A hybrid rule/model-based finite-state framework for normalizing SMS messages

Richard Beaufort[1] **Sophie Roekhaut**[2] **Louise-Amélie Cougnon**[1] **Cédrick Fairon**[1]
(1) CENTAL, Université catholique de Louvain – 1348 Louvain-la-Neuve, Belgium
{richard.beaufort,louise-amelie.cougnon,cedrick.fairon}@uclouvain.be
(2) TCTS Lab, Université de Mons – 7000 Mons, Belgium
sophie.roekhaut@umons.ac.be

Abstract

In recent years, research in natural language processing has increasingly focused on normalizing SMS messages. Different well-defined approaches have been proposed, but the problem remains far from being solved: best systems achieve a 11% Word Error Rate. This paper presents a method that shares similarities with both spell checking and machine translation approaches. The normalization part of the system is entirely based on models trained from a corpus. Evaluated in French by 10-fold-cross validation, the system achieves a 9.3% Word Error Rate and a 0.83 BLEU score.

1 Introduction

Introduced a few years ago, Short Message Service (SMS) offers the possibility of exchanging written messages between mobile phones. SMS has quickly been adopted by users. These messages often greatly deviate from traditional spelling conventions. As shown by specialists (Thurlow and Brown, 2003; Fairon et al., 2006; Bieswanger, 2007), this variability is due to the simultaneous use of numerous coding strategies, like phonetic plays (*2m1* read 'demain', "tomorrow"), phonetic transcriptions (*kom* instead of 'comme', "like"), consonant skeletons (*tjrs* for 'toujours', "always"), misapplied, missing or incorrect separators (*j esper* for 'j'espère', "I hope"; *j'croibi1k*, instead of 'je crois bien que', "I am pretty sure that"), etc. These deviations are due to three main factors: the small number of characters allowed per text message by the service (140 bytes), the constraints of the small phones' keypads and, last but not least, the fact that people mostly communicate between friends and relatives in an informal register.

Whatever their causes, these deviations considerably hamper any standard natural language processing (NLP) system, which stumbles against so many Out-Of-Vocabulary words. For this reason, as noted by Sproat et al. (2001), an SMS normalization must be performed *before* a more conventional NLP process can be applied. As defined by Yvon (2008), "SMS normalization consists in rewriting an SMS text using a more conventional spelling, in order to make it more readable for a human or for a machine."

The SMS normalization we present here was developed in the general framework of an SMS-to-speech synthesis system[1]. This paper, however, only focuses on the normalization process.

Evaluated in French, our method shares similarities with both spell checking and machine translation. The machine translation-like module of the system performs the true normalization task. It is entirely based on models learned from an SMS corpus and its transcription, aligned at the character-level in order to get parallel corpora. Two spell checking-like modules surround the normalization module. The first one detects unambiguous tokens, like URLs or phone numbers, to keep them out of the normalization. The second one, applied on the normalized parts only, identifies non-alphabetic sequences, like punctuations, and labels them with the corresponding token. This greatly helps the system's print module to follow the basic rules of typography.

This paper is organized as follows. Section 2 proposes an overview of the state of the art. Section 3 presents the general architecture of our system, while Section 4 focuses on how we learn and combine our normalization models. Section 5 evaluates the system and compares it to

[1]The *Vocalise* project.
See cental.fltr.ucl.ac.be/team/projects/vocalise/.

previous works. Section 6 draws conclusions and considers some future possible improvements of the method.

2 Related work

As highlighted by Kobus et al. (2008b), SMS normalization, up to now, has been handled through three well-known NLP metaphors: spell checking, machine translation and automatic speech recognition. In this section, we only present the pros and cons of these approaches. Their results are given in Section 5, focused on our evaluation.

The spell checking metaphor (Guimier de Neef et al., 2007; Choudhury et al., 2007; Cook and Stevenson, 2009) performs the normalization task on a word-per-word basis. On the assumption that most words should be correct for the purpose of communication, its principle is to keep In-Vocabulary words out of the correction process. Guimier de Neef et al. (2007) proposed a rule-based system that uses only a few linguistic resources dedicated to SMS, like specific lexicons of abbreviations. Choudhury et al. (2007) and Cook and Stevenson (2009) preferred to implement the noisy channel metaphor (Shannon, 1948), which assumes a communication process in which a sender emits the intended message W through an imperfect (noisy) communication channel, such that the sequence O observed by the recipient is a noisy version of the original message. On this basis, the idea is to recover the intended message W hidden behind the sequences of observations O, by maximizing:

$$\begin{aligned} W_{max} &= \arg\max P(W|O) \qquad (1) \\ &= \arg\max \frac{P(O|W)\,P(W)}{P(O)} \end{aligned}$$

where $P(O)$ is ignored because constant, $P(O|W)$ models the channel's noise, and $P(W)$ models the language of the source. Choudhury et al. (2007) implemented the noisy channel through a Hidden-Markov Model (HMM) able to handle both graphemic variants and phonetic plays as proposed by (Toutanova and Moore, 2002), while Cook and Stevenson (2009) enhanced the model by adapting the channel's noise $P(O|W, \mathit{wf})$ according to a list of predefined observed word formations $\{\mathit{wf}\}$: stylistic variation, word clipping, phonetic abbreviations, etc. Whatever the system, the main limitation of the spell checking approach is the excessive confidence it places in word boundaries.

The machine translation metaphor, which is historically the first proposed (Bangalore et al., 2002; Aw et al., 2006), considers the process of normalizing SMS as a translation task from a source language (the SMS) to a target language (its standard written form). This standpoint is based on the observation that, on the one side, SMS messages greatly differ from their standard written forms, and that, on the other side, most of the errors cross word boundaries and require a wide context to be handled. On this basis, Aw et al. (2006) proposed a statistical machine translation model working at the phrase-level, by splitting sentences into their k most probable phrases. While this approach achieves really good results, Kobus et al. (2008b) make the assertion that a phrase-based translation can hardly capture the lexical creativity observed in SMS messages. Moreover, the translation framework, which can handle many-to-many correspondences between sources and targets, exceeds the needs of SMS normalization, where the normalization task is almost deterministic.

Based on this analysis, Kobus et al. (2008b) proposed to handle SMS normalization through an automatic speech recognition (ASR) metaphor. The starting point of this approach is the observation that SMS messages present a lot of phonetic plays that sometimes make the SMS word (*sré*, *mwa*) closer to its phonetic representation ([sʁe], [mwa]) than to its standard written form (*serai*, “will be”, *moi*, “me”). Typically, an ASR system tries to discover the best word sequence within a lattice of weighted phonetic sequences. Applied to the SMS normalization task, the ASR metaphor consists in first converting the SMS message into a phone lattice, before turning it into a word-based lattice using a phoneme-to-grapheme dictionary. A language model is then applied on the word lattice, and the most probable word sequence is finally chosen by applying a best-path algorithm on the lattice. One of the advantages of the grapheme-to-phoneme conversion is its intrinsic ability to handle word boundaries. However, this step also presents an important drawback, raised by the authors themselves: it prevents next normalization steps from knowing what graphemes were in the initial sequence.

Our approach, which is detailed in Sections 3 and 4, shares similarities with both the spell checking approach and the machine translation principles, trying to combine the advantages of these methods, while leaving aside their drawbacks: like in spell checking systems, we detect unambiguous units of text as soon as possible and try to rely on word boundaries when they seem *reliable enough*; but like in the machine translation task, our method intrinsically handles word boundaries in the normalization process if needed.

3 Overview of the system

3.1 Tools in use

In our system, all lexicons, language models and sets of rules are compiled into finite-state machines (FSMs) and combined with the input text by *composition* ($\circ$). The reader who is not familiar with FSMs and their fundamental theoretical properties, like composition, is urged to consult the state-of-the-art literature (Roche and Schabes, 1997; Mohri and Riley, 1997; Mohri et al., 2000; Mohri et al., 2001).

We used our own finite-state tools: a finite-state machine library and its associated compiler (Beaufort, 2008). In conformance with the format of the library, the compiler builds finite-state machines from weighted rewrite rules, weighted regular expressions and n-gram models.

3.2 Aims

We formulated four constraints before fixing the system's architecture. First, special tokens, like URLs, phones or currencies, should be identified as soon as possible, to keep them out of the normalization process.

Second, word boundaries should be taken into account, as far as they seem *reliable enough*. The idea, here, is to base the decision on a learning able to catch frequent SMS sequences to include in a dedicated In-Vocabulary (IV) lexicon.

Third, any other SMS sequence should be considered as Out-Of-Vocabulary (OOV), on which in-depth rewritings may be applied.

Fourth, the basic rules of typography and typesettings should be applied on the normalized version of the SMS message.

3.3 Architecture

The architecture depicted in Figure 1 directly relies on these considerations. In short, an SMS message first goes through three SMS modules, which normalize its noisy parts. Then, two standard NLP modules produce a morphosyntactic analysis of the normalized text. A last module, finally, takes advantage of this linguistic analysis either to print a text that follows the basic rules of typography, or to synthesize the corresponding speech signal.

Because this paper focuses on the normalization task, the rest of this section only presents the SMS modules and the "smart print" output. The morphosyntactic analysis, made of state-of-the-art algorithms, is described in (Beaufort, 2008), and the text-to-speech synthesis system we use is presented in (Colotte and Beaufort, 2005).

3.3.1 SMS modules

SMS preprocessing. This module relies on a set of manually-tuned rewrite rules. It identifies paragraphs and sentences, but also some

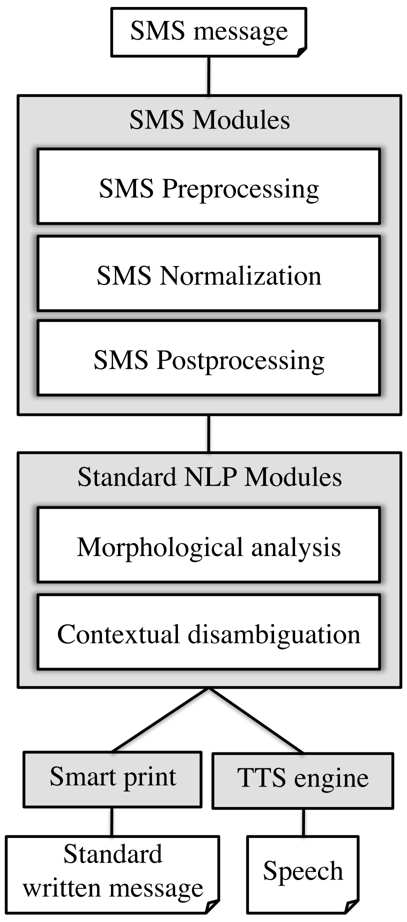

Figure 1: Architecture of the system

unambiguous tokens: URLs, phone numbers, dates, times, currencies, units of measurement and, last but not least in the context of SMS, smileys[2]. These tokens are kept out of the normalization process, while any other sequence of characters is considered – and labelled – as noisy.

SMS normalization. This module only uses models learned from a training corpus (cf. Section 4). It involves three steps. First, an SMS-dedicated lexicon look-up, which differentiates between known and unknown parts of a noisy token. Second, a rewrite process, which creates a lattice of weighted solutions. The rewrite model differs depending on whether the part to rewrite is known or not. Third, a combination of the lattice of solutions with a language model, and the choice of the best sequence of lexical units. At this stage, the normalization as such is completed.

SMS postprocessing. Like the preprocessor, the postprocessor relies on a set of manually-tuned rewrite rules. The module is only applied on the normalized version of the noisy tokens, with the intention to identify any non-alphabetic sequence and to isolate it in a distinct token. At this stage, for instance, a point becomes a 'strong punctuation'. Apart from the list of tokens already managed by the preprocessor, the postprocessor handles as well numeric and alphanumeric strings, fields of data (like bank account numbers), punctuations and symbols.

3.3.2 Smart print

The smart print module, based on manually-tuned rules, checks either the kind of token (chosen by the SMS pre-/post-processing modules) or the grammatical category (chosen by the morphosyntactic analysis) to make the right typography choices, such as the insertion of a space after certain tokens (URLs, phone numbers), the insertion of two spaces after a strong punctuation (point, question mark, exclamation mark), the insertion of two carriage returns at the end of a paragraph, or the upper case of the initial letter at the beginning of the sentence.

[2]Our list contains about 680 smileys.

4 The normalization models

4.1 Overview of the normalization algorithm

Our approach is an approximation of the noisy channel metaphor (cf. Section 2). It differs from this general framework, because we adapt the model of the channel's noise depending on whether the noisy token (our sequence of observations) is In-Vocabulary or Out-Of-Vocabulary:

$$P(O|W) = \begin{cases} P_{IV}(O|W) & \text{if } O \in IV \\ \\ P_{OOV}(O|W) & \text{else} \end{cases} \qquad (2)$$

Indeed, our algorithm is based on the assumption that applying different normalization models to IV and OOV words should both improve the results and reduce the processing time.

For this purpose, the first step of the algorithm consists in composing a noisy token T with an FST Sp whose task is to differentiate between sequences of IV words and sequences of OOV words, by labelling them with a special IV or OOV marker. The token is then split in n segments sg_i according to these markers:

$$\{sg\} = \text{Split}(T \circ Sp) \qquad (3)$$

In a second step, each segment is composed with a rewrite model according to its kind: the IV rewrite model R_{IV} for sequences of IV words, and the OOV rewrite model R_{OOV} for sequences of OOV words:

$$sg'_i = \begin{cases} sg_i \circ R_{IV} & \text{if } sg_i \in IV \\ \\ sg_i \circ R_{OOV} & \text{else} \end{cases} \qquad (4)$$

All rewritten segments are then concatenated together in order to get back the complete token:

$$T = \odot_{i=1}^{n}(sg'_i) \qquad (5)$$

where $\odot$ is the concatenation operator.

The third and last normalization step is applied on a complete sentence S. All tokens T_j of S are concatenated together and composed with the lexical language model LM. The result of this composition is a word lattice, of which we take the most probable word sequence S' by applying a best-path algorithm:

$$S' = \text{BestPath}(\ (\odot_{j=1}^{m} T_j) \circ LM\ ) \qquad (6)$$

where m is the number of tokens of S. In S', each noisy token T_j of S is mapped onto its most probable normalization.

4.2 The corpus alignment

Our normalization models were trained on a French SMS corpus of 30,000 messages, gathered in Belgium, semi-automatically anonymized and manually normalized by the Catholic University of Louvain (Fairon and Paumier, 2006). Together, the SMS corpus and its transcription constitute *parallel corpora* aligned at the message-level.

However, in order to learn pieces of knowledge from these corpora, we needed a string alignment at the *character-level*.

One way of implementing this string alignment is to compute the edit-distance of two strings, which measures the minimum number of operations (substitutions, insertions, deletions) required to transform one string into the other (Levenshtein, 1966). Using this algorithm, in which each operation gets a cost of 1, two strings may be aligned in different ways with the same global cost. This is the case, for instance, for the SMS form *kozer* ([koze]) and its standard transcription *causé* ("talked"), as illustrated by Figure 2. However, from a linguistic standpoint, alignment (1) is preferable, because corresponding *graphemes* are aligned on their *first character*.

In order to automatically choose this preferred alignment, we had to distinguish the three edit-operations, according to the characters to be aligned. For that purpose, probabilities were required. Computing probabilities for each operation according to the characters to be aligned was performed through an iterative algorithm described in (Cougnon and Beaufort, 2009). In short, this algorithm gradually learns the best way of aligning strings. On our parallel corpora, it converged after 7 iterations and provided us with a result from which the learning could start.

(1)	ko_ser causé_	(2)	k_oser causé_
(3)	ko_ser caus_é	(4)	k_oser caus_é

Figure 2: Different equidistant alignments, using a standard edit-cost of 1. Underscores ('_') mean *insertion* in the upper string, and *deletion* in the lower string.

4.3 The split model Sp

In natural language processing, a word is commonly defined as "a sequence of alphabetic characters between separators", and an IV word is simply a word that belongs to the lexicon in use.

In SMS messages however, separators are surely indicative, but not reliable. For this reason, our definition of the *word* is far from the previous one, and originates from the string alignment. After examining our parallel corpora aligned at the character-level, we decided to consider as a word "the longest sequence of characters parsed without meeting the same separator on both sides of the alignment". For instance, the following alignment

```
J esper_ k___tu va_
J'espère que tu vas
```

(*I hope that you will*)

is split as follows according to our definition:

J esper_	k___tu	va_
J'espère	que tu	vas

since the separator in "`J esper`" is different from its transcription, and "`ktu`" does not contain any separator. Thus, this SMS sequence corresponds to 3 SMS words: [*J esper*], [*ktu*] and [*va*].

A first parsing of our parallel corpora provided us with a list of SMS sequences corresponding to our IV lexicon. The FST Sp is built on this basis:

$$Sp = (\ S^{*}\ (I|O)\ \ (\ S^{+}(I|O)\ )^{*}\ S^{*}\ ) \circ G \quad (7)$$

where:

- I is an FST corresponding to the lexicon, in which IV words are mapped onto the IV marker.
- O is the complement of I[3]. In this OOV lexicon, OOV sequences are mapped onto the OOV marker.
- S is an FST corresponding to the list of separators (any non-alphabetic and non-numeric character), mapped onto a SEP marker.

[3]Actually, the true complement of I accepts sequences with separators, while these sequences were removed from O.

- G is an FST able to detect consecutive sequences of IV (resp. OOV) words, and to group them under a unique IV (resp. OOV) marker. By gathering sequences of IVs and OOVs, SEP markers disappear from Sp.

Figure 3 illustrates the composition of Sp with the SMS sequence *J esper kcv b1* (*J'espère que ça va bien*, "I hope you are well"). For the example, we make the assumption that *kcv* was never seen during the training.

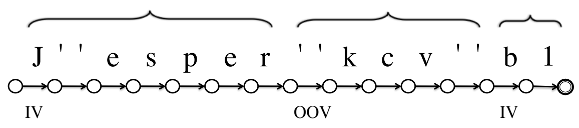

Figure 3: Application of the split model Sp. The OOV sequence starts and ends with separators.

4.4 The IV rewrite model R_{IV}

This model is built during a second parsing of our parallel corpora. In short, the parsing simply gathers all possible normalizations for each SMS sequence put, by the first parsing, in the IV lexicon. Contrary to the first parsing, this second one processes the corpus *without taking separators into account*, in order to make sure that *all* possible normalizations are collected.

Each normalization $\bar{w}$ for a given SMS sequence w is weighted as follows:

$$p(\bar{w}|w) = \frac{\text{Occ}(\bar{w}, w)}{\text{Occ}(w)} \quad (8)$$

where $\text{Occ}(x)$ is the number of occurrences of x in the corpus. The FST R_{IV} is then built as follows:

$$R_{IV} = {S_{IV}}^{*}\ IV_R\ (\ {S_{IV}}^{+}\ IV_R\ )^{*}\ {S_{IV}}^{*} \quad (9)$$

where:

- IV_R is a weighted *lexicon* compiled into an FST, in which each IV sequence is mapped onto the list of its possible normalizations.
- S_{IV} is a weighted *lexicon* of separators, in which each separator is mapped onto the list of its possible normalizations. The deletion is often one of the possible normalization of a separator. Otherwise, the deletion is added and is weighted by the following smoothed probability:

$$p(\texttt{DEL}|w) = \frac{0.1}{\text{Occ}(w) + 0.1} \quad (10)$$

4.5 The OOV rewrite model R_{OOV}

In contrast to the other models, this one is not a regular expression made of weighted lexicons. It corresponds to a set of *weighted rewrite rules* (Chomsky and Halle, 1968; Johnson, 1972; Mohri and Sproat, 1996) learned from the alignment. Developed in the framework of generative phonology, rules take the form

$$\phi \rightarrow \psi : \lambda \_ \rho \ / \ w \quad (11)$$

which means that the replacement $\phi \rightarrow \psi$ is only performed when ϕ is surrounded by λ on the left and ρ on the right, and gets the weight w. However, in our case, rules take the simpler form

$$\phi \rightarrow \psi \ / \ w \quad (12)$$

which means that the replacement $\phi \rightarrow \psi$ is always performed, whatever the context.

Inputs of our rules (ϕ) are sequences of 1 to 5 characters taken from the SMS side of the alignment, while outputs (ψ) are their corresponding normalizations. Our rules are sorted in the reverse order of the length of their inputs: rules with longer inputs come first in the list.

Long-to-short rule ordering reduces the number of proposed normalizations for a given SMS sequence for two reasons:

1. the firing of a rule with a longer input blocks the firing of any shorter sub-rule. This is due to a constraint expressed on lists of rewrite rules: a given rule may be applied only if no more specific and relevant rule has been met higher in the list;
2. a rule with a longer input usually has fewer alternative normalizations than a rule with a shorter input does, because the longer SMS sequence likely occurred paired with fewer alternative normalizations in the training corpus than did the shorter SMS sequence.

Among the wide set of possible sequences of 2 to 5 characters gathered from the corpus, we only kept in our list of rules the sequences that allowed at least one normalization *solely made of IV words*. It is important to notice that here, we refer to the standard notion of *IV word*: while gathering the candidate sequences from the corpus, we systematically checked each word of the normalizations against a lexicon of French standard written forms. The lexicon we used contains about 430,000 inflected forms and is derived from Morlex[4], a French lexical database.

Figure 4 illustrates these principles by focusing on 3 input sequences: *aussi*, *au* and *a*. As shown by the Figure, all rules of a set dedicated to the same input sequence (for instance, *aussi*) are optional (?→), except the last one, which is obligatory (→). In our finite-state compiler, this convention allows the application of all concurrent normalizations on the same input sequence, as depicted in Figure 5.

In our real list of OOV rules, the input sequence a corresponds to 231 normalizations, while *au* accepts 43 normalizations and *aussi*, only 3. This highlights the interest, in terms of efficiency, of the long-to-short rule ordering.

4.6 The language model

Our language model is an n-gram of lexical forms, smoothed by linear interpolation (Chen and Goodman, 1998), estimated on the normalized part of our training corpus and compiled into a weighted FST LM_w.

At this point, this FST cannot be combined with our other models, because it works on *lexical units* and not on *characters*. This problem is solved by composing LM_w with another FST L, which represents a lexicon mapping each input word, considered as a string of characters, onto the same output words, but considered here as a lexical unit. Lexical units are then permanently removed from the language model by keeping only the first projection (the input side) of the composition:

$$LM = \text{FirstProjection}(\ L \circ LM_w\ ) \qquad (13)$$

In this model, special characters, like punctuations or symbols, are represented by their categories (light, medium and strong punctuations, question mark, symbol, etc.), while special tokens, like URLs or phone numbers, are handled as token values (URL, phone, etc.) instead of as sequences of characters. This reduces the complexity of the model.

As we explained earlier, tokens of a same sentence S are concatenated together at the end of the second normalization step. During this concatenation process, sequences corresponding to special tokens are automatically replaced by their token values. Special characters, however,

```
"aussi" ?-> "au si" / 8.4113 (*)
"aussi" ?-> "ou si" / 6.6743 (*)
"aussi" ->  "aussi" / 0.0189 (*)
...
...
"au" ?-> "ow" / 14.1787
...
"au" ?-> "ôt" / 12.5938
"au" ?-> "du" / 12.1787 (*)
"au" ?-> "o" / 11.8568
...
"au" ?-> "on" / 10.8568 (*)
...
"au" ?-> "aud" / 9.9308
"au" ?-> "aux" / 6.1731 (*)
"au" ->  "au" / 0.0611 (*)
...
...
"a" ?-> "a d" / 17.8624
"a" ?-> "ation" / 17.8624
"a" ?-> "âts" / 17.8624
...
"a" ?-> "ablement" / 16.8624
"a" ?-> "anisation" / 16.8624
...
"a" ?-> "u" / 15.5404
"a" ?-> "y a" / 15.5404
...
"a" ?-> "abilité" / 13.4029
"a" ?-> "à-" / 12.1899
"a" ?-> "ar" / 11.5225
"a" ?-> \DEL / 9.1175
"a" ?-> "ça" / 6.2019
"a" ?-> "à" / 3.5013
"a" ->  "a" / 0.3012
```

Figure 4: Samples from the list of OOV rules. Rules' weights are negative logarithms of probabilities: smaller weights are thus better. Asterisks indicate normalizations solely made of French IV words.

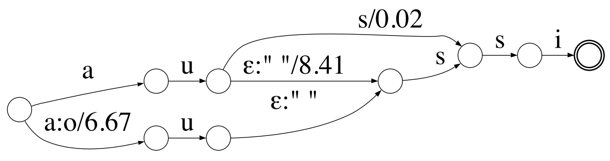

Figure 5: Application of the OOV rules on the input sequence *aussi*. All normalizations corresponding to this sequence were allowed, while rules corresponding to shorter input sequences were ignored.

[4]See http://bach.arts.kuleuven.be/pmertens/.

are still present in S. For this reason, S is first composed with an FST *Reduce*, which maps each special character onto its corresponding category:

$$S \circ Reduce \circ LM \qquad (14)$$

5 Evaluation

The performance and the efficiency of our system were evaluated on a MacBook Pro with a 2.4 GHz Intel Core 2 Duo CPU, 4 GB 667 MHz DDR2 SDRAM, running Mac OS X version 10.5.8.

The evaluation was performed on the corpus of 30,000 French SMS presented in Section 4.2, by ten-fold cross-validation (Kohavi, 1995). The principle of this method of evaluation is to split the initial corpus into 10 subsets of equal size. The system is then trained 10 times, each time leaving out one of the subsets from the training corpus, but using only this omitted subset as test corpus.

The language model of the evaluation is a 3-gram. We did not try a 4-gram. This choice was motivated by the experiments of Kobus et al. (2008a), who showed on a French corpus comparable to ours that, if using a larger language model is always rewarded, the improvement quickly decreases with every higher level and is already quite small between 2-gram and 3-gram.

Table 1 presents the results in terms of efficiency. The system seems efficient, while we cannot compare it with other methods, which did not provide us with this information.

Table 2, part 1, presents the performance of our approach (*Hybrid*) and compares it to a trivial copy-paste (*Copy*). The system was evaluated in terms of BLEU score (Papineni et al., 2001), Word Error Rate (WER) and Sentence Error Rate (SER). Concerning WER, the table presents the distribution between substitutions (Sub), deletions (Del) and insertions (Ins). The copy-paste results just inform about the real deviation of our corpus from the traditional spelling conventions, and highlight the fact that our system is still at pains to significantly reduce the SER, while results in terms of WER and BLEU score are quite encouraging.

Table 2, part 2, provides the results of the state-of-the-art approaches. The only results truly comparable to ours are those of Guimier de Neef et al. (2007), who evaluated their approach on the same corpus as ours[5]; clearly, our method outperforms theirs. Our results also seem a bit better than those of Kobus et al. (2008a), although the comparison with this system, also evaluated in French, is less easy: they combined the French corpus we used with another one and performed a single validation, using a bigger training corpus (36.704 messages) for a test corpus quite similar to one of our subsets (2.998 SMS). Other systems were evaluated in English, and results are more difficult to compare; at least, our results seem in line with them.

[5]They performed an evaluation without ten-fold cross-validation, because their rule-based system did not need any training.

	mean	**dev.**
bps	1836.57	159.63
ms/SMS (140b)	76.23	22.34

Table 1: Efficiency of the system.

The analysis of the normalizations produced by our system pointed out that, most often, errors are contextual and concern the gender (*quel(le)*, “what”), the number (*bisou(s)*, “kiss”), the person (*[tu t’]inquiète(s)*, “you are worried”) or the tense (*arrivé/arriver*, “arrived”/“to arrive”). That contextual errors are frequent is not surprising. In French, as mentioned by Kobus et al. (2008b), n-gram models are unable to catch this information, as it is generally out of their scope.

On the other hand, this analysis confirmed our initial assumptions. First, special tokens (URLs, phones, etc.) are not modified. Second, agglutinated words are generally split (*Pensa ms* $\rightarrow$ *Pense à mes*, “think to my”), while misapplied separators tend to be deleted (*G t* $\rightarrow$ *J’étais*, “I was”). Of course, we also found some errors at word boundaries (*[il] l’arrange* $\rightarrow$ *[il] la range*, “[he] arranges” $\rightarrow$ “[he] pits in order”), but they were fairly rare.

6 Conclusion and perspectives

In this paper, we presented an SMS normalization framework based on finite-state machines and developed in the context of an SMS-to-speech synthesis system. With the intention to avoid wrong modifications of special tokens and to handle word boundaries as easily as possible, we designed a method that shares similarities with both spell checking and machine translation. Our

	1. Our approach				2. State of the art					
	Ten-fold cross-validation, French				French			English		
	Copy		*Hybrid*		*Guimier*	*Kobus 2008*		*Aw*	*Choud.*	Cook
	$\bar{x}$	σ	$\bar{x}$	σ	*2007*	1	2*	*2006*	*2006***	*2009***
Sub.	25.90	1.65	6.69	0.45		11.94				
Del.	8.24	0.74	1.89	0.31		2.36				
Ins.	0.46	0.08	0.72	0.10		2.21				
WER	34.59	2.37	**9.31**	0.78		16.51	**10.82**		41.00	44.60
SER	85.74	0.87	**65.07**	1.85		76.05				
BLEU	0.47	0.03	**0.83**	0.01	**0.736**		**0.8**	**0.81**		

$\bar{x}$=mean, σ=standard deviation

Table 2: Performance of the system. $^{(*)}$ Kobus 2008-1 corresponds to the ASR-like system, while Kobus 2008-2 is a combination of this system with a series of open-source machine translation toolkits. $^{(**)}$ Scores obtained on noisy data only, out of the sentence's context.

normalization algorithm is original in two ways. First, it is entirely based on models learned from a training corpus. Second, the rewrite model applied to a noisy sequence differs depending on whether this sequence is known or not.

Evaluated by ten-fold cross-validation, the system seems efficient, and the performance in terms of BLEU score and WER are quite encouraging. However, the SER remains too high, which emphasizes the fact that the system needs several improvements.

First of all, the model should take phonetic similarities into account, because SMS messages contain a lot of phonetic plays. The phonetic model, for instance, should know that *o*, *au*, *eau*, ..., *aux* can all be pronounced [o], while *è*, *ais*, *ait*, ..., *aient* are often pronounced [ɛ]. However, unlike Kobus et al. (2008a), we feel that this model must avoid the normalization step in which the graphemic sequence is converted into phonemes, because this conversion prevents the next steps from knowing which graphemes were in the initial sequence. Instead, we propose to *learn* phonetic similarities from a dictionary of words with phonemic transcriptions, and to build *graphemes-to-graphemes* rules. These rules could then be automatically weighted, by learning their frequencies from our aligned corpora. Furthermore, this model should be able to allow for timbre variation, like [e]–[ɛ], in order to allow similarities between graphemes frequently confused in French, like *ai* ([e]) and *ais*/*ait*/*aient* ([ɛ]). Last but not least, the graphemes-to-graphemes rules should be contextualized, in order to reduce the complexity of the model.

It would also be interesting to test the impact of another lexical language model, learned on non-SMS sentences. Indeed, the lexical model must be learned from sequences of *standard written forms*, an obvious prerequisite that involves a major drawback when the corpus is made of SMS sentences: the corpus must first be *transcribed*, an expensive process that reduces the amount of data on which the model will be trained. For this reason, we propose to learn a lexical model from non-SMS sentences. However, the corpus of external sentences should still share two important features with the SMS language: it should mimic the oral language and be as spontaneous as possible. With this in mind, our intention is to gather sentences from Internet forums. But not just any forum, because often forums share another feature with the SMS language: their language is noisy. Thus, the idea is to choose a forum asking its members to pay attention to spelling mistakes and grammatical errors, and to avoid the use of the SMS language.

Acknowledgments

This research was funded by grants no. 716619 and 616422 from the Walloon Region of Belgium, and supported by the Multitel research centre.

We sincerely thank our anonymous reviewers for their insightful and helpful comments on the first version of this paper.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proc. COLING/ACL 2006*.

Srinivas Bangalore, Vanessa Murdock, and Giuseppe Riccardi. 2002. Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system. In *Proc. the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA.

Richard Beaufort. 2008. *Application des machines à etats finis en synthèse de la parole. Sélection d'unités non uniformes et correction orthographique*. Ph.D. thesis, FUNDP, Namur, Belgium, March. 605 pages.

Markus Bieswanger. 2007. abbrevi8 or not 2 abbrevi8: A contrastive analysis of different space and time-saving strategies in English and German text messages. In *Texas Linguistics Forum*, volume 50.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report 10-98, Computer Science Group, Harvard University.

Noam Chomsky and Morris Halle. 1968. *The sound pattern of English*. Harper and Row, New York, NY.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar1, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10(3):157–174.

Vincent Colotte and Richard Beaufort. 2005. Linguistic features weighting for a text-to-speech system without prosody model. In *Proc. Interspeech'05*, pages 2549–2552.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proc. Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78.

Louise-Amélie Cougnon and Richard Beaufort. 2009. SSLD: a French SMS to standard language dictionary. In Sylviane Granger and Magali Paquot, editors, *Proc. eLexicography in the 21st century: New applications, new challenges (eLEX 2009)*. Presses Universitaires de Louvain. To appear.

Cédrick Fairon and Sébastien Paumier. 2006. A translated corpus of 30,000 French SMS. In *Proc. LREC 2006*, May.

Cécrick. Fairon, Jean R. Klein, and Sébastien Paumier. 2006. *Le langage SMS: étude d'un corpus informatisé à partir de l'enquête Faites don de vos SMS à la science*. Presses Universitaires de Louvain. 136 pages.

Emilie Guimier de Neef, Arnaud Debeurme, and Jungyeul Park. 2007. TILT correcteur de SMS: évaluation et bilan quantitatif. In *Actes de TALN 2007*, pages 123–132, Toulouse, France.

C. Douglas Johnson. 1972. *Formal aspects of phonological description*. Mouton, The Hague.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008a. Normalizing SMS: are two metaphors better than one? In *Proc. COLING 2008*, pages 441–448, Manchester, UK.

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

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. IJCAI'95*, pages 1137–1143.

Vladimir Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics*, 10:707–710.

Mehryar Mohri and Michael Riley. 1997. Weighted determinization and minimization for large vocabulary speech recognition. In *Proc. Eurospeech'97*, pages 131–134.

Mehryar Mohri and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *Proc. ACL'96*, pages 231–238.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2001. Generic ϵ-removal algorithm for weighted automata. *Lecture Notes in Computer Science*, 2088:230–242.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL 2001*, pages 311–318.

Emmanuel Roche and Yves Schabes, editors. 1997. *Finite-state language processing*. MIT Press, Cambridge.

Claude E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.

Richard Sproat, A.W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Crispin Thurlow and Alex Brown. 2003. Generation txt? The sociolinguistics of young people's text-messaging. *Discourse Analysis Online*, 1(1).

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proc. ACL'02*, pages 144–151.

François Yvon. 2008. Reorthography of SMS messages. Technical Report 2008, LIMSI/CNRS, Orsay, France.