Paraphrasing 4 Microblog Normalization

Wang Ling **Chris Dyer** **Alan W Black** **Isabel Trancoso**
L^{2}F Spoken Systems Lab, INESC-ID, Lisbon, Portugal
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
Instituto Superior Técnico, Lisbon, Portugal
{lingwang,cdyer,awb}@cs.cmu.edu
isabel.trancoso@inesc-id.pt

Abstract

Compared to the edited genres that have played a central role in NLP research, microblog texts use a more informal register with nonstandard lexical items, abbreviations, and free orthographic variation. When confronted with such input, conventional text analysis tools often perform poorly. Normalization — replacing orthographically or lexically idiosyncratic forms with more standard variants — can improve performance. We propose a method for learning normalization rules from machine translations of a parallel corpus of microblog messages. To validate the utility of our approach, we evaluate extrinsically, showing that normalizing English tweets and then translating improves translation quality (compared to translating unnormalized text) using three standard web translation services as well as a phrase-based translation system trained on parallel microblog data.

1 Introduction

Microblogs such as Twitter, Sina Weibo (a popular Chinese microblog service) and Facebook have received increasing attention in diverse research communities (Han and Baldwin, 2011; Hawn, 2009, *inter alia*). In contrast to traditional text domains that use carefully controlled, standardized language, microblog content is often informal, with less adherence to conventions regarding punctuation, spelling, and style, and with a higher proportion of dialect or pronouciation-derived orthography. While this diversity itself is an important resource for studying, e.g., sociolinguistic variation (Eisenstein et al., 2011; Eisenstein, 2013), it poses challenges to NLP applications developed for more formal domains. If retaining variation due to sociolinguistic or phonological factors is not crucial, **text normalization** can improve performance on downstream tasks (§2).

This paper introduces a data-driven approach to learning normalization rules by conceiving of normalization as a kind of paraphrasing and taking inspiration from the bilingual pivot approach to paraphrase detection (Bannard and Callison-Burch, 2005) and the observation that translation is an inherently "simplifying" process (Laviosa, 1998; Volansky et al., 2013). Starting from a parallel corpus of microblog messages consisting of English paired with several other languages (Ling et al., 2013), we use standard web machine translation systems to *re-translate* the non-English segment, producing $\langle \text{English original}, \text{English MT} \rangle$ pairs (§3). These are our normalization examples, with MT output playing the role of normalized English. Several techniques for identifying high-precision normalization rules are proposed, and we introduce a character-based normalization model to account for predictable character-level processes, like repetition and substitution (§4). We then describe our decoding procedure (§5) and show that our normalization model improve translation quality for English–Chinese microblog translation (§6).[1]

2 Why Normalize?

Consider the English tweet shown in the first row of Table 1 which contains several elements that NLP

[1]The datasets used in this paper are available from http://www.cs.cmu.edu/~lingwang/microtopia.

Table 1: Translations of an English microblog message into Mandarin, using three web translation services.

orig.	*To DanielVeuleman yea iknw imma work on that*
MT^1	啊iknw DanielVeuleman伊马工作，
MT^2	DanielVeuleman 是iknw 凋谢关于工作，
MT^3	到DanielVeuleman是的iknw imma这方面的工作

systems trained on edited domains may not handle well. First, it contains several nonstandard abbreviations, such as, *yea*, *iknw* and *imma* (abbreviations of *yes*, *I know* and *I am going to*). Second, there is no punctuation in the text although standard convention would dictate that it should be used. To illustrate the effect this can have, consider now the translations produced by Google Translate,[2] Microsoft Bing,[3] and Youdao,[4] shown in rows 2–4. Even with no knowledge of Chinese, it is not hard to see that all engines have produced poor translations: the abbreviation *iknw* is left translated by all engines, and *imma* is variously deleted, left untranslated, or transliterated into the meaningless sequence 伊马 (pronounced *yī mǎ*).

While normalization to a form like *To Daniel Veuleman: Yes, I know. I am going to work on that.* does indeed lose some information (information important for an analysis of sociolinguistic or phonological variation clearly goes missing), it expresses the propositional content of the original in a form that is more amenable to processing by traditional tools. Translating the normalized form with Google Translate produces 要丹尼尔*Veuleman*：是的，我知道。我打算在那工作。, which is a substantial improvement over all translations in Table 1.

3 Obtaining Normalization Examples

We want to treat normalization as a supervised learning problem akin to machine translation, and to do so, we need to obtain pairs of microblog posts and their normalized forms. While it would be possible to ask annotators to create such a corpus, it would be quite expensive to obtain large numbers of examples. In this section, we propose a method for creating normalization examples without any human annotation, by leveraging existing tools and data resources.

Table 2: Translations of Chinese original post to English using web-based service.

orig.	To DanielVeuleman yea iknw imma work on that
orig.	对DanielVeuleman说，是的，我知道，我正在向那方面努力
MT^1	Right DanielVeuleman say, yes, I know, I'm Xiangna efforts
MT^2	DanielVeuleman said, Yes, I know, I'm that hard
MT^3	Said to DanielVeuleman, yes, I know, I'm to that effort

The English example sentence in Table 1 was selected from the **μtopia parallel corpus** (Ling et al., 2013), which consists of *self-translated* messages from Twitter and Sina Weibo (i.e., each message contains a translation of itself). Row 2 of Table 2 shows the Mandarin self-translation from the corpus. The key observation is what happens when we *automatically* translate the Mandarin version back into English. Rows 3–5 shows automatic translations from three standard web MT engines. While not perfect, the translations contain several correctly normalized subphrases. We will use such *re-translations* as a source of (noisy) normalization examples. Since such self-translations are relatively numerous on microblogs, this technique can provide a large amount of data.

Of course, to motivate this paper, we argued that NLP tools — like the very translation systems we propose to use — often fail on unnormalized input. Is this a problem? We argue that it is not for the following two reasons.

Normalization in translation. Work in translation studies has observed that translation tends to be a *generalizing* process that "smooths out" author- and work-specific idiosyncrasies (Laviosa, 1998; Volansky et al., 2013). Assuming this observation is robust, we expect that dialectal variant forms found in microblogs to be normalized in translation. Therefore, if the parallel segments in our microblog parallel corpus did indeed originate through a translation process (rather than, e.g., being generated as two independent utterances from a bilingual), we may then state the following assumption about the distribution of variant forms in a parallel segment

[2]`http://translate.google.com/`
[3]`http://www.bing.com/translator`
[4]`http://fanyi.youdao.com/`

$\langle \mathbf{e}, \mathbf{f} \rangle$: *if **e** contains nonstandard lexical variants, then **f** is likely to be a normalized translation using with fewer nonstandard lexical variants (and vice-versa).*

Uncorrelated orthographic variants. Any written language has the potential to make creative use of orthography: alphabetic scripts can render approximations of pronunciation variants; logographic scripts can use homophonic substitutions. However, the kinds of innovations used in particular languages will be language specific (depending on details of the phonology, lexicon, and orthography of the language). However, for language pairs that differ substantially in these dimensions, it may not always be possible (or at least easy) to preserve particular kinds of nonstandard orthographic forms in translation. Consider the (relatively common) pronoun-verb compounds like *iknw* and *imma* from our motivating example: since Chinese uses a logographic script without spaces, there is no obvious equivalent.

3.1 Variant–Normalized Parallel Corpus

For the two reasons outlined above, we argue that we will be able to translate back into English using MT, even when the underlying English part of the parallel corpus has a great deal of nonstandard content. We leverage this fact to build the normalization corpus, where the original English tweet is treated as the variant form, and the automatic translation obtained from another language is considered a potential normalization.[5]

Our process is as follows. The microblog corpus of Ling et al. (2013) contains sentence pairs extracted from Twitter and Sina Weibo, for multiple language pairs. We use all corpora that include English as one of the languages in the pair. The respective non-English side is translated into English using different translation engines. The different sets we used and the engines we used to translate are shown in Table 3. Thus, for each original English post **o**, we obtain n paraphrases $\{\mathbf{p}_i\}_{i=1}^{n}$, from n different translation engines.

[5]We additionally assume that the translation engines are trained to output more standardized data, so there will be additional normalizing effect from the machine translation system.

Table 3: Corpora Used for Paraphrasing.

Lang. Pair	Source	Segs.	MT Engines
ZH-EN	Weibo	800K	Google, Bing, Youdao
ZH-EN	Twitter	113K	Google, Bing, Youdao
AR-EN	Twitter	114K	Google, Bing
RU-EN	Twitter	119K	Google, Bing
KO-EN	Twitter	78K	Google, Bing
JA-EN	Twitter	75K	Google, Bing

3.2 Alignment and Filtering

Our parallel microblog corpus was crawled automatically and contains many misaligned sentences. To improve precision, we attempt to find the similarity between the (unnormalized) original and each of the normalizations using an alignment based on the one used in METEOR (Denkowski and Lavie, 2011), which computes the best alignment between the original tweet and each of the normalizations but modified to permit domain-specific approximate matches. To address lexical variants, we allow fuzzy word matching, that is, we allow lexically similar, such as *yea* and *yes* to be aligned (similarity is determined by the Levenshtein distance). We also perform phrasal matchings, such as *ikwn* to *i know*. To do so, we extend the alignment algorithm from word to phrasal alignments. More precisely, given the original post **o** and a candidate normalization **n**, we wish to find the optimal segmentation producing a good alignment. A segmentation $\mathbf{s} = \langle s_1, \ldots, s_{|\mathbf{s}|} \rangle$ is a sequence of segments that aligns as a block to a source word. For instance, for the sentence *yea iknw imma work on that*, one possible segmentation could be $s_1 =$*yea ikwn*, $s_2 =$*imma* and $s_3 =$*work on that*.

Model. We define the score of an alignment **a** and segmentation **s** in using a model that makes semi-Markov independence assumptions, similar to the work in (Bansal et al., 2011), $u(\mathbf{a}, \mathbf{s} \mid \mathbf{o}, \mathbf{n}) =$

$$\prod_{i=1}^{|\mathbf{s}|} \Big[u_e(s_i, a_i \mid \mathbf{n}) \times u_t(a_i \mid a_{i-1}) \times u_\ell(|s_i|) \Big]$$

In this model, the maximal scoring segmentation and alignment can be found using a polynomial time dynamic programming algorithm. Each segment can be aligned to any word or segment in **o**. The aligned segment for s_k is defined as a_k. For the score of a segment correspondence $u_e(s, a \mid \mathbf{n})$, we assume that this can be estimated using the lexical similarity between segments, which we define to be $1 - \frac{L(s_k, a_k)}{\max\{|s_k|, |a_k|\}}$, where $L(x, y)$ denotes the Levenshtein distance between strings x and y, normalized by the highest possible distance between those segments.

For the alignment score u_t, we assume that the relative order of the two sequences will be mostly monotonous. Thus, we approximate u_t with the following density $pos_s(a_k) - pos_e(a_{k-1}) \sim \mathcal{N}(1, 1)$, where the pos_s is the index of the first word in the segment and pos_e the one of the last word.

After finding the Viterbi alignments, we compute the similarity measure $\tau = \frac{|A|}{|A|+|U|}$, used in (Resnik and Smith, 2003), where $|A|$ and $|U|$ are the number of words that were aligned and unaligned, respectively. In this work, we extract the pair if $\tau > 0.2$.

4 Normalization Model

From the normalization corpus, we learn a normalization model that generalizes the normalization process. That is, from the data we observe that *To DanielVeuleman yea iknw imma work on that* is normalized to *To Daniel Veuleman: yes, I know. I am going to work on that.* However, this is not useful, since the chances of the exact sentence *To DanielVeuleman yea iknw imma work on that* occurring in the data is low. We wish to learn a process to convert the original tweet into the normalized form.

There are two mechanisms that we use in our model. The first (§4.1) learns word–word and phrase–phrase mappings. That is, we wish to find that *DanielVeuleman* is normalized to *Daniel Veuleman*, that *iknw* is normalized to *I know* and that *imma* is normalized to *I am going*. These mappings are more useful, since whenever *iknw* occurs in the data, we have the option to normalize it to *I know*. The second (§4.2) learns character sequence mappings. If we look at the normalization *DanielVeuleman* to *Daniel Veuleman*, we can see that it is only applicable when the exact word *DanielVeuleman* occurs. However, we wish to learn that it is uncommon for the letters *l* and *v* to occur in the same word sequentially, so that be can add missing spaces in words that contain the *lv* character sequence, such as normalizing *phenomenalvoter* to *phenomenal voter*.

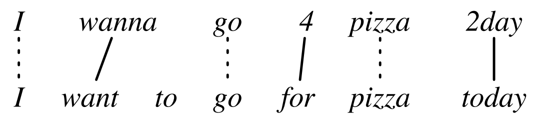

Figure 1: Variant–normalized alignment with the variant form above and the normalized form below; solid lines show potential normalizations, while dashed lines represent identical translations.

However, there are also cases where this is not true, for instance, in the word *velvet*, we do not wish to separate the letters *l* and *v*. Thus, we shall describe the process we use to decide when to apply these transformations.

4.1 From Sentences To Phrases

The process to find phrases from sentences has been throughly studied in Machine Translation. This is generally done in two steps, Word Alignments and Phrase Extraction.

Alignment. The first step is to find the word-level alignments between the original post and its normalization. This is a well studied problem in MT, referred as Word Alignment (Brown et al., 1993). Many alignment models have been proposed, such as, the HMM-based word alignment models (Vogel et al., 1996) and the IBM models (Och and Ney, 2003). Generally, a symmetrization step is performed, where the bidirectional alignments are combined heuristically. In our work, we use the fast aligner proposed in (Dyer et al., 2013) to obtain the word alignments. Figure 1 shows an example of an word aligned pair of a tweet and its normalization.

Phrase Extraction. The phrasal extraction step (Ling et al., 2010), uses the word aligned sentences and extracts phrasal mappings between the original tweet and its normalization, named phrase pairs. For instance, in Figure 1, we would like to extract the phrasal mapping from *go 4* to *go for*, so that we learn that the word *4* in the context of *go* is normalized to the proposition *for*. To do this, the most common approach is to use the template proposed in (Och and Ney, 2004), which allows phrase pairs to be extracted, if there is at least one word alignment within the pair, and there are no

Table 4: Fragment of the phrase normalization model built, for each original phrase **o**, we present the top-3 normalized forms ranked by $f(\mathbf{n} \mid \mathbf{o})$.

Original (**o**)	Normalization (**n**)	$f(\mathbf{n} \mid \mathbf{o})$
wanna	want to	0.4679
wanna	will	0.0274
wanna	going to	0.0114
4	4	0.5641
4	for	0.01795
go 4	go for	1.0000

words inside the pair that are aligned to words not in the pair. For instance, in the example above, the phrase pair that normalizes *wanna* to *want to* would be extracted, but the phrase pair normalizing *wanna* to *want to go* would not, because the word *go* in the normalization is aligned to a word not in the pair.

Phrasal Features. After extracting the phrase pairs, a model is produced with features derived from phrase pair occurrences during extraction. This model is equivalent to phrasal translation model in MT, but we shall refer to it as the normalization model. For a phrase pair $\langle \mathbf{o}, \mathbf{n} \rangle$, where **o** is the original phrase, and **n** is the normalized phrase, we compute the normalization relative frequency $f(\mathbf{n} \mid \mathbf{o}) = \frac{C(\mathbf{n},\mathbf{o})}{C(\mathbf{o})}$, where $C(\mathbf{n}, \mathbf{o})$ denotes the number of times **o** was normalized to **n** and $C(\mathbf{o})$ denotes the number of times **o** was seen in the extracted phrase pairs. Table 4 gives a fragment of the normalization model. The columns represent the original phrase, its normalization and the probability, respectively.

In Table 4, we observe that the abbreviation *wanna* is normalized to *want to* with a relatively high probability, but it can also be normalized to other equivalent expressions, such as *will* and *going to*. The word *4* by itself has a low probability to be normalized to the preposition *for*. This is expected, since this decision cannot be made without context. However, we see that the phrase *go 4* is normalized to *go for* with a high probability, which specifies that within the context of *go*, *4* is generally used as a preposition.

4.2 From Phrases to Characters

While we can learn lexical variants that are in the corpora using the phrase model, we can only address word forms that have been observed in the corpora.

Table 5: Fragment of the character normalization model where examples representative of the lexical variant generation process are encoded in the model.

Original (**o**)	Normalization (**n**)	$f(\mathbf{n} \mid \mathbf{o})$
o o o	o o	0.0223
o o o	o	0.0439
s	c	0.0331
z	s	0.0741
s h	c h	0.019
2	t o	0.014
4	f o r	0.0013
0	o	0.0657
i n g f o r	i n g <space> f o r	0.4545
g f	g <space> f	0.01028

This is quite limited, since we cannot expect all the word forms to be present, such as all the possible orthographic errors for the word *cat*, such as *catt*, *kat* and *caaaat*. Thus, we will build a character-based model that learns the process lexical variants are generated at the subword level.

Our character-based model is similar to the phrase-based model, except that, rather than learning word-based mappings from the original tweet and the normalization sentences, we learn character-based mappings from the original phrases to the normalizations of those phrases. Thus, we extract the phrase pairs in the phrasal normalization model, and use them as a training corpora. To do this, for each phrase pair, we add a start token, <*start*>, and a end token, <*end*>, at the beginning and ending of the phrase pair. Afterwards, we separate all characters by space and add a space token <*space*> where spaces were originally. For instance, the phrase pair normalizing *DanielVeuleman* to *Daniel Veuleman* would be converted to <*start*> *d a n i e l v e u l e m a n* <*end*> and <*start*> *d a n i e l* <*space*> *v e u l e m a n* <*end*>.

Character-based Normalization Model - To build the character-based model, we proceed using the same approach as in the phrasal normalization model. We first align characters using Word Alignment Models, and then we perform phrase extraction to retrieve the phrasal character segments, and build the character-based model by collecting statistics. Once again, we provide examples of entries in the model in Table 5.

We observe that many of the normalizations dealt with in the previous model by memorizing phrases are captured with string transformations. For instance, from phrase pairs such as *tooo* to *too* and *sooo* to *so*, we learn that sequences of *o*'s can be reduced to 2 or 1 *o*. Other examples include orthographic substitutions, such as *2* for *to* and *4* for *for* (as found in *2gether*, *2morrow*, *4ever* and *4get*). Moreover, orthographic errors can be generated from mistaking characters with similar phonetic properties, such as, *s* to *c*, *z* to *s* and *sh* to *ch*, generating lexical variants such as *reprecenting*. Finally, we learn that the number *0* that resembles the letter *o*, can be used as a replacement, as in *g00d*. Finally, we can see that the rule *ingfor* to *ing for* attempts to find segmentation errors, such as *goingfor*, where a space between *going* and *for* was omitted.[6]

5 Normalization Decoder

In section 4, we built two models to learn the process of normalization, the phrase-based model and the character-based model. In this section, we describe the decoder we used to normalize the sentences.

The advantage of the phrase-based model is that it can make decisions for normalization based on context. That is, it contains phrasal units, such as, *go 4*, that determine, when the word *4* should be normalized to the preposition *for* and when to leave it as a number. However, it cannot address words that are unseen in the corpora. For instance, if the word form *4ever* is not seen in the training corpora, it is not be able to normalize it, even if it has seen the word *4get* normalized to *forget*. On the other hand, the character-based model learns subword normalizations, for instance, if we see the word *nnnnno* normalized to *no*, we can learn that repetitions of the letter *n* are generally shorted to *n*, which allows it to generate new word forms. This model has strong generalization potential, but the weakness of the character-based model is that it fails to consider the context of the normalization that the phrase-based model uses to make normalization decisions. Thus, our goal in this section is describe a decoder that uses both models to improve the quality of the normalizations.

5.1 Phrasal Decoder

We use Moses, an off-the-shelf phrase-based MT system (Koehn et al., 2007), to "translate" the original tweet its normalized form using the phrasal model (§4.1). Aside form the normalization probability, we also use the common features used in MT. These are the reverse normalization probability, the lexical and reverse lexical probabilities and the phrase penalty. We also use the MSD reordering model proposed in (Koehn et al., 2005), which adds reordering features.[7] The final score of each phrase pair is given as a sum of weighted log features. The weights for these features are optimized using MERT (Och, 2003). In our work, we sampled 150 tweets randomly from Twitter and normalized them manually, and used these samples as development data for MERT. As for the character-based model features, we simply rank the training phrase pairs by their relative frequency the $f(\mathbf{n} \mid \mathbf{o})$, and use the top-1000 phrase pairs as development set. Finally, a language model is required during decoding as a prior, since it defines the type of language that is produced by the output. We wish to normalized to formal language, which is generally better processed by NLP tools. Thus, for the phrase model, we use the English NIST dataset composed of 8M sentences in English from the news domain to build a 5-gram Kneser-Ney smoothed language model.

5.2 Character and Phrasal Decoder

We now turn to how to apply the character-based (§4.2), together with the phrasal model. For this model, we again use Moses, treating each character as a "word". The simplest way to combine both methods is first to decode the input **o** sentence with the character-based decoder, normalizing each word independently and then normalizing the resulting output using the phrase-based decoder, which enables the phrase model to score the outputs of the character model in context.

[6]Note that this captures the context in which such transformations are likely to occur: there are not many words that contain the sequence *ingfor*, so the probability that these should be normalized by inserting a space is high. On the other hand, we cannot assume that if we observe the sequence *gf*, we can safely separate these with a space. This is because, there are many words that contain this sequence, such as the abbreviation of *gf* (*girlfriend*), *dogfight*, and *bigfoot*.

[7]Reordering helps find lexical variants that are generated by transposing characters, such as, *mabye* to *maybe*.

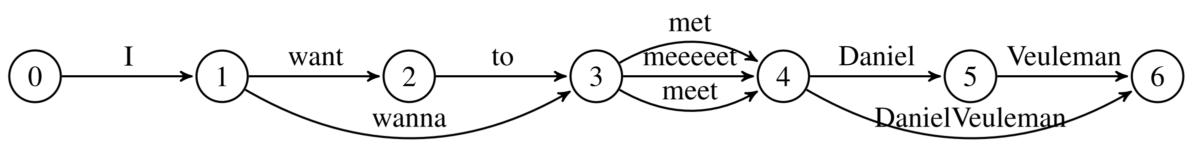

Figure 2: Example output lattice of the character-based decoder, for the sentence *I wanna meeeeet DanielVeuleman*.

Our process is as follows. Given the input sentence **o**, with the words $o_1, \ldots, o_m$, where m is the number of words in the input, we generate for each word o_i a list of n-best normalization candidates $z^1_{o_i}, \ldots, z^n_{o_i}$. We further filter the candidates using two criteria. We start by filtering each candidate $z^j_{o_i}$ that occurs less frequently than the original word o_i. This is motivated by our observation that lexical variants occur far less than the respective standard form. Second, we build a corpus of English language Twitter consisting of 70M tweets, extract the unigram counts, and perform Brown clustering (Brown et al., 1992) with $k = 3000$ clusters. Next, we calculate the cluster similarity between o_i and each surviving candidate, $z^j_{o_i}$. We filter the candidate if the similarity is less than 0.8. The similarity between two clusters represented as *bit strings*, $S[c(o_i), c(z^j_{o_i})]$, calculated as:

$$S(x, y) = \frac{2 \cdot |lpm\{x, y)\}|}{|x| + |y|},$$

where lpm computes the longest common prefix of the contexts and $|x|$ is the length of the bit string.[8] If a candidate contains more than one word (because a space was inserted), we set its count as the minimum count among its words. To find the cluster for multiple word units, we concatenate the words together, and find the cluster with the resulting word if it exists. This is motivated by the fact that it is common for missing spaces to exist in microblog corpora, generating new word forms, such as *wantto*, *goingfor*, and given a large enough corpora as the one we used, these errors occur frequently enough to be placed in the correct cluster. In fact, the variants such as *wanna* and *tmi*, occur in the same clusters as the words *wantto* and *toomuchinformation*.

Remaining candidates are combined into a word lattice, enabling us to perform lattice-based decoding with the phrasal model (Dyer et al., 2008). Figure 2, provides an example of such a lattice for the variant sentence *I wanna meeeet DanielVeuleman*.

5.3 Learning Variants from Monolingual Data

Until now, we learned normalizations from pairs of original tweets and their normalizations. We shall now describe a process to leverage monolingual documents to learn new normalizations, since the monolingual data is far easier to obtain than parallel data. This process is similar to the work in (Han et al., 2012), where confusion sets of contextually similar words are built initially as potential normalization candidates. We again use the $k = 3000$ Brown clusters,[9] and this time consider the contents of each cluster as a set of possible normalization variants. For instance, we find that the cluster that includes the word *never*, also includes the variant forms *neverrrr*, *neva* and *nevahhh*. However, the cluster also contains non-variant forms, such as *gladly* and *glady*. Thus, we want to find that *neverrrr* maps to *never*, while *glady* maps to *gladly* in the same cluster. Our work differs from previous work in that, rather than defining features manually, we use our character-based decoder to find the mappings between lexical variants and their normalizations.

For every word type w_i in cluster $c(w_i) = \{w_1, \ldots, w_n\}$, we generate a set of possible candidates for each word $w^1_i, \ldots, w^m_i$. Then, we build a directed acyclic graph (DAG), where every word. We add an edge between w_i and w_j, if w_i can be decoded into w_j using the character model from the previous section, and also if w_i occurs less than w_j; the second condition guarantees that the graph will be acyclic. Sample graphs are shown in Figure 3.

Afterwards, we find the number of paths between all nodes in the graph (this can be computed efficiently in $O(|V| + |E|)$ time). Then, for each word

[8]Brown clusters are organized such that more words with more similar distributions share common prefixes.

[9]The Brown clustering algorithm groups words together based on contextual similarity.

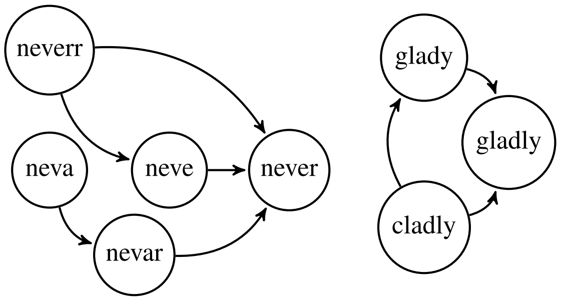

Figure 3: Example DAGs, built from the cluster containing the words *never* and *gladly*.

w_i, we find the w_j to which it has the highest number of paths to and extract the normalization of w_i to w_j. In case of a tie, we choose the word w_j that occurs more often in the monolingual corpora. This is motivated by the fact that normalizations are transitive. Thus, even if *neva* cannot be decoded directly to *never*, we can use *nevar* as an intermediate step to find the correct normalization. This is performed for all the clusters, and the resulting dictionary of lexical variants mapped to their standard forms is added to the training data of the character-based model.

6 Experiments

We evaluate our normalization model intrinsically by testing whether our normalizations more closely resemble standardized data, and then extrinsically by testing whether we can improve the translation quality of in-house as well as online Machine Translation systems by normalizing the input.

6.1 Setup

We use the gold standard by Ling et al. (2013), composed by 2581 English-Mandarin microblog sentence pairs. From this set, we randomly select 1290 pairs for development and 1291 pairs for testing.

The normalizer model is trained on the corpora extracted and filtered in section 3, in total, there were 1.3M normalization pairs used during training. The test sentences are normalized using four different setups. The first setup leaves the input sentence unchanged, which we call **No Norm**. The second uses the phrase-based model to normalize the input sentence, which we will denote **Norm+phrase**. The third uses the character-based model to output lattices, and then decodes with the phrase based model, which we will denote **Norm+phrase+char**. Finally, we test the same model after adding the training data extracted using monolingual documents, which we will refer as **Norm+phrase+char+mono**.

To test the normalizations themselves, we used Google Translate to translate the Mandarin side of the 1291 test sentence pairs back to English and use the original English tweet. While, this is by itself does not guarantee that the normalizations are correct, since the normalizations could be syntactically and semantically incorrect, it will allow us to check whether the normalizations are closer to those produced by systems trained on news data. This experiment will be called **Norm**.

As an application and extrinsic evaluation for our normalizer, we test if we can obtain gains on the MT task on microblog data by using our normalizer prior to translation. We build two MT systems using Moses. Firstly, we build a out-of-domain model using the full 2012 NIST Chinese-English dataset (approximately 8M sentence pairs), which is dataset from the news domain, and we will denote this system as **Inhouse+News**. Secondly, we build a in-domain model using the 800K sentence pairs from μtopia corpora (Ling et al., 2013). We also add the NIST dataset to improve coverage. We call this system **Inhouse+News+Weibo**. To train these systems, we use the Moses phrase-based MT system with standard features (Koehn et al., 2003). For reordering, we use the MSD reordering model (Axelrod et al., 2005). As the language model, we train a 5-gram model with Kneser-ney smoothing using a 10M tweets from twitter. Finally, the weights were tuned using MERT (Och, 2003). As for online systems, we consider the systems used to generate the paraphrase corpora in section 3, which we will denote as **Online A**, **Online B** and **Online C**[10]

The normalization and MT results are evaluated with BLEU-4 (Papineni et al., 2002) comparing the produced translations or normalizations with the appropriate reference.

6.2 Results

Results are shown in Table 6. In terms of the normalizations, we observe a much better match between

[10]The names of the systems are hidden to not violate the privacy issues in the terms and conditions of these online systems.

Table 6: Normalization and MT Results. Rows denote different normalizations, and columns different translation systems, except the first column (Norm), which denotes the normalization experiment. Cells display the BLEU score of that experiment.

Condition	Norm	Moses (News)	Moses (News+Weibo)	Online A	Online B	Online C
baseline	19.90	15.10	24.37	20.09	17.89	18.79
norm+phrase	21.96	15.69	24.29	20.50	18.13	18.93
norm+phrase+char	22.39	15.87	24.40	20.61	18.22	19.08
norm+phrase+char+mono	**22.91**	**15.94**	**24.46**	**20.78**	**18.37**	**19.21**

the normalized text with the reference, than the original tweets. In most cases, adding character-based models improves the quality of the normalizations.

We observe that better normalizations tend to lead to better translations. The relative improvements are most significant, when moving from No Norm to **norm+phrase** normalization. This is because, we are normalizing words that are not seen in general MT system's training data, but occur frequently in microblog data, such as *wanna* to *want to*, *u* to *you* and *im* to *i'm*. The only exception is in the **Inhouse+News+Weibo** system, where the normalization deteriorates the results. This is to be expected, since this system is trained on the same microblog data used to learn the normalizations. However, we can observe on **norm+phrase+char** that if we add the character-based model, we can observe improvements for this system as well as for all other ones. This is because the model is actually learning normalizations that are unseen in the data. Some examples of these normalization include, normalizing *lookin* to *looking*, *nutz* to *nuts* and *maimi* to *miami* but also separating *peaceof* to *peace of*. The fact that these improvements are obtained for all systems is strong evidence that we are actually producing good normalizations, and not overfitting to one of the systems that we used to generate our data. The gains are much smaller from **norm+phrase** to **norm+phrase+char**, since the improvements we obtain come from normalizing less frequent words. Finally, we can obtain another small improvement by adding monolingual data to the character-based model in **norm+phrase+char+mono**.

7 Related Work

Most of the work in microblog normalization is focused on finding the standard forms of lexical variants (Yang and Eisenstein, 2013; Han et al., 2013; Han et al., 2012; Kaufmann, 2010; Han and Baldwin, 2011; Gouws et al., 2011; Aw et al., 2006). A lexical variant is a variation of a standard word in a different lexical form. This ranges from minor or major spelling errors, such as *jst*, *juxt* and *jus* that are lexical variants of *just*, to abbreviations, such as *tmi* and *wanna*, which stand for *too much information* and *want to*, respectively. Jargon can also be treated as variants, for instance *cday* is a slang word for *birthday*, in some groups.

There are many rules that govern the process lexical variants are generated. Some variants are generated from orthographic errors, caused by some mistake from the user when writing. For instance, the variants *representin*, *representting*, or *reprecenting* can be generated by a spurious letter swap, insertion or substitution by the user. One way to normalize these types of errors is to attempt to insert, remove and swap words in a lexical variant until a word in a dictionary of standard words is found (Kaufmann, 2010). Contextual features are another way to find lexical variants, since variants generally occur in the same context as their standard form. This includes orthographic errors, abbreviations and slang. However, this is generally not enough to detect lexical variants, as many words share similar contexts, such as *already*, *recently* and *normally*. Consequently, contextual features are generally used to generate a confusion set of possible normalizations of a lexical variant, and then more features are used to find the correct normalization (Han et al., 2012). One simple approach is to compute the Levenshtein distance to find lexical similarities between words, which would effectively capture the mappings between *representting*, *reprecenting* and *representin* to *representing*. However, a pronunciation model (Tang et al., 2012) would be needed to find the mapping between *g8*, *2day* and *4ever* to *great*, *today* and *forever*, respectively. Moreover, visual character similarity features would be required to find the mapping between *g00d* and ι to *good* and *i*.

Clearly, learning this process is a challenging task, and addressing each different case individually would require vast amounts of resources. Furthermore, once we change the language to normalize to another language, the types of rules that generate lexical variants would radically change and a new set of features would have to be engineered. We believe that to be successful in normalizing microblogs, the process to learn new lexical variants should be learned from data, making as few assumptions as possible. We learn our models without using any type of predefined features, such as phonetic features or lexical features. In fact, we will not assume that most words and characters map to themselves, as it is assumed in methods using the Levenshtein distance (Kaufmann, 2010; Han et al., 2012; Wang and Ng, 2013). All these mappings are learned from our data. Furthermore, in the work above, the dictionaries built using these methods assume that lexical variants are mapped to standard forms in a word-to-word mapping. Thus, variants such as *wanna*, *gonna* and *imma* are not normalizable, since they are normalized to multiple words *want to*, *going to* and *I am gonna*. Moreover, there are segmentation errors that occur from missing spaces, such as *sortof* and *goingfor*, which also map to more than one word to *sort of* and *going for*. These cases shall also be addressed in our work.

Wang and Ng (2013) argue that microblog normalization is not simply to map lexical variants into standard forms, but that other tasks, such as punctuation correction and missing word recovery should be performed. Consider the example tweet *you free?*, while there are no lexical variants in this message, the authors consider that it is the normalizer should recover the missing article *are* and normalize this tweet to *are you free?*. To do this, the authors train a series of models to detect and correct specific errors. While effective for narrow domains, training models to address each specific type of normalization is not scalable over all types of normalizations that need to be performed within the language, and the fact that a set of new models must be implemented for another language limits the applicability of this work.

Another strong point of the work above is that a decoder is presented, while the work on building dictionaries only normalize out of vocabulary (OOV) words. The work on (Han et al., 2012) trains a classifier to decide whether to normalize a word or not, but is still preconditioned on the fact that the word in question is OOV. Thus, lexical variants, such as, *4* and *u*, with the standard forms *for* and *you*, are left untreated, since they occur in other contexts, such as *u* in *u s a*. Inspired by the work above, we also propose a decoder based on the existing off-the-self decoder Moses (Koehn et al., 2007).

Finally, the work in (Xu et al., 2013) obtains paraphrases from Twitter, by finding tweets that contain common entities, such as *Obama*, that occur during the same period by matching temporal expressions. The resulting paraphrase corpora can also be used to train a normalizer.

8 Conclusion

We introduced a data-driven approach to microblog normalization based on paraphrasing. We build a corpora of tweets and their normalizations using parallel corpora from microblogs using MT techniques. Then, we build two models that learn generalizations of the normalization process, one the phrase level and on the character level. Then, we build a decoder that combines both models during decoding. Improvements on multiple MT systems support the validity of our method.

In future work, we shall attempt to build normalizations for other languages. We shall also attempt to learn an unsupervised normalization model with only monolingual data, similar to the work for MT in (Ravi and Knight, 2011).

Acknowledgements

The PhD thesis of Wang Ling is supported by FCT – Fundação para a Ciência e a Tecnologia, under project SFRH/BD/51157/2010. This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2013.

The authors also wish to express their gratitude to the anonymous reviewers for their comments and insight.

References

[Aw et al.2006] AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of the ACL*, COLING-ACL '06, pages 33–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Axelrod et al.2005] Amittai Axelrod, Ra Birch Mayne, Chris Callison-burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *In Proc. International Workshop on Spoken Language Translation (IWSLT*.

[Bannard and Callison-Burch2005] Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Bansal et al.2011] Mohit Bansal, Chris Quirk, and Robert C. Moore. 2011. Gappy phrasal alignment by agreement. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1308–1317, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Brown et al.1992] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

[Brown et al.1993] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19:263–311, June.

[Denkowski and Lavie2011] Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland, July. Association for Computational Linguistics.

[Dyer et al.2008] Chris Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of HLT-ACL*.

[Dyer et al.2013] Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL-HLT*, pages 644–648.

[Eisenstein et al.2011] Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1365–1374, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Eisenstein2013] Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of NAACL-HLT*, pages 359–369.

[Gouws et al.2011] Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, EMNLP '11, pages 82–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Han and Baldwin2011] Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 368–378, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Han et al.2012] Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 421–432, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Han et al.2013] Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):5.

[Hawn2009] Carleen Hawn. 2009. Take two aspirin and tweet me in the morning: how twitter, facebook, and other social media are reshaping health care. *Health affairs*, 28(2):361–368.

[Kaufmann2010] M. Kaufmann. 2010. Syntactic Normalization of Twitter Messages. *studies*, 2.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. 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, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

[Koehn et al.2005] Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White. 2005. Edinburgh system description for the 2005 nist mt evaluation. In *Proceedings of Machine Translation Evaluation Workshop 2005*.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-burch, Richard Zens, Rwth

Aachen, Alexandra Constantin, Marcello Federico, Nicola Bertoldi, Chris Dyer, Brooke Cowan, Wade Shen, Christine Moran, and Ondrej Bojar. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

[Laviosa1998] Sara Laviosa. 1998. Core patterns of lexical use in a comparable corpus of English lexical prose. *Meta*, 43(4):557–570.

[Ling et al.2010] Wang Ling, Tiago Luís, João Graça, Luísa Coheur, and Isabel Trancoso. 2010. Towards a general and extensible phrase-extraction algorithm. In *IWSLT '10: International Workshop on Spoken Language Translation*, pages 313–320, Paris, France.

[Ling et al.2013] Wang Ling, Guang Xiang, Chris Dyer, Alan Black, and Isabel Trancoso. 2013. Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting on Association for Computational Linguistics*, ACL '13. Association for Computational Linguistics.

[Och and Ney2003] Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

[Och and Ney2004] Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, December.

[Och2003] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ravi and Knight2011] Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *ACL*, pages 12–21.

[Resnik and Smith2003] Philip Resnik and Noah A Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

[Tang et al.2012] Hao Tang, Joseph Keshet, and Karen Livescu. 2012. Discriminative pronunciation modeling: A large-margin, feature-rich approach. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 194–203. Association for Computational Linguistics.

[Vogel et al.1996] S. Vogel, H. Ney, and C. Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

[Volansky et al.2013] Vered Volansky, Noam Ordan, and Shuly Wintner. 2013. On the features of translationese. *Literary and Linguistic Computing*.

[Wang and Ng2013] Pidong Wang and Hwee Ng. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *Proceedings of NAACL-HLT 2013*, NAACL '13. Association for Computational Linguistics.

[Xu et al.2013] Wei Xu, Alan Ritter, and Ralph Grishman. 2013. Gathering and generating paraphrases from twitter with application to normalization. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 121–128, Sofia, Bulgaria, August. Association for Computational Linguistics.

[Yang and Eisenstein2013] Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proc. of EMNLP*.