Multimodular Text Normalization of Dutch User-Generated Content

SARAH SCHULZ, Ghent University
GUY DE PAUW, University of Antwerp
ORPHÉE DE CLERCQ, BART DESMET, and VÉRONIQUE HOSTE,
Ghent University
WALTER DAELEMANS, University of Antwerp
LIEVE MACKEN, Ghent University

As social media constitutes a valuable source for data analysis for a wide range of applications, the need for handling such data arises. However, the nonstandard language used on social media poses problems for natural language processing (NLP) tools, as these are typically trained on standard language material. We propose a text normalization approach to tackle this problem. More specifically, we investigate the usefulness of a multimodular approach to account for the diversity of normalization issues encountered in user-generated content (UGC). We consider three different types of UGC written in Dutch (SNS, SMS, and tweets) and provide a detailed analysis of the performance of the different modules and the overall system. We also apply an extrinsic evaluation by evaluating the performance of a part-of-speech tagger, lemmatizer, and named-entity recognizer before and after normalization.

CCS Concepts: • **General and reference → Evaluation**; • **Information systems → Social networking sites**; **Data cleaning**; • **Human-centered computing → Social networking sites**; • **Computing methodologies → Natural language processing**; **Machine translation**; **Phonology / morphology**; *Information extraction*; *Language resources*; • **Applied computing → Language translation;**

Additional Key Words and Phrases: Social media, text normalization, user-generated content

ACM Reference Format:
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 Trans. Intell. Syst. Technol. 7, 4, Article 61 (July 2016), 22 pages.
DOI: http://dx.doi.org/10.1145/2850422

1. INTRODUCTION

With the advent of Web 2.0, user interaction on the Internet has become common practice. According to Murugesan [2007], Web 2.0 is a conglomerate of technologies and strategies aimed at online user participation: it is highly dynamic and characterized by a productive user community. The online content that these users produce is called

The research described in this article was funded by IWT-SBO grant 120007 (AMiCA (http://www.amicaproject.be) and IWT-SBO grant 110067 (PARIS (http://www.parisproject.be).
Authors' addresses: S. Schulz, O. De Clercq, B. Desmet, V. Hoste, and L. Macken, Department of Translation, Interpreting, and Communication, Ghent University, Groot-Brittanniëlaan 45, 9000 Gent, Belgium; emails: {orphee.declercq, bart.desmet, veronique.hoste, lieve.macken}@ugent.be; G. De Pauw and W. Daelemans, Computational Linguistics and Psycholinguistics Research Center, University of Antwerp, Prinsstraat 13, 2000 Antwerp, Belgium; emails: {guy.depauw, walter.daelemans}@uantwerpen.be.
Author's current address: Sarah Schulz, Institute for Natural Language Processing, University of Stuttgart, Pfaffenwaldring 5B, 70569 Stuttgart, Germany; email: sarah.schulz@ims.uni-stuttgart.de.

user-generated content (UGC). Van Dijk [2009] discusses this new concept of *user* and his or her role and participation mechanisms in the virtual world. Phenomena known from face-to-face interaction are taken over into the virtual space and adjusted to it. Riegner [2007], for example, describes how the concept of word-of-mouth is adopted in cyberspace.

Since data generated in a variety of new online media holds potential for commercial and sociological applications, a need has emerged to automatically analyze it. From a commercial perspective, UGC has attracted the interest of research in a variety of text mining applications [Cortizo et al. 2012], including sentiment and opinion mining [Paltoglou and Thelwall 2012], which is used, for example, in user-tailored advertising [Aven et al. 2009]. But similar methods can also be used to automatically trace harmful content on social media [Peersman et al. 2011; Desmet and Hoste 2014; Van Hee et al. 2015]. This is especially important for the protection of teenagers and fits an urgent need. Royen et al. [2015] describe the harmfulness of cyberbullying on social network sites and state the need for prevention methods. It goes without saying that solving these tasks requires a deep linguistic processing of the text at hand.

However, the automatic analysis of UGC poses a problem for natural language processing (NLP), as the kind of language used in social media highly differs from standard language. Eisenstein [2013] even goes as far as to call it *bad language*. The noisy nature of UGC complicates the task of automatically processing this valuable data source, because the performance of standard NLP tools significantly decreases on social media data [Melero et al. 2012; Eisenstein 2013]. This is because these tools have originally been developed for standard language and, as a consequence, cannot deal with many of the peculiarities encountered in UGC.

Two different computational approaches have been suggested to tackle this problem [Han et al. 2013]: tool adaptation and text normalization. Tool adaptation aims at including UGC data into the training process. As such, tools are made robust with respect to the text type at hand. Work in this field has been performed by, among others, Ritter et al. [2011] for named entity recognition (NER), Gimpel et al. [2011] for part-of-speech (POS) tagging, and Foster et al. [2011] for parsing. A disadvantage of this approach is that it is nontransferable, which means that every single tool would have to be adapted individually. The other approach is text normalization, which envisages to first bring nonstandard language closer to the "norm" (i.e., better conforming to the standard rules of spelling, grammar, and punctuation of a particular language). In this way, standard NLP tools can be applied in a next step.

In this article, we follow the latter approach and introduce a multimodular system for text normalization. We have developed several task-specific modules to solve the different normalization problems that can be encountered in UGC (see the following), and more general modules, that try to tackle all normalization issues in one step. To assess the suitability of the different modules, we evaluated the performance of each module separately. As we assume that text normalization with a multimodular system covering a variety of approaches can outperform simple individual approaches, we combine the output of the different modules in several ways and analyze the overall performance of our system. We furthermore reveal the impact of text normalization on different NLP tools by comparing the output of NER, POS tagging, and lemmatization on UGC before and after normalizing.

The remainder of this article is structured as follows. Section 2 discusses the characteristics of UGC, and Section 3 gives an overview of related work on text normalization. Section 4 presents our modular approach to text normalization. In Section 5, we introduce the datasets that were used for the experiments, analyze the experimental results, and illustrate the usefulness of text normalization on three NLP tasks. In Section 6, we give concluding remarks and directions for future research.

2. UGC: A CHALLENGE FOR NLP

What started as online chatting on a PC and text messaging on cell phones has now evolved into a continuous stream of content that is being produced online using a variety of devices. This evolution has led to the creation of a language that often strongly deviates from standard language, characterized by abbreviations, omissions, spelling mistakes, and grammatically incorrect utterances.

Eisenstein [2013] relates these phenomena to text input affordance, which might vary depending on the input method used (e.g., mobile phone keyboard vs. touch screen keyboard vs. computer keyboard). He also notes that the type of social media application (e.g., online chat, Internet forum, or social network status updates) influences the language used. In addition, social variables, such as age [Rosenthal and McKeown 2011], ethnicity [Eisenstein et al. 2011], and location [Wing and Baldridge 2011; Eisenstein et al. 2010] can influence wording and writing style. VandeKerckhove and Nobels [2010], for example, observe regional variation in UGC and discuss the example of different graphemic realizations of words ending with *-en* in Flemish (*-en*, *-n*, or *-e*) and attribute these to different phonetic realizations depending on the regional dialect. They conclude that the large variety of dialects in Flanders leads to a strong variation in the graphemic realization of words in UGC.

VandeKerckhove and Nobels [2010] relate the language phenomena in UGC to two writing principles: *write as you speak* and *write as fast as possible*. Along the same line, De Clercq et al. [2013] divide the language deviations of UGC into three linguistically motivated categories, namely abbreviations and orthographic and phonetic variants. Very typical of UGC is the large number of *abbreviations*, which can be explained by different factors: space limitations (e.g., in Twitter posts or SMS) and time limitations. As VandeKerckhove and Nobels [2010] point out, the Internet is a medium in which communication is fast. Nevertheless, most abbreviations are easy to understand, as they occur either frequently or are straightforward in a specific context. Social media users most commonly abbreviate *facebook* as *fb*, react to funny content with *lol* (laughing out loud), or talk about their *bf* (boyfriend) or *gf* (girlfriend). Quickly produced text also leads to *typos* and other orthographic issues. Uppercasing is often ignored, or unconventionally used to emphasize something or to convey a specific emotion. Letter transpositions can be observed due to fast typing and a lack of correction. Again, the frequency of these error types varies strongly depending on the social media application used.

The tendency to "write as you speak" can be observed across languages. It seems as if users mimic direct social interaction online by using phonetically motivated realizations of words. In English, this is largely realized by using homophonous graphemic variants of a word like *r* for *are* or *dey* for *they*. In Dutch, words are often transformed or even fused on the basis of the regional pronunciation of the user. This leads to variants such as *zoiso* instead of *sowieso* (in any case) or *kheb* instead of *ik heb* (I have). Very typical of UGC is also that emotions are often orthographically expressed. This can be done in the form of flooding (i.e., the repetition of characters), capitalization, and the productive use of emoticons.

Each of these characteristics contributes to the challenge of linguistically processing this type of text using standard NLP tools. In the next section, we overview some research efforts that have attempted to automatically *normalize* such nonstandard data.

3. TEXT NORMALIZATION: RELATED WORK

Originally, *text normalization* referred to a preprocessing step for text-to-speech synthesis. It dealt with domain-specific problems that were often solved using handcrafted rules. As such, the expected input was limited to a few a priori known patterns [Taylor et al. 1998], and the normalization problems were often restricted to words without context, which could therefore easily be solved at the token level using rules. Sproat et al. [2001] were the first to extend this technique by treating normalization as a language modeling problem and to propose a taxonomy of normalization types. This was done based on four rather distinct text types, newspaper articles, real estate ads, daily digests from a mailing list, and recipes. Their work marked the beginning of more intricate text normalization research.

In more recent years, text normalization has been studied in the framework of UGC. As previously explained, this genre can be characterized by many issues that are not limited to the word level, and very often context is needed to normalize correctly. Moreover, UGC is an umbrella term that covers different text types, such as SMS, tweets, and chat logs. As a consequence, the frequency and density of normalization problems also varies strongly depending on the social media application used. Han and Baldwin [2011] and Baldwin et al. [2013], for example, observe that English Twitter is more dissimilar compared to other forms of social media, such as blogs and comments.

Previous research on UGC normalization has been performed on diverse languages using different techniques. Kobus et al. [2008b] introduced three metaphors to refer to these normalization approaches: the spell checking, translation, and automatic speech recognition (ASR) metaphor.

The *spell checking* metaphor leaves correct words untouched and only performs normalization on the incorrect words. Choudhury et al. [2007] use a hidden Markov model trained on SMS data to find the most probable mapping from an erroneous word to its standard equivalent, thus treating UGC as a noisy version of standard language. Closely related is the use of a dictionary containing both standard and out-of-vocabulary (OOV) entries to this purpose. In this respect, Gouws et al. [2011] suggest a method for the extraction of frequent domain-specific lexical variants, which can serve as a basis for rule-based normalization systems. Such a system is described in Clark and Araki [2011]. They normalize English tweets as a preprocessing step for machine translation from English to Japanese, based on a database of frequent erroneous words in Twitter posts and pattern matching rules. Since the coverage of these dictionaries often poses a problem, Han et al. [2012] introduces a method to automatically compile a large dictionary.

The *translation* metaphor treats social media language as the source language and standard language as the target language. As in general statistical machine translation (SMT), a translation model is trained on parallel data. This model is then combined with a language model to transform a noisy input string into a string that is closer to the standard. The advantage of using SMT is that it directly makes use of contextual information during translation. This approach is described in Aw et al. [2006], who use phrase-based machine translation to normalize English SMS data, and by Kaufmann and Kalita [2010] to normalize English tweets.

Pennell and Liu [2011] were the first to also perform machine translation at the character level, as well as Tiedemann [2012], who uses this technique to translate between closely related languages. Applied to abbreviation normalization, they find that character-based machine translation is more robust to new abbreviations. Li and Liu [2012] likewise describe a character-level machine translation approach to normalizing tweets and extend it to also work with character blocks to improve on the automatic alignments. They also suggest a two-step MT approach that converts tokens into phonemes and phonemes into dictionary words, thereby incorporating the sensibility that people *write as they speak*. De Clercq et al. [2013] show an improvement using character-based over token-based models for text normalization when applying this technique to the entire range of normalization problems and not only to abbreviations. At the same time, Ling et al. [2013] introduce an approach based on paraphrasing by also building two translation models: one on the token- and one on the character level.

Combining these two in a subsequent decoding step proved beneficial for normalizing English tweets.

Text found in social media also shares features with spoken language, and the metaphor of ASR utilizes this similarity. Here, text encountered in social media is treated as an alphabetic approximation of a phonetic string and is brought to a standardized written form using techniques from ASR. Kobus et al. [2008a] propose an ASR-like system for text normalization based on this idea and, like Li and Liu [2012], combine it with SMT-like approaches to normalize French SMS messages. This metaphor has mostly been merged with other techniques to boost performance. Xue et al. [2011] show that combining phonetic with orthographic and contextual information, together with acronym expansion, works well for microtext normalization when combined in a multichannel model. For their automatic dictionary construction, Baldwin et al. [2013] similarly rely on the morphophonemic similarity between standard and ill-formed tokens, which leads them to use both edit distance and phonemic transcription to create word candidates, which are subsequently ranked by a trigram language model.

Some approaches fall beyond the scope of these metaphors, such as the character-level sequence labeling technique described in Li and Liu [2012, 2014], which uses a variety of phonetic, syllabic, and orthographic features to construct likely abbreviations for words in a dictionary. This information is then used during testing as a reverse lookup table to suggest expansions of observed OOV words. A similar approach is suggested in Liu et al. [2012] that learns character transformations on the basis of token-word pairs that were collected in an unsupervised fashion. Liu et al. [2012] also suggest a cognitive-sensitive visual priming technique that favors candidate words that are frequently used and bear an orthographic similarity to the token.

A log-linear model is proposed by Yang and Eisenstein [2013] that scores the conditional probability of a source and target sequence by means of language modeling of the latter and log-likelihood maximization of the former. They report state-of-the-art F-scores that improve on previous research efforts on the same dataset [Han and Baldwin 2011; Liu et al. 2012]. Another log-linear approach, albeit over a series of replacement generators on the character level, is presented in Zhang et al. [2013], who evaluate the technique extrinsically by comparing the performance of a dependency parser on nonnormalized, gold standard, and automatically normalized data.

With such a wide variety of techniques at our disposal, *system combination* seems promising for text normalization. Yvon [2010] describes a normalization device based on finite state transducers using a phonetic representation as an intermediate step. He concludes that the two systems perform better on different aspects of the task and that combining these two modules works best. A similar method is presented in Beaufort et al. [2010], who combine both spell checking and machine translation approaches on French data, which leads to good results. They conclude, however, that including phonetic information into the system is crucial.

Li and Liu [2012] demonstrate state-of-the-art performance using a rule-based combination of a variety of techniques. In later work [Li and Liu 2014], the rule-based approach is abandoned for a discriminative reranking technique that operates on the word level, as well as on the sentence level. Similar to Liu et al. [2012], they also report good results when performing sentence-level Viterbi decoding through the incorporation of a language model. Finally, Wang and Ng [2013] report good results using a novel beam-search decoder that iteratively produces normalized sentence candidates according to several hypothesis producers and consequently evaluates these sentences on the basis of language model scores and a set of count feature functions.

For our approach, we assume that to find a way to automatically normalize highly diverse texts containing a wide variety of normalization issues, a multimodular system is needed. Moreover, we utilize different techniques to interpret the metaphors (e.g., we include three techniques focusing on different spelling errors and implement different MT approaches both on the token and character level). As such, we end up with a multimodular system that should be able to tackle the full normalization task. Other than the approaches described earlier, we do not just combine two of the metaphors but apply all three of them. In addition, in contrast to research efforts such as Yang and Eisenstein [2013] or Li and Liu [2014], we do not consider the nonstandard tokens to be known in advance and consider their identification an integral and nontrivial part of the normalization task. We are the first to apply such an exhaustive approach on diverse genres of Dutch UGC.[1]

4. A MULTIMODULAR APPROACH TOWARD NORMALIZATION

Our multimodular UGC normalization system relies on a partly cascaded, partly parallelized architecture and consists of three main layers:

(1) A preprocessing layer, in which the input text is split into sentences, is tokenized, and flooding (word lengthening) is corrected.
(2) A suggestion layer, in which each module generates suggestions either for tokens (i.e., the token-based modules) or for a sentence as a whole (i.e., the context-based modules). Most of the token-based modules focus on well-understood normalization issues (e.g., abbreviations, compounds, split words). Context-based modules can operate on the word level, as well as the character level, and differ from token-based modules in that they can look beyond word boundaries to make normalization decisions.
(3) A decision layer, in which the best combination of suggestions is chosen from the pool of suggestions.

By combining token-based and context-based modules, we try to combine the best of different methods. The architecture of the multimodular UGC normalization system is depicted in Figure 1.

4.1. Preprocessing Layer

The preprocessing layer consists of two modules. A first module splits the text into sentences and tokens, a task for which we adapted the rule-based tokenizer of Treetagger [Schmid 1994] to cope with UGC-specific phenomena such as e-mail addresses, hyperlinks, and emoticons. Whereas previous work focused on the tokenization of Twitter posts [O'Connor et al. 2010; Bontcheva et al. 2013], we investigate different genres of UGC, requiring us to build a more general tokenizer, covering a wider range of smileys, emoticons, and other tokenization issues.

A second phenomenon dealt with in the preprocessing layer is character flooding (i.e., the repetition of the same character or character sequences), which is often used in UGC to express emotion, as illustrated in the following example. To reduce the number of OOV words in subsequent modules, we limit the number of repetitions to maximally two for all characters except for the vowel "e," where a maximum of three is allowed. The flooding correction module makes use of the Hunspell spell checker[2] to generate the most probable correction and to ensure that correct words are not overcorrected. The module corrects repeated characters and character combinations in the following way:

[1]Since our system works on Dutch text, we will illustrate various parts using Dutch examples with an English translation.
[2]http://hunspell.sourceforge.net/.

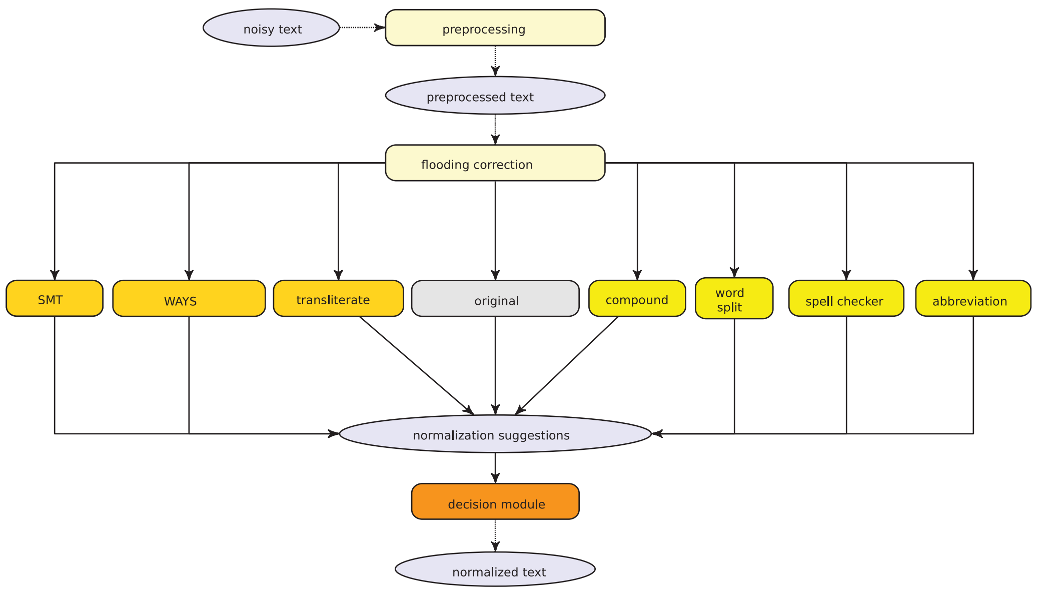

Fig. 1. Multilayer architecture of the UGC normalization system with the preprocessing layer on top, the context-based modules on the left-hand side, the token-based modules on the right-hand side, and the decision module on the bottom.

```
jij hebt egggggt zoooooooooooooooooooo onwijse mooie lipjes ...
jij hebt eggt zoo onwijse mooie lipjes ...
En: you have really such incredibly beautiful little lips ...
```

Note that the flooding-corrected version still contains normalization problems. The flooding *o* is incorrectly substituted by *zoo* (zoo) and not by *zo* (such), as both words do exist in Dutch. The Dutch adverb *echt* (really) should be spelled with *ch* instead of *g*.

4.2. Suggestion Layer

The suggestion layer comprises a variety of modules that have been conceived to account for the different normalization issues encountered in UGC (compare to Section 3). The included modules can be divided into two main groups. The first group contains the *token-based modules*, which are responsible for a specific type of issues. The second group comprises *context-based modules*, which can correct a variety of normalization problems.

The token-based modules are designed to solve specific normalization problems. They are not expected to return normalized sentences but to find a solution to just one problem—more specifically, to tackle abbreviations and various misspellings.

—ABBREVIATION module: Language used in UGC often shares certain abbreviations and uniform ways of reference, such as hash tags in Twitter posts. Therefore, lookup approaches can cover a reasonable number of issues. The ABBREVIATION module relies on a dictionary of about 350 frequent abbreviations appearing in social media texts, such as *lol* (laughing out loud) and *aub* for *alstublieft* (thank you).[3]

[3]This dictionary is available for download at http://www.lt3.ugent.be/amica/chat_abbreviations_dutch.

—SPELL CHECKING modules: These modules account for normalization problems such as typos (e.g., the transposition in *spelne*, which should be *spelen* (play)) or orthographic mistakes such as the omission of diacritics (e.g., as in *cafe*, which should be *café*). We include three modules in the suggestion layer that relate to the spell checking metaphor.

We use a plain SPELL CHECKER,[4] which uses Levenshtein distance to suggest the most probable correction. The SPELL CHECKER can correct minor misspellings in a word like *gzien* to *gezien* (seen) or *zowiezo* to *sowieso* (in any case).

The second module that uses the spell checker is the COMPOUND module. It checks whether words that have been written as two separate words should have been written together. It verifies all token bigrams and can solve cases such as split verbs (e.g., *langs komen* to *langskomen* (drop in)), a phenomenon that frequently occurs in Dutch.

The WORD SPLIT module is the opposite of the COMPOUND module and splits words that have been erroneously written together. In UGC, words are often concatenated to save space. The WORD SPLIT module is based on the compound-splitter module of Moses [Koehn et al. 2007] and has been trained on the Corpus Gesproken Netherlands (CGN) [Oostdijk 2000]. Problems such as *misje* to *mis je* (miss you) or *perse* to *per se* (at any price) can be solved.

A problem related to the spell checking approach is the limited coverage of the word list upon which the spell checker is based. To improve the coverage, we extended the spell checker's dictionary with a word list containing about 2.3 million words compiled from a Dutch Wikipedia corpus. Considering the highly productive nature of UGC, this partly alleviates the problem of OOV words.

The context-based modules have a wider range of responsibilities. They cover a variety of normalization issues and can solve phonologically motivated problems, as well as spelling mistakes and abbreviations. Their main strength is that they use contextual information during normalization:

—SMT modules: Following previous experiments described in De Clercq et al. [2013], the SMT models have been trained on the token and character level using Moses [Koehn et al. 2007]. We include a token-unigram–based module, a character-unigram–based module, a character-bigram–based module, and a combination of a token-based– and a character-unigram–based module that is reported to perform best in De Clercq et al. [2013]. The combination follows a cascaded approach, which means that we first process a sentence with the token-unigram–based module and subsequently forward the output of this module to the character-unigram–based module. The token model can solve problems of rather frequent shortenings, such as *ng* to *nog* (still) or *na* to *naar* (to). Character-based models on the other hand, tend to solve problems such as character transposition, but also problems across tokens like fusions as in *kheb* and *ik heb* (I have). Additionally, they may offer better generalization, as they can learn productive alterations and correct them in words that do not occur in the training data.

—TRANSLITERATE module: This module approaches the normalization task as a transliteration problem to be solved using a discriminative sequence labeler. The normalization problem is defined on the level of the grapheme, not unlike the SMT-character-unigram module. It uses the manually annotated data of the training corpus (see Section 5) as an information source to build a supervised machine learning classifier in which each grapheme in the nonnormalized input sequence is associated with a

[4]Hunspell: http://hunspell.sourceforge.net/.

class in the output sequence. This class can be empty (deletion), the input grapheme itself, or a sequence of graphemes, potentially also containing word boundaries (insertion). This is illustrated in the following example:

```
kebda ni gedaan
ik heb dat niet gedaan
En: I did not do that
```

In preprocessing, we first align these sequences using a dynamic programming script based on Wagner and Fisher [1974] so that they are of equal length:

```
+k +eb+da+ ni++ gedaaan
ik heb dat niet gedaa+n
```

This data is consequently presented as training material to a memory-based learner [Daelemans and van den Bosch 2005] that learns to associate the individual input graphemes with a contextually appropriate output class (input/output with "-" indicating a word boundary):

```
k/ik- e/he b/b- d/d a/at -/- n/n i/iet -/- g/g e/e d/d a/a a/a a/+ n/n
```

The classifier takes different types of context into account: the input characters on the left and the right of the current input character, but also the already transliterated output characters on the left.

—WAYS module: The WAYS module (*write as you speak*) attempts to model idiosyncrasies of UGC in which users write words as they speak, such as *kep* as the contracted representation of the expression *ik heb*, or *ma* instead of *maar*. The module is built as two machine learning classifiers: a grapheme-to-phoneme converter (G2P) and a consecutive phoneme-to-grapheme converter (P2G). We used the phonetic transcriptions of the CGN corpus [Oostdijk 2000] to train our machine learning classifiers. CGN contains 136,000 transcribed sentences using graphemes and phonemes, as illustrated in the following example:

```
die net daar in de zee ligt zeg maar
di nEt tAr In d ze lIxt sEx mar
En: which is lying there in the sea say
```

Similar to the TRANSLITERATE module, preprocessing involves aligning the sequences of graphemes so that the input and output sequence are of equal length:

```
die net daar in de zee ligt zeg maar
di+ nEt tA+r In d@ ze+ lIxt sEx ma+r
```

This is used as training material for the aforementioned memory-based learner, which now converts a sequence of graphemes into phonemes as follows:

```
d/d i/i e/+ -/- n/n e/E t/t -/- d/t a/A a/+ r/r -/- i/I n/n -/- d/d e/@
-/- z/z e/e e/+ -/- l/l i/I g/X t/t -/- z/s e/E g/x -/- m/m a
```

Likewise, a memory-based learner was built that converts a sequence of phonemes back into graphemes.

Finally, as a high percentage of tokens do not contain normalization problems and should therefore not be changed, we also include the ORIGINAL input token in the word candidate list to ensure that we do not lose correct tokens in the input text. Therefore, the original module just adds the original token to the list of suggestions.

Table I. Number of Tokens of the Training, Development, and Test Sets Listed by Subgenre

Subgenre	Train	Dev1	Dev2	Test	All
SMS balanced	6,665	1,137	1,138	2,150	11,090
SMS all	9,689	1,137	1,138	2,150	14,114
SNS balanced	5,706	929	829	1,701	9,165
SNS all	40,363	929	829	1,701	41,875
TWE	6,471	1,008	1,054	2,119	10,652
Total balanced	18,842	3,074	3,021	5,970	30,907
Total all	56,523	3,074	3,021	5,970	68,588

4.3. Decision Layer

Since we have no a priori knowledge about the nature of a normalization problem, each sentence is sent to all modules of the suggestion layer. Each module is allowed to output only one suggestion. It is the task of the decision module to choose the most probable combination of suggestions to build a well-formed sentence. To tackle this combinatorial problem, the decision module makes use of the Moses decoder [Koehn et al. 2007], whose task it is to find the highest scoring normalized sentence corresponding to a given UGC input sentence.

As in general SMT, the decoder makes use of a language model and a phrase table. The language model has been built from a combination of four corpora using KenLM [Heafield 2011] (see Section 5.2 for more details). The phrase table is a lookup table containing words and word sequences along with the normalization suggestions generated by the modules. The decoder can be tuned by allocating weights to the language model and phrase table, setting penalties for phrase reordering and sentence length. We also included features in the phrase table that indicated which module(s) generated a specific normalization suggestion. These features can be tuned as well. We assume that the normalization suggestions of certain modules are more reliable than others and expect their feature weights to be higher after tuning. All tuning was performed on the development data (see Section 5 for a description of the datasets).

5. EVALUATION

5.1. Dataset

The language encountered in UGC differs among different social media applications [Baldwin et al. 2013]. To account for this variety, we include three different types of social media content in our corpus, namely texts from Twitter (TWE) accompanying a Flemish TV show,[5] texts from the social networking site Netlog[6] (SNS), and short messages (SMS) from the Flemish part of SoNaR [Reynaert et al. 2010].

Table I gives an overview of the size of our experimental corpus. To measure the cross-genre performance of our normalization system, we also compiled a genre-balanced dataset, which includes an approximately equal number of tokens from each of the subgenres.

We split our corpus into a training, development, and test set, setting aside about 60% for training, 20% for development, and 20% for testing. We use half of the development set for tuning the individual modules (Dev1) and the other for tuning the overall system (Dev2).

[5]The Voice of Flanders.
[6]http://nl.netlog.com/; the SNS data is a combination of the Netlog datasets of De Clercq et al. [2013] and Kestemont et al. [2012].

Table II. Data Statistics of the Three Genres of UGC: The Number of Messages and the Number of Tokens Before and After Normalization, Together with the Overall Expansion Rate (Left-Hand Side); Normalization Effort Expressed in the Number of Operations on Character Level (Right-Hand Side)

Genre	Msg (#)	Before	After	%	INS (#)	DEL (#)	SUB (#)	TR (#)
SMS	1,000	14,114	14,663	3.89	3,624	605	627	57
SNS	1,505	25,670	25,913	0.94	4,170	5,270	1,372	52
TWE	246	10,652	10,633	−0.18	1,104	394	270	9

Table III. Overview of Corpora Used for Language Modeling

Corpus	Sentences	Words
CGN [Oostdijk 2000]	985,609	6,765,336
SoNaR [Oostdijk 2008]	197,493	3,581,182
Open Subtitles Dutch (OSD)	11,788,416	90,147,315
Training set (TS)	3,721	56,523

All data have been manually normalized and annotated following the guidelines described in De Clercq et al. [2013]. All operations that are necessary to transform the anomalous text into standard language have been added to the data. These operations are as follows:

—*Insertions (INS)*: stappe → stappen (step)
—*Deletions (DEL)*: schatjeeeee → schatje (honey, darling)
—*Substitutions (SUB)*: egt → echt (really)
—*Transpositions (TR)*: ftoo → foto (photo).

This fine-grained annotation facilitates the analysis of normalization issues that are present in the data. Interannotator agreement was calculated between the two fully normalized versions for the SMS genre, which is the genre that includes the highest number of normalization problems. This was done by calculating the accuracy of taking one annotator as the gold standard to score the annotations of the other. This results in an accuracy of 0.967 for both annotators. If we compare this to the nonnormalized accuracy score (i.e., 0.839), we conclude that we have a nearly perfect interannotator agreement.

The normalization effort calculated on a part of our data can be inspected in Table II. The left-hand side of the table shows the number of messages and the number of tokens included in the corpus per subgenre before and after normalization, as well as the expansion rate. On the right-hand side, the number of individual operations that have to be performed to reach the normalized version are shown. The large number of insertions hint at a high rate of abbreviations and phonologically realized words in our data, whereas deletions can be mainly attributed to flooding. Substitutions and transpositions roughly correspond to spelling problems. The slight decrease in tokens observed in Twitter data is due to words that are spread over multiple tokens in the original text that should actually be written as one word.

5.2. Modeling UGC Language

Apart from normalization problems, UGC language differs from standard language in terms of word choice, syntax, and style as well. As the language model is a core element of the SMT modules and the decision module, we want to build a high-quality language model that fits the data that needs to be normalized as well as possible.

We have built language models from three corpora and combinations thereof. The corpora, listed in Table III, were all chosen because of their relative closeness to the target domain (i.e., they all contain a high degree of spoken language features). To

Table IV. Evaluation Results of the Tokenization Module

Genre	SMS		SNS		TWE		ALL	
Metric	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Tokenization	0.98	0.97	0.98	0.98	0.99	0.99	0.98	0.98

maximize this similarity, we also added all training data of our UGC corpus. We used KenLM [Heafield 2011] to evaluate the perplexity of different language models trained on different combinations with respect to our development corpus (Dev1). We varied the order of the models from three grams up to six grams but could not observe any improvements above the order of 5. A five-gram language model, built on the combination of all corpora, obtained the lowest perplexity of 7.4 and was used in the experiments.

5.3. Evaluation Metrics

We evaluated our results using standard evaluation measures for lexical normalization (i.e., word error rate (WER) and precision and recall calculated at the token level). WER is a commonly used metric in speech recognition and machine translation evaluation. It takes into account the number of insertions, deletions, and substitutions that are needed to transform the suggested string into the manually normalized string and is computed as follows:

$$\text{WER} = \frac{\text{Insertions + Substitutions + Deletions}}{\text{\# Tokens in the manual reference}}$$

Besides WER, we also calculate precision and recall, which are widely used metrics in information retrieval. They give information about the degree of over- and under-generation in the suggested string. Precision and recall are computed as follows:

$$\text{Precision} = \frac{\text{\# Correct tokens}}{\text{\# Tokens in the suggestion}}$$

$$\text{Recall} = \frac{\text{\# Correct tokens}}{\text{\# Tokens in the manual reference}}$$

As the token-based evaluation metrics are rather strict and do not reward improvements that are not entirely correct (e.g., the suggestion *antworden* (correct form: *antwoorden* (answer)) for the anomalous form *antwrdn*), we also report character error rate (CER). This is inherently the same formula as for WER, but instead of tokens it looks at characters. As we want to focus on the performance of the normalization modules, we take as input the manually tokenized and automatically flooding-corrected version of the data and each time compare the output with the gold standard dataset.

We evaluated the performance of the tokenizer and sentence splitting component in a separate experiment, in which we compared the automatically and manually tokenized strings. Tokenization in UGC is known to be a difficult task due to the productive use of emoticons, punctuation for emphasis, and the appearance of concatenated words. The results in Table IV show high precision scores, ranging between 0.98 and 0.99 for the three UGC genres. Recall scores are equally high, ranging between 0.97 and 0.99. Given that this preprocessing step comes before a whole range of normalization modules, high precision is important. We assume that some unsolved tokenization problems might find a solution during the normalization process. A notable problem for the tokenizer are cases in which words are strung together, such as *teveel*, which should be tokenized into *te veel* (too much); this is also a problem for which a dedicated word split module was designed in the next layer.

Table V. Performance of the Filtering Methods

Module	SMS		SNS		TWE		ALL	
Metric	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Bigram LM	0.93	0.84	0.96	0.85	0.98	0.92	0.95	0.87
NER	0.65	0.69	0.38	0.39	0.93	0.39	0.76	0.58

5.4. Experiments

Our experiments are structured in two main parts. First, we investigate the performance of *each module separately* by presenting precision, recall, WER, and CER scores. Because some modules were specifically designed to solve a certain type of normalization issue (compare to Section 4.2), we also performed a task-specific evaluation for them (i.e., the compound, abbreviation, and word split module). This was done by evaluating each module with respect to its responsibility range, which was manually annotated. In a next step, we evaluate the overall performance of our *multimodular system* using the same evaluation metrics. In this setup, we apply different settings by weighting the individual modules differently for the decision-making process, by adding more information about the necessity to normalize a token, and by using different training sets.

Since only a portion of the tokens in the input sentence exhibit normalization issues, we experimented with filtering the module suggestions to assess the impact on performance (both of individual modules and of the combined systems). The reasoning behind this is that we want to filter out unlikely suggestions to avoid overcorrection during the normalization process, namely tokens that do not include any normalization problem should not be changed for the worse. We use two filters: a classifier trained on a bigram language model and a named entity recognizer. The classifier is trained on a simple bigram language model compiled from the data described in Section 5.2. We look up each token of the input sentence in the context of the preceding and subsequent token and only retain normalization suggestions for tokens for which we cannot find both bigrams in the language model.

The second filtering mechanism aims at detecting named entities (NEs). NEs typically consist of OOV words that should not be normalized. It is therefore important to recognize them as such to avoid overcorrection. NER in tweets is a far from trivial problem [Liu et al. 2013]: NEs in UGC often have different characteristics than in standard texts (NEs frequently lack capitalization or are introduced with specific characters, such as @ or #), and thus we developed a dedicated NER tool [Schulz 2014]. The NER tool is hybrid in the sense that it uses gazetteer lookup and classification. The gazetteers contain a variety of NEs. Moreover, it includes a simple pattern-matching rule to find words with a capitalized first letter that does not appear at the beginning of a sentence. Given the productive nature of NEs, we also added a dedicated conditional random field classifier trained on the training set of our corpus.

Table V shows the results of these two types of filtering for the three genres. For both techniques, we compared the output of the filtering with the gold standard. The precision obtained with bigram filtering is high, ranging between 0.93 and 0.98, whereas the recall scores range between 0.84 and 0.92. The precision of the NER module is high for Twitter data and reasonable for SMS. For SNS, we observe a large number of tokens mistakenly classified as NEs. This could be attributed to a nonstandard usage of uppercase and lowercase letters.

5.4.1. Module-Specific Evaluation. The evaluation scores for all modules are presented in Table VI. Baseline scores are calculated by comparing the manually tokenized original input with the gold standard normalized text. A first observation is that, in general, the performance varies significantly between the different UGC genres. The highest

Table VI. WER, CER, Precision, and Recall of the General Modules with and without Filtering of Suggestions

Module	SMS			SNS			TWE			ALL			
Metric	Pre	Rec	WER	Pre	Rec	WER	Pre	Rec	WER	Pre	Rec	WER	CER
Baseline	81.6	78.2	21.8	79.7	76.0	24.2	96.3	96.2	4.3	86.6	84.1	16.1	7.7
Without Filtering													
SMT Token	87.5	87.0	12.3	84.1	81.7	18.0	96.2	96.1	4.1	89.8	88.9	10.8	6.0
SMT Unigram	92.6	92.0	7.5	86.5	85.6	14.1	96.6	96.7	3.7	92.4	92.0	**7.7**	4.9
SMT Bigram	92.8	91.5	8.3	86.5	84.7	14.7	95.0	95.1	5.4	92.0	91.0	8.9	6.4
SMT Cascaded	89.9	90.3	9.6	86.0	85.4	14.0	96.2	96.4	3.8	91.1	91.2	8.2	5.1
WAYS	68.9	65.9	34.2	63.8	60.8	40.2	73.7	73.1	28.4	69.2	67.0	33.6	17.4
Transliterate	90.0	88.9	11.1	84.0	81.6	19.1	94.0	93.9	6.8	89.9	88.8	11.6	6.4
Spell checking	81.1	77.7	22.0	75.6	72.1	27.7	94.0	93.8	6.6	84.5	82.1	17.9	7.7
Abbreviation	82.3	79.1	20.9	79.7	76.6	23.7	96.3	96.2	4.1	86.8	84.6	15.5	7.4
Compound	81.9	74.7	29.6	80.2	73.0	31.0	96.9	91.1	15.4	87.0	80.2	18.5	9.3
Word Split	78.2	76.1	24.3	78.6	75.3	25.2	91.7	93.1	7.3	83.2	82.1	18.4	7.9
With Filtering													
SMT Token	86.6	85.0	14.9	83.9	81.2	18.7	96.5	96.3	4.1	89.5	88.1	11.9	6.3
SMT Unigram	89.1	87.6	12.0	85.4	84.4	15.4	96.3	96.4	4.0	90.8	90.0	10.0	5.4
SMT Bigram	88.6	86.9	12.9	85.1	83.0	16.3	95.7	95.7	4.8	90.3	89.1	10.8	5.9
SMT Cascaded	88.2	87.0	12.7	85.4	84.3	15.2	96.3	96.4	4.0	90.6	89.7	10.2	6.4
WAYS	79.8	76.4	23.7	74.7	71.2	29.7	92.2	91.9	8.5	83.1	80.7	19.7	11.7
Transliterate	87.3	85.7	14.4	83.7	81.4	18.9	96.0	96.0	4.4	89.6	88.3	12.0	6.4
Spell checking	82.2	78.8	20.9	78.9	75.3	24.5	95.5	95.3	5.1	86.3	83.9	16.1	9.2
Abbreviation	82.4	79.2	20.9	80.0	76.7	23.5	96.3	96.2	4.1	87.0	84.7	15.8	6.9
Compound	81.7	78.1	22.1	80.0	75.0	25.0	96.4	95.9	4.8	86.7	83.9	16.5	7.8
Word Split	79.2	76.8	23.5	78.6	75.6	24.9	94.5	95.0	5.4	84.8	83.1	17.3	7.8

scores are obtained on the Twitter data, followed by SMS and SNS. This variation can be explained by the difference in density of normalization problems, which is in line with the data statistics that were presented in Table II.

Table VI also illustrates that the SMT and TRANSLITERATE modules reveal particularly high performance. The character-based SMT modules outperform all other modules, with and without filtering. It performs best with a WER reduction of almost 50% over the input text. CER shows a similar tendency. The strength of the character-based SMT modules lies in resolving concatenations such as *keb* to *ik heb*, whereas the token-based module is doing well in resolving frequent abbreviations. The TRANSLITERATE module also shows good normalization capabilities. Even though it does not contain any mechanism to prevent OOV words on the output side, it is able to resolve quite a few issues of compounding and cliticization.

We observe that without filtering, four modules are never able to beat the baseline (WAYS, SPELL CHECKING, COMPOUND, and WORD SPLIT). These are modules that typically overcorrect, and as a result, we observe some moderate improvements after applying filtering.

Although the WAYS module is able to model some aspects of write-as-you-speak effects, its usability on our data is rather limited. Correct words in the input sequence are very often converted erroneously through the processing chain. Furthermore, it is by definition not able to convert abbreviated forms, such as *ff* for *effe*, which are plentiful in our data. Finally, write-as-you-speak effects are very dependent on regional varieties of Dutch. As a results, a single pronunciation model capturing all such regional variants is just not tractable. Specialized region-specific WAYS modules may obtain better results.

Since the COMPOUND, ABBREVIATION, and WORD SPLIT modules have been designed with a specific normalization issue in mind, these modules have a specific range of

Table VII. Number of Problems for Which Each Specialized Module Is Responsible (RES), Has Solved Correctly (COR), and Has Overcorrected (OVER)

Module	SMS			SNS			TWE		
	RES	COR	OVER	RES	COR	OVER	RES	COR	OVER
Without Filtering									
Compound	2	1	96	7	4	60	8	4	136
Abbrev.	94	27	2	46	14	4	18	1	0
Word Split	10	0	57	26	2	15	0	0	80
With Filtering									
Compound	2	0	3	7	1	6	8	1	8
Abbrev.	94	27	0	46	13	0	18	1	0
Word Split	10	0	38	26	2	10	0	0	31

responsibilities (compare to Section 4.2). Table VII gives an impression of the absolute number of problems for which each module is responsible, based on a manual analysis and the actual performance with and without filtering. Besides the fact that the type of problems encountered in the three UGC genres differs considerably, we can also observe that some specialized problems are rather infrequent in our data, such as the small amount of compounding issues. We will now discuss the results of those three modules in closer detail.

Without filtering, the Compound module can only solve about half of the problems of its responsibility range. In addition, we notice that it returns a lot of incorrect suggestions. As mentioned earlier, the number of problematic compounds is small in our test set. In total, only 17 problems with compounds need to be solved, including very uncommon compounds such as *songkeuze* (song choice) and *dragqueen* (drag queen). Introducing filtering leads to a drastic decrease in the number of overcorrections, but it also harms the ability of the module to solve problematic compounds correctly.

We can observe that the Abbreviation module is able to solve around 30% of the issues in the SMS and SNS genre, and 5% in the TWE genre. If we translate these numbers into precision and recall, we can see that it achieves a high precision and a rather low recall (averaged over all genres, we reach a precision of 0.90 and a recall of 0.22). This high precision can be attributed to the lookup approach on which it is based. The low recall points to a coverage issue of the dictionary. A manual analysis revealed, for example, that abbreviations such as *Hvj*, which stands for *hou van jou* (I love you) or *ipv* for *in plaats van* (instead of), remained uncorrected, as they are not yet included in our dictionary, although they are highly frequent in Dutch. Nevertheless, high precision means that the module does not harm the overall performance of our system. Extending the dictionaries represented in this module could lead to a more valuable module contributing well to the normalization success. It is also worthy of mention that the filtering method works well for this module, because no overcorrections remain after filtering.

Finally, the Word Split module has the lowest performance of all. This can be attributed to the modules' inherent capacity to only split a word into two when those two words are actually existing and correct words. As a consequence, it cannot split words containing additional normalization problems. Typical examples are *kzit*, which has to be split into *k* and *zit*. Since *k* has to be transformed into *ik* to build the correct bigram *ik zit* (I sit), the module cannot cope with those problems. The same problem occurs in fused words such as *loveyouuu*, where the second token is anomalous. Again, we see that this module accounts for a large number of overcorrections. Introducing filtering leads to a decrease, although not as outspoken as it was for the previous module.

To conclude, we can state that the actual responsibility range of the COMPOUND and WORD SPLIT modules looks rather limited. However, to evaluate the complementarity of the different modules, we also manually checked the number of unique suggestions that each module (without filtering) proposes on the development dataset. This revealed that even these three worst-performing modules each return unique correct suggestions. We therefore decided to keep all modules in our multimodular system and leave it up to the decision module to select the best suggestion. This is certainly not a trivial task. The two modules that suggest the highest number of unique correct suggestions (WAYS and SPELL CHECKING each offer 16) also generate the highest number of unique incorrect suggestions (respectively, 1,571 and 594).

5.4.2. Multimodular System. Having explored the performance of all modules separately, we also evaluated the interaction of all of our modules in combination. As described in Section 4.3, we include features that provide information on which module(s) generated a normalization suggestion into the decoding process using the Moses decoder. Initially, these features were uniformly weighted (setting 1), but after further tuning on the development set (settings 2 through 5), we bias the decoder to trust certain modules more than others. It is important to note that these are overall module weights that do not take into account the particular normalization issue at hand.

Since we observed that filtering improved the output of some of the modules that tend to overcorrect, we also experiment with two different approaches to include this filtering in our system. In one setting, which we label "hard filtering" (setting 3), we remove suggestions for tokens that according to the filters should not be normalized. In the second approach (setting 4), "soft filtering" is applied by adding this filtering information in the form of two additional features (NER and bigram LM) to the decoding process. The weights for these two additional features are tuned alongside other decoder parameters. In a last evaluation scenario, we built a system using all of our training data using the best settings of the previous experiments (i.e., with tuning and soft filtering) and compare the results of the all-data-in setting to an all-data-in baseline. In sum, we have thus set up five evaluation scenarios:

(1) Genre-balanced system without tuning
(2) Genre-balanced system with tuning
(3) Genre-balanced system with tuning and hard filtering
(4) Genre-balanced system with tuning and soft filtering
(5) All-data-in system with tuning and soft filtering.

For the evaluation of the entire system, we decided to focus on minimizing WER. The first baseline is again calculated on the original, manually tokenized data. As a second baseline, we took the single best-performing module (SMT UNIGRAM). A combined approach should in any case beat the second baseline to show that a combination of modules leads to an improvement over a single module approach.

The results in Table VIII show that the genre-balanced system without tuning (setting 1) improves WER on the entire test set by about 30% over the first baseline and reaches high recall and precision scores. Model tuning (setting 2) improves results noticeably by lowering WER to 7.2%, a decrease of more than 50% over the baseline. This experimental setup beats the best-performing single module, which has a WER of 7.7%.

To gain some insight into the contribution of the different modules to the overall system, we inspected the feature weights of the modules. The weights do not entirely correlate with the ranking of the performance of the modules in terms of WER, but they do reveal the same tendency. The highest weight is allocated to the SMT modules.

Table VIII. Precision, Recall, and WER of the Normalization in Five Different Settings for Each Genre and on the Entire Test Set

System	SMS			SNS			TWE			ALL			
	Prec	Rec	WER	Prec	Rec	WER	Prec	Rec	WER	Prec	Rec	WER	CER
Baseline	81.6	78.2	21.8	79.7	76.0	24.2	96.3	96.2	4.3	86.6	84.1	16.1	7.7
SMT Uni	92.6	92.0	7.5	86.5	85.6	14.1	96.6	96.7	3.7	92.4	92.0	7.7	4.9
1	89.6	87.3	12.8	84.9	81.8	18.4	96.7	96.5	3.9	91.0	89.2	11.0	6.1
2	92.2	92.2	7.5	87.5	87.5	12.4	97.0	97.2	3.2	92.8	92.8	7.2	4.9
3	88.7	87.6	12.1	86.1	85.8	14.0	96.2	96.4	3.9	90.8	90.4	9.7	5.5
4	91.3	92.7	7.0	87.6	87.4	12.6	96.9	97.1	3.2	93.1	92.9	7.1	4.8
SMT Uni all	92.9	92.2	7.4	88.1	87.8	12.0	95.8	96.2	4.1	93.4	92.5	7.4	4.6
5	93.5	93.0	6.7	89.1	88.2	11.5	95.9	96.3	4.0	93.2	92.9	6.9	4.7

Table IX. Oracle Recall Values for the Tuned, Soft-Filtered Genre-Unbalanced System Compared to the Recall Values Achieved by the System in This Setting without Oracle

Genre	SMS	SNS	TWE	ALL
Oracle	96.2	93.7	98.2	96.3
5	93.0	88.2	96.3	92.9

The abbreviation module, which shows reasonable performance, gets the third highest weight. As expected, modules that highly overgenerate receive a low weight.

Interestingly, we cannot show an overall improvement in WER over setting 2 by adding hard filtering (setting 3). It especially impairs results for the SMS test data, which contain the highest number of normalization issues. This means that hard filtering removes too many correct suggestions for anomalous words. The CER values slightly improve by hard filtering, which can be explained by the limitation of overcorrection.

Soft filtering (setting 4) performs better in comparison to hard filtering on all genres. It appears that adding filtering information as decoding features to be tuned achieves slightly better results than when such filtering is absent (setting 2) for SMS and achieves the best scores for all data among the genre-balanced systems. This shows that flagging a token that contains a normalization problem by the bigram language model or an NE adds valuable information to the decoding process.

Adding more training data (setting 5) introduces a slight bias toward SNS data. The performance for TWE slightly suffers, whereas the performance for SNS and SMS noticeably improves, as we substantially extend the training set for SNS. The WER calculated on the entire test set is the lowest among all systems. Comparing to the SMT Unigram module with all training data as a baseline, we achieve significantly better results with our multimodular system. Significance has been calculated using the Monte Carlo algorithm [Efron and Tibshirani 1986] with a resulting 95% confidence interval of 1.19 and 1.22 of difference in mean using 10,000 test suites.

Since we cannot presuppose that the decision module always picks the right suggestion even if it is provided by the modules, we also calculated the upper bound performance for system setting 5, which assumes a perfectly working decision module. These oracle values are shown in Table IX.

A first observation is that our system almost reaches the upper bound of 96.3 with an actual recall of 92.9, which means that the decision module performs really well. Nevertheless, the oracle values also show that not all normalization issues are handled by the modules of the suggestion layer. A manual inspection of the tokens for which no correct suggestions are provided shows that those tokens often contain more than

Table X. Performance of Different NLP Tools Before and After Normalization with the All-Data-In Multimodular System

Metric	WER	Accuracy		F-score
Task	NORM	POS	LEMMA	NER
Gold standard	—	79.8	90.2	20.7
Before normalization	24.6	66.1	71.5	18.5
All-data-in system Tuned module weights, soft filtering	14.9	73.5	80.7	20.4

one normalization issue. An example is *tuurlyk* for *natuurlijk* (of course), which is not only shortened but also has the homophones *ij* and *y* exchanged. Therefore, a spell checking approach or a machine translation approach will probably struggle to solve such issues, as they deviate too strongly from the standard form. The problem of multiple corrections within one word could possibly be solved by a sieve technique in which modules are called consecutively instead of in parallel.

5.5. The Bigger Picture: Extrinsic Evaluation and Portability

Since the main motivation for text normalization is to counter the drop in performance of NLP tools on nonstandard text, we also performed an extrinsic evaluation of our approach, similar to the work described in Zhang et al. [2013]. We evaluated the performance of a POS tagger (POS), a named entity recognizer (NER), and lemmatizer (LEMMA) [van de Kauter et al. 2013] before and after normalization (NORM) on a test set from a subgenre that had not been included in training. Therefore, we additionally annotated 918 posts (7,610 tokens) from the social network ask.fm[7] for these four tasks.

We used the best-working multimodular system including all training data with soft filtering (setting 5) to normalize the posts. As can be seen in Table X, for the normalization of this new subgenre, the system performs much better than the baseline (WER of 24.6).

To assess the impact of normalization on other NLP tasks, we include the results for our gold standard data to set the upper bound that we can reach with perfect normalization and calculate the accuracy and F-score. For all three tasks (POS tagging, lemmatization, and NER), we observe a clear improvement after normalization with an accuracy of 73.5% (after normalization) versus 66.1% (before normalization) for POS tagging and an accuracy of 80.7% (after normalization) versus 71.5% (before normalization) for lemmatization.

The performance improvement for NER, on the other hand, is very modest. The low scores of NER on the gold standard dataset further illustrate that NER is a difficult task in UGC.

6. CONCLUSION

Automatic normalization of UGC is a complex task with many challenges. In this article, we worked with three different types of Dutch UGC, namely SMS, blog and forum posts, and tweets. As can be seen in the expansion rate before and after normalization (Table II) and the baseline WER scores (Table VIII), the normalization effort for the different subgenres varies considerably, with tweets being easier to normalize than SMS and posts on social network sites.

To account for the diversity of normalization problems, we implemented eight different modules that make use of three well-known metaphors for normalization: spell checking, speech recognition, and machine translation. The module-specific evaluation showed that especially the modules belonging to the machine translation metaphor

[7]http://ask.fm/.

(the SMT and TRANSLITERATE modules) perform well. However, as even the low-performing modules generated unique suggestions, we built a multimodular system based on all modules.

The real challenge of the multimodular system is the selection of the best (combination of) candidates from the pool of suggestions, which is the task of the decision module. We stored all normalization suggestions in a phrase table and made use of the Moses decoder to tackle this problem. In contrast to previous research efforts that were limited to language model-based decoding, we use the phrase table infrastructure provided by Moses and add additional features to it that encode information about which module(s) generated a normalization suggestion. These features were tuned on the development set, thus permitting the decoder to learn to trust certain modules more than others. Furthermore, we experimented with two types of filtering (hard and soft filtering) to reduce overcorrection. The oracle values showed that the decision module obtains a high performance despite the large number of suggestions.

Since the main motivation for text normalization is the improvement of the performance of state-of-the-art NLP tools on UGC data, we also performed an extrinsic evaluation on data normalized by our system on yet another type of UGC, namely posts from ask.fm. We demonstrated that automatic normalization indeed improves the performance of POS tagging, lemmatization, and NER. However, the performance level of the standard NLP tools on UGC data (after normalization and even on the gold standard data) is still far below the performance level of those tools on standard language. This might be due to the high degree of syntactic anomalies and English words in Dutch UGC, which our system at this moment is not able to tackle.

A manual inspection of the remaining normalization issues indicates that our system struggles with normalizing words that contain multiple normalization problems. To overcome this problem, a sieve architecture as suggested in Raghunathan et al. [2010] could be attempted, passing on tokens from module to module, resolving issues subsequently.

A current trend in research on text normalization deals with compiling training data (token-word pairs) using unsupervised learning techniques, as illustrated in Liu et al. [2012], Li and Liu [2014], and Yang and Eisenstein [2013]. Other promising research attempts to minimize the need for manually annotated training data by using character embeddings [Chrupala 2014] that allow for maximum utilization of annotated data. Future work will investigate the applicability of these techniques for the normalization task of Dutch UGC, described in this article.

ACKNOWLEDGMENTS

The authors would like to thank Arda Tezcan for his helpful contributions to this research.

REFERENCES

Brandy Lee Aven, David Anthony Burgess, Jonathan Frank Haynes, James Raymond Merino, and Paul Cameron Moore. 2009. Using Product and Social Network Data to Improve Online Advertising. U.S. Patent App. 11/965,509.

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of the COLING/ACL Main Conference Poster Sessions (COLING-ACL'06)*. 33–40.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP'13)*. 356–364.

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

Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark Greenwood, Diana Maynard, and Niraj Aswani. 2013. TwitIE: An open-source information extraction pipeline for microblog text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'13)*. 83–90.

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.

Grzegorz Chrupala. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14), Vol. 2: Short Papers*. 680–686.

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, 2–11.

José Carlos Cortizo, Francisco Carrero, Iván Cantador, José Antonio Troyano, and Paolo Rosso. 2012. Introduction to the special section on search and mining user-generated content. *ACM Transactions on Intelligent Systems and Technology* 3, 4, 65:1–65:3.

Walter Daelemans and Antal van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press, Cambridge, UK.

Orphée De Clercq, Sarah Schulz, Bart Desmet, Els Lefever, and Véronique Hoste. 2013. Normalization of dutch user-generated content. In *Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP'13)*. 179–188.

Bart Desmet and Véronique Hoste. 2014. Recognising suicidal messages in Dutch social media. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*. 830–835.

Bradley Efron and Robert Tibshirani. 1986. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science* 1, 1, 54–75.

Jacob Eisenstein. 2013. What to do about bad language on the Internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 359–369.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP'10)*. 1277–1287.

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, Vol. 1 (HLT'11)*. 1365–1374.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. From news to comment: Resources and benchmarks for parsing the language of Web 2.0. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*. 893–901.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers, Vol. 2 (HLT'11)*. 42–47.

Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the 1st Workshop on Unsupervised Learning in NLP (EMNLP'11)*. 82–90.

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, Vol. 1 (HLT'11)*. 368–378.

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)*. 421–432.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology* 4, 1, 5:1–5:27.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the 6th Workshop on Statistical Machine Translation (WMT'11)*. 187–197.

Max Kaufmann and Jugal Kalita. 2010. Syntactic normalization of Twitter messages. In *Proceedings of the International Conference on Natural Language Processing*.

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 8th International Conference on Language Resources and Evaluation (LREC'12)*. 1569–1572.

Catherine Kobus, Yvon François, and Damnati Géraldine. 2008a. Normalizing SMS: Are two metaphors better than one? In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*. 441–448.

Catherine Kobus, Yvon François, and Damnati Géraldine. 2008b. Transcrire les SMS comme on reconnaît la parole. In *Actes de la Conférence sur le Traitement Automatique des Langues (TALN'08)*. 128–138.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions (ACL'07)*. 177–180.

Chen Li and Yang Liu. 2012. Improving text normalization using character-blocks based models and system combination. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING'12)*. 1587–1602.

Chen Li and Yang Liu. 2014. Improving text normalization via unsupervised model and discriminative reranking. In *Proceedings of the ACL 2014 Student Research Workshop*. 86–93.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 73–84.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Vol. 1: Long Papers*. 1035–1044.

Xiaohua Liu, Furu Wei, Shaodian Zhang, and Ming Zhou. 2013. Named entity recognition for tweets. *ACM Transactions on Intelligent Systems and Technology* 4, 1, 3:1–3:15.

Maite Melero, Marta R. Costa-Juss, Judith Domingo, Montse Marquina, and Mart Quixal. 2012. Holaaa!! Writin like u talk is kewl but kinda hard 4 NLP. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*. 3794–3800.

San Murugesan. 2007. Understanding Web 2.0. *IT Professional* 9, 4, 34–41.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory search and topic summarization for Twitter. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*. AAAI Press, Washington, DC.

Nelleke Oostdijk. 2000. The spoken Dutch corpus. Overview and first evaluation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC'00)*. 887–893.

Nelleke Oostdijk. 2008. SoNaR: STEVIN Nederlandstalig Referentiecorpus. Retrieved March 12, 2016, from http://lands.let.ru.nl/projects/SoNaR/.

Georgios Paltoglou and Mike Thelwall. 2012. Twitter, MySpace, Digg: Unsupervised sentiment analysis in social media. *ACM Transactions on Intelligent Systems and Technology* 3, 4, 66:1–66:19.

Claudia Peersman, Walter Daelemans, and Leona Van Vaerenbergh. 2011. Predicting age and gender in online social networks. In *Proceedings of the 3rd International Workshop on Search and Mining User-Generated Contents (SMUC'11)*. ACM, New York, NY, 37–44.

Deana L. Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of SMS abbreviations. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*. 974–982.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP'10)*. 492–501.

Martin Reynaert, Nelleke Oostdijk, Orphe De Clercq, Henk van den Heuvel, and Franciska de Jong. 2010. Balancing SoNaR: IPR versus processing issues in a 500-million-word written Dutch reference corpus. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*. 2693–2698.

Cate Riegner. 2007. Word of mouth on the Web: The impact of Web 2.0 on consumer purchase decisions. *Journal of Advertising Research.* 47, 4, 436–437.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*. 1524–1534.

Sara Rosenthal and Kathleen McKeown. 2011. Age prediction in blogs: A study of style, content, and online behavior in pre- and post-social media generations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (HLT'11)*. 763–772.

Kathleen Van Royen, Karolien Poels, Walter Daelemans, and Heidi Vandebosch. 2015. Automatic monitoring of cyberbullying on social networking sites: From technological feasibility to desirability. *Telematics and Informatics* 32, 1, 89–97.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*. 44–49.

Sarah Schulz. 2014. Named entity recognition for user-generated content. In *Proceedings of the ESSLLI 2014 Student Session*. 207–2018.

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.

Paul Taylor, Alan Black, and Richard Caley. 1998. The architecture of the festival speech synthesis system. In *Proceedings of the 3rd ESCA/COCOSDA Workshop on Speech Synthesis*. 147–151.

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 (EACL'12)*. 141–151.

Marjan van de Kauter, Geert Coorman, Els Lefever, Bart Desmet, Lieve Macken, and Véronique Hoste. 2013. LeTs preprocess: The multilingual LT3 linguistic preprocessing toolkit. *Computational Linguistics in the Netherlands Journal* 3, 103–120.

José van Dijk. 2009. Users like you? Theorizing agency in user generated content. *Media, Culture and Society* 31, 1, 41–58.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *Proceedings of Recent Advances in Natural Language Processing (RANLP'15)*.

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

Robert A. Wagner and Michael J. Fisher. 1974. The string-to-string correction problem. *Journal of the ACM* 21, 1, 168–173.

Pidong Wang and Hwee Tou Ng. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 471–481.

Benjamin P. Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 1 (HLT'11)*. 955–964.

Zhenzhen Xue, Dawei Yin, and Brian D. Davison. 2011. Normalizing microtext. In *Proceedings of the AAAI-11 Workshop on Analyzing Microtext*, Vol. WS-11-05. 74–79.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 61–72.

François Yvon. 2010. Rewriting the orthography of SMS messages. *Natural Language Engineering* 16, 2, 133–159.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive parser-centric text normalization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Vol. 1: Long Papers*. 1159–1168.

Received November 2014; revised August 2015; accepted November 2015