A

Lexical Normalisation for Social Media Text

Bo Han,♠♡ Paul Cook,♡ and Timothy Baldwin♠♡
♠ NICTA Victoria Research Laboratory
♡ Department of Computing and Information Systems, The University of Melbourne

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 lexical variants. Our method uses a classifier to detect lexical variants, 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. This paper is an extension of Han and Baldwin [2011] and Han et al. [2012], with significantly expanded experimentation over the original paper.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms: Text analysis

Additional Key Words and Phrases: Lexical normalisation, Short text message, microblog, text preprocessing

ACM Reference Format:
Han, B., Cook, P.,Baldwin, T. 2011. Lexical Normalisation of Short Text Messages. ACM Trans. Intell. Syst. Technol. V, N, Article A (January YYYY), 27 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Micro-blogging services, such as Twitter,[1] are highly attractive for information extraction and text mining purposes, as they offer large volumes of real-time data. According to Twitter [2011], 2, 65, and 200 million messages were posted per day in 2009, 2010, and first half of 2011, respectively, and the number is still growing. Messages from Twitter have shown to have utility in applications such as disaster detection [Sakaki et al. 2010], sentiment analysis [Jiang et al. 2011; González-Ibáñez et al. 2011], and event discovery [Weng and Lee 2011; Benson et al. 2011]. 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

[1] www.twitter.com

Author's addresses: B. Han, Department of Computing and Information Systems, University of Melbourne, VIC 3010 Australia; P. Cook, Department of Computing and Information Systems, University of Melbourne, VIC 3010 Australia. T. Baldwin, Department of Computing and Information Systems, University of Melbourne, VIC 3010 Australia.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

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"),[2] 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.

One way to minimise the performance drop of current tools and make full use of Twitter data, is to re-train text processing tools on this new domain [Gimpel et al. 2011; Liu et al. 2011; Ritter et al. 2011]. An alternative approach is to preprocess messages to produce a more-standard rendering of these lexical variants. For example, *se u 2morw!!!* would be normalised to *see you tomorrow!* The normalisation approach is especially attractive as a preprocessing step for applications which rely on keyword match or word frequency statistics, such as topic trend analysis. For example, *earthqu*, *eathquake*, and *earthquakeee* — all attested in a Twitter corpus — have the standard form *earthquake*; by normalising these types to their standard form, better coverage can be achieved for keyword-based methods, and better word frequency estimates can be obtained. We will collectively refer to individual instances of typos, ad hoc abbreviations, unconventional spellings, phonetic substitutions and other causes of lexical deviation as "lexical variants".

The normalisation task is challenging. It has similarities with spell checking [Peterson 1980], but differs in that lexical variants in text messages are often intentionally generated, 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 lexical variants — i.e. distinguishing lexical variants from out-of-vocabulary words that do not require normalisation — is rather difficult, due at least in part to the short length of, and noise encountered in, Twitter messages.

In this paper, we focus on the task of lexical normalisation of English Twitter and SMS messages, in which out-of-vocabulary (OOV) tokens are normalised to their in-vocabulary (IV) standard form, i.e. a standard form that is in a dictionary. We further restrict the task to be a one-to-one normalisation in which one OOV token is normalised to one IV word.

The remainder of this article is organised as follows: after reviewing relevant related work in Section 2, we offer a more formal definition of the task of lexical normalisation in Section 3. In Section 4 we analyse a number of linguistic properties of SMS and Twitter messages to gain some insight into the task of lexical normalisation. We then propose a context-sensitive method for lexical normalisation, and further present a manually-annotated lexical normalisation dataset based on Twitter data. We compare the performance of our proposed normalisation method to a number of baselines and benchmark approaches. In Section 5 we consider the task of lexical variant detection, an often-overlooked aspect of lexical normalisation. We propose and evaluate a token-based method for this task. Our findings indicate that poor performance on lexical variant detection leads to poor performance in real-world normalisation tasks. In response to this finding, in Section 6 we propose a purely type-based dictionary-lookup approach to normalisation focusing on context insensitive lexical variants. It combines the detection and normalisation of lexical variants into a single step. In Section 7 we then show that this new approach achieves state-of-the-art performance. We consider the impact of lexical normalisation on downstream processing in Section 8. In experi-

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

ments on part-of-speech tagging, we find some evidence that pre-normalisation of the input text can lead to performance improvements. Finally, we offer some concluding remarks in Section 9.

This article is a revised and extended version of Han and Baldwin [2011] and Han et al. [2012]. This article further includes the following additional material, not included in either of those papers: (1) experiments on the impact of normalisation in an applied setting (Section 8); (2) baselines for context sensitive lexical normalisation based on a web-based language model and a spell checker (Section 4.4.1), and a discussion of the limitations of such approaches; (3) an analysis of the impact of lexical variants on context-sensitive normalisation (Section 7.2.4).

2. RELATED WORK

The noisy channel model [Shannon 1948] has traditionally been the primary approach to tackling text normalisation. Given a token t, lexical normalisation is the task of finding $\arg\max P(s|t) \propto \arg\max P(t|s)P(s)$, where s is the standard form, i.e. an IV word. Standardly in lexical normalisation, t is assumed to be an OOV token, relative to a fixed dictionary. In practice, not all OOV tokens should be normalised; i.e. only lexical variants (e.g. *tmrw* "tomorrow") should be normalised and tokens that are OOV but otherwise not lexical variants (e.g. *iPad* "iPad") should be unchanged. Most work in this area focuses only on the normalisation task itself, oftentimes assuming that the task of lexical variant detection has already been completed.

Various approaches have been proposed to estimate the error model, $P(t|s)$. For example, in work on spell-checking, Brill and Moore [2000] improve on a standard edit-distance approach by considering multi-character edit operations; Toutanova and Moore [2002] build on this by incorporating phonological information. Li et al. [2006] utilise distributional similarity [Lin 1998] to correct misspelled search queries.

In text message normalisation, Choudhury et al. [2007] model the letter transformations and emissions using a hidden Markov model [Rabiner 1989]. Cook and Stevenson [2009] and Xue et al. [2011] propose multiple simple error models, each of which captures a particular way in which lexical variants are formed, such as phonetic spelling (e.g. *epik* "epic") or clipping (e.g. *walkin* "walking"). Nevertheless, optimally weighting the various error models in these approaches is challenging.

Without pre-categorising lexical variants into different types, Liu et al. [2011] collect Google search snippets from carefully-designed queries from which they then extract noisy lexical variant–standard form pairs. These pairs are used to train a conditional random field [Lafferty et al. 2001] to estimate $P(t|s)$ at the character level. One shortcoming of querying a search engine to obtain training pairs is it tends to be costly in terms of time and bandwidth. Here we exploit microblog data directly to derive $(\text{lexical variant}, \text{standard form})$ pairs, instead of relying on external resources. In more recent work, Liu et al. [2012] endeavour to improve the accuracy of top-n normalisation candidates by integrating human cognitive inference, character-level transformations and spell checking in their normalisation model. The encouraging results shift the focus to reranking and promoting the correct normalisation to the top-*1* position. However, like much previous work on lexical normalisation, this work assumes perfect lexical variant detection.

Aw et al. [2006] and Kaufmann and Kalita [2010] consider normalisation as a machine translation task using off-the-shelf tools. Essentially, this is also a noisy channel model, which regards lexical variants and standard forms as a source language and a target language, and translates from one to the other. The advantage of these methods are they do not assume that lexical variants have been pre-identified; however, these methods do rely on large quantities of labelled training data, which is not available for microblogs. In this paper, we extensively discuss lexical normalisation methods for

short text message, and compare our proposed methods with benchmarks using different evaluation metrics.

3. SCOPING LEXICAL NORMALISATION

Popular micro-blogs such as Twitter attract users from diverse language backgrounds, and are therefore highly multilingual. Preliminary language identification shows that English tweets account for less than 50% of the overall data. In this research, we focus exclusively on the English lexical normalisation task, leaving the more general task of multilingual lexical normalisation for future work.

We define the lexical normalisation 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, and assuming that *gonna* is in-vocabulary).

An immediate implication of our task definition is that lexical variants which happen to coincide with an IV word (e.g. *can't* spelt as *cant*) are outside the scope of this research. We also consider deabbreviation of acronyms and initialisms (e.g. *imo* "in my opinion") to largely fall outside the scope of text normalisation, as such abbreviated forms 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 correspond to a single token in their standard lexical form.

Given this definition, a necessary pre-processing step for lexical normalisation is the identification of candidate tokens for normalisation. Here we examine all tokens that consist of alphanumeric characters, and categorise them into IVs and OOVs, relative to a dictionary, and take the OOVs as candidates for lexical normalisation. Note, however, that the OOVs will include lexical variants, but will also include other word types, such as neologisms and proper nouns, that happen to not be listed in the dictionary being used. One challenge for lexical normalisation is therefore to distinguish between OOV words that should not be normalised (such as *hopeable* and *WikiLeaks*, which are not included in the dictionary we use in our experiments) and lexical variants requiring normalisation such as typos (e.g. *earthquak* "earthquake"), register-specific single-word abbreviations (e.g. *lv* "love"), and phonetic substitutions (e.g. *2morrow* "tomorrow"). Note that many previous approaches to normalisation — discussed in Section 2 — have made the simplifying assumption that lexical variants have already been identified. In this article, we begin by considering lexical normalisation making this assumption in Section 4.2. We then address the issue of identifying lexical variants from amongst OOVs in Section 5, and show that this task is challenging, and that poor performance at this task negatively impacts overall normalisation performance. In Section 6 we then propose a new dictionary-based approach to normalisation that avoids such problems associated with lexical variant identification.

Throughout this paper, we use the GNU aspell dictionary (v6.06)[3] to determine whether a token is OOV. In tokenising the text, 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.[4] The language filtering of Twitter to automatically identify English tweets

[3]We remove all one character tokens, except *a* and *I*, and treat *RT* as an IV word.
[4]`http://www.noslang.com`

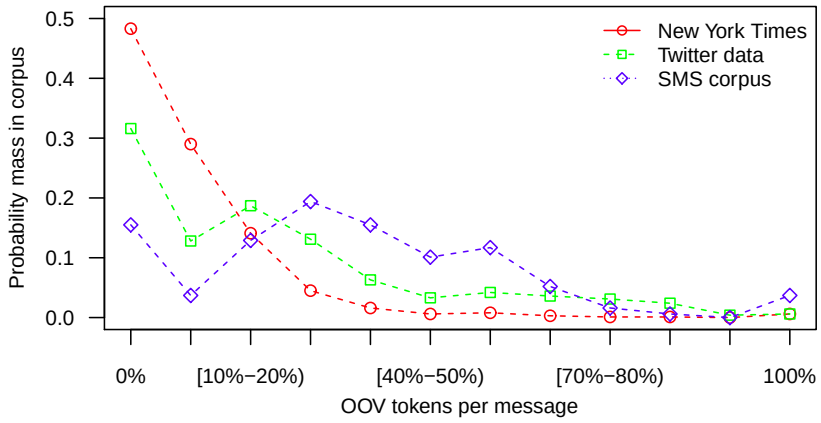

Fig. 1. Out-of-vocabulary word distribution in English Gigaword (NYT), Twitter and SMS data

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.

4. LEXICAL NORMALISATION

4.1. OOV Word Distribution and Type analysis

To get a sense of the relative need for lexical normalisation, we perform an 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 OOV token proportion, and plot the probability mass contained in each bin for a given text type. The three corpora we compare are the New York Times (NYT),[5] SMS,[6] and Twitter.[7] The results are presented in Figure 1.

Both SMS and Twitter have a relatively flat distribution, with Twitter having a particularly long tail: around 15% of tweets have 50% or more OOV tokens. Therefore, many OOV words in SMS and Twitter co-occur, and this makes context modeling difficult. In contrast, NYT shows a more Zipfian distribution, despite the large number of proper nouns 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 two lists of OOV terms — those found exclusively in SMS, and those found only in Twitter — and sorted each list by frequency. Manual analysis of high-frequency items in each list revealed that OOV words found only in SMS were largely personal names, while the Twitter-specific set, on the other hand, contained a more-heterogeneous col-

[5]Based on 44 million sentences from English Gigaword [David Graff 2003]
[6]Based on 12.6 thousand SMS messages from How and Kan [2005] and Choudhury et al. [2007].
[7]Based on 1.37 million tweets collected from the Twitter streaming API from Aug to Oct 2010, and filtered for monolingual English messages using `langid.py` [Baldwin and Lui 2010].

Table I. Categorisation of lexical variants

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

lection of OOVs including more lexical variants. Despite the difference in size of these datasets, this finding suggests that Twitter is a richer source of lexical variants, and hence a noisier data source, and that text normalisation for Twitter needs to be more nuanced than for SMS.

To further analyse the lexical variants 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 variants, and broke them down into categories, as listed in Table I. "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). "Other" is the remainder of the instances, which is predominantly made up of occurrences of spaces having been deleted between words (e.g. *sucha* "such a").[8] 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 I. Acknowledging other categorisation methods based on lexical variant format process [Thurlow 2003; Cook and Stevenson 2009; Xue et al. 2011], our classification is shaped to coordinate the downstream normalisation.

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

4.2. Token-based Normalisation Approach

The proposed lexical normalisation strategy involves two general steps: (1) confusion set generation, where we identify IV normalisation candidates for a given lexical variant type; (2) candidate selection, where we select the best standard form of the given lexical variant from the candidates generated in (1).

4.2.1. Confusion Set Generation. In generation of possible normalisation candidates, the following steps are utilised. First, inspired by Kaufmann and Kalita [2010], any repetitions of more than 3 letters are reduced back to 3 letters (e.g. *cooool* is reduced to *coool*). Second, IV words within a threshold of T_c in terms of character edit distance of a given OOV word are considered, a heuristic widely used in spell checkers. Third, the double metaphone algorithm [Philips 2000] is used to decode the pronunciation of all IV words; IV words within an edit distance of T_p of a 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 II, we list the recall and average size of the

[8]Which we don't touch, in accordance with our task definition.

ALGORITHM 1: Confusion Set Generation

Input: An OOV word (oov), edit distance thresholds for characters (T_c) and phonemic codes (T_p), a dictionary of IV words ($DICT$), and the proportion of candidates to retain after ranking by language model (R_{lm})
Output: Confusion set for OOV word (C_{set})

```
oov = RemoveRepetitions(oov);
C_set ← {};
forall the iv ∈ DICT do
    if CharacterEditDistance(oov, iv) ≤ T_c or PhonemicEditDistance(oov, iv) ≤ T_p then
        C_set ← C_set ∪ {iv};
    end
end
C_list = RankByTrigamModelScoreDesc(C_set);
num_list = GetLength(C_list) * R_lm;
index = 0;
C_set ← {};
repeat
    C_set ← C_set ∪ {C_list[index]}
    index++
until index ≥ num_list;
return C_set;
```

Table II. Recall and average number of candidates for different confusion set generation strategies

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

confusion set generated by the final two strategies with different threshold settings, based on our evaluation dataset (see Section 4.3).

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, and causes expensive computational cost. Furthermore, a smaller confusion set is easier for the downstream candidate selection as well. 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.

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. We truncate the ranking to the top 10% of candidates in the experiments, the recall drops back to 84% with a 90% reduction in candidates. The confusion set generation process is summarised in Algorithm 1.

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

4.2.2. Candidate Selection. We select the most likely candidate from the previously generated confusion set as the basis of normalisation using both lexical string similarity and contextual information. The information is combined linearly 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 non-linearly transformed to $\frac{1}{exp(ED)}$ so that smaller numbers correspond to higher similarity, as with the subsequence-based methods.

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 *gainst* "against" and *talkin* "talking". We calculate the ratio of the LCS over the maximum string length between lexical variant and the candidate, since the lexical variant can be either longer or shorter than (or the same size as) the standard form. For example, *mve* can represent either *me* or *move*, depending on context. We normalise these ratios so that the sum over candidates for each measure is made to be 1, 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 lexical variants 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 dependency features that are not restricted by contiguity.

First, we use the Stanford parser [Klein and Manning 2003; de Marneffe et al. 2006] to extract dependencies from the NYT corpus (see Section 4.1). 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 dependency 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. [9]

Although text messages are of a different genre to edited newswire text, we assume that words in the two genres participate in 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 lexical variants, 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.

[9]The confidence score is derived from the proportion of dependency tuples. For example, assume context word CW and OOV word O, and IV normalisation candidates for O of A and B which form dependency tuples (CW, A, +1), (CW, B, +1) and occur 200 and 300 times respectively in the corpus. The confidence score for A and B would be calculated as 0.4 and 0.6, respectively.

4.3. Dataset and Evaluation Metrics

The aim of our experiments is to compare the effectiveness of different methodologies over two datasets of short messages: (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 automatically pre-identified, and the annotators were requested to determine: (a) whether each OOV word was a lexical variant or not; and (b) in the case of tokens judged as lexical variants, what the standard form was, subject to the task definition outlined in Section 3. The total number of lexical variants contained in the SMS and Twitter datasets were 3849 and 1184, respectively.[10]

As discussed in Sections 2 and 3, much previous work on SMS data has assumed perfect lexical variant detection and focused only on the identification of standard forms. Here we also assume perfect detection of lexical variants in order to compare our proposed approach to previous methods. We consider token-level precision, recall and F-score ($\beta = 1$), and also evaluate using BLEU [Papineni et al. 2002] over the normalised form of each message. We consider the latter measure because SMT-based approaches to normalisation (which we compare our proposed method against) can lead to perturbations of the token stream, vexing evaluation using standard precision, recall and F-score.

$$P = \frac{\text{\# correctly normalised tokens}}{\text{\# normalised tokens}}$$

$$R = \frac{\text{\# correctly normalised tokens}}{\text{\# tokens requiring normalisation}}$$

$$F = \frac{2PR}{P+R}$$

4.4. Baselines and Benchmarks

4.4.1. Baselines. We compare our proposed approach to normalisation to some off-the-shelf tools and simple methods. As the first baseline, we use the Ispell spell checker to correct lexical variants.[11] We also consider a web-based language modeling approach to normalisation. For a given lexical variant, we first use our method for candidate set generation (Section 4.2.1) to identify plausible normalisation candidates. We then identify the lexical variant's left and right context tokens, and use the Web 1T 5-gram corpus [Brants and Franz 2006]. to determine the most frequent 3-gram (one word to each of the left and right of the lexical variant) or 5-gram (two words to each of the left and right). Lexical normalisation takes the form of simply identifying the centre word of the highest-frequency n-gram which matches the left/right context, and where the centre word is a member of the lexical variant's candidate set. Finally, we also consider a simple dictionary lookup method using the Internet slang dictionary (Section 3) where we substitute any usage of an OOV having an entry in the dictionary by its listed standard form.

4.4.2. Benchmarks. We further compare our proposed method against previous methods, which we take as benchmarks. We reimplemented the state-of-art noisy channel model of Cook and Stevenson [2009] and the SMT approach of Aw et al. [2006], widely used in SMS normalisation. We implement the SMT approach in Moses [Koehn et al.

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

[11] We use Ispell 3.1.20 with the `-w/-S` options to get the most probable correct word.

Table III. Candidate selection effectiveness over different datasets (*SC* = spell checker; *LM3* = 3-gram language model; *LM5* = 5-gram language model; *DL* = dictionary lookup; *NC* = SMS noisy channel model [Cook and Stevenson 2009]; *MT* = SMT [Aw et al. 2006]; *WS* = word similarity; *CS* = context support; *WC* = WS + DS; *DWC* = DL + WS + DS)

Dataset	*Eval*	*SC*	*LM3*	*LM5*	*DL*	*NC*	*MT*	*WS*	*CS*	*WC*	*DWC*
SMS	P	0.209	0.116	0.556	**0.927**	0.465	—	0.521	0.116	0.532	0.756
	R	0.134	0.064	0.017	0.597	0.464	—	0.520	0.116	0.531	**0.754**
	F	0.163	0.082	0.033	0.726	0.464	—	0.520	0.116	0.531	**0.755**
	BLEU	0.607	0.763	0.746	0.801	0.746	0.700	0.764	0.612	0.772	**0.876**
Twitter	P	0.277	0.110	0.324	**0.961**	0.452	—	0.551	0.194	0.571	0.753
	R	0.179	0.068	0.020	0.460	0.452	—	0.551	0.194	0.571	**0.753**
	F	0.217	0.083	0.037	0.622	0.452	—	0.551	0.194	0.571	**0.753**
	BLEU	0.788	0.779	0.766	0.861	0.857	0.728	0.878	0.797	0.884	**0.934**

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 the distribution of the SMS corpus. The 10-fold cross-validated BLEU score over this data is 0.81.

4.5. Results and Analysis

In Table III, we compare our method (*DWC*) with the baselines and benchmarks discussed above. Additionally, we determine the relative effectiveness of the component methods of our approach, namely dictionary lookup (*DL*), word similarity (*WS*), context support (*CS*), and combined word similarity and context support (*WC*).

From Table III, we see that the general performance of our proposed method over Twitter is better than that over the SMS dataset. To better understand this, we examined the annotations in the SMS corpus, and found them to be less conservative than ours, due to the different task specification. In our annotations, the annotators were instructed to only normalise lexical variants if they had high confidence of how to normalise, as with *talkin* "talking". For lexical variants where they couldn't be certain of the standard form, the tokens were left untouched. However, in the SMS corpus, annotations such as *sammis* is mistakenly recognised as a variant of "same", which actually represents a person name in the context. This leads to a performance drop for most of methods over the SMS corpus.

Among all the baselines in Table III, the spell checker (*SC*) performs marginally better than language model-based approaches (*LM3* and *LM5*), but is inferior to the dictionary lookup method, and receives the lowest BLEU score of all methods over the SMS dataset.

Both web n-gram approaches are relatively ineffective at lexical normalisation. The primary reason for this can be attributed to the simplicity of the context modelling. Comparing the different-order language models, it is evident that longer n-grams (i.e. more highly-specified context information) support normalisation with higher precision. Nevertheless, lexical context in Twitter data is noisy: many OOV words are surrounded by mentions of other Twitter users, hashtags, URLs and other OOV words, which are uncommon in other text genres. In the web n-gram approach, OOV words are mapped to the `<UNK>` flag in the Web 1T corpus construction process, leading to a loss of context information. Even the relaxed context constraints of the trigram method suffer from data sparseness, as indicated by the low recall. In fact, due to the temporal mismatch between the web n-gram corpus (harvested in 2006) and the Twitter data (harvested in late 2010), lexical variant contexts are often missing in the web n-gram data, limiting the performance of the web n-gram model for normalisation. Without the candidate filtering based on confusion sets, we observed that the web n-gram approach generated fluent-sounding normalisation candidates (e.g. *back, over, in, soon, home* and *events*) for *tomoroe* in *coming tomoroe* ("coming tomorrow") but which lack

semantic felicity with the original OOV word. This demonstrates the importance of candidate filtering as proposed.

The dictionary lookup method ("DL") unsurprisingly achieves the best precision, but the recall on Twitter is not competitive. Twitter normalisation clearly cannot be tackled with such a small-scale dictionary lookup approach, although it is an effective pre-processing strategy when combined with other wider-coverage normalisation approaches. Nevertheless, because of the very high precision of the dictionary lookup method, we will reconsider such an approach, but on a much larger-scale, in Section 6.

The noisy channel method of Cook and Stevenson [2009] (NC) shares similar features with our word similarity method (*WS*). However, when word similarity and context support are combined (*WC*), our method outperforms NC by about 7% and 12% in F-score over the SMS and Twitter datasets, respectively. This can be explained as follows. First, NC is type-based, so all token instances of a given lexical variant will have the same normalisation. However, the same lexical variant can correspond to different IV words, 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*. Our word similarity method does not make the assumption that each lexical variant has a unique standard form. Second, NC was developed specifically for SMS normalisation, based on observations about how lexical variants are typically formed in text messages, e.g. clipping is quite frequent in SMS. In Twitter, word lengthening for emphasis, such as *moviiie* "movie", is common, but this is not the case for text messages; NC therefore performs poorly on such lexical variants.

The SMT approach is relatively stable on the two datasets, but performs well below our method. This is due to the limitations of the training data: we obtain the lexical variants and their standard forms from the SMS corpus, but the lexical variants 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 lexical variants are not recognised and are therefore not normalised. This shows the shortcoming of supervised data-driven approaches that require annotated data to cover all possibilities of lexical variants in Twitter.

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 lexical variants emanate from morphophonemic variations. However, when combined with context support, the performance improves over word similarity 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 (*DWC*), in which lexical variants are first looked up in the slang dictionary, and only if no match is found do we apply our normalisation method.

As is common in research on text normalisation [Gouws et al. 2011; Liu et al. 2012], throughout this section we have assumed perfect detection of lexical variants. This is, of course, not practical for real-world applications, and in the following section we consider the task of identifying lexical variants.

5. TOKEN-BASED LEXICAL VARIANT DETECTION

A real-world end-to-end normalisation solution must be able to identify which tokens are lexical variants and require normalisation. In this section, we explore a context fitness-based approach for lexical variant detection. The task is to determine whether a given OOV word in context is a lexical variant or not, relative to its confusion set. To the best of our knowledge, we are the first to target the task of lexical variant 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 dependency-based features generated in Section 4.2.1.

To judge context fitness, we first train a linear kernel SVM classifier [Fan et al. 2008] on clean Twitter data, i.e. the subset of Twitter messages without OOV words (discussed in Section 4.2.1). Each target word is represented by a vector with dimensions corresponding to the IV words within a context window of three words to either side of the target, together with their relative positions in the form of `(word1,word2,position)` tuples, and with the feature value for a particular dimension set to the score for the corresponding tuple in the dependency bank. These vectors form the positive training exemplars. Negative exemplars are automatically constructed by replacing target words with highly-ranked candidates from their confusion set. For example, we extract a positive instance for the target word *book* with a dependency feature corresponding to the the tuple `(book,hotel,`-2`)`. A highly-ranked confusion of *book* is *hook*. We therefore form a negative instance for *hook* with a feature for the tuple `(hook,hotel,`-2`)`. In training, it is possible for the exact same feature vector to occur as both positive and negative exemplars. To prevent the positive exemplars from becoming contaminated through the automatic negative-instance generation, we remove all negative instances in such cases. The `(word1,word2,position)` features are sparse and sometimes lead to conservative results in lexical variant detection. That is, without valid features, the SVM classifier tends to label uncertain cases as correct (i.e. not requiring normalisation) rather than as lexical variants. This is arguably the right approach to normalisation, in choosing to under- rather than over-normalise in cases of uncertainty. This artificially-generated data is not perfect; however, this approach is appealing because the classifier does not require any manually-annotated data, as all training exemplars are constructed automatically.

To predict whether a given OOV word is a lexical variant, we form a feature vector as above for each of its confusion candidates. If the number of the OOV's candidates predicted to be positive by the model is greater than a threshold t_d, we consider the OOV to be a lexical variant; otherwise, the OOV is not deemed to be a lexical variant. We experiment with varying settings of $t_d \in \{1, 2, ..., 10\}$. Note that in an end-to-end normalisation system, for an OOV predicted to be a lexical variant, we would pass all its confusion candidates (not just those classified positively) to the candidate selection step; however, the focus of this section is only on the lexical variant detection task.

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, dependency-based features tolerate the noisy context word, e.g. given a lexical variant *seee*, its confusion candidate "see" can form `(see, film, +2)` in *film to seeee*, but not `(see, flm, +2)`. If we tolerate the context word variations assuming *flm* is "film", `(see, flm, +2)` would be also counted as `(see, film, +2)`.

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 test the impact of the w_d and t_d values on lexical variant detection effectiveness for Twitter messages, based on dependencies from either the NYT, or the Spinn3r blog corpus (Blog: Burton et al. [2009]), a large corpus of blogs which we also processed and parsed like Twitter data. The results for precision, recall and F-score are presented in Figure 2.

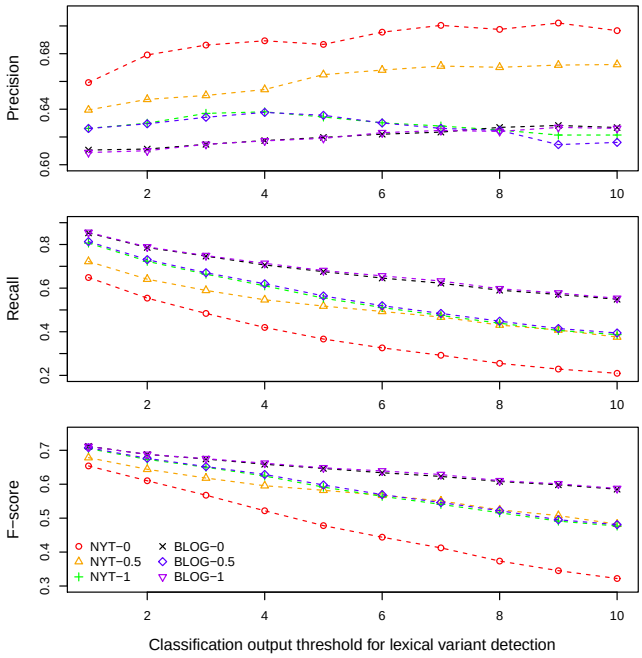

Fig. 2. Lexical variant detection precision, recall and F-score

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 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 between the two corpora with the best F-score is slight (when $t_d = 1$ and $w_d = 0.5$ on Blog corpus). The lexical variant proportion among all OOV words in the Twitter dataset is 55%. Overall, the best F-score is 71.2%, with a precision of 61.1% and recall of 85.3%. Clearly there is significant room for improvements in these results. One quick solution is to find as many named entities as possible to filter out the non-lexical variant OOV words. Owning to the extensive editing, sheer volume of data and up-to-date content, we choose Wikipedia article titles as a source of non-lexical variants which contains many named entities. However, the results in preliminary experiments with this data source do not lead to any improvement. By analysing the results, we found that terms from Wikipedia article titles are inappropriate for our task because they include many lexical variants such as *u* and *hw* which cause a decrease in recall.

We further consider the performance of a lexical variant detection method based on the Internet slang dictionary. In particular, if an OOV type has an entry in this dictionary, we consider all token instances of that type to be lexical variants; if a type is not in this dictionary, instances of that type are considered to be non-lexical variant OOVs. This very simple method achieves precision, recall, and F-score of 95.2%, 45.3%, and 61.4%, respectively. Although the performance of this dictionary-based method is

substantially below that of our best-performing method, we are encouraged by the very high precision of this method, particularly because of the previously-noted importance of not over-normalising.

The challenges of lexical variant detection, along with the high precision of dictionary-based methods at both lexical normalisation and lexical variant detection, lead us to consider an alternative dictionary-based approach to normalisation.

6. DICTIONARY-BASED TYPE NORMALISATION

6.1. Motivation and Feasibility Analysis

As discussed in the Section 4.5, dictionary lookup approaches to normalisation have been shown to have high precision but low recall. Frequent (lexical variant, standard form) pairs such as (u, you) are typically included in the dictionaries used by such methods, while less-frequent items such as $(g0tta, gotta)$ are generally omitted. Because of the degree of lexical creativity and large number of non-standard forms observed on Twitter, a wide-coverage normalisation dictionary would be expensive to construct manually. Based on the assumption that lexical variants occur in similar contexts to their standard forms, however, it should be possible to automatically construct a normalisation dictionary with wider coverage than is currently available.

Dictionary lookup is a type-based approach to normalisation, i.e. every token instance of a given type will always be normalised in the same way. However, lexical variants can be ambiguous, e.g. *y* corresponds to "you" in *yeah, y r right! LOL* but "why" in *AM CONFUSED!!! y you did that?*. Nevertheless, the relative occurrence of ambiguous lexical variants is small [Liu et al. 2011], and it has been observed that while shorter variants such as *y* are often ambiguous, longer variants tend to be unambiguous. For example *bthday* and *4eva* are unlikely to have standard forms other than "birthday" and "forever", respectively. Therefore, the normalisation lexicons we produce will only contain entries for OOVs with character length greater than a specified threshold, which are likely to have an unambiguous standard form.

Recently, Gouws et al. [2011] produced a small normalisation lexicon based on distributional similarity and string similarity [Lodhi et al. 2002]. Our method adopts a similar strategy using distributional/string similarity, but instead of constructing a small lexicon for preprocessing, we build a much wider-coverage normalisation dictionary and opt for a fully lexicon-based end-to-end normalisation approach. In contrast with the normalisation dictionaries which focus on very frequent lexical variants, we focus on moderate-frequency lexical variants of a minimum character length, which tend to have unambiguous standard forms; our intention is to produce normalisation lexicons that are complementary to those currently available. Furthermore, we investigate the impact of a variety of contextual and string similarity measures on the quality of the resulting lexicons. In summary, our dictionary-based normalisation approach is a lightweight end-to-end method which performs both lexical variant detection and normalisation, and thus is suitable for practical online pre-processing, despite its simplicity.

6.2. Word Type Normalisation

Our method for constructing a normalisation dictionary is as follows:

Input:. Tokenised English tweets
1.. Extract (OOV, IV) pairs based on distributional similarity.
2.. Re-rank the extracted pairs by string similarity.
Output:. A list of (OOV, IV) pairs ordered by string similarity; select the top-n pairs for inclusion in the normalisation lexicon.

In Step 1, we leverage large volumes of Twitter data to identify the most distributionally-similar IV type for each OOV type. The result of this process is a set of (OOV, IV) pairs, ranked by distributional similarity. The extracted pairs will include $(\text{lexical variant}, \text{standard form})$ pairs, such as $(tmrw, tomorrow)$, but will also contain false positives such as $(Tusday, Sunday)$ — *Tusday* is a lexical variant, but its standard form is not "Sunday" — and $(Youtube, web)$ — *Youtube* is an OOV named entity, not a lexical variant. Nevertheless, lexical variants are typically formed from their standard forms through regular processes [Thurlow 2003; Cook and Stevenson 2009; Xue et al. 2011] — e.g. the omission of characters — and from this perspective, *Sunday* and *web* are not plausible standard forms for *Tusday* and *Youtube*, respectively. In Step 2, we therefore capture this intuition in re-ranking the extracted pairs by string similarity. The top-n items in this re-ranked list then form the normalisation lexicon, which is based only on development data.

Although computationally-expensive to build, this dictionary can be created offline. Once built, it then offers a very fast approach to normalisation.

We can only reliably compute distributional similarity for types that are moderately frequent in a corpus. Nevertheless, many lexical variants are sufficiently frequent to be able to compute distributional similarity, and can potentially make their way into our normalisation lexicon. This approach is not suitable for normalising low-frequency lexical variants, nor is it suitable for shorter lexical variant types which — as discussed in Section 6 — are more likely to have an ambiguous standard form. Nevertheless, previously-proposed normalisation methods that can handle such phenomena also rely in part on a normalisation lexicon. The normalisation lexicons we create can therefore be easily integrated with previous approaches to form hybrid normalisation systems.

6.3. Contextually Similar Pair Generation

Our objective is to extract contextually-similar (OOV, IV) pairs from a large-scale collection of microblog data. Fundamentally, the surrounding words define the primary context, but there are different ways of representing context and different similarity measures we can use, which may influence the quality of generated normalisation pairs.

Intuitively, distributional similarity measures the context proximity of two words in a corpus, as follows: (1) represent a word's context by its surrounding words in a (large) feature vector. Each entry in the vector represents a particular word, usually in the form of a word frequency. (2) calculate the similarity between two context vectors based on some distance/similarity measure. For instance, *tmrw* and *tomorrow* in the following tweets share a number of context words in the vector, like *see*, *you* and *school*. This suggests they are distributionally similar.

— I don't wanna go to school *tmrw*
— okay off to work now . paipai . see you guys *tmrw* (:
— No school *tomorrow* or Tuesday woot!!!
— ah i can't wait to see you *tomorrow*

In representing the context, we experimentally explore the following factors: (1) context window size (from 1 to 3 tokens on both sides); (2) n-gram order of the context tokens (unigram, bigram, trigram); (3) whether context words are indexed for relative position or not; and (4) whether we use all context tokens, or only IV words. Because high-accuracy linguistic processing tools for Twitter are still under exploration [Liu et al. 2011; Gimpel et al. 2011; Ritter et al. 2011; Foster et al. 2011], we do not consider richer representations of context, for example, incorporating information about part-of-speech tags or syntax. We also experiment with a number of simple but widely-used geometric and information theoretic distance/similarity measures. In particular, we

use Kullback–Leibler (KL) divergence [Kullback and Leibler 1951], Jensen–Shannon (JS) divergence [Lin 1991], Euclidean distance and Cosine distance.

We use a corpus of 10 million English tweets to do parameter tuning over, and a larger corpus of tweets in the final candidate ranking. All tweets were collected from September 2010 to January 2011 via the Twitter API.[12] From the raw data we extract English tweets using an improved language identification tool [Lui and Baldwin 2011],[13] and then apply a simplified Twitter tokeniser (adapted from O'Connor et al. [2010]). We use the Aspell dictionary (v6.06)[14] to determine whether a word is IV, and only include in our normalisation dictionary OOV tokens with at least 64 occurrences in the corpus and character length ≥ 4, both of which were determined through empirical observation. For each OOV word type in the corpus, we select the most similar IV type to form (OOV, IV) pairs. To further narrow the search space, we only consider IV words which are morphophonemically similar to the OOV type, based on parameter tuning from Section 4.2.1 over the top-30% of most frequent IV words in the confusion set.

In order to evaluate the generated pairs, we randomly selected 1000 OOV words from the 10 million tweet corpus. We set up an annotation task on Amazon Mechanical Turk,[15] presenting five independent annotators with each word type (with no context) and asking for corrections where appropriate. For instance, given *tmrw*, the annotators would likely identify it as a non-standard variant of "tomorrow". For correct OOV words like *Wikileaks*, on the other hand, we would expect them to leave the word unchanged. If 3 or more of the 5 annotators make the same suggestion (in the form of either a canonical spelling or leaving the word unchanged), we include this in our gold standard for evaluation. In total, this resulted in 351 lexical variants and 282 correct OOV words, accounting for 63.3% of the 1000 OOV words. These 633 OOV words were used as (OOV, IV) pairs for parameter tuning. The remainder of the 1000 OOV words were ignored on the grounds that there was not sufficient consensus amongst the annotators.[16]

Contextually-similar pair generation aims to include as many correct normalisation pairs as possible. We evaluate the quality of the normalisation pairs using "Cumulative Gain" (CG):

$$\text{CG} = \sum_{i=1}^{N'} rel'_i$$

Suppose there are N' correct generated pairs (oov_i, iv_i), each of which is weighted by rel'_i, the frequency of oov_i to indicate its relative importance; for example, $(\textit{thinkin}, \textit{thinking})$ has a higher weight than $(\textit{g0tta}, \textit{gotta})$ because *thinkin* is more frequent than *g0tta* in our corpus. In this evaluation we don't consider the position of normalisation pairs, and nor do we penalise incorrect pairs. Instead, we push distinguishing between correct and incorrect pairs into the downstream re-ranking step in which we incorporate string similarity information.

[12] `https://dev.twitter.com/docs/streaming-api/methods`

[13] A much-updated version of the language identification method used to construct the lexical normalisation dataset, trained over a larger sample of datasets, with feature selection based on the notion of domain generalisation.

[14] `http://aspell.net/`

[15] `https://www.mturk.com/mturk/welcome`

[16] Note that the objective of this annotation task is to identify lexical variants that have agreed-upon standard forms irrespective of context, as a special case of the more general task of lexical normalisation (where context may or may not play a significant role in the determination of the normalisation).

Table IV. The five best parameter combinations in the exhaustive search of parameter combinations

Rank	Window size	n-gram	Positional index?	Lex. choice	Sim/distance measure	$\log(\text{CG})$
1	± 3	2	Yes	All	KL divergence	19.571
2	± 3	2	No	All	KL divergence	19.562
3	± 2	2	Yes	All	KL divergence	19.562
4	± 3	2	Yes	IVs	KL divergence	19.561
5	± 2	2	Yes	IVs	JS divergence	19.554

Table V. Parameter sensitivity analysis measured as $\log(\text{CG})$ for correctly-generated pairs. We tune one parameter at a time, using the default (underlined) setting for other parameters; the non-exhaustive best-performing setting in each case is indicated in **bold**.

Window size		n-gram		Positional index?		Lexical choice		Similarity/distance measure	
± 1	19.325	1	19.328	Yes	**19.328**	IVs	**19.335**	KL divergence	**19.328**
± 2	19.327	2	**19.571**	No	19.263	All	19.328	Euclidean	19.227
± 3	**19.328**	3	19.324					JS divergence	19.311
								Cosine	19.170

Given the development data and CG, we run an exhaustive search of parameter combinations over our development corpus. The five best parameter combinations are shown in Table IV. We notice the CG is almost identical for the top combinations. As a context window size of 3 incurs a heavy processing and memory overhead over a size of 2, we use the 3rd-best parameter combination for subsequent experiments, namely: context window of ± 2 tokens, token bigrams, positional index, and KL divergence as our distance measure.

To better understand the sensitivity of the method to each parameter, we perform a post-hoc parameter analysis relative to a default setting (as underlined in Table V), altering one parameter at a time. The results in Table V show that bigrams outperform other n-gram orders by a large margin (note that the evaluation is based on a $\log$ scale), and information-theoretic measures are superior to the geometric measures. Furthermore, it also indicates using the positional indexing better captures context. However, there is little to distinguish context modelling with just IV words or all tokens. Similarly, the context window size has relatively little impact on the overall performance, supporting our earlier observation from Table IV.

6.4. Pair Re-ranking by String Similarity

Once the contextually-similar (OOV, IV) pairs are generated using the selected parameters in Section 6.3, we further re-rank this set of pairs in an attempt to boost morphophonemically-similar pairs like $(\textit{bananaz}, \textit{bananas})$, and penalise noisy pairs like $(\textit{paninis}, \textit{beans})$.

Instead of using the small 10 million tweet corpus, from this step onwards, we use a larger corpus of 80 million English tweets (collected over the same period as the development corpus) to develop a larger-scale normalisation dictionary. This is because once pairs are generated, re-ranking based on string comparison is much faster. We only include in the dictionary OOV words with a token frequency > 15 to include more OOV types than in Section 6.3, and again apply a minimum length cutoff of 4 characters.

To measure how well our re-ranking method promotes correct pairs and demotes incorrect pairs (including both OOV words that should not be normalised, e.g. $(\textit{Youtube}, \textit{web})$, and incorrect normalisations for lexical variants, e.g. $(\textit{bcuz}, \textit{cause})$), we modify our evaluation metric from Section 6.3 to evaluate the *ranking* at different points, using Discounted Cumulative Gain (DCG@N: Jarvelin and Kekalainen [2002]):

$$\text{DCG@}N = rel_1 + \sum_{i=2}^{N} \frac{rel_i}{\log_2(i)}$$

where rel_i again represents the frequency of the OOV, but it can be a gain (a positive number) or loss (a negative number), depending on whether the ith pair is correct or incorrect. Because we also expect correct pairs to be ranked higher than incorrect pairs, DCG@N takes both factors into account.

Given the generated pairs and the evaluation metric, we first consider three baselines: no re-ranking (i.e. the final ranking is that of the contextual similarity scores), and re-rankings of the pairs based on the frequencies of the OOVs in the Twitter corpus, and the IV unigram frequencies in the Google Web 1T corpus [Brants and Franz 2006] to get less-noisy frequency estimates. We also compared a variety of re-rankings based on a number of string similarity measures that have been previously considered in normalisation work (reviewed in Section 2). We experiment with a series of string similarity measures: standard edit distance [Levenshtein 1966], which is the minimum number of character-level insertions/deletions/substitutions to transform one string to another (from a lexical variant to its normalisation in this paper); edit distance over double metaphone codes (phonetic edit distance: [Philips 2000]); longest common subsequence ratio over the consonant edit distance of the paired words (hereafter, denoted as consonant edit distance: [Contractor et al. 2010]); a string subsequence kernel [Lodhi et al. 2002], which measures common character subsequences of length n between (OOV, IV) pairs. Because it is computationally expensive to calculate similarity for larger n, we choose n=2, following Gouws et al. [2011].

In Figure 3, we present the DCG@N results for each of our ranking methods at different rank cut-offs. Ranking by OOV frequency is motivated by the assumption that lexical variants are frequently used by social media users. This is confirmed by our findings that lexical pairs like $(\textit{goin}, \textit{going})$ and $(\textit{nite}, \textit{night})$ are at the top of the ranking. However, many proper nouns and named entities are also used frequently and ranked at the top, mixed with lexical variants like $(\textit{Facebook}, \textit{speech})$ and $(\textit{Youtube}, \textit{web})$. In ranking by IV word frequency, we assume the lexical variants are usually derived from frequently-used IV equivalents, e.g. $(\textit{abou}, \textit{about})$. However, many less-frequent lexical variant types have high-frequency (IV) normalisations. For instance, the highest-frequency IV word *the* has more than 40 OOV lexical variants, such as *tthe* and *thhe*. These less-frequent types occupy the top positions, reducing the cumulative gain. Compared with these two baselines, ranking by default contextual similarity scores delivers promising results. It successfully ranks many more intuitive normalisation pairs at the top, such as $(\textit{2day}, \textit{today})$ and $(\textit{wknd}, \textit{weekend})$, but also ranks some incorrect pairs highly, such as $(\textit{needa}, \textit{gotta})$.

The string similarity-based methods perform better than our baselines in general. Through manual analysis, we found that standard edit distance ranking is fairly accurate for lexical variants with low edit distance to their standard forms, e.g. $(\textit{thinkin}, \textit{thinking})$. Because this method is based solely on the number of character edits, it fails to identify heavily-altered variants like $(\textit{tmrw}, \textit{tomorrow})$. Consonant edit distance favours pairs with longer common subsequences, and therefore places many longer words at the top of the ranking. Edit distance over double metaphone codes (phonetic edit distance) performs particularly well for lexical variants that include character repetitions — commonly used for emphasis on Twitter — because such repetitions do not typically alter the phonetic codes. Compared with the other methods, the string subsequence kernel delivers encouraging results. As N (the lexicon size cut-off) increases, the performance drops more slowly than the other methods. Although this method fails to rank heavily-altered variants such as $(\textit{4get}, \textit{forget})$ highly, it typically works well for longer words. Given that we focus on longer OOVs (specifically those longer than 4 characters), this ultimately isn't a great handicap.

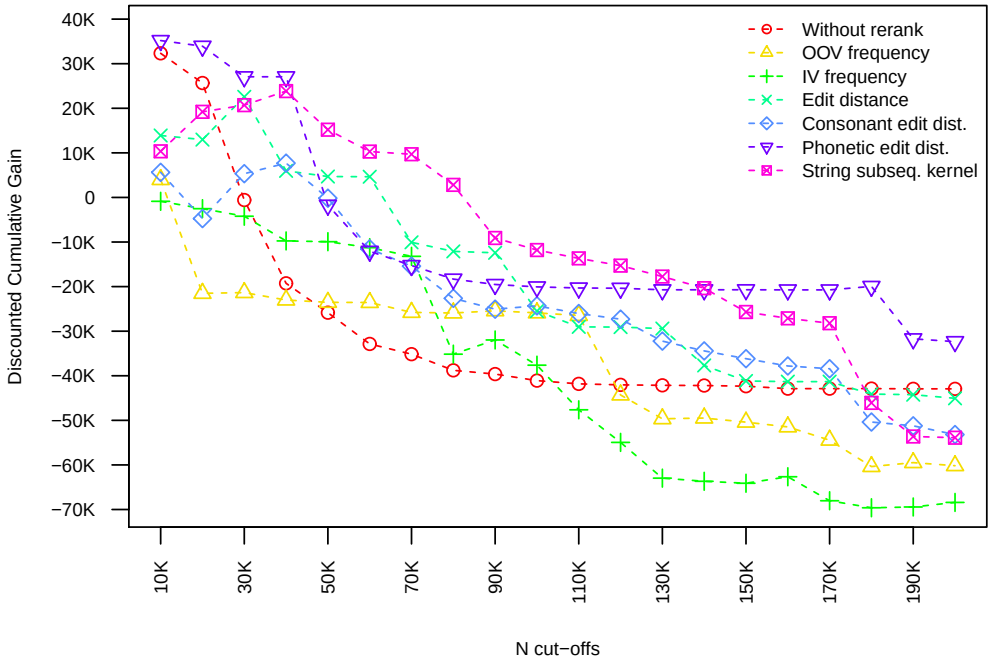

Fig. 3. Re-ranking based on different string similarity methods.

7. EVALUATION OF DICTIONARY-BASED NORMALISATION

Given the re-ranked pairs from Section 6.4, here we apply them to a token-level normalisation task, once again using the normalisation dataset from Section 4.3.

7.1. Metrics

We use the same standard evaluation metrics of precision (P), recall (R) and F-score (F) as detailed in Section 4.3. In addition, we also consider the false alarm rate (FA) and word error rate (WER), as shown below. FA measures the negative effects of applying normalisation: a good approach to normalisation should not (incorrectly) normalise tokens that are already in their standard form and do not require normalisation.[17] WER, like F-score, shows the overall benefits of normalisation, but unlike F-score, measures how many token-level edits are required for the output to be the same as the ground truth data. In general, dictionaries with a high F-score/low WER and low FA are preferable.

$$FA = \frac{\text{\# incorrectly normalised tokens}}{\text{\# normalised tokens}}$$

$$WER = \frac{\text{\# token edits needed after normalisation}}{\text{\# all tokens}}$$

[17] FA + P $\leq$ 1 because some lexical variants might be incorrectly normalised.

Table VI. Normalisation results using our derived dictionaries (contextual similarity (C-dict); double metaphone rendering (DM-dict); string subsequence kernel scores (S-dict)), the dictionary of Gouws et al. [2011] (GHM-dict), the Internet slang dictionary (HB-dict) in Section 4.5, and combinations of these dictionaries. Furthermore, we combine the dictionaries with the normalisation method of Gouws et al. [2011] (GHM-norm) and the combined unsupervised approach in (HB-norm) Section 4.2.2. In addition, we also compare the context sensitive normalisation on cleaned text after the lexicon lookup normalisation attached with *.

Method	Precision	Recall	F-Score	False Alarm	Word Error Rate
C-dict	0.474	0.218	0.299	0.298	0.103
DM-dict	0.727	0.106	0.185	0.145	0.102
S-dict	0.700	0.179	0.285	0.162	0.097
HB-dict	0.915	0.435	0.590	0.048	0.066
GHM-dict	**0.982**	0.319	0.482	**0.000**	0.076
HB-dict+S-dict	0.840	0.601	0.701	0.090	0.052
GHM-dict+S-dict	0.863	0.498	0.632	0.072	0.061
HB-dict+GHM-dict	0.920	0.465	0.618	0.045	0.063
HB-dict+GHM-dict+S-dict	0.847	0.630	**0.723**	0.086	**0.049**
GHM-dict+GHM-norm	0.338	0.578	0.427	0.458	0.135
HB-dict+GHM-dict+S-dict+GHM-norm	0.406	0.715	0.518	0.468	0.124
HB-dict+HB-norm	0.515	0.771	0.618	0.332	0.081
HB-dict+GHM-dict+S-dict+HB-norm	0.527	0.789	0.632	0.332	0.079
HB-dict+GHM-dict+S-dict+HB-norm*	0.528	**0.791**	0.633	0.332	0.079

7.2. Results and Analysis

We select the three best re-ranking methods, and best cut-off N for each method, based on the highest DCG@N value for a given method over the development data, as presented in Figure 3. Namely, they are string subsequence kernel (S-dict, N=40,000), double metaphone edit distance (DM-dict, N=10,000) and default contextual similarity without re-ranking (C-dict, N=10,000).[18]

We evaluate each of the learned dictionaries in Table VI. We also compare each dictionary with the performance of the manually-constructed Internet slang dictionary (HB-dict) used in Section 4.5, the small automatically-derived dictionary of Gouws et al. [2011] (GHM-dict), and combinations of the different dictionaries. In addition, the contribution of these dictionaries in hybrid normalisation approaches is presented, in which we first normalise OOVs using a given dictionary (combined or otherwise), and then apply the normalisation method of Gouws et al. [2011] based on consonant edit distance (GHM-norm), or the approach of Han and Baldwin [2011] based on the summation of many unsupervised approaches (HB-norm), to the remaining OOVs. Results are shown in Table VI, and discussed below.

7.2.1. Individual Dictionaries. Overall, the individual dictionaries derived by the re-ranking methods (DM-dict, S-dict) perform better than those based on contextual similarity (C-dict) in terms of precision and false alarm rate, indicating the importance of re-ranking. Even though C-dict delivers higher recall — indicating that many lexical variants are correctly normalised — this is offset by its high false alarm rate, which is particularly undesirable in normalisation. Because S-dict has better performance than DM-dict in terms of both F-score and WER, and a much lower false alarm rate than C-dict, subsequent results are presented using S-dict only.

Both HB-dict and GHM-dict achieve better than 90% precision with moderate recall. Compared to these methods, S-dict is not competitive in terms of either precision or recall. This result seems rather discouraging. However, considering that S-dict is an automatically-constructed dictionary targeting lexical variants of varying frequency, it is not surprising that the precision is worse than that of HB-dict — which is manually-

[18]We also experimented with combining ranks using Mean Reciprocal Rank. However, the combined rank didn't improve performance on the development data. We plan to explore other ranking aggregation methods in future work.

constructed — and GHM-dict — which includes entries only for more-frequent OOVs for which distributional similarity is more accurate. Additionally, the recall of S-dict is hampered by the restriction on lexical variant token length of 4 characters.

7.2.2. Combined Dictionaries. Next we look to combining HB-dict, GHM-dict and S-dict. In combining the dictionaries, a given OOV word can be listed with different standard forms in different dictionaries. In such cases we use the following preferences for dictionaries — motivated by our confidence in the normalisation pairs of the dictionaries — to resolve conflicts: HB-dict $>$ GHM-dict $>$ S-dict.

When we combine dictionaries in the second section of Table VI, we find that they contain complementary information: in each case the recall and F-score are higher for the combined dictionary than any of the individual dictionaries. The combination of HB-dict+GHM-dict produces only a small improvement in terms of F-score over HB-dict (the better-performing dictionary) suggesting that, as claimed, HB-dict and GHM-dict share many frequent normalisation pairs. HB-dict+S-dict and GHM-dict+S-dict, on the other hand, improve substantially over HB-dict and GHM-dict, respectively, indicating that S-dict contains markedly different entries to both HB-dict and GHM-dict. The best F-score and WER are obtained using the combination of all three dictionaries, HB-dict+GHM-dict+S-dict. Furthermore, the difference between the results using HB-dict+GHM-dict+S-dict and HB-dict+GHM-dict is statistically significant ($p < 0.01$), based on the computationally-intensive Monte Carlo method of Yeh [2000], demonstrating the contribution of S-dict.

7.2.3. Hybrid Approaches. So far, we have discussed using approaches using context sensitive normalisation (Section 4.2) and dictionary-based type normalisation (Section 6). The methods of Gouws et al. [2011] (i.e. GHM-dict+GHM-norm) and our proposed token-based hybrid approach (i.e. HB-dict+HB-norm) have lower precision and higher false alarm rates than the dictionary-based approaches; this is largely caused by lexical variant detection errors. When doing such comparisons, we report results that do not assume perfect detection of lexical variants, unlike the original published results in each case. Using all dictionaries in combination with these methods — HB-dict+GHM-dict+S-dict+GHM-norm and HB-dict+GHM-dict+S-dict+HB-norm — gives some improvements, but the false alarm rates remain high. A larger dictionary helps in improving the F-score and reducing the WER.

7.2.4. Impact of Context. As mentioned in Section 4.5, the disappointing performance of context features is partially attributable to noisy contexts, as neighbouring lexical variants mutually reduce the usable context of each other. To counter this effect, we apply context-sensitive token-based normalisation on the basis of the already partially normalised text (through our best dictionary) and compare its performance with token-based normalisation using the original unnormalised text, as shown in the last two rows of Table VI. This quantifies the relative impact of dictionary-based pre-normalisation on context-sensitive normalisation.

The results indicate that partial pre-normalisation has only a very slight effect. Analysis of the two methods led to the finding that only 45 tokens were altered by the context-sensitive normalisation. That is, most lexical variants are already normalised by the lexicon in pre-normalisation, and it is not surprising that the context-sensitive lexical normalisation step had little impact.

We further analysed the 45 instances which the context-sensitive normalisation modified, and found that cleaned text does indeed help in context-sensitive normalisation, as shown in Table VII. When presented with the noisy context *sorryy*, the lexical variant *im* is incorrectly normalised to *him*, however, when the context is cleaned — i.e. *sorryy* is restored to *sorry* — *im* is correctly normalised to (“i’m”), as both the lan-

Table VII. Example where cleaned text helps context sensitive normalisation

Data label	messages
Noisy input message	@username damn that sucks *im sorryy*) :
Normalisation on original message	@username damn that sucks *him* sorry) :
Normalisation on cleaned message	@username damn that sucks i'm sorry) :
Correct normalised message (oracle)	@username damn that sucks i'm sorry) :

Table VIII. Error types in the combined dictionary (HB-dict+GHM-dict+S-dict)

Error type	OOV	Standard form	
		Dict.	Gold
(a) plurals	*playe*	*players*	*player*
(b) negation	*unlike*	*like*	*dislike*
(c) possessives	*anyones*	*anyone*	*anyone's*
(d) correct OOVs	*iphone*	*phone*	*iphone*
(e) test data errors	*durin*	*during*	*durin*
(f) ambiguity	*siging*	*signing*	*singing*

Table IX. S-dict normalisation results broken down according to OOV token length. Recall is presented both over the subset of instances of length $\geq N$ in the data ("Recall ($\geq N$)"), and over the entirety of the dataset ("Recall (all)"); "#Variants" is the number of token instances of the indicated length in the test dataset.

Length cut-off (N)	#Variants	Precision	Recall ($\geq N$)	Recall (all)	False Alarm
$\geq$4	556	0.700	0.381	0.179	0.162
$\geq$5	382	0.814	0.471	0.152	0.122
$\geq$6	254	0.804	0.484	0.104	0.131
$\geq$7	138	0.793	0.471	0.055	0.122

guage model-based and dependency-based context feature strongly support the usage of "i'm sorry". Other encouraging cases where pre-normalisation with the dictionary aids context-sensitive normalisation are shown in Table VII. In the bulk of cases, however, the updated context led to a different, but still incorrect, normalisation candidate, as compared to the simple case of no pre-normalisation. Clearly, therefore, more work can be done at the interface between the two methods.

Despite the limitations of a pure dictionary-based approach to normalisation — discussed in Section 6.2 — the current best practical approach to normalisation is to use a lexicon, combining hand-built and automatically-learned normalisation dictionaries.

7.3. Discussion and Error Analysis

We first manually analyse the errors in the combined dictionary (HB-dict+GHM-dict+S-dict) and give examples of each error type in Table VIII. The most frequent word errors are caused by slight morphological variations, including plural forms (a), negations (b), possessive cases (c), and OOVs that are correct and do not require normalisation (d). In addition, we also notice some missing annotations where lexical variants are skipped by human annotations but captured by our method (e). Ambiguity (f) definitely exists in longer OOVs, however, these cases do not appear to have a strong negative impact on the normalisation performance. An example of a remaining miscellaneous error is *bday* "birthday", which is mis-normalised as *day*.

To further study the influence of OOV word length relative to the normalisation performance, we conduct a fine-grained analysis of the performance of the derived dictionary (S-dict) in Table IX, broken down across different OOV word lengths. The results generally support our hypothesis that our method works better for longer OOV words. The derived dictionary is much more reliable for longer tokens (length 5, 6, and 7 characters) in terms of precision and false alarm. Although the recall is relatively

modest, in the future we intend to improve recall by mining more normalisation pairs from larger collections of microblog data.

8. EXTRINSIC EVALUATION OF LEXICAL NORMALISATION

Having proposed a number of approaches to lexical normalisation, and evaluating those methods directly, we now evaluate the impact of normalisation in an applied setting. When NLP tools trained on more conventional text are applied to social media, their performance is hampered in part due to the presence of lexical variants (as discussed in Section 1). We therefore hypothesise that the performance of such tools might improve if lexical normalisation is applied after tokenisation, and before subsequent processing.

In this section we test the above hypothesis on a Twitter part-of-speech (POS) tagging task. We choose POS tagging for the following reasons: (1) the impact of lexical normalisation is readily-observed, as it is easy to compare the POS tags for the original and normalised texts; (2) off-the-shelf part-of-speech taggers are available for both more-conventional text [Toutanova et al. 2003] and social media [Gimpel et al. 2011]; (3) a human-annotated Twitter POS tagging dataset is publicly available [Gimpel et al. 2011].

The Part-of-speech tagging dataset of Gimpel et al. [2011] consists of 1827 tokenised and annotated messages from Twitter.[19] 500 messages — referred to as the test set — are held out for test purposes, with the rest of the data being used for training and development, as described in Gimpel et al. [2011]. For each message in the test set, we apply the best-performing dictionary-based normalisation method from Section 7.2, namely HB-dict+GHM-dict+S-dict. However, when substituting words, we also consider information about the case of the original tokens, as this information is known to be important for Twitter POS tagging [Gimpel et al. 2011]. Specifically, the case of the first and last characters of the normalised word form are set to the case of the first and last characters of the original token, respectively. All the other characters of the normalised form are set to the case of the middle character of the original token. For example, *Todei*, *WKEND* and *tmrw* are normalised as "Today", "WEEKEND" and "tomorrow", respectively.

We compare the performance of the Twitter-specific POS tagger ("$\text{POS}_{\text{twitter}}$") to that of a standard off-the-shelf tool, the Stanford POS tagger ("$\text{POS}_{\text{Stanford}}$": Toutanova et al. [2003]). However, these taggers use different tagsets: $\text{POS}_{\text{twitter}}$ uses a much more coarse-grained tagset than the Penn Treebank POS tagset that is used by $\text{POS}_{\text{Stanford}}$. We are interested in the performance of a conventional off-the-shelf tool, and therefore do not re-train $\text{POS}_{\text{Stanford}}$ on the POS-annotated tweets. Instead, we manually devised a lossy mapping from the fine-grained $\text{POS}_{\text{Stanford}}$ tagset to that of $\text{POS}_{\text{twitter}}$. In this mapping, finer-grained tags unique to $\text{POS}_{\text{Stanford}}$ (e.g. *VBP* and *VBN*) are mapped to coarser-grained $\text{POS}_{\text{twitter}}$ tags (e.g. *V*).[20]

We apply $\text{POS}_{\text{twitter}}$ and $\text{POS}_{\text{Stanford}}$ to the test set, both with and without first applying normalisation. We use accuracy to measure performance, excluding tokens in the gold-standard with a Twitter-specific POS tag from the calculation (as there is no way of reliably mapping onto them from the Penn POS tagset). Results are shown in Table X. First, we compare the performance of $\text{POS}_{\text{Stanford}}$ on the original tweets to its performance on the normalised tweets. The accuracy on normalised text is 1.6 percentage points higher than that on the original text. In total, 108 more tokens are correctly

[19] `http://ark-tweet-nlp.googlecode.com/files/twpos-data-v0.2.tar.gz`

[20] The test set provides tokenised tweets, but contains some tokenisation errors, e.g. *"Success* is tokenised as a single token, instead of as the pair of tokens *"* and *Sucess*. In the small number of such cases we manually correct the tokens output from the POS tagger to be consistent with the test set tokenisation.

Table X. Comparison of accuracy of $\text{POS}_{\text{twitter}}$ (a Twitter POS tagger) and $\text{POS}_{\text{Stanford}}$ (a general-purpose POS tagger) applied to the original and normalised tweets in the test set. The total number of correct tags is also shown.

Tagger	Text	% accuracy	# correct tags
$\text{POS}_{\text{Stanford}}$	original	68.4	4753
$\text{POS}_{\text{Stanford}}$	normalised	70.0	4861
$\text{POS}_{\text{twitter}}$	original	95.2	6819
$\text{POS}_{\text{twitter}}$	normalised	94.7	6780

tagged when lexical normalisation is used. We observe that most of this improvement is for nouns and verbs. Although the improvement in performance is small, it is statistically significant ($p < 0.01$) using the method of Yeh [2000]. Furthermore, only 275 tokens in the test set are normalised (i.e. the remaining tokens are unchanged through normalisation). It could be the case that more normalisation would lead to a greater improvement in POS tagging performance for noisier text containing more lexical variants.

We further consider the impact of normalisation on $\text{POS}_{\text{twitter}}$, the Twitter-specific tagger. In this case the performance on pre-normalised tweets drops slightly over that for the original messages, indicating that normalising the input hurts the performance of $\text{POS}_{\text{twitter}}$. This is somewhat expected because some features used by $\text{POS}_{\text{twitter}}$ are derived from noisy tokens: when the input is normalised, some of these features are not present, e.g. features capturing cooccurrence with lexical variants.

In summary, these preliminary comparisons show the influence of normalisation on the task of POS tagging for Twitter. In terms of cost, using a conventional off-the-shelf tool (e.g. $\text{POS}_{\text{Stanford}}$) with normalisation is the cheapest option, and would obviate the need for the development of a Twitter-specific tool such as $\text{POS}_{\text{twitter}}$. Not surprisingly, building a tool specific to the target domain yields the best performance. However, this comes at the substantial overhead of developing a specific tagset, manually annotating training data, and developing the tagger itself.

9. CONCLUSION AND FUTURE WORK

In this paper, we have proposed the task of normalising lexical variants to their canonical forms for short text messages in social media, e.g. Twitter and SMS data. We first analyse the in-domain OOV word types and distributions, and propose a detection-and-normalisation approach using both contextual and string similarity information. The proposed method beat other benchmark methods in token-based normalisation, however, most of these conventional methods suffer from the poor performance of lexical variant detection, which makes them less practical in real normalisation. Encouraged by the dictionary-based normalisation performance, we moved on to use contextual/string similarity information to build a type-based normalisation lexicon with particular focus on context-insensitive lexical variants. Although the proposed type-based method has the limitation that it cannot capture context or disambiguate different usages of the same token, in empirical evaluation, we showed it to achieve state-of-the-art accuracy, with broader coverage than existing dictionaries and reasonable precision. Furthermore, this dictionary-lookup approach combines the detection and normalisation of lexical variants into a simple, lightweight solution which is suitable for processing high-volume microblog feeds.

Given the encouraging results of the dictionary-based method, we primarily intend to improve the quantity and quality of the mined lexicon in future work, e.g. (1) enriching our dictionary entries by leveraging the constantly-growing volume of microblog data; and (2) exploring alternative ways to combine distributional and string similarity for better quality lexical candidate generation. Ultimately, context-sensitive nor-

malisation is much more powerful than the lexicon-based approach, and we also plan to explore context-sensitive methods for token-based normalisation given different potential normalisations. We only consider one-to-one token level normalisation in this paper, and plan to further expand the scope of the task to include many-to-many token normalisation.

Acknowledgements

We would like to thank anonymous reviewers for their insightful comments and valuable suggestions.

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

AW, A., ZHANG, M., XIAO, J., AND SU, J. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Sydney, Australia, 33–40.

BALDWIN, T. AND LUI, M. 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*. Los Angeles, USA, 229–237.

BENSON, E., HAGHIGHI, A., AND BARZILAY, R. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, 389–398.

BRANTS, T. AND FRANZ, A. 2006. Web 1T 5-gram Version 1.

BRILL, E. AND MOORE, R. C. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Hong Kong, 286–293.

BURTON, K., JAVA, A., AND SOBOROFF, I. 2009. The ICWSM 2009 Spinn3r Dataset. In *Proceedings of the Third Annual Conference on Weblogs and Social Media*. San Jose, USA.

CHOUDHURY, M., SARAF, R., JAIN, V., MUKHERJEE, A., SARKAR, S., AND BASU, A. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition 10*, 157–174.

CONTRACTOR, D., FARUQUIE, T. A., AND SUBRAMANIAM, L. V. 2010. Unsupervised cleansing of noisy text. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Beijing, China, 189–196.

COOK, P. AND STEVENSON, S. 2009. An unsupervised model for text message normalization. In *CALC '09: Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*. Boulder, USA, 71–78.

DAVID GRAFF, C. C. 2003. *English Gigaword*. Linguistic Data Consortium, Philadelphia, USA. `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T05`.

DE MARNEFFE, M., MACCARTNEY, B., AND MANNING, C. D. 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.

FAN, R.-E., CHANG, K.-W., HSIEH, C.-J., WANG, X.-R., AND LIN, C.-J. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research 9*, 1871–1874.

FOSTER, J., ÇETINOGLU, O., WAGNER, J., ROUX, J. L., HOGAN, S., NIVRE, J., HOGAN, D., AND VAN GENABITH, J. 2011. #hardtoparse: POS Tagging and Parsing the Twitterverse. In *Analyzing Microtext: Papers from the 2011 AAAI Workshop*. AAAI Workshops Series, vol. WS-11-05. San Francisco, CA, USA, 20–25.

GIMPEL, K., SCHNEIDER, N., O'CONNOR, B., DAS, D., MILLS, D., EISENSTEIN, J., HEILMAN, M., YOGATAMA, D., FLANIGAN, J., AND SMITH, N. A. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, 42–47.

GONZÁLEZ-IBÁÑEZ, R., MURESAN, S., AND WACHOLDER, N. 2011. Identifying sarcasm in Twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*. Portland, Oregon, USA, 581–586.

GOUWS, S., HOVY, D., AND METZLER, D. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*. Edinburgh, Scotland, UK, 82–90.

HAN, B. AND BALDWIN, T. 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*. Portland, Oregon, USA, 368–378.

HAN, B., COOK, P., AND BALDWIN, T. 2012. Automatically constructing a normalisation dictionary for microblogs. To appear in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL 2012)*. Jeju, Korea.

HOW, Y. AND KAN, M.-Y. 2005. Optimizing predictive text entry for short message service on mobile phones. In *Human Computer Interfaces International (HCII 05)*. Las Vegas, USA.

IZUMI, E., UCHIMOTO, K., SAIGA, T., SUPNITHI, T., AND ISAHARA, H. 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*. Sapporo, Japan, 145–148.

JARVELIN, K. AND KEKALAINEN, J. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems 20,* 4.

JIANG, L., YU, M., ZHOU, M., LIU, X., AND ZHAO, T. 2011. Target-dependent Twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. Portland, Oregon, USA, 151–160.

KAUFMANN, J. AND KALITA, J. 2010. Syntactic normalization of Twitter messages. In *International Conference on Natural Language Processing*. Kharagpur, India.

KLEIN, D. AND MANNING, C. D. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*. Whistler, Canada, 3–10.

KOEHN, P., HOANG, H., BIRCH, A., CALLISON-BURCH, C., FEDERICO, M., BERTOLDI, N., COWAN, B., SHEN, W., MORAN, C., ZENS, R., DYER, C., BOJAR, O., CONSTANTIN, A., AND HERBST, E. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Prague, Czech Republic, 177–180.

KULLBACK, S. AND LEIBLER, R. A. 1951. On information and sufficiency. *Annals of Mathematical Statistics 22*, 49–86.

LAFFERTY, J. D., MCCALLUM, A., AND PEREIRA, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA, 282–289.

LEVENSHTEIN, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady 10*, 707–710.

LI, M., ZHANG, Y., ZHU, M., AND ZHOU, M. 2006. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Sydney, Australia, 1025–1032.

LIN, D. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*. Montreal, Quebec, Canada, 768–774.

LIN, J. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory 37,* 1, 145–151.

LIU, F., WENG, F., AND JIANG, X. 2012. A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*. Jeju, Republic of Korea.

LIU, F., WENG, F., WANG, B., AND LIU, Y. 2011. Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, 71–76.

LIU, X., ZHANG, S., WEI, F., AND ZHOU, M. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, 359–367.

LODHI, H., SAUNDERS, C., SHAWE-TAYLOR, J., CRISTIANINI, N., AND WATKINS, C. 2002. Text classification using string kernels. *J. Mach. Learn. Res. 2*, 419–444.

LUI, M. AND BALDWIN, T. 2011. Cross-domain feature selection for language identification. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand, 553–561.

O'CONNOR, B., KRIEGER, M., AND AHN, D. 2010. TweetMotif: Exploratory search and topic summarization for Twitter. In *Proceedings of the 4th International Conference on Weblogs and Social Media (ICWSM 2010)*. Washington, USA, 384–385.

PAPINENI, K., ROUKOS, S., WARD, T., AND ZHU, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, USA, 311–318.

PETERSON, J. L. 1980. Computer programs for detecting and correcting spelling errors. *Commun. ACM 23*, 676–687.

PHILIPS, L. 2000. The double metaphone search algorithm. *C/C++ Users Journal 18*, 38–43.

RABINER, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77,* 2, 257–286.

RITTER, A., CHERRY, C., AND DOLAN, B. 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*. Los Angeles, USA, 172–180.

RITTER, A., CLARK, S., MAUSAM, AND ETZIONI, O. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK, 1524–1534.

SAKAKI, T., OKAZAKI, M., AND MATSUO, Y. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*. Raleigh, North Carolina, USA, 851–860.

SHANNON, C. E. 1948. A mathematical theory of communication. *Bell System Technical Journal 27*, 379–423, 623–656.

SPROAT, R., BLACK, A. W., CHEN, S., KUMAR, S., OSTENDORF, M., AND RICHARDS, C. 2001. Normalization of non-standard words. *Computer Speech and Language 15,* 3, 287 – 333.

STOLCKE, A. 2002. Srilm - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*. Denver, USA, 901–904.

SUN, G., CONG, G., LIU, X., LIN, C.-Y., AND ZHOU, M. 2007. Mining sequential patterns and tree patterns to detect erroneous sentences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic, 81–88.

THURLOW, C. 2003. Generation txt? The sociolinguistics of young people's text-messaging. *Discourse Analysis Online 1,* 1.

TOUTANOVA, K., KLEIN, D., MANNING, C. D., AND SINGER, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL '03. Association for Computational Linguistics, Edmonton, Canada, 173–180.

TOUTANOVA, K. AND MOORE, R. C. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, USA, 144–151.

TWITTER. 2011. 200 million tweets per day. Retrived at August 17th, 2011.

WENG, J. AND LEE, B.-S. 2011. Event detection in Twitter. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*. Barcelona, Spain.

WONG, W., LIU, W., AND BENNAMOUN, M. 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*. Sydney, Australia, 83–89.

XUE, Z., YIN, D., AND DAVISON, B. D. 2011. Normalizing microtext. In *Proceedings of the AAAI-11 Workshop on Analyzing Microtext*. San Francisco, USA, 74–79.

YEH, A. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*. Saarbrücken, Germany, 947–953.

Online Appendix to: Lexical Normalisation for Social Media Text

Bo Han,♠♡ Paul Cook,♡ and Timothy Baldwin♠♡
♠ NICTA Victoria Research Laboratory
♡ Department of Computing and Information Systems, The University of Melbourne

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000