Benefits of Data Augmentation for NMT-based Text Normalization of User-Generated Content

Claudia Matos Veliz, Orphée De Clercq and Véronique Hoste
LT^3, Language and Translation Technology Team - Ghent Univeristy
Groot-Brittanniëlaan 45, 9000, Ghent, Belgium
Firstname.Lastname@UGent.be

Abstract

One of the most persistent characteristics of written user-generated content (UGC) is the use of non-standard words. This characteristic contributes to an increased difficulty to automatically process and analyze UGC. Text normalization is the task of transforming lexical variants to their canonical forms and is often used as a pre-processing step for conventional NLP tasks in order to overcome the performance drop that NLP systems experience when applied to UGC. In this work, we follow a Neural Machine Translation approach to text normalization. To train such an encoder-decoder model, large parallel training corpora of sentence pairs are required. However, obtaining large data sets with UGC and their normalized version is not trivial, especially for languages other than English. In this paper, we explore how to overcome this data bottleneck for Dutch, a low-resource language. We start off with a small publicly available parallel Dutch data set comprising three UGC genres and compare two different approaches. The first is to manually normalize and add training data, a money and time-consuming task. The second approach is a set of data augmentation techniques which increase data size by converting existing resources into synthesized non-standard forms. Our results reveal that, while the different approaches yield similar results regarding the normalization issues in the test set, they also introduce a large amount of over-normalizations.

1 Introduction

Social media text are considered important language resources for several NLP tasks (Van Hee et al., 2017; Pinto et al., 2016; Zhu et al., 2014). However, one of their most persistent characteristics is the use non-standard words. Social media texts are considered a type of written user-generated content (UGC) in which several language variations can be found as people often tend to write as they speak and/or write as fast as possible (Vandekerckhove and Nobels, 2010). For instance, it is typical to express emotions by the use of symbols or lexical variation. This can be done in the form of the repetition of characters or flooding (*wooooow*), capitalization (*YEY!*), and the productive use of emoticons. In addition, the use of homophonous graphemic variants of a word, abbreviations, spelling mistakes or letter transpositions are also used regularly (Eisenstein et al., 2014).

Since NLP tools have originally been developed for and trained on standard language, these non-standard forms adversely affect their performance. One of the computational approaches which has been suggested to overcome this problem is text normalization (Sproat et al., 2001). This approach envisages transforming the lexical variants to their canonical forms. In this way, standard NLP tools can be applied in a next step after normalization (Aw et al., 2006). Please note that for some NLP applications, e.g. sentiment analysis, it might be beneficial to keep some 'noise' in the data. For example, the use of flooding or capital letters could be a good indicator of the emotion present in the text (Van Hee et al., 2017). However, for applications aiming at information extraction from text, normalization is needed to help to improve the performance of downstream NLP tasks (Schulz et al., 2016). Kobus et al. (2008) introduced three metaphors to refer to these normalization approaches: the spell checking, automatic speech recognition and machine translation metaphors.

In this paper, the focus will be on the machine translation metaphor. One of the most conventional approaches is to use Statistical Machine Translation (SMT) techniques (Kaufmann, 2010; De Clercq et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2016), in particular using

Proceedings of the 2019 EMNLP Workshop W-NUT: The 5th Workshop on Noisy User-generated Text, pages 275–285
Hong Kong, Nov 4, 2019.

the Moses toolkit (Koehn et al., 2007). However, neural networks have proven to outperform many state-of-the-art systems in several NLP tasks (Young et al., 2018). Especially the encoder-decoder model with an attention mechanism (Bahdanau et al., 2014) for recurrent neural networks (RNN) has lead to a new paradigm in machine translation, i.e., Neural MT (NMT) (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014; Luong et al., 2015; Sennrich et al., 2016a).

Many works have adopted and applied these techniques to the normalization task (Ikeda et al., 2016; Mandal and Nanmaran, 2018; Lusetti et al., 2018) some of them outperforming the SMT approach. However, it is well-known that these neural systems require a huge amount of data in order to perform properly (Ikeda et al., 2016; Saito et al., 2017). When it comes to translation these data even have to be parallel and should thus consist of aligned source and target sentences. Unfortunately, when it comes to UGC text normalization there is a lack of parallel corpora in which UGC is considered the source language and its standardized form the target language. Furthermore, the problem even exacerbates when working with low-resourced languages.

In this work, we follow an NMT approach to tackle text normalization of Dutch UGC and explore how to overcome this parallel data bottleneck for Dutch, a low-resource language. We start off with a publicly available tiny parallel Dutch data set comprising three UGC genres and compare two different approaches. The first one is to manually normalize and add training data, a money and time-consuming task. The second approach consists in a set of data augmentation techniques which increase data size by converting existing resources into synthesized non-standard forms. Our results reveal that the different setups resolve most of the normalization issues and that automatic data augmentation mainly helps to reduce the number of over-generalizations produced by the NMT approach.

In the following section, we discuss related work on MT-based text normalization as well as data augmentation techniques. In section 3, we discuss the two approaches to augment the available data: manual annotations of new sentence pairs and augmentation techniques. The data used for our experiments are also explained in detail. Section 4 gives an overview of the experiments and results, whereas section 5 concludes this work and offers prospects for future work.

2 Related Work

Previous research on UGC text normalization has been performed on diverse languages using different techniques ranging from hand-crafted rules (Chua et al., 2018) to deep learning approaches (Ikeda et al., 2016; Sproat and Jaitly, 2016; Lusetti et al., 2018). Three different metaphors were introduced by Kobus et al. (2008) to refer to these normalization approaches. That is the automatic speech recognition (ASR), spell checking, and translation metaphors. The ASR approach exploits the similarity between social media text and spoken language. Several works have followed this methodology, mostly combining it with the others (Beaufort and Roekhaut, 2010; Xue et al., 2011; Han and Baldwin, 2011). In the spell checking approach, corrections from noisy to standard words occurs at the word level. Some approaches have treated the problem by using dictionaries containing standard and non-standard words (Clark and Araki, 2011). However, the success of this kind of systems highly depends on the coverage of the dictionary. Since social media language is highly productive and new terms constantly appear, it is very challenging and expensive to continuously keep such a dictionary up to date.

In this work, we consider the normalization task as a Machine Translation problem and treat noisy UGC text as the source language and its normalized form as the target language. In the past, several works have also used this approach and there are two leading paradigms: Statistical and Neural Machine Translation.

Statistical Machine Translation (SMT) models, especially those trained at the character-level, have proven highly effective for the task because they capture well intra-word transformations (Pennell and Liu, 2011). Besides, they have the advantage of being effective when small training data is provided, thanks to their small vocabulary size. Kaufmann (2010), for example, followed a two step approach for the normalization of English tweets. First, they pre-processed the tweets to remove as much noise as possible, and then they used Moses[1] to convert them into standard English. Moses is a statistical machine translation package which can produce high quality translations from one lan-

[1] http://statmt.org/moses/

Source Sentence	Target Sentence	English Translation
iz da muzieksgool vnavnd ? kwt da niemr .	is dat muziekschool vanavond ? ik weet dat niet meer .	is that music school tonight? I don't know that anymore.
wa is je msn k en e nieuwe msn omda k er nie meer op graal . xxx	wat is je msn ik heb een nieuwe msn omdat ik er niet meer op geraak . xx	what is your msn i have a new msn because i can't get it anymore. xx
@renskedemaessc dm me je gsmnummer eens ;-)	<user> doormail me je gsmnummer eens <emoji>	<user> mail me your cellphone number once <emoji>

Table 1: Source and target pairs as parallel data for a machine translation approach.

guage into another (Koehn et al., 2007). De Clercq et al. (2013) proposed a phrase-based method to normalize Dutch UGC comprising various genres. In a preprocessing step they handled emoticons, hyperlinks, hashtags and so forth. Then they worked in two steps: first at the word level and then at the character level. This approach revealed good results across various genres of UGC. However, a high number of phonetic alternations remained unresolved.

Recently, neural networks have proven to outperform many state-of-the-art systems in several NLP tasks (Young et al., 2018). The encoder-decoder model for recurrent neural networks (RNN) was developed in order to address the sequence-to-sequence nature of machine translation and obtains good results for this task (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014; Luong et al., 2015). The model consist of two neural networks: an encoder and a decoder. The encoder extracts a fixed-length representation from a variable-length input sentence, and the decoder generates a correct translation from this representation. Some works on text normalization have followed the same approach (Lusetti et al., 2018; Cho et al., 2014).

In 2016, Sproat and Jaitly (Sproat and Jaitly, 2016) presented a challenge to the research community: given a large corpus of *written* text aligned to its normalized *spoken* form, train an RNN to learn the correct normalization function. Although their work focuses on the Text to Speech (TTS) use case of text normalization, they compared prior work of text normalization for TTS (Rao et al., 2015; William Chan, 2016) and also discuss the problems that arise when using neural networks for text normalization. They made clear that although RNNs were often capable to produce surprisingly good results and learn some complex mappings, they are prone to make errors like reading the wrong number, or substituting *hours* for *gigabytes*. This makes them risky to apply in a TTS system. Lusetti et al. (2018) performed NMT text normalization over Swiss German WhatsApp messages and compared it to a state-of-the-art SMT system. They revealed that integrating language models into an encoder-decoder framework can outperform the character-level SMT methods for that language.

Although the encoder-decoder model has shown its effectiveness in large datasets, it is much less effective when only a small number of sentence pairs is available (Sennrich et al., 2016b; Zoph et al., 2016). Automatic data augmentation is commonly used in vision and speech and can help train more robust models, particularly when using smaller datasets (Chatfield et al., 2014; Taylor and Nitschke, 2017). Fadaee et al. (2017) present Translation Data Augmentation (TDA), a method to improve the translation quality for low resource pairs (English to German and German to English). Their approach generates new sentence pairs containing rare words in new, synthetically created contexts.

Data augmentation techniques have therefore also been applied to outperform text normalization for low-resourced languages. For example, Ikeda et al. (2016) performed text normalization at the character level for Japanese text and proposed a method for data augmentation using handcrafted rules. Their method transformed existing resources into synthesized non-standard forms using the rules proposed in the work of Sasano et al. (2015). They proved that the use of the synthesized corpus improved the performance of Japanese text normalization. Saito et al. (2017) also proposed two methods for data augmentation in order to improve text normalization. Unlike the previous work, the proposed method did not use prior knowledge to generate synthetic data at the

character and morphological level. Instead, they proposed a two-level data augmentation model that converted standard sentences to dialect sentences by using extracted morphological conversion patterns. Their experiments using an encoder-decoder model for Japanese, performed better than SMT with Moses after the data augmentation.

3 Methodology

Our objective is to go from noisy to standard text and we tackle this normalization problem using an NMT approach. Sequence-to-Sequence (seq2seq) models have been used for a variety of NLP tasks including machine translation obtaining state-of-the-art results (Luong et al., 2015; Young et al., 2018). As in general MT, a translation model is trained on parallel data consisting of pairs (x, y) of source sentences/words (= social media text) and their corresponding target equivalents (= standard text). Table 1 lists some examples of the noisy data we are dealing with.

3.1 NMT Approach

In this approach, both input and output sentences are going in and out of the model. As described in the literature overview, the model consist of two neural networks: an encoder and decoder (See Figure 1).

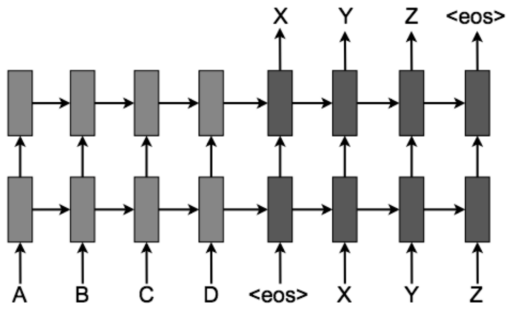

Figure 1: Encoder-decoder architecture. The light-color nodes represent the encoder and the dark-color ones the decoder. Image taken from Luong et al. (2015).

The encoder extracts a fixed-length representation from a variable-length input sentence (*A B C D*), and the decoder generates a correct translation from this representation (*X Y Z*). In the figure, <eos> marks the end of a sentence. The encoder-decoder model is trained on a parallel corpus consisting of source sentences aligned with their normalized form.

We relied on OpenNMT[2] to train our encoder-decoder model. OpenNMT is an open source (MIT) initiative for neural machine translation and neural sequence modeling (Klein et al., 2017). The main system is implemented in the Lua/Torch mathematical framework, and can easily be extended using Torch's internal standard neural network components. We used the version of the system with the basic architecture which consists of an encoder using a simple LSTM recurrent neural network. The decoder applies attention over the source sequence and implements input feeding (Luong et al., 2015).

3.2 Evaluation Metric

To evaluate the results of the normalization, we calculated Word Error Rate (WER) and Character Error Rate (CER) over the three genres. WER is a metric derived from the Levenshtein distance (Levenshtein, 1966), working at the word level. Character Error Rate (CER), instead, works at the character level. These metrics take into account the number of insertions (INS), deletions (DEL) and substitutions (SUBS) that are needed to transform the suggested string into the manually normalized string. The metrics are computed as follows:

$$WER = \frac{INS_w + DEL_w + SUBS_w}{N_w}$$

$$CER = \frac{INS_c + DEL_c + SUBS_c}{N_c}$$

where N_w is the number of words in the reference and N_c represents the number of characters.

The higher the value, the higher the number of normalization operations needed to obtain the target sentence.

3.3 Overcoming Data Sparsity

Since our focus is on Dutch, our starting point is an existing Dutch corpus comprising three UGC genres, which were manually normalized (Schulz et al., 2016). The genres represented in this corpus are the following:

Tweets (TWE), which were randomly selected from the social network.

Message board posts (SNS), which were sampled from the social network Netlog, which was

[2]http://opennmt.net

a Belgian social networking website targeted at youngsters.

Text messages (SMS), which were sampled from the Flemish part of the SoNaR corpus (Treurniet et al., 2012).

This corpus is, to our knowledge, the only freely available parallel normalization dataset for Dutch.

Table 2 presents the number of parallel sentences in each genre and the number of words before and after normalization[3]. The WER and CER values computed between the original and target parallel sentence pairs are also shown. These values were calculated per sentence and averaged over the data set. As can be observed, the Dutch tweets (TWE) required hardly any normalization (a WER of 0.09 and a CER of 0.047). This can be explained by the fact that this platform has mainly been mainly adopted by professionals in Belgium who write in a more formal style (Schulz et al., 2016).

Genre	# Sent.	# Words		WER	CER
		Src	Tgt		
TWE	841	12951	12867	0.09	0.047
SNS	770	11670	11913	0.25	0.116
SMS	801	13063	13610	0.27	0.117

Table 2: Dutch parallel corpora data statistics.

On the other hand, we observe that the text messages (SMS) required most normalization (with a WER and CER score of 0.27 and 0.117, respectively). Table 2 also reveals that the corpus amounts to only a few hundred parallel sentences. NMT models often fail when insufficient data is provided (Ikeda et al., 2016; Saito et al., 2017). Because of that, we believe that the mentioned data would not be enough to successfully train a RNN model.

Under these conditions, we decided to experimentally verify which approach works best in order to overcome this data sparsity problem. Our objective is to find out whether annotating more data or using a set of data augmentation techniques is more beneficial and leads to better results.

Collecting More Data

First, we sampled and manually annotated ten thousand additional sentences for each of the three genres. We sampled SMS from the Flemish part of the SoNaR corpus (Treurniet et al., 2012). For the TWE genre, new data were retrieved by crawling Twitter using the Twiqs software[4], which was specifically designed to crawl Dutch tweets from Twitter. We used emoticons as keyword for the crawling process in order to collect text in which noisy words were present to some extent. For the SNS genre we relied on text collected from the social networking site ASKfm (Van Hee et al., 2018), where users can create profiles and ask or answer questions, with the option of doing so anonymously. ASKfm data typically consists of question-answer pairs published on a user's profile.

For all genres we made sure that there were no duplicates in the texts. Each message was also lowercased and tokenized using the NLTK tokenizer[5] prior to annotation, and the annotators also had to check the tokenization of the sentences. Besides this, hashtags, usernames and emoticons were replaced with a placeholder. All data was then normalized following the procedure described in Schulz et al. (2016). Table 3 shows the size of the newly annotated data in terms of the number of sentences and tokens for each genre. The WER and CER values give an insight into the normalization needed for each of the genres.

Genre	# Sent.	# Words		WER	CER
		Src	Tgt		
TWE	5190	124578	122165	0.10	0.062
SNS	8136	108127	110326	0.23	0.094
SMS	7626	111393	113846	0.15	0.067

Table 3: Parallel corpora data statistics after new annotations.

As can be derived from Table 3, TWE remains the least noisy genre, whereas the SNS genre is now the noisiest one with higher WER and CER values.

Applying Data Augmentation Techniques

A less time-consuming way to overcome data sparsity is to investigate the added value of data augmentation. A rudimentary way to augment the existing data, is to simply add monolingual standard data to both the source and target side of the corpus. This implies providing a large number of standard Dutch sentences from which the model can learn the standard word use. This type of data augmentation, however, is very primitive and, although it could probably provide good training

[3] All data was normalized following the procedure described in Schulz et al. (2016)

[4] https://github.com/twinl/crawler

[5] https://www.nltk.org/api/nltk.tokenize.html

data to learn the standard form of sentences, the non-canonical form of the words would be heavily (and even more so than in the beginning) underrepresented in the training data.

In order to address this problem, we took our initial parallel data as starting point for the data augmentation. Since we want to obtain more instances of non-canonical words and their corresponding normalized forms, we first relied on pretrained embeddings to replace the standard words by similar ones. These embeddings where trained on Flemish newspapers and magazines data collected from 1999 to 2014 in the Mediargus Archives, which can be accessed through GoPress Academic[6].

	Sentences
Source	jaaa sws *toch* <emoji> *hij is echt leuk*
Target	ja sowieso *toch* <emoji> *hij is echt leuk*
Embed.	jaaa sws **toch** <emoji> **hijzelf** is **wel tof**
	jaaa sws **toch** <emoji> hij **blijft** echt leuk
	jaaa sws **maar** <emoji> hij is **gewoon** leuk
	jaaa sws **dus** <emoji> hij is **inderdaad tof**
English	yes anyway **but/so** <emoji> **he/himself** is **really/just/indeed nice/cool**

Table 4: Data augmentation using pretrained embeddings.

Using this technique, we produced synthesized similar sentences containing the original user-generated text in it. In Table 4, we illustrate this augmentation technique, starting from the user-generated text *jaaa sws toch :) hij is echt leuk* (*yes anyway <emoji> he is really nice*). It is important to emphasize that we only replaced words that were already in their standard form in both the source and target side of the corpus. The replacements were made using the most similar words from the embeddings, using a similarity threshold with a value equal or greater than 0.5.

In the upper part of Table 4 the standard words in the source and target sentences are placed in cursive. In the middle and lower part, the bold words show the replacement for the standard words based on the embeddings. Please note that this replacement sometimes caused a (slight) change in the semantics of the sentence, as in *jaaa sws toch <emoji> hij blijft echt leuk*. However, since we are only dealing with lexical normalization, we argue that this is not an issue for the task.

In a second step, we applied an additional data augmentation technique which produces new abbreviations on the source side of the parallel corpus. We made use of a dictionary of about 350 frequent abbreviations appearing in social media texts, such as *lol* (*laughing out loud*) and *aub* for *alstublieft* (*you are welcome*) (Schulz et al., 2016). We went through every sentence in the newly augmented dataset and duplicated every sentence pair in which a standard word or phrase appeared in the dictionary. In the new source sentence, this standard word was then replaced by its corresponding abbreviation. For those cases in which a standard word had several abbreviation forms, a new original sentence for each of the abbreviation forms was generated.

Table 5 exemplifies this technique. It first lists two examples of newly generated sentences using the embeddings after which the abbreviations step is applied, leading to two additional new sentences.

	Sentences
Source	jaaa sws *toch* <emoji> *hij is echt leuk*
Target	ja sowieso *toch* <emoji> *hij is echt leuk*
Embed.	jaaa sws **maar** <emoji> hij is **gewoon** leuk
	jaaa sws **dus** <emoji> hij is **inderdaad tof**
Abbr.	jaaa sws **maar** <emoji> hij is ***gwn*** leuk
	jaaa sws **dus** <emoji> hij is ***idd*** **tof**
English	yes anyway **but/so** <emoji> he is **really/just/indeed nice/cool**

Table 5: Data augmentation using dictionary of abbreviations.

3.4 Experimental Setup

To conduct the experiments, both approaches were applied in several steps and combinations.

The first round of experiments **(Setup 0)**, which we will consider as the baseline system, consisted in applying the NMT architecture to the original small Dutch parallel corpus, as presented in Table 2. We performed 10-fold cross validation experiments to ensure that each part of the dataset would be used once as training and test set. For the remaining experiments (see Setup 1 to 3 below), this entire small dataset was used as held-out test set in order to find out which approach works best for overcoming data sparsity.

	INS	DEL	SUBS	SUM
TWE	1118	934	355	2407
SNS	2483	2238	1021	5742
SMS	4209	508	758	5475

Table 6: Operations needed at the character level to normalize the test set for each genre.

[6]https://bib.kuleuven.be/english/ub/searching/collections/belgian-press-database-gopress-academic

Table 6 shows the number of insertions (INS), deletions (DEL) and substitutions (SUBS) that are needed to transform the source sentences of the test set into manually normalized ones. The last column of the table shows the overall number of operations that would need to be solved by the systems.

The new annotations as presented in Table 3 were used for training **(Setup 1)**. In a next step, we trained the NMT model on the data obtained by first applying the embeddings technique to the newly annotated data **(Setup 2)** and then together with the abbreviation step **(Setup 3)**.

	#Sent.	# Words	
TWE		**Src**	**Tgt**
1	5190	124578	122165
2	697441	20692213	20416466
3	853465	26316110	26002836
SNS		**Src**	**Tgt**
1	8136	108127	110326
2	577281	21120858	21499465
3	835091	70337827	71257870
SMS		**Src**	**Tgt**
1	7626	111393	113846
2	615195	15356594	15669683
3	766946	22066532	22651464

Table 7: Parallel corpora data statistics for each experimental setup.

Table 7 shows the number of parallel training sentences and words in the original and target sides of each setup for each genre.

4 Experiments

As we explained before, the goal of our work is to experimentally verify which approach works best to overcome the data sparsity problem for a low-resourced language such as Dutch UGC: annotating more data or using data augmentation techniques.

Figure 2 presents the WER results for each genre with the different experimental setups and Figure 3 the CER results. As expected, the 10-fold cross validation experiments on the tiny original datasets (Setup 0) leads to bad results. The system's output consisted of sentences of the type *<emoji> , de , , , . . . <emoji>*. These are random repetitions of the most represented tokens in the training data like *ik* (*I* in English), punctuation marks or *<emoji>* labels. The system was thus unable to learn from such a small dataset.

Annotating more data (Setup 1) consistently results in better WER scores, with a more than 50% WER and CER reduction for the SMS genre. The data augmentation techniques (Setup 2 and 3) seem to further affect the normalization results. For the SNS and SMS genre, the WER and CER significantly decreases with the data augmentation technique relying on embeddings. However, when more noise is introduced to the data set (Setup 3) this adversely affects the WER and CER values for all genres.

However, these data augmentation techniques only seem to work when working with similar data. Recall that for the annotation of the new training data, similar data were collected for both the SMS and SNS genre. However, given the lack of noise in the Twitter data, we opted for another strategy to collect the tweets, viz. we used emoticons as keyword to query Twitter. This has resulted in a different style of Twitter corpus which comprises more noisy and longer tweets (see CER in Table 2 vs. Table 3). This difference in training data seems to be amplified during data augmentation. Whereas the use of more training data yields a large WER reduction (from 1.16 to 0.72 WER and 1.37 to 0.74 CER), this effect of adding more data is undone when augmenting the training data with similar data. However, the overall CER and WER results are still better than when just relying on the small original dataset (Setup 0).

When comparing the results of Setups 2 and 3 with the original WER values of our test set (see Table 2) we observe that the normalization task was not solved at all as the WER values are almost always higher. Only for the SMS genre, the results are somewhat similar, with WERs of 0.25 and 0.27 for setups 2 and 3 respectively.

4.1 Error Analysis

Table 8 shows some examples of sentences which were normalized using the proposed NMT approach.

However, the approach still produces a large number of odd normalizations. For example, in the second sentence in Table 8, the system was unable to correctly normalize some of the words. For instance, the word *byyyy* (*there*) was incorrectly normalized as the verb *gedragen* (*behave*). Furthermore, the system also produced odd translations of words that already were in their standard form. For example, the word *tammy*, which is a proper name, is changed into *toevallig* (*accidentally*) and the word *u* (*you*) was duplicated in

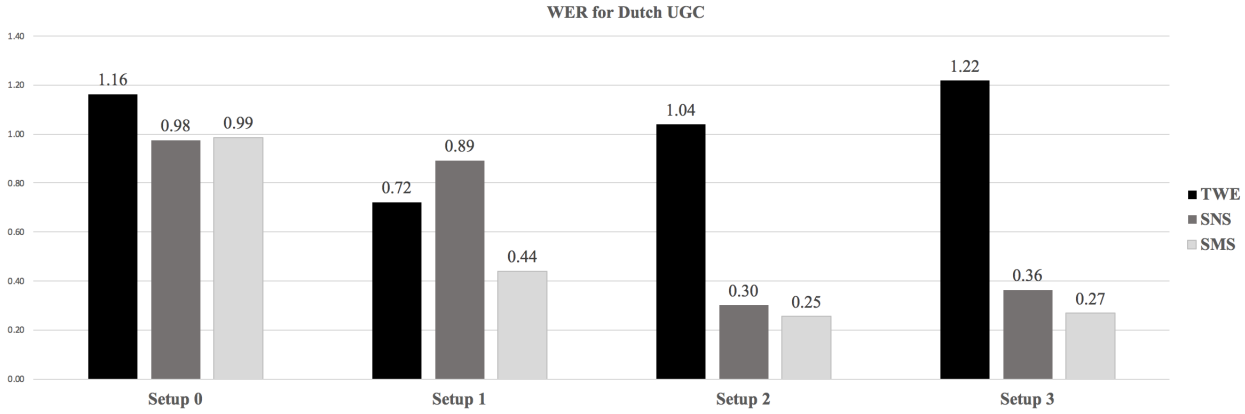

Figure 2: Experimental WER results for each genre using the different setups.

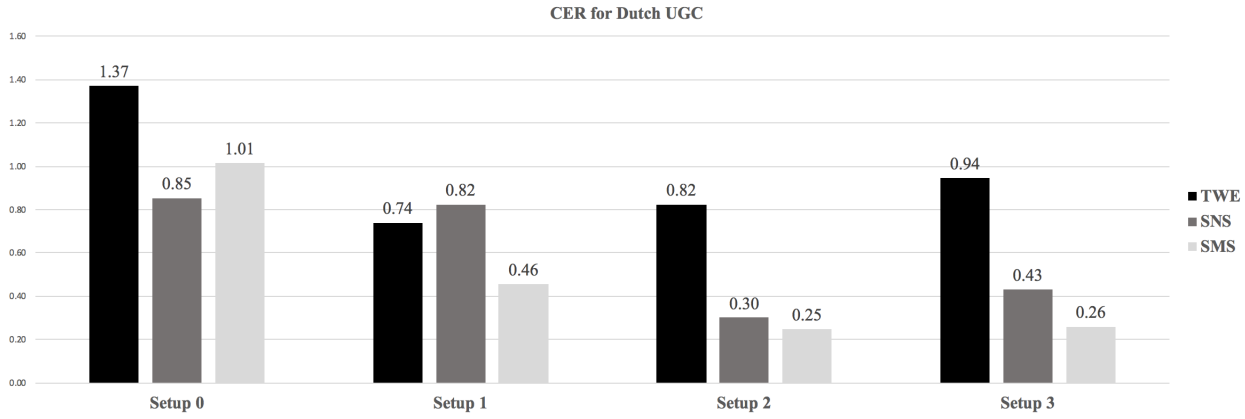

Figure 3: Experimental CER results for each genre using the different setups.

src sent.	haha zekr mdaje n vriend vn matti zyt ! <emoji> x
norm sent.	haha zeker omdat je een vriend van matti zijt ! <emoji> x
tgt sent.	haha zeker omdat je een vriend van matti zijt ! <emoji> x
English	haha certainly because you are a friend of matti !<emoji> x
src sent.	eeh ik sta dr nie byyyy <emoji> l y mis je
norm sent.	he ik sta er niet gedragen <emoji> love you mis je
tgt sent.	he ik sta er niet bij <emoji> love you mis je
English	hey i am not there <emoji> love you miss you
src sent.	jonge tammy , k u
norm sent.	jongen toevallig , ik u u
tgt sent.	jongen tammy , ik u
English	boy tammy, me you

Table 8: Examples of original (src), predicted (norm) and target (tgt) sentences using the NMT approach. An English translation is also provided.

with encoder-decoder models (Sproat and Jaitly, 2017).

TWE	**INS**	**DEL**	**SUBS**	**SUM**
Test	1118	934	355	2407
Setup 0	1101	930	351	2382
Setup 1	1099	924	330	2353
Setup 2	1105	917	326	2348
Setup 3	1094	903	325	2322
SNS	**INS**	**DEL**	**SUBS**	**SUM**
Test	2483	2238	1021	5742
Setup 0	2454	2225	992	5671
Setup 1	2468	2220	995	5683
Setup 2	2366	2092	968	5426
Setup 3	2366	2141	971	5478
SMS	**INS**	**DEL**	**SUBS**	**SUM**
Test	4209	508	758	5475
Setup 0	4107	505	727	5339
Setup 1	4094	483	700	5277
Setup 2	4099	470	709	5278
Setup 3	4090	468	710	5268

Table 9: Number of solved operations at the character level after normalization for each genre.

the normalized sentence. So, while the *k* (*I*) was correctly normalized, two errors were introduced through normalization. This is a common issue

Ideally, the number of operations after normalization should be reduced to zero. As can be derived from the Table 9 many cases where correctly

normalized by the systems. However, we can also observe that at the same time a large number of over-normalizations is introduced (Figures 2 and 3). Regarding data augmentation, we conclude that these techniques mainly help to reduce the number of over-normalizations.

5 Conclusions and Future Work

In this article, we have applied text normalization to Dutch written user-generated content from different genres: text messages, message board posts and tweets. We followed a Neural Machine Translation approach to solve the task and investigated different data augmentation techniques to tackle the problem of data sparsity. Results show that for most of the genres, augmenting the data by using pretrained embeddings helped to reduce the errors introduced by the NMT approach. On the other hand, for most of the genres Setup 0, i.e. training the NMT system on the small in-domain data set, solved most of the normalization problems in the test set.

Regarding the quality of the normalization, despite many of the non-standard words being correctly normalized, the system also produced odd translations, which is a common error using encoder-decoder architectures (Sproat and Jaitly, 2016). This is reflected in the number of over-normalizations that are produced by the system. With respect to these errors, we believe that following a modular approach that helps to solve the remaining errors, instead of only using NMT, could lead to a better performance.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. *Proceedings of the COLING/ACL on Main conference poster sessions* -, (July):33–40.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint*, pages 1–15.

Richard Beaufort and Sophie Roekhaut. 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics*, 1(July):770–779.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. *BMVC 2014 - Proceedings of the British Machine Vision Conference 2014*, pages 1–11.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *International Conference on Learning Representations ICLR*.

Mason Chua, Daan Van Esch, Noah Coccaro, Eunjoon Cho, Sujeet Bhandari, and Libin Jia. 2018. Text Normalization Infrastructure that Scales to Hundreds of Language Varieties. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, pages 1353–1356, Miyazaki, Japan. European Language Resource Association.

Eleanor Clark and Kenji Araki. 2011. Text normalization in social media: Progress, problems and applications for a pre-processing system of casual English. *Procedia - Social and Behavioral Sciences*, 27(Pacling):2–11.

Orphée De Clercq, Sarah Schulz, Bart Desmet, and Véronique Hoste. 2014. Towards Shared Datasets for Normalization Research. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1218–1223.

Orphée De Clercq, Sarah Schulz, Bart Desmet, Els Lefever, and Véronique Hoste. 2013. Normalization of Dutch User-Generated Content. *Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013)*, pages 179–188.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2014. Diffusion of lexical change in social media. *PLoS ONE*, 9(11):1–13.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-Resource neural machine translation. *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 2:567–573.

Bo Han and Timothy Baldwin. 2011. Lexical Normalisation of Short Text Messages : Makn Sens a #twitter. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 368–378.

Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Japanese Text Normalization with Encoder-Decoder Model. *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 129–137.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction. pages 1546–1556.

Max Kaufmann. 2010. Syntactic Normalization of Twitter Messages. *International conference on natural language processing*, 2:1–7.

Guillaume Klein, Yoon Kim, Yuntian Deng, Josep Crego, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source Toolkit for Neural Machine Translation. *arXiv preprint*.

Catherine Kobus, Francois Yvon, and Geéraldine Damnati. 2008. Normalizing SMS: are two metaphors better than one? *Proceedings of the 22nd International Conference on Computational Linguistics*, 1(August):441–448.

Philipp Koehn, Hiue Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, and Brooke Cowan. 2007. Moses: Open source toolkit for statistical machine translation. *Prague: Proceedings of 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions.*, (June):177–180.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions. *Soviet physics doklady*, 10(8):707–710.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *arXiv preprint*.

Massimo Lusetti, Tatyana Ruzsics, Anne Göhring, Tanja Samardi Samardžic, and Elisabeth Stark. 2018. Encoder-Decoder Methods for Text Normalization. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 18–28.

Soumil Mandal and Karthick Nanmaran. 2018. Normalization of Transliterated Words in Code-Mixed Data Using Seq2Seq Model & Levenshtein Distance. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 49–53, Brussels, Belgium. Association for Computational Linguistics.

D Pennell and Y Liu. 2011. A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations. *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 974–982.

Alexandre Pinto, Hugo Gonalo Oliveira, and Ana Oliveira Alves. 2016. Comparing the performance of different NLP toolkits in formal and social media text. *OpenAccess Series in Informatics*, 51(3):31–316.

Kanishka Rao, Fuchun Peng, Hasim Sak, and Francoise Beaufays. 2015. Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2015-Augus, pages 4225–4229.

Itsumi Saito, Jun Suzuki, Kyosuke Nishida, and Kugatsu Sadamitsu. 2017. Improving Neural Text Normalization with Data Augmentation at Character- and Morphological Levels. *Proceedings of the The 8th International Joint Conference on Natural Language Processing*, pages 257–262.

Ryohei Sasano, Sadao Kurohashi, and Manabu Okumura. 2015. A Simple Approach to Unknown Word Processing in Japanese Morphological Analysis. *Journal of Natural Language Processing*, 21(6):1183–1205.

Sarah Schulz, Guy De Pauw, Orphée De Clercq, Bart Desmet, Véronique Hoste, Walter Daelemans, and Lieve Macken. 2016. Multimodular Text Normalization of Dutch User-Generated Content. *ACM Transactions on Intelligent Systems and Technology*, 7(4):1–22.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 3:1715–1725.

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

Richard Sproat and Navdeep Jaitly. 2016. RNN Approaches to Text Normalization: A Challenge. *Computing Research Repository (CoRR)*, abs/1611.0.

Richard Sproat and Navdeep Jaitly. 2017. An RNN model of text normalization. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2017-Augus:754–758.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems*, pages 3104–3112.

Luke Taylor and Geoff Nitschke. 2017. Improving Deep Learning using Generic Data Augmentation. *arXiv preprint*.

Maaske Treurniet, Henk van den Heuvel, Nelleke Oostdijk, and Orphée De Clercq. 2012. Collection of a corpus of Dutch SMS. *Proceedings of the Eight*

Conference of International Language Resources and Evaluation., pages 2268–2273.

Cynthia Van Hee, Gilles Jacobs, Chris Emmery, Bart Desmet, Els Lefever, Ben Verhoeven, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2018. Automatic detection of cyberbullying in social media text. *Plos One*, 13(10):1–21.

Cynthia Van Hee, Marjan Van De Kauter, Orphée De Clercq, Els Lefever, Bart Desmet, and Véronique Hoste. 2017. Noise or music? Investigating the usefulness of normalisation for robust sentiment analysis on social media data. *Revue Traitement Automatique des Langues*, 58(1):63–87.

Reinhild Vandekerckhove and Judith Nobels. 2010. Code eclecticism : Linguistic variation and code alternation in the chat language of Flemish teenagers. *Journal of Sociolinguistics*, 14(5):657–677.

Quoc Le Oriol Vinyals William Chan, Navdeep Jaitly. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2016*, pages 4960–4964.

Zhenzhen Xue, Dawei Yin, and Bd Davison. 2011. Normalizing Microtext. *Analyzing Microtext*, pages 74–79.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75.

Linhong Zhu, Aram Galstyan, James Cheng, and Kristina Lerman. 2014. Tripartite graph clustering for dynamic sentiment analysis on social media. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1531–1542.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer Learning for Low-Resource Neural Machine Translation. pages 1568–1575.