moore (2000) characterise the error model by computing the product of operation probabilities on slice-by-slice string edits. Toutanova and Moore (2002) improve the model by incorporating pronunciation information. Choudhury et al. (2007) model the word-level text generation process for SMS messages, by considering graphemic/phonetic abbreviations and unintentional typos as hidden Markov model (HMM) state transitions and emissions, respectively (Rabiner, 1989). Cook and Stevenson (2009) expand the error model by introducing inference from different erroneous formation processes, according to the sampled error distribution. While the noisy channel model is appropriate for text normalisation, $P(T|S)$, which encodes the underlying error production process, is hard to approximate accurately. Additionally, these methods make the strong assumption that a token $t_i \in T$ only depends on $s_i \in S$, ignoring the context around the token, which could be utilised to help in resolving ambiguity.

Statistical machine translation (SMT) has been proposed as a means of context-sensitive text normalisation, by treating the ill-formed text as the source language, and the standard form as the target language. For example, Aw et al. (2006) propose a phrase-level SMT SMS normalisation method with bootstrapped phrase alignments. SMT approaches tend to suffer from a critical lack of training data, however. It is labor intensive to construct an annotated corpus to sufficiently cover ill-formed words and context-appropriate corrections. Furthermore, it is hard to harness SMT for the lexical normalisation problem, as even if phrase-level re-ordering is suppressed by constraints on phrase segmentation, word-level re-orderings within a phrase are still prevalent.

Some researchers have also formulated text normalisation as a speech recognition problem. For example, Kobus et al. (2008) firstly convert input text tokens into phonetic tokens and then restore them to words by phonetic dictionary lookup. Beaufort et al. (2010) use finite state methods to perform French SMS normalisation, combining the advantages of SMT and the noisy channel model. Kaufmann and Kalita (2010) exploit a machine translation approach with a preprocessor for syntactic (rather than lexical) normalisation.

Predominantly, however, these methods require large-scale annotated training data, limiting their adaptability to new domains or languages. In contrast, our proposed method doesn't require annotated data. It builds on the work on SMS text normalisation, and adapts it to Twitter data, exploiting multiple data sources for normalisation.

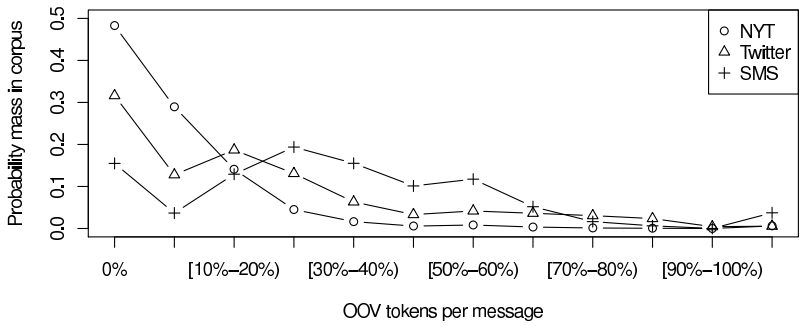

Figure 1: Out-of-vocabulary word distribution in English Gigaword (NYT), Twitter and SMS data

3 Scoping Text Normalisation

3.1 Task Definition of Lexical Normalisation

We define the task of text normalisation to be a mapping from "ill-formed" OOV lexical items to their standard lexical forms, focusing exclusively on English for the purposes of this paper. We define the task as follows:

- only OOV words are considered for normalisation;
- normalisation must be to a single-token word, meaning that we would normalise *smokin* to *smoking*, but not *imo* to *in my opinion*; a side-effect of this is to permit lower-register contractions such as *gonna* as the canonical form of *gunna* (given that *going to* is out of scope as a normalisation candidate, on the grounds of being multi-token).

Given this definition, our first step is to identify candidate tokens for lexical normalisation, where we examine all tokens that consist of alphanumeric characters, and categorise them into in-vocabulary (IV) and out-of-vocabulary (OOV) words, relative to a dictionary. The OOV word definition is somewhat rough, because it includes neologisms and proper nouns like *hopeable* or *WikiLeaks* which have not made their way into the dictionary. However, it greatly simplifies the candidate identification task, at the cost of pushing complexity downstream to the word detection task, in that we need to explicitly distinguish between correct OOV words and ill-formed OOV words such as typos (e.g. *earthquak* "earthquake"), register-specific single-word abbreviations (e.g. *lv* "love"), and phonetic substitutions (e.g. *2morrow* "tomorrow").

An immediate implication of our task definition is that ill-formed words which happen to coincide with an IV word (e.g. the misspelling of *can't* as *cant*) are outside the scope of this research. We also consider that deabbreviation largely falls outside the scope of text normalisation, as abbreviations can be formed freely in standard English. Note that single-word abbreviations such as *govt* "government" are very much within the scope of lexical normalisation, as they are OOV and match to a single token in their standard lexical form.

Throughout this paper, we use the GNU aspell dictionary (v0.60.6)[2] to determine whether a token is OOV. In tokenising the text, hyphenanted tokens and tokens containing apostrophes (e.g. *take-off* and *won't*, resp.) are treated as a single token. Twitter mentions (e.g. *@twitter*), hashtags (e.g. *#twitter*) and urls (e.g. *twitter.com*) are excluded from consideration for normalisation, but left in situ for context modelling purposes. Dictionary lookup of Internet slang is performed relative to a dictionary of 5021 items collected from the Internet.[3]

3.2 OOV Word Distribution and Types

To get a sense of the relative need for lexical normalisation, we perform analysis of the distribution of OOV words in different text types. In particular, we calculate the proportion of OOV tokens per message (or sentence, in the case of edited text), bin the messages according to the OOV token proportion, and plot the probability mass contained in each bin for a given text type. The three corpora we compare

[2]We remove all one character tokens, except *a* and *I*, and treat *RT* as an IV word.

[3]`http://www.noslang.com`

are the New York Times (NYT),[4] SMS,[5] and Twitter.[6] The results are presented in Figure 1.

Both SMS and Twitter have a relatively flat distribution, with Twitter having a particularly large tail: around 15% of tweets have 50% or more OOV tokens. This has implications for any context modelling, as we cannot rely on having only isolated occurrences of OOV words. In contrast, NYT shows a more Zipfian distribution, despite the large number of proper names it contains.

While this analysis confirms that Twitter and SMS are similar in being heavily laden with OOV tokens, it does not shed any light on the relative similarity in the makeup of OOV tokens in each case. To further analyse the two data sources, we extracted the set of OOV terms found exclusively in SMS and Twitter, and analysed each. Manual analysis of the two sets revealed that most OOV words found only in SMS were personal names. The Twitter-specific set, on the other hand, contained a heterogeneous collection of ill-formed words and proper nouns. This suggests that Twitter is a richer/noisier data source, and that text normalisation for Twitter needs to be more nuanced than for SMS.

To further analyse the ill-formed words in Twitter, we randomly selected 449 tweets and manually analysed the sources of lexical variation, to determine the phenomena that lexical normalisation needs to deal with. We identified 254 token instances of lexical normalisation, and broke them down into categories, as listed in Table 1. "Letter" refers to instances where letters are missing or there are extraneous letters, but the lexical correspondence to the target word form is trivially accessible (e.g. *shuld* "should"). "Number Substitution" refers to instances of letter–number substitution, where numbers have been substituted for phonetically-similar sequences of letters (e.g. *4* "for"). "Letter&Number" refers to instances which have both extra/missing letters and number substitution (e.g. *b4* "before"). "Slang" refers to instances of Internet slang (e.g. *lol* "laugh out loud"), as found in a slang dictionary (see Section 3.1). "Other" is the remainder of the instances, which is predominantly made up of occurrences of spaces having being deleted between words (e.g. *sucha* "such a"). If a given instance belongs to multiple error categories (e.g. "Letter&Number" and it is also found in a slang dictionary), we classify it into the higher-occurring category in Table 1.

Category	Ratio
Letter&Number	2.36%
Letter	72.44%
Number Substitution	2.76%
Slang	12.20%
Other	10.24%

Table 1: Ill-formed word distribution

From Table 1, it is clear that "Letter" accounts for the majority of ill-formed words in Twitter, and that most ill-formed words are based on morphophonemic variations. This empirical finding assists in shaping our strategy for lexical normalisation.

4 Lexical normalisation

Our proposed lexical normalisation strategy involves three general steps: (1) confusion set generation, where we identify normalisation candidates for a given word; (2) ill-formed word identification, where we classify a word as being ill-formed or not, relative to its confusion set; and (3) candidate selection, where we select the standard form for tokens which have been classified as being ill formed. In confusion set generation, we generate a set of IV normalisation candidates for each OOV word type based on morphophonemic variation. We call this set the confusion set of that OOV word, and aim to include all feasible normalisation candidates for the word type in the confusion set. The confusion candidates are then filtered for each token occurrence of a given OOV word, based on their local context fit with a language model.

4.1 Confusion Set Generation

Revisiting our manual analysis from Section 3.2, most ill-formed tokens in Twitter are morphophonemically derived. First, inspired by Kaufmann and Kalita (2010), any repititions of more than 3 letters are reduced back to 3 letters (e.g. *cooool* is re-

[4]Based on 44 million sentences from English Gigaword.

[5]Based on 12.6 thousand SMS messages from How and Kan (2005) and Choudhury et al. (2007).

[6]Based on 1.37 million tweets collected from the Twitter streaming API from Aug to Oct 2010, and filtered for monolingual English messages; see Section 5.1 for details of the language filtering methodology.

Criterion	Recall	Average Candidates
$T_c \leq 1$	40.4%	24
$T_c \leq 2$	76.6%	240
$T_p = 0$	55.4%	65
$T_p \leq 1$	83.4%	1248
$T_p \leq 2$	91.0%	9694
$T_c \leq 2 \vee T_p \leq 1$	88.8%	1269
$T_c \leq 2 \vee T_p \leq 2$	92.7%	9515

Table 2: Recall and average number of candidates for different confusion set generation strategies

duced to *coool*). Second, IV words within a threshold T_c character edit distance of the given OOV word are calculated, as is widely used in spell checkers. Third, the double metaphone algorithm (Philips, 2000) is used to decode the pronunciation of all IV words, and IV words within a threshold T_p edit distance of the given OOV word under phonemic transcription, are included in the confusion set; this allows us to capture OOV words such as *earthquick* "earthquake". In Table 2, we list the recall and average size of the confusion set generated by the final two strategies with different threshold settings, based on our evaluation dataset (see Section 5.1).

The recall for lexical edit distance with $T_c \leq 2$ is moderately high, but it is unable to detect the correct candidate for about one quarter of words. The combination of the lexical and phonemic strategies with $T_c \leq 2 \vee T_p \leq 2$ is more impressive, but the number of candidates has also soared. Note that increasing the edit distance further in both cases leads to an explosion in the average number of candidates, with serious computational implications for downstream processing. Thankfully, $T_c \leq 2 \vee T_p \leq 1$ leads to an extra increment in recall to 88.8%, with only a slight increase in the average number of candidates. Based on these results, we use $T_c \leq 2 \vee T_p \leq 1$ as the basis for confusion set generation.

Examples of ill-formed words where we are unable to generate the standard lexical form are clippings such as *fav* "favourite" and *convo* "conversation".

In addition to generating the confusion set, we rank the candidates based on a trigram language model trained over 1.5GB of clean Twitter data, i.e. tweets which consist of all IV words: despite the prevalence of OOV words in Twitter, the sheer volume of the data means that it is relatively easy to collect large amounts of all-IV messages. To train the language model, we used SRILM (Stolcke, 2002) with the `-<unk>` option. If we truncate the ranking to the top 10% of candidates, the recall drops back to 84% with a 90% reduction in candidates.

4.2 Ill-formed Word Detection

The next step is to detect whether a given OOV word in context is actually an ill-formed word or not, relative to its confusion set. To the best of our knowledge, we are the first to target the task of ill-formed word detection in the context of short text messages, although related work exists for text with lower relative occurrences of OOV words (Izumi et al., 2003; Sun et al., 2007). Due to the noisiness of the data, it is impractical to use full-blown syntactic or semantic features. The most direct source of evidence is IV words around an OOV word. Inspired by work on labelled sequential pattern extraction (Sun et al., 2007), we exploit large-scale edited corpus data to construct dependency-based features.

First, we use the Stanford parser (Klein and Manning, 2003; de Marneffe et al., 2006) to extract dependencies from the NYT corpus (see Section 3.2). For example, from a sentence such as *One obvious difference is the way they look*, we would extract dependencies such as `rcmod(way-6,look-8)` and `nsubj(look-8,they-7)`. We then transform the dependencies into relational features for each OOV word. Assuming that *way* were an OOV word, e.g., we would extract dependencies of the form `(look,way,+2)`, indicating that *look* occurs 2 words after *way*. We choose dependencies to represent context because they are an effective way of capturing key relationships between words, and similar features can easily be extracted from tweets. Note that we don't record the dependency type here, because we have no intention of dependency parsing text messages, due to their noisiness and the volume of the data. The counts of dependency forms are combined together to derive a confidence score, and the scored dependencies are stored in a dependency bank.

Given the dependency-based features, a linear kernel SVM classifier (Fan et al., 2008) is trained on clean Twitter data, i.e. the subset of Twitter messages without OOV words. Each word is repre-

sented by its IV words within a context window of three words to either side of the target word, together with their relative positions in the form of `(word1,word2,position)` tuples, and their score in the dependency bank. These form the positive training exemplars. Negative exemplars are automatically constructed by replacing target words with highly-ranked candidates from their confusion set. Note that the classifier does not require any hand annotation, as all training exemplars are constructed automatically.

To predict whether a given OOV word is ill-formed, we form an exemplar for each of its confusion candidates, and extract `(word1,word2,position)` features. If all its candidates are predicted to be negative by the model, we mark it as correct; otherwise, we treat it as ill-formed, and pass all candidates (not just positively-classified candidates) on to the candidate selection step. For example, given the message *way yu lookin shuld be a sin* and the OOV word *lookin*, we would generate context features for each candidate word such as `(way,looking,-2)`, and classify each such candidate.

In training, it is possible for the exact same feature vector to occur as both positive and negative exemplars. To prevent positive exemplars being contaminated from the automatic generation, we remove all negative instances in such cases. The `(word1,word2,position)` features are sparse and sometimes lead to conservative results in ill-formed word detection. That is, without valid features, the SVM classifier tends to label uncertain cases as correct rather than ill-formed words. This is arguably the right approach to normalisation, in choosing to under- rather than over-normalise in cases of uncertainty.

As the context for a target word often contains OOV words which don't occur in the dependency bank, we expand the dependency features to include context tokens up to a phonemic edit distance of 1 from context tokens in the dependency bank. In this way, we generate dependency-based features for context words such as *seee* "see" in `(seee, flm, +2)` (based on the target word *flm* in the context of *flm to seee*). However, expanded dependency features may introduce noise, and we therefore introduce expanded dependency weights $w_d \in \{0.0, 0.5, 1.0\}$ to ameliorate the effects of noise: a weight of $w_d = 0.0$ means no expansion, while 1.0 means expanded dependencies are indistinguishable from non-expanded (strict match) dependencies.

We separately introduce a threshold $t_d \in \{1, 2, ..., 10\}$ on the number of positive predictions returned by the detection classifier over the set of normalisation candidates for a given OOV token: the token is considered to be ill-formed iff t_d or more candidates are positively classified, i.e. predicted to be correct candidates.

4.3 Candidate Selection

For OOV words which are predicted to be ill-formed, we select the most likely candidate from the confusion set as the basis of normalisation. The final selection is based on the following features, in line with previous work (Wong et al., 2006; Cook and Stevenson, 2009).

Lexical edit distance, phonemic edit distance, prefix substring, suffix substring, and the longest common subsequence (LCS) are exploited to capture morphophonemic similarity. Both lexical and phonemic edit distance (ED) are normalised by the reciprocal of $exp(ED)$. The prefix and suffix features are intended to capture the fact that leading and trailing characters are frequently dropped from words, e.g. in cases such as *ish* and *talkin*. We calculate the ratio of the LCS over the maximum string length between ill-formed word and the candidate, since the ill-formed word can be either longer or shorter than (or the same size as) the standard form. For example, *mve* can be restored to either *me* or *move*, depending on context. We normalise these ratios following Cook and Stevenson (2009).

For context inference, we employ both language model- and dependency-based frequency features. Ranking by language model score is intuitively appealing for candidate selection, but our trigram model is trained only on clean Twitter data and ill-formed words often don't have sufficient context for the language model to operate effectively, as in *bt* "but" in *say 2 sum1 bt nt gonna say* "say to someone but not going to say". To consolidate the context modelling, we obtain dependencies from the dependency bank used in ill-formed word detection. Although text messages are of a different genre to edited newswire text, we assume they form similar

dependencies based on the common goal of getting across the message effectively. The dependency features can be used in noisy contexts and are robust to the effects of other ill-formed words, as they do not rely on contiguity. For example, *uz* "use" in *i did #tt uz me and yu*, dependencies can capture relationships like `aux(use-4, do-2)`, which is beyond the capabilities of the language model due to the hashtag being treated as a correct OOV word.

5 Experiments

5.1 Dataset and baselines

The aim of our experiments is to compare the effectiveness of different methodologies over text messages, based on two datasets: (1) an SMS corpus (Choudhury et al., 2007); and (2) a novel Twitter dataset developed as part of this research, based on a random sampling of 549 English tweets. The English tweets were annotated by three independent annotators. All OOV words were pre-identified, and the annotators were requested to determine: (a) whether each OOV word was ill-formed or not; and (b) what the standard form was for ill-formed words, subject to the task definition outlined in Section 3.1. The total number of ill-formed words contained in the SMS and Twitter datasets were 3849 and 1184, respectively.[7]

The language filtering of Twitter to automatically identify English tweets was based on the language identification method of Baldwin and Lui (2010), using the EuroGOV dataset as training data, a mixed unigram/bigram/trigram byte feature representation, and a skew divergence nearest prototype classifier.

We reimplemented the state-of-art noisy channel model of Cook and Stevenson (2009) and SMT approach of Aw et al. (2006) as benchmark methods. We implement the SMT approach in Moses (Koehn et al., 2007), with synthetic training and tuning data of 90,000 and 1000 sentence pairs, respectively. This data is randomly sampled from the 1.5GB of clean Twitter data, and errors are generated according to distribution of SMS corpus. The 10-fold cross-validated BLEU score (Papineni et al., 2002) over this data is 0.81.

[7]The Twitter dataset is available at `http://www.csse.unimelb.edu.au/research/lt/resources/lexnorm/`.

In addition to comparing our method with competitor methods, we also study the contribution of different feature groups. We separately compare dictionary lookup over our Internet slang dictionary, the contextual feature model, and the word similarity feature model, as well as combinations of these three.

5.2 Evaluation metrics

The evaluation of lexical normalisation consists of two stages (Hirst and Budanitsky, 2005): (1) ill-formed word detection, and (2) candidate selection. In terms of detection, we want to make sense of how well the system can identify ill-formed words and leave correct OOV words untouched. This step is crucial to further normalisation, because if correct OOV words are identified as ill-formed, the candidate selection step can never be correct. Conversely, if an ill-formed word is predicted to be correct, the candidate selection will have no chance to normalise it.

We evaluate detection performance by token-level precision, recall and F-score ($\beta = 1$). Previous work over the SMS corpus has assumed perfect ill-formed word detection and focused only on the candidate selection step, so we evaluate ill-formed word detection for the Twitter data only.

For candidate selection, we once again evaluate using token-level precision, recall and F-score. Additionally, we evaluate using the BLEU score over the normalised form of each message, as the SMT method can lead to perturbations of the token stream, vexing standard precision, recall and F-score evaluation.

5.3 Results and Analysis

First, we test the impact of the w_d and t_d values on ill-formed word detection effectiveness, based on dependencies from either the Spinn3r blog corpus (Blog: Burton et al. (2009)) or NYT. The results for precision, recall and F-score are presented in Figure 2.

Some conclusions can be drawn from the graphs. First, higher detection threshold values (t_d) give better precision but lower recall. Generally, as t_d is raised from 1 to 10, the precision improves slightly but recall drops dramatically, with the net effect that the F-score decreases monotonically. Thus, we use a

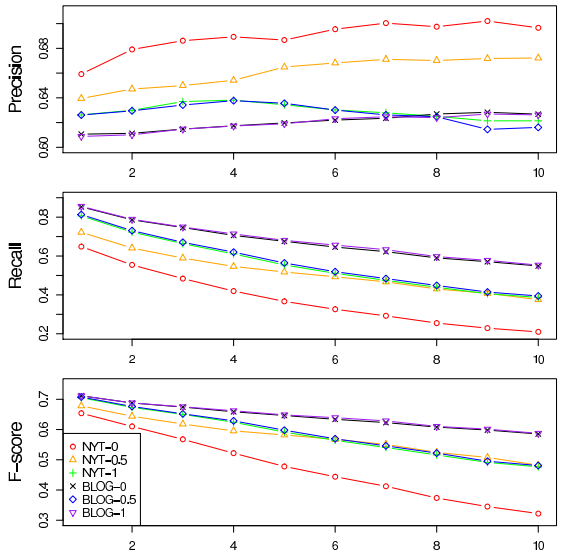

Figure 2: Ill-formed word detection precision, recall and F-score

smaller threshold, i.e. $t_d = 1$. Second, there are differences between the two corpora, with dependencies from the Blog corpus producing slightly lower precision but higher recall, compared with the NYT corpus. The lower precision for the Blog corpus appears to be due to the text not being as clean as NYT, introducing parser errors. Nevertheless, the difference in F-score between the two corpora is insignificant. Third, we obtain the best results, especially in terms of precision, for $w_d = 0.5$, i.e. with expanded dependencies, but penalised relative to non-expanded dependencies.

Overall, the best F-score is 71.2%, with a precision of 61.1% and recall of 85.3%, obtained over the Blog corpus with $t_d = 1$ and $w_d = 0.5$. Clearly there is significant room for immprovements in these results. We leave the improvement of ill-formed word detection for future work, and perform evaluation of candidate selection for Twitter assuming perfect ill-formed word detection, as for the SMS data.

From Table 3, we see that the general performance of our proposed method on Twitter is better than that on SMS. To better understand this trend, we examined the annotations in the SMS corpus, and found them to be looser than ours, because they have different task specifications than our lexical normalisation. In our annotation, the annotators only normalised ill-formed word if they had high confidence of how to normalise, as with *talkin* "talking". For ill-formed words where they couldn't be certain of the standard form, the tokens were left untouched. However, in the SMS corpus, annotations such as *sammis* "same" are also included. This leads to a performance drop for our method over the SMS corpus.

The noisy channel method of Cook and Stevenson (2009) shares similar features with word similarity ("WS"), However, when word similarity and context support are combined ("WS+CS"), our method outperforms the noisy channel method by about 7% and 12% in F-score over SMS and Twitter corpora, respectively. This can be explained as follows. First, the Cook and Stevenson (2009) method is type-based, so all token instances of a given ill-formed word will be normalised identically. In the Twitter data, however, the same word can be normalised differently depending on context, e.g. *hw* "how" in *so hw many time remaining so I can calculate it?* vs. *hw* "homework" in *I need to finish my hw first*. Second, the noisy channel method was developed specifically for SMS normalisation, in which clipping is the most prevalent form of lexical variation, while in the Twitter data, we commonly have instances of word lengthening for emphasis, such as *moviiie* "movie". Having said this, our method is superior to the noisy channel method over both the SMS and Twitter data.

The SMT approach is relatively stable on the two datasets, but well below the performance of our method. This is due to the limitations of the training data: we obtain the ill-formed words and their standard forms from the SMS corpus, but the ill-formed words in the SMS corpus are not sufficient to cover those in the Twitter data (and we don't have sufficient Twitter data to train the SMT method directly). Thus, novel ill-formed words are missed in normalisation. This shows the shortcoming of supervised data-driven approaches that require annotated data to cover all possibilities of ill-formed words in Twitter.

The dictionary lookup method ("DL") unsurprisingly achieves the best precision, but the recall on Twitter is not competitive. Consequently, the Twitter normalisation cannot be tackled with dictionary lookup alone, although it is an effective preprocessing strategy when combined with more ro-

Dataset	*Evaluation*	*NC*	*MT*	*DL*	*WS*	*CS*	*WS+CS*	*DL+WS+CS*
SMS	Precision	0.465	—	**0.927**	0.521	0.116	0.532	0.756
	Recall	0.464	—	0.597	0.520	0.116	0.531	**0.754**
	F-score	0.464	—	0.726	0.520	0.116	0.531	**0.755**
	BLEU	0.746	0.700	0.801	0.764	0.612	0.772	**0.876**
Twitter	Precision	0.452	—	**0.961**	0.551	0.194	0.571	0.753
	Recall	0.452	—	0.460	0.551	0.194	0.571	**0.753**
	F-score	0.452	—	0.622	0.551	0.194	0.571	**0.753**
	BLEU	0.857	0.728	0.861	0.878	0.797	0.884	**0.934**

Table 3: Candidate selection effectiveness on different datasets (*NC* = noisy channel model (Cook and Stevenson, 2009); *MT* = SMT (Aw et al., 2006); *DL* = dictionary lookup; *WS* = word similarity; *CS* = context support)

bust techniques such as our proposed method, and effective at capturing common abbreviations such as *gf* "girlfriend".

Of the component methods proposed in this research, word similarity ("WS") achieves higher precision and recall than context support ("CS"), signifying that many of the ill-formed words emanate from morphophonemic variations. However, when combined with word similarity features, context support improves over the basic method at a level of statistical significance (based on randomised estimation, $p < 0.05$: Yeh (2000)), indicating the complementarity of the two methods, especially on Twitter data. The best F-score is achieved when combining dictionary lookup, word similarity and context support ("DL+WS+CS"), in which ill-formed words are first looked up in the slang dictionary, and only if no match is found do we apply our normalisation method.

We found several limitations in our proposed approach by analysing the output of our method. First, not all ill-formed words offer useful context. Some highly noisy tweets contain almost all misspellings and unique symbols, and thus no context features can be extracted. This also explains why "CS" features often fail. For such cases, the method falls back to context-independent normalisation. We found that only 32.6% ill-formed words have all IV words in their context windows. Moreover, the IV words may not occur in the dependency bank, further decreasing the effectiveness of context support features. Second, the different features are linearly combined, where a weighted combination is likely to give better results, although it also requires a certain amount of well-sampled annotations for tuning.

6 Conclusion and Future Work

In this paper, we have proposed the task of lexical normalisation for short text messages, as found in Twitter and SMS data. We found that most ill-formed words are based on morphophonemic variation and proposed a cascaded method to detect and normalise ill-formed words. Our ill-formed word detector requires no explicit annotations, and the dependency-based features were shown to be somewhat effective, however, there was still a lot of room for improvement at ill-formed word detection. In normalisation, we compared our method with two benchmark methods from the literature, and achieved that highest F-score and BLEU score by integrating dictionary lookup, word similarity and context support modelling.

In future work, we propose to pursue a number of directions. First, we plan to improve our ill-formed word detection classifier by introducing an OOV word whitelist. Furthermore, we intend to alleviate noisy contexts with a bootstrapping approach, in which ill-formed words with high confidence and no ambiguity will be replaced by their standard forms, and fed into the normalisation model as new training data.

Acknowledgements

NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT centre of Excellence programme.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of the 21st International Con-*

ference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 33–40, Sydney, Australia.

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *HLT '10: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 229–237, Los Angeles, USA.

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

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293, Hong Kong.

Kevin Burton, Akshay Java, and Ian Soboroff. 2009. The ICWSM 2009 Spinn3r Dataset. In *Proceedings of the Third Annual Conference on Weblogs and Social Media*, San Jose, USA.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10:157–174.

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

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11:87–111.

Yijue How and Min-Yen Kan. 2005. Optimizing predictive text entry for short message service on mobile phones. In *Human Computer Interfaces International (HCII 05)*, Las Vegas, USA.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, pages 145–148, Sapporo, Japan.

Joseph Kaufmann and Jugal Kalita. 2010. Syntactic normalization of Twitter messages. In *International Conference on Natural Language Processing*, Kharagpur, India.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pages 3–10, Whistler, Canada.

Catherine Kobus, Franois Yvon, and Graldine Damnati. 2008. Transcrire les SMS comme on reconnat la parole. In *Actes de la Confrence sur le Traitement Automatique des Langues (TALN'08)*, pages 128–138.

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*, pages 177–180, Prague, Czech Republic.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, USA.

James L. Peterson. 1980. Computer programs for detecting and correcting spelling errors. *Commun. ACM*, 23:676–687, December.

Lawrence Philips. 2000. The double metaphone search algorithm. *C/C++ Users Journal*, 18:38–43.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of Twitter conversations. In *HLT '10: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, Los Angeles, USA.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

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

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *International Conference on Spo-*

ken Language Processing, pages 901–904, Denver, USA.

Guihua Sun, Gao Cong, Xiaohua Liu, Chin-Yew Lin, and Ming Zhou. 2007. Mining sequential patterns and tree patterns to detect erroneous sentences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 81–88, Prague, Czech Republic.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 144–151, Philadelphia, USA.

Twitter. 2010. Big goals, big game, big records. `http://blog.twitter.com/2010/06/big-goals-big-game-big-records.html`. Retrieved 4 August 2010.

Wilson Wong, Wei Liu, and Mohammed Bennamoun. 2006. Integrated scoring for spelling error correction, abbreviation expansion and case restoration in dirty text. In *Proceedings of the Fifth Australasian Conference on Data Mining and Analytics*, pages 83–89, Sydney, Australia.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 947–953, Saarbrücken, Germany.