Normalization of Dutch User-Generated Content

Orphée De Clercq[1,2], **Sarah Schulz**[3], **Bart Desmet**[1,2], **Els Lefever**[1,2] **and Véronique Hoste**[1,3]

[1] LT[3], Language and Translation Technology Team – University College Ghent
Groot-Brittanniëlaan 45, 9000 Ghent, Belgium
[2] Department of Applied Mathematics and Computer Science – Ghent University
Krijgslaan 281 (S9), 9000 Ghent, Belgium
[3] Department of Lingusitics – Ghent University
Blandijnberg 1, 9000 Ghent, Belgium

Firstname.Lastname@UGent.be

Abstract

This paper describes a phrase-based machine translation approach to normalize Dutch user-generated content (UGC). We compiled a corpus of three different social media genres (text messages, message board posts and tweets) to have a sample of this recent domain. We describe the various characteristics of this noisy text material and explain how it has been manually normalized using newly developed guidelines. For the automatic normalization task we focus on text messages, and find that a cascaded SMT system where a token-based module is followed by a translation at the character level gives the best word error rate reduction. After these initial experiments, we investigate the system's robustness on the complete domain of UGC by testing it on the other two social media genres, and find that the cascaded approach performs best on these genres as well. To our knowledge, we deliver the first proof-of-concept system for Dutch UGC normalization, which can serve as a baseline for future work.

1 Introduction

In the past two decades, many resources have been invested to develop state-of-the-art text processing tools for Dutch[1]. Similar to other reported languages, these tools, which have all been developed with standard text in mind, show a significant drop in performance when applied to user-generated content (UGC). This is for example the case when applying parsing (Foster et al., 2011) or named entity recognition (Liu et al., 2011b; Ritter et al., 2011) to Twitter data. Typical problems that hinder automatic text processing include the use and productivity of abbreviations, deliberate misspellings, phonetic text, colloquial and ungrammatical language use, lack of punctuation and inconsistent capitalization.

No systems currently exist to automatically normalize Dutch noisy text into its standard equivalent. In order to develop a system which can handle different types of user-generated content, we collected and studied three social media genres: text messages, message board posts and tweets. In this paper, we investigate the viability of adopting a character-based machine translation approach to the normalization task. This is different from previous research investigating MT approaches for normalization, which has mainly focused on token-based translation (Aw et al., 2006; Kobus et al., 2008).

For our experiments we first focus on the genre that poses the largest number of normalization challenges in our corpus, namely text messages, in order to have a proof of concept. We will show that a cascaded SMT system with a token-based module followed by a transliteration at the character level yields the best results, i.e. a 63% drop in word error rate. In this cascade, the first module aims at obtaining high precision, thus presenting high-confidence translations. The second module further improves this output by generalizing over character mappings.

To conclude, we applied this proof-of-concept system tuned for text messages to the other genres and observed similar improvements.

The paper is structured as follows. After the literature overview (Section 2) we discuss the social

[1] Among others, in the framework of the STEVIN programme, see Spijns and Odijk (2013) for an overview.

Proceedings of Recent Advances in Natural Language Processing, pages 179–188,
Hissar, Bulgaria, 7-13 September 2013.

media genres used, their characteristics and how these have been normalized in Section 3. The set-up and experiments are presented in Section 4. We examine the results in Section 5, perform a qualitative error analysis in Section 6 to end with some conclusions and prospects for future work in Section 7.

2 Related Work

Traditionally, the task of text normalization is a crucial first step for every text-to-speech system, in which specific numbers and digit sequences, acronyms, etc. need to be rewritten in order to have them pronounced correctly. A thorough overview of the main characteristics and bottlenecks can be found in Sproat et al. (2001).

More recently, however, the surge of user-generated content has introduced new problems such as non-existent abbreviations and deliberate misspellings. This reality combined with the need to process UGC data has revived the interest in normalization techniques. In this regard, we can define three dominant approaches to transfer noisy into standard text. These are referred to as the spell-checking, machine translation and speech recognition metaphors (Kobus et al., 2008).

The most intuitive way of normalizing text would be to approach the problem as a spell-checking one where noisy text has to be transformed to standard text using noisy channel models. Choudhury et al. (2007), for example, proposed a supervised noisy channel model using Hidden Markov Models to calculate the probability of less frequent words. Extensions to this approach were made by studying word processes (Cook and Stevenson, 2009), adapting weighted finite-state machines and rewrite rules (Beaufort et al., 2010) or by adding other elements such as orthographic, phonetic and contextual factors (Xue et al., 2011).

Another approach is using statistical machine translation (SMT) techniques for text normalization. Previous work in this field has mostly focused on phrase-based machine translation at the word level. Aw et al. (2006) were the first to compare dictionary substitution using frequencies with phrase-based machine translation. They revealed that SMT improves BLEU scores for English SMS translation. Also working on English text, Raghunathan et al. (2009) confirmed that using an SMT system outperforms a dictionary look-up, most notably when used on an out-of-domain test set.

Kobus et al. (2008) followed the same approach but combined the machine translation features with a speech recognition approach using HMMs on a French corpus. They concluded that the two systems perform better on different aspects of the task and that combining these two modules works best.

A different way of approaching normalization is the work by Liu et al. (2011a; 2012). They propose a cognition-driven text normalization system using an unsupervised approach. By observing and simulating human techniques for the normalization task, they avoid dependence on human annotations. They construct a broad-coverage system to enable better word-coverage, using three key components: enhanced letter transformation, visual priming and string/phonetic similarity.

If we consider normalization, the task intuitively has a lot in common with transliteration tasks for which character-based SMT systems have proven adequate (Vilar et al., 2007). Pennell and Liu (2011) were the first to study character-based normalization. They, however, limited their approach by only focusing on abbreviations.

In this paper, we propose a cascaded model that follows a machine translation approach and tries to tackle the full range of normalization problems.

3 Three Genres of UGC

In order to normalize using a machine translation system, and to evaluate the performance, it is essential to build a gold standard data set that can serve as training and test material. As far as we know, no such data set is currently available for Dutch.

3.1 Corpus Compilation

To ensure that our corpus is representative of the domain of user-generated content (UGC), we decided to include three different social media genres: text messages (SMS), message board posts from a social networking site (SNS) and tweets (TWE). As text messages, we sampled 1,000 messages from the Flemish part of the SoNaR corpus (Treurniet et al., 2012), aimed at a balanced spread of two characteristics: age and region. In order to also include longer streams of UGC, 1,505 message board posts were randomly selected from the social networking site Netlog, which is popular amongst Belgian teenagers. In order to take into

	ORIGINAL	NORMALIZED	TRANSLATED
SMS	Oguz ! Edde me Jana gesproke ? En ze flipt lyk omdak ghsmoord heb .. !	Oh gods ! Heb je met Jana gesproken ? En ze flipt gelijk omdat ik gesmoord heb ... !	Oh god ! Did you speak to Jana ? And she's flipping because I smoked ... !
SNS	schaaaat , Je komt wel boven die Blo , je et em nii nodig wie jou laat gaan is gwn DOM :p Iloveyouuuu hvj	schat , Je komt wel boven die Blo , je hebt hem niet nodig wie jou laat gaan is gewoon dom :p I love you hou van je	honey, You'll get over that Blo, you don't need him whoever lets you go is just stupid :p I love you I love you
TWE	@minnebelle top ! Tis voor m'n daddy !	@minnebelle top ! Het is voor m'n daddy !	@minnebelle great ! It is for my daddy !

Table 1: Examples of UGC from the three social media genres representing the original utterance, its normalized version and an English translation

account the vast amount of normalization research done on Twitter data, we also included 246 randomly selected tweets. It is to be noted, however, that average Twitter content in Belgium differs from that in English-speaking countries or The Netherlands, because Twitter has mainly been adopted amongst professionals. An example of each genre can be found in the left column of Table 1.

These examples clearly illustrate the main characteristics of Dutch UGC, most of which are similar to previously reported problems in other languages (Baron, 2003; Beaufort et al., 2010).

Some of the more well-known problems include the omission of words or characters, e.g. the omission of the final *n* in *gesproke* (Eng: *spoke* versus *spoken*). The frequent use of abbreviations and acronyms, such as *gwn, hvj* (Eng: *LOL*), which are highly productive. Moreover, many utterances deviate from the standard spelling at the lexical level, such as *lyk* instead of *gelijk* (Eng: *luv* versus *love*) or by writing colloquially, e.g. *et em* instead of *hebt hem* (Eng: *you iz* vs *you are*). In UGC, emotions are also expressed by using flooding (repetition of the same character or sequence, *baaaaaaby*), emoticons (*:p*) and capitalized letters (*STUPID*).

More specific to the Dutch language is the concatenation of tokens which leads to the elimination of clitics and pronouns (*Edde* instead of *Heb je*, *khou* instead *ik hou*, *Tis* instead of *Het is*). Moreover, the influence of the English-speaking world on Belgium and the fact that it is a trilingual country often leads to various languages within a single utterance, which are often adapted to Dutch aspects (*Oguz, daddy, we are forever*).

3.2 Corpus Annotation

All text material was annotated by two annotators, independently of each other using newly developed normalization guidelines. These guidelines, tailored for Dutch, have been drawn up in close collaboration with the developers of the Chatty Corpus (Kestemont et al., 2012).

The guidelines can be roughly divided into two parts. The first part consists of the actual text normalization and comprises three steps: clearing all obvious tokenization problems, stating the different normalization operations and writing down the full normalized version. We allow four different operations: insertions, deletions, substitutions and transpositions; examples of tokens requiring these operations are given below (in English).

- INS: spoke (spoken), sis (sister)
- DEL: baaaaabyyyy (baby)
- SUB: iz (is), stoopid (stupid)
- TRANS: liek (like)

Insertions allow to indicate missing characters in a string. Deletions are used when characters should be deleted from a certain string. Substitutions are used when a character has been replaced with another similar one. Finally, transpositions are used when a combination of characters should be switched within one string.

The second part consists of flagging additional information that might be useful for automatic processing purposes. Within each utterance the

Genre	#	Before	After	%	#INS	#DEL	#SUB	#TRANS
SMS	1000	16630	17194	3.39	3622	338	547	57
SNS	1505	31513	32221	2.25	4165	1500	1692	57
TWE	246	3276	3357	2.47	923	67	127	4

Table 2: Data statistics of the three genres of UGC. The left-hand side shows the number of tokens before and after normalization and the increase in %. The right-hand side visualizes the actual normalization effort expressed in the number of operations.

annotators were asked to indicate the end of a thought (to account for missing punctuation), regional words, foreign words and named entities. They could also flag words that are ungrammatical, stressed, part of a compound, used as interjections or words that require consecutive normalization operations.

To check the reliability of our annotation guidelines, the two annotators each normalized the 1,000 text messages. We estimated the inter-annotator reliability by computing the word error rate (cf. infra) between the two fully normalized versions. The WER was 0.048, which indicates near-perfect overlap.

In order to give an idea of the normalization effort required, we present some data statistics for each genre in Table 2. The left-hand side visualizes the increase in the number of tokens before and after normalization in absolute numbers and percentage-wise. On the right one can see the actual normalization effort, which is expressed by the number of individual operations. The normalized versions of the previously mentioned examples can be found in the middle column of Table 1 and their translation to English in the right column.

For the experiments presented in this paper we work with the first part of the manual normalization (ignoring flagging information such as ends of thought). We chose to focus on SMS, because it was the noisiest data in our corpus, with a token increase of 3.39% (see Table 2).

4 System Architecture

Using SMT for noisy text normalization can be done at various levels of granularity. The advantage of working at the token level is that the high-frequency words and abbreviations can be translated in context, which outperforms a simple dictionary look-up (Raghunathan and Krawczyk, 2009). However, working at the character level allows one to generalize over character mappings which makes the system more robust (Pennell and Liu, 2011).

Prior to any sort of learning, we adapted our tokenizer to be able to handle emoticons, hyperlinks, hashtags and at-replies. Similar to Beaufort et al. (2010), we devised some rewrite rules: we decided to tackle the flooding of characters before translating in order to avoid too many confounding factors. Characters and character sequences were allowed to occur twice consecutively, at maximum. A higher number of repetitions was reduced to two. The validity of this approach was checked by running the rewrite module on the CELEX database (Baayen et al., 1995), which contains 381,292 valid Dutch words, including inflections. Only two (highly infrequent) entries were changed by the module, which confirms that it virtually does not overnormalize.

After this preliminary preprocessing, the noisy text is processed by two modules. First, the standard phrase-based SMT approach at the token level is used to ensure the translation of the more frequently used abbreviations (such as *fb* for *facebook* and other highly frequent normalization problems, e.g. *tht* for *that*). Afterwards, the translated text is split into characters and a translation at the character level takes place. This intuitively makes sense, because transformations at the character level are more likely to be reproduced than a combination of possible transformations at the word level. Trying to generalize such character transformations at the word level would probably fail due to data sparseness. We worked with both character unigram and bigram translation models. Bigrams supposedly have the advantage that one character of context across phrase boundaries is used in the selection of translation alternatives from the phrase table (Tiedemann, 2012). This means that more precise translations will be suggested.

For our experiments we first focus on the individual performance we can achieve within the

SMS genre, after which we test this approach on the other genres to see whether it is possible to create a robust system that can process all three UGC genres.

To evaluate our approach, both the Word Error Rate (WER) and BLEU scores were calculated. WER, an evaluation metric that is based on edit distance at the word level, is very well suited for the evaluation of NLP tasks where the input and output strings are closely related. As a consequence, the metric is used for the evaluation of optical character recognition (Kolak et al., 2003), grapheme-to-phoneme conversion (Demberg et al., 2007), diacritization (Schlippe et al., 2008) and vocalization of Arabic (Kübler and Mohamed, 2008). The BLEU metric, which has been specifically designed for measuring machine translation quality, measures the n-gram overlap between the translation being evaluated and a set of target translations. We therefore believe that BLEU is less appropriate for evaluation in the current set-up, but we include it for comparison's sake (as other systems mention it such as Aw et al. (2006), Kobus et al. (2008), etc.).

5 Experimental Set-up and Results

For all experiments, we used the Moses SMT system (Koehn et al., 2007). As a target corpus for our language model, we used the Spoken Dutch Corpus (Corpus Gesproken Nederlands, CGN (Oostdijk, 2000)) since spoken language could better reflect the language used in UGC. The target training data was also added to the model. All language models were built using the SRILM toolkit (Stolcke, 2002) with Witten-Bell discounting which has been proven to work well on small data sets (Tiedemann, 2012)

We experimented with different translation models. The token-level translation model was each time built using Moses with standard settings and a 5-gram language model. For the character-level model the same Moses setting was used. For the language model we experimented with different sizes of n on our training data, 5 - 7 - 10 - 15, and found that a 10-gram language model gave the best results.

For the first set of experiments (Section 5.1), training was performed on the SMS data, which was divided into three data sets: 625 messages for training, 125 for development and 125 for testing. In order to estimate the system's robustness to unseen genres, the SMS-tuned system was tested on the other two genres, 125 SNS posts and 125 tweets (Section 5.2).

5.1 Results on SMS

This is, to our knowledge, the first study on Dutch text normalization, so there is no basis for comparison to other systems. Figure 1 present a visual overview of the different set-ups' performance on the Dutch SMS data. We start by reporting the difference between the original source and target text (A) as well as a baseline where only the rewrite rules have been applied (B). We see that a moderate improvement in WER, from 21.70 to 21.47%, already occurs by eliminating flooding.

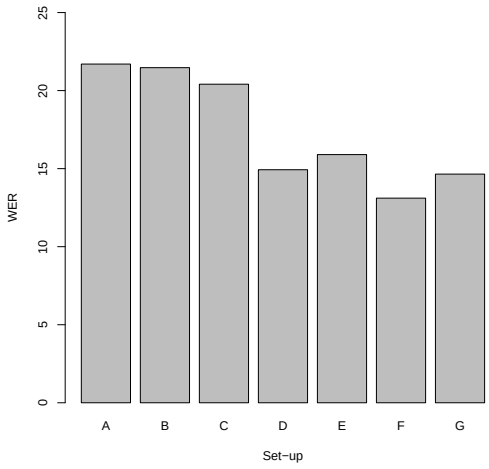

Figure 1: Visualisation of the WER reduction on the SMS data set using our seven different set-ups

In the next step, the various translation models were trained and tested and we clearly see that all the following results outperform the baseline. The token-based model (C) accounts for a moderate improvement but clearly the character-based models, both with unigrams (D) and bigrams (E), perform much better. Introducing the unigram and bigram cascaded models leads to the best results (F and G). The best result is reached by the cascaded unigram model (F). This model has a WER of 13.11 which is a 63% drop in word error rate over the baseline and 56% over the non-cascaded word level SMT.

If we perform the same analysis on the BLEU results (Figure 2), we observe a different tendency. Clearly, the token-based model (C) accounts here for the best performance whereas the cascaded un-

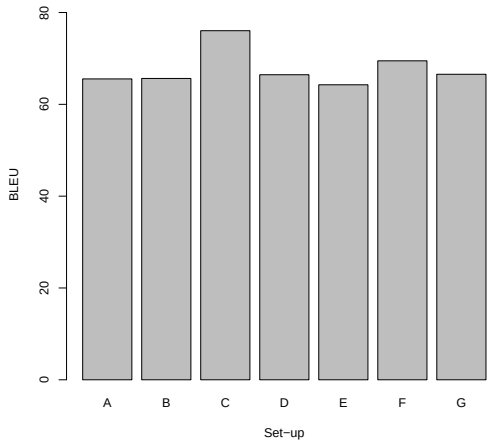

Figure 2: Visualisation of BLEU on the SMS data set using our seven different set-ups

igram model (F) only achieves the second best result. This could be explained by the inherent differences between the metrics. WER is based on the edit distance whereas BLEU measures n-gram overlap. This means that the output of the unigram cascaded model can be closer - but not perfect - to the reference than the output from the token model. If we consider the example below from our data we see that the token model was not able to find the correct version, whereas the cascaded unigram output is already a bit closer. If we would then feed this closer version back into our token model it should be able to resolve it correctly[2]. This insight could be used to improve our system by extending the cascaded unigram module with another run of the token-based system in future work.

- original: *laatk* – target: *laat ik*
- output C: *laatk* – output F: *laat k* – output C based on output F: *laat ik*

All experimental results on the SMS data, expressed both in WER and BLEU, can be found in Table 3.

5.2 Results on three genres of UGC

By testing our translation models tuned for text messages on the other two genres, we aim to verify the robustness of our approach. These results can be found in Table 3.

[2]Proof of this was found in the output of our token-based system.

Applying the baseline system with rewrite rules gives the same minor positive effect for SNS as for SMS, compared to the original source and target text. For tweets, on the other hand, no improvement is noted. Upon closer inspection of the Twitter data, not a single instance of flooding was found, which explains this status quo.

When comparing the other models, the same evolution in word error rate can be observed. For each genre, the best WER reduction over the baseline is reached with the cascaded unigram model, namely 63% for SMS, 39% for SNS and 28% for TWE. For the SNS data, the cascaded unigram and bigram translation models give an equal performance.

6 Error Analysis

We performed a qualitative error analysis of our best performing set-up, i.e. the cascaded unigram approach (F). After close inspection of the output on the SMS test data we learned that the system was able to locate and resolve 172 of the 320 words requiring normalization. Besides this, the system also generated 51 false positives, which leads to a precision of 77.13%, a recall of 55.66% and thus an overall F-measure of 64.66%.

In order to gain more insights, the instances our system missed were classified in two ways. We first inspected which types of operations seem most difficult to resolve (cf. Section 3.2).

Operations	Total required	Absolute # missed	Relative # missed
INS	549	270	49%
DEL	28	20	71%
SUB	55	30	54%
TRANS	11	6	54%

Table 4: Absolute number of the operations missed at the character-level together with the relative number when compared to the total number of operations

Since one word may need multiple or different operations[3], this was calculated at the character level. Table 4 presents the number of operations missed by our system both in absolute and relative numbers.

At first sight, especially the deletions seem hard to resolve, followed by the substitutions and trans-

[3]For example *sis* requires three insertions and *luv* requires both a substitution and an insertion.

Training Set-ups	Testing					
	SMS		SNS		TWE	
	WER	BLEU	WER	BLEU	WER	BLEU
A. Original	21.70	65.54	20.41	66.03	13.26	76.10
B. Baseline	21.47	65.64	20.36	65.93	13.26	76.10
C. Token-level only	20.41	76.04	25.03	73.26	19.03	78.32
D. Unigram only	14.93	66.45	15.41	64.02	13.52	66.29
E. Bigram only	15.90	64.26	15.17	63.94	14.08	65.50
F. Cascaded unigram	**13.11**	69.48	**14.59**	65.17	**10.35**	72.25
G. Cascaded bigram	14.65	66.55	**14.59**	64.79	10.36	72.25

Table 3: Results of the different set-ups on the SMS genre

positions. When taking the absolute numbers into account, however, proportionally these classes are much less frequent than the total number of insertions needed (549 to be exact). Apparently our system is able to resolve most of these insertions (i.e. 51%). On closer inspection, however, we found that the system is especially good in normalizing shorter words requiring only one or two insertions, such as *eb* for *heb*, *nie* for *niet*, and not in building longer words such as *gr* for *groetjes*. If we extrapolate this finding to the number of insertions needed at the word level, we indeed discovered that at the word level 60% gets successfully resolved. Another observation at the word level is that words requiring different types of operations are difficult for our system: only 44% is successfully replaced.

The second error classification consists of a more linguistically motivated subdivision. Inspired by the work of Androutsopoulos (2007), we defined three categories: abbreviation (ABBR), phonetic (PHON) and orthographic (ORTH) issues. Examples of some instances our system missed following this classification are presented in Table 5.

Classes	Output	Correct
ABBR	aug	augustus
PHON	hebk	heb ik
ORTH	uan	van

Table 5: Missed instances according to error classification 2

This classification is visualized in Figure 3, where we see that especially resolving phonetic problems seems difficult for our system, i.e. 103 instances. In order to better grasp this we had a closer look at the various phonetic issues and

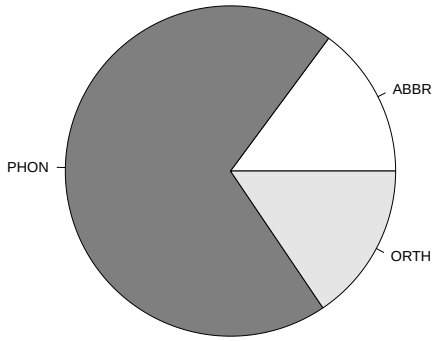

Figure 3: Pie chart visualizing the number of missed instances according to the second error classification

further classified these into fusions (concatenations of words, 25%), omissions (missing characters, 43%), equivalents (characters referring to the same sound, 26%) and onomatopoeias (sounds like, 6%). Especially the omission of characters seems problematic, which is consistent with the high number of missed insertions (i.e. 270 characters).

This error analysis indicates that our system might benefit from including other modules besides machine translation. The orthographic issues might probably be resolved using a spell checker, whereas the phonetic ones, especially the equivalents, might benefit from grapheme-to-phoneme conversion.

As far as the hypercorrections are concerned

(our system generated 51 false positives), we found that 15 of these are actually named entities or foreign words which should not be normalized at all. This is why we are also thinking of expanding our preprocessing module so that these words can be filtered out before processing them with the other modules.

7 Conclusion and Future Work

In this paper, we have discussed a cascaded machine translation approach to normalize Dutch user-generated content (UGC). Three social media genres have been collected and normalized using newly developed guidelines. After a short description of the main normalization errors and characteristics of this particular domain, we investigated the viability of an SMT approach at the character level.

Experiments on text messages, the genre requiring most normalization, revealed that a cascaded model where a token-based module is followed by a translation at the character level yields the best results. Testing this model on two other genres revealed the same trend, which indicates that this approach is robust across genres. To our knowledge, we have developed the first proof-of-concept system for Dutch UGC normalization, which can serve as a baseline for future work. A first error analysis revealed that our best system already reaches an F-measure of 64.66%. Looking at the different operations, insertions occur most frequently. Moreover, it appears that our system is best at resolving smaller words requiring only one or two insertions. When we analyzed the output in a different way, especially the high number of phonetic alternations remaining unresolved drew our attention.

For future work we believe that incorporating other modules into our system will further increase the overall performance. Considering the error analysis, we feel that a combination of the three metaphors (machine translation, spell checking and speech recognition) might produce an optimal combination of various features. Moreover, sometimes we would like to introduce a second round through some modules to tackle module-specific problems. In order to really evaluate the ability to generalize over multiple genres we are currently training and testing our system on the individual text genres. Since we aim to make a system that can handle UGC, we also envisage to combine our three genres and thus experiment on the full set. First experiments have revealed that this does indeed increase overall performance. For now, we have only focused on normalization at the lexical level, so colloquial and ungrammatical language usage also presents an interesting alley for future work. Since previous work on English text normalization using MT approaches at the character-level has only focussed on abbreviations (Pennell and Liu, 2011), we would also like to investigate whether our methodology can be applied to English noisy text as well.

We are looking for ways to make our data sets publicly available.

Acknowledgments

We would like to thank our annotators for their hard work. and the reviewers for their valuable comments. This work was funded by three projects: PARIS[4], SUBTLE[5] and AMiCA[6].

References

Jannis Androutsopoulos. 2007. Neue Medien neue Schriftlichkeit? *Mitteilungen des Deutschen Germanistenverbandes*, 1(7):72–97.

AiTi Aw, Zhang Min, Xiao Juan, and Su Jian. 2006. A Phrase-based Statistical Model for SMS Text Normalization. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 33–40, Sydney, Australia.

Harald R. Baayen, Richard Piepenbrock, and Leon Gulikers. 1995. The CELEX Lexical Database. Linguistic Data Consortium. CD-ROM.

Naomi S. Baron. 2003. Language of the internet. *The Stanford Handbook for Language Engineers*, pages 59–127.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of ACL*, pages 770–779.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proceedings of the NAACL HLT Workshop on Computational Approaches to Linguistic Creativity*.

Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological Constraints and Morphological Preprocessing for Grapheme-to-Phoneme Conversion. In *45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*,

[4]www.parisproject.be

[5]http://lt3.hogent.be/en/projects/subtle/

[6]http://www.amicaproject.be/

Prague, Czech Republic. Association for Computational Linguistics.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. #hard-toparse: POS tagging and parsing the twitterverse. In *Proceedings of the AAAI workshop on Analyzing Microtext*, pages 20–25.

Mike Kestemont, Claudia Peersman, Benny De Decker, Guy De Pauw, Kim Luyckx, Roser Morante, Frederik Vaassen, Janneke van de Loo, and Walter Daelemans. 2012. The netlog corpus. a resource for the study of flemish dutch internet language. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may.

Catherine Kobus, Yvon François, and Damnati Géraldine. 2008. Normalizing SMS: are two metaphors better than one? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 441–448, Manchester, UK.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Okan Kolak, William J. Byrne, and Philip Resnik. 2003. A Generative Probabilistic OCR Model for NLP Applications. In *HLT-NAACL 2003: conference combining Human Language Technology conference series and the North American Chapter of the Association for Computational Linguistics conference series*, Edmonton, Canada.

Sandra Kübler and Emad Mohamed. 2008. Memory-based vocalization of Arabic. In *Proceedings of the LREC Workshop on HLT and NLP within the Arabic World.*, Marrakech, Morocco.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011a. Insertion, deletion or substitution? Normalizing text messages without pre-categorization nor supervision. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 71–76.

Xiaohua Liu, Furu Wei Shaodian Zhang, and Ming Zhou. 2011b. Recognizing named entities in tweets. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 359–367.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1035–1044.

Nelleke Oostdijk. 2000. The spoken dutch corpus. Outline and rst evaluation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 887–894.

Deana L. Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of SMS abbreviations. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Karthik Raghunathan and Stefan Krawczyk. 2009. CS224N: Investigating SMS Text Normalization using Statistical Machine Translation. Technical report, Stanford University: Department of Computer Science.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of Empirical Methods for Natural Language Processing EMNLP*, pages 1524–1534.

Monojit Choudhury Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupan Basu. 2007. Investigating and modeling the structure of texting language. *International Journal on Document Analysis and Recognition*, 10(3):157–174.

Tim Schlippe, ThuyLinh Nguyen, and Stephan Vogel. 2008. Diacritization as a machine translation problem and as a sequence labeling problem. In *MT at work: Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA-2008)*, pages 270–278, Waikiki, Hawai'i.

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.

Peter Spyns and Jan Odijk. 2013. *Essential Speech and Language Technology for Dutch, Theory and Applications of Natural Language Processing*. Springer, Berlin.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002*, pages 901–904.

Jörg Tiedemann. 2012. Character-based pivot translation for under-resourced languages and domains. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 141–151, Avignon, France.

Maaske Treurniet, Orphée De Clercq, Henk van den Heuvel, and Nelleke Oostdijk. 2012. Collection of a Corpus of Dutch SMS. In *Proceedings of the 8th Language Resources and Evaluation Conference (LREC'12)*, pages 2268–2273, Istanbul, Turkey.

David Vilar, Jan-Thorsten Peter, and Herman Ney. 2007. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39, Prague, Czech Republic.

Zhenzhen Xue, Dawei Yin, and Brian D. Davison. 2011. Normalizing microtext. In *Proceedings of the AAAI Workshop on Analyzing Microtext*, pages 74–79.