Natural Language Engineering **16** (2): 133–159. © Cambridge University Press 2010
doi:10.1017/S1351324909990258

Rewriting the orthography of SMS messages

FRANÇOIS YVON
LIMSI-CNRS and Université Paris Sud 11, Paris, France
e-mail: yvon@limsi.fr

(*Received 30 December 2008; revised 20 July 2009; 28 October 2009; accepted 20 July 2009*)

Abstract

Electronic written texts used in computer-mediated interactions (emails, blogs, chats, and the like) contain significant deviations from the norm of the language. This paper presents the detail of a system aiming at normalizing the orthography of French SMS messages: after discussing the linguistic peculiarities of these messages and possible approaches to their automatic normalization, we present, compare, and evaluate various instanciations of a normalization device based on weighted finite-state transducers. These experiments show that using an intermediate phonemic representation and training, our system outperforms an alternative normalization system based on phrase-based statistical machine translation techniques.

1 Introduction

The rapid dissemination of electronic communication devices, such as emails, Short Messaging Systems (SMSs), chat rooms, instant messaging programs, and blogs, has triggered the emergence of new forms of written texts (see e.g. Crystal 2001; Véronis and Guimier de Neef 2006). Addressed to relatives or peers, written on the spur of the moment using basic interfaces, each with its specific constraints, such as computer keyboards, personal digital assistants (PDAs), and mobile phones keypads, these electronic messages are characterized by massive and systematic deviations from the orthographic norm, as well as by an unconventional use of alphabetical symbols. In contrast with normal spelling, letters and punctuation marks are used not only to conventionally encode a phonetic content but also to introduce meta-discourse and to signal emotions, verbal effects (e.g., laughters), or attitudes (such as humor, derision, and emphasis). If each medium enforces its own set of constraints and promotes idiosyncratic forms of writings, these new types of texts nonetheless share a lot of commonalities. To effectively process these messages, it is thus necessary to develop robust language preprocessing tools, capable of dealing with the extreme form of 'noise' they contain. In the present study, we focus more specifically on

We wish to thank the reviewers for many insightful comments on earlier versions of this paper. We are also grateful to C. Kobus, G. Damnati, and N. Palhavi for their contributions to our initial attempts at rewriting the orthography of SMS messages. We are also grateful to M. Adda-Decker for her comments on draft versions of this text.

SMSs, which, because of the paucity of their input interface (mobile phone keypads) and the space limitations, seem to constitute the most challenging type of data. To a large extent, the techniques we present in the current paper are also applicable to other types of electronic messages.

The 'SMS language' (or 'texting language') has been the subject of several linguistic studies (notably, for French, Anis 2001; Fairon, Klein, and Paumier 2006), which have emphasized its main characteristics, notably the extraordinary orthographic variability of lexical forms. In brief, this variability results from the mixing of several encoding systems: in SMS, the usual alphabetic system competes with a more 'phonetic' type of writing (e.g., *rite* for *right*[1]), as well as traces of a 'consonantic' orthography (vowels are deleted, as in *wrk* for *work* or *cn* for *can*), and with non-conventional use of letters or numbers, sometimes used to encode the phonetic value of their spelling, as in *ani1* for *anyone*. These processes can also be mixed as in *Rtst* (for *artist*) or *bcum* (for *become*). This variability is also the result of an informal style of communication, which licenses many deviations from the typographic (suppression of word separators and of quotes, erratic use of punctuation marks, and the like), orthographic (simplification of repeated consonants, omission of accents, use of non-conventional abbreviations), and grammatical (absence of case distinction, non-respect of agreement or tense markers, and the like) prescriptions, notwithstanding truly unintentional typos. Finally, practitioners of the texting language excel in devising acronyms that condense, sometimes in a radical way, multiword units: this is for instance the case with *afair*, which stands for *as far as I recall*. The following message, extracted from our database, illustrates the complexity of deciphering natural SMS texts (the message is on the first line, the normalized version on the second line, and an English translation on the third one):

c como kon fai pr ls k do
c'est comment qu'on fait pour les cadeaux
what do we do for the gifts

This example condenses several typical difficulties, such as the use of the abbeviation '*c*' which phonetically approximates '*c'est*', the unusual spelling '*como*' for '*comment*', and the superfluous space in the middle of '*kdo*', which is itself a rebus form for *cadeaux*. As a result, SMS messages contain an abnormally high rate of out-of-vocabulary (OOV) forms,[2] and the ambiguity of existing word forms is increased, two factors that contribute to degrade the performance of natural

[1] We will illustrate this general presentation of the SMS language using examples taken from English messages, even though our systems deal with French messages. As far as we can see, the same kinds of deviations from the orthographic norm are observed in both languages, albeit in different proportions. A more thorough comparison of both languages certainly remains to be carried out.

[2] For instance, the noisiest type of texts considered in Sproat *et al.* (2001), namely, real estate classified ads, contains 43.4% of the so-called nonstandard words; by comparison, our corpus of SMS contains approximately 70% of OOV word forms.

language processing (NLP) tools. Recovering a normalized orthography seems thus to be a necessary preprocessing step for many real-world NLP applications, such as text-to-speech, machine translation (MT), or text mining applications (filtering, routing, information retrieval, and the like).

In spite of their growing importance in day-to-day communication, these short messages have received relatively little attention from the NLP community[3] so far: see, for English, Aw *et al.* (2006) and Choudhury *et al.* (2007), both of which address the problem with statistical learning techniques, and, for French, Guimier de Neef, Debeurme and Park (2007), which details a complete pipeline of hand-crafted, symbolic modules. In fact, the problem of normalizing SMS shares a lot with other NLP applications and can be addressed from several viewpoints. The first, maybe the most natural angle, is to make an analogy with the spell-checking problem. This problem has been extensively studied in the past, and a variety of statistical approaches are readily available, most notably the 'noisy channel' approach (see e.g. Church and Gale 1991; Golding and Schabes 1996; Brill and Moore 2000; Toutanova and Moore 2002). An alternative metaphor is the translation metaphor: under this view, the SMS language is considered as a foreign language, and the normalization task is accomplished using standard (statistical) translation techniques (Brown *et al.* 1990; Koehn, Och, and Marcu 2003). Both views have their own merits and their limitations, which we will discuss shortly. A third metaphor, proposed in Kobus, Yvon, and Damnati (2008a), draws inspiration from some similarities between the SMS language and speech and notably from the fact that in SMS, word separators are much more unstable than in conventional writings. To recover missing word segmentations, these authors propose to resort to techniques that are commonly used in speech recognition systems, most notably the use of weighted finite-state transducers for representing both a phonemic transduction model and a statistical language model. The resulting system however proves to be significantly worse than a system employing statistical MT techniques; combining both approaches provides a very substantial gain over each individual system.

The work reported in the current study explores the potential of integrated finite-state approaches to the SMS normalization problem, presenting a complete architecture for this task with all the necessary details. As such, it continues a fruitful line of research on text normalization techniques, which have a long-standing history in the area of text-to-speech applications and have been recently revitalized by the need to process noisier and more diverse texts. Text normalization has several facets: to recover sentence and paragraph boundaries, to restore proper capitalization and punctuation, to provide an appropriate textual form for nonstandard words, and to correct spelling errors, among others. These problems are studied, notably in Mikeev (2000), Clark (2003), and Zhu *et al.* (2007), all of which present integrated architectures for simultaneously solving these problems. The technical solutions

[3] A couple of on-line SMS-to-English translation systems are accessible on the Internet; see for instance `http://www.transl8it.com/` and `http://www.lingo2word.com/`; 'Netspeak' dictionaries, again for English, also abound. The situation is more or less comparable for French; see for instance `http://www.traducteur-sms.com/`.

advocated in these papers, based on trainable finite-state devices, have a lot in common with ours. Our problem is however somewhat different, because of several peculiarities of SMS messages: (i) they are very short (approximately twenty-one tokens on average), which makes it unnecessary to fix issues related to sentence or paragraph boundaries; (ii) as mentioned earlier, the proportion of OOV tokens is remarkably high, which implies that techniques for normalizing the orthography must be used; (iii) most misspellings are deliberate and sometimes even systematic (rather than unintended), which suggests that part of this orthographic variation can be modeled; (iv) as is the case for speech transcripts, the notion of what constitute a sentence is often unclear, which makes it difficult to recover sentence boundaries and to properly address the related problem of restoring a proper casing. Consequently, the primary focus of the current work will be the normalization of word tokenization and orthography, and we will entirely disregard issues related to sentence breaks, punctuation, and casing. In this respect, our problem is more similar to one of the tasks considered in Sproat *et al.* (2001), which consists of expanding unconventional abbreviations occurring in classified adds. The authors of this study use, as we do, weighted finite-state transducers: we however find that the material considered here is both noisier and much more varied, which prevents us to use, as they do, domain-dependent vocabularies or language models.

The main novelty of the current work is thus neither methodological nor technical but primarily stems from the unusual nature of the SMS messages, whose extreme noisiness makes them sometimes as difficult to decipher as if they were written in a foreign language. In particular, we show that a great deal of variation in spelling can be captured using well-established finite-state technologies and discuss the issues that remain unsolved. More specifically, we introduce various add-ons to the architecture proposed in Kobus *et al.* (2008a): in particular, we consider adding a letter-based error model that complements the phonemic model, to optimize these models using a training step and to automatically extract an exception dictionary. We also provide a systematic evaluation of the individual merits of these innovations. Most of the gain is achieved through training, whereas the addition of an error model does not seem to help much. Our best system, which incorporates all these novel features, proves to be significantly better than a baseline using statistical MT techniques.

This paper is organized as follows: In Section 2, we discuss three metaphors for the normalization of SMS; we then give a detailed presentation of our implementation of a generic SMS normalization system. The general principles are introduced in Section 3, and specific implementation details are reported in Section 4. A comparative evaluation of various implementations is conducted in Section 5, where we also contrast our results with those obtained using two baseline systems. This section also includes an analysis of the unsolved SMS normalization errors. Section 6 presents a summary of the main findings and discusses future prospects.

2 Three approaches (plus one) to SMS normalization problem

In this section, we set the general problem of SMS normalization and discuss, on the basis of the analysis of SMS examples, the relevance of various NLP approaches to this task.

In a nutshell, SMS normalization consists of rewriting an SMS text using a more conventional spelling, in order to make it more readable for a human or for a machine. Cast in such terms, SMS normalization is related to the many problems related to normalizing and/or correcting the spelling of noisy inputs.

2.1 The 'spell-checking' metaphor

A first approach to the problem considers each input token as 'noisy' observation of the correct word form(s): normalization is thus viewed as a spell-checking task. The spell-checking problem has received considerable attention in the past, and a variety of correction techniques have been explored in the past (Kukich 1992; Mitton 1996). In this context, noisy channel models (see e.g. Church and Gale 1991; Brill and Moore 2000; Toutanova and Moore 2002) constitute one of the predominant and most successful approaches. Under this paradigm, correction is performed on a word-per-word basis and concerns primarily OOV tokens[4] : the general assumptions are that most words are correctly spelled and that in-vocabulary words should preferably be left untouched. In this context, the best correction(s) w of an erroneous word v is retrieved from the dictionary by combining the individual context-independent probability of w with an error model probability, which computes the probability of mistyping v for w, on the basis of the surface similarity between both forms. The key component here is the error model, which should capture not only orthographic neighbors (Brill and Moore 2000) but also phonetic similarities (Toutanova and Moore 2002).

This framework easily extends to the case in which several words should simultaneously be corrected: it is simply a matter of exploring the lattice of all possible corrections, which can be reranked using conventional tools such as statistical language models. This is basically the idea behind the `reac` system (Simard and Deslauriers 2001), which recovers the correct accentuation of unaccented French texts using an error model, complemented with a statistical language model.

As far as SMSs are concerned, this approach is essentially the one advocated in Choudhury *et al.* (2007). The authors propose to model the joint probability of observing the word w represented by the character sequence $c_1, \ldots, c_l$ by a hidden Markov model (HMM) whose topology takes into account both 'graphemic' variants (such as typos and omissions of repeated letters) and 'phonemic' variants (i.e. spellings that resemble the word's pronunciation). This HMM is initialized by considering the word's received orthography and phonology, with additional transitions to account for the possibility of inserting, substituting, or deleting symbols. For the most frequent words, the various parameters associated with these transitions are estimated on a training corpus; various heuristics are then used to plug these values into the HMMs that model the less frequent word types.

[4] The detection and correction of spelling errors that result in *an existing word form*, such as *there* for *their*, also remains a challenging problem (Golding and Schabes 1996; Golding and Roth 1999).

2.2 The 'translation' metaphor

A second approach to the problem consists of adopting the translation metaphor: using this analogy, the SMS language is just another foreign language, and normalization can be viewed as a MT task.

Using MT tools might be regarded as 'an overkill' (Choudhury *et al.* 2007), considering the close relationships between source and target languages. Furthermore, learning the kinds of many-to-many correspondences between source and target sentences that make up for the high translation accuracy of phrase-based systems might be seen as introducing an unnecessary complexity, as SMSs tend to be shorter, in terms of characters and words, than their normalized counterparts. This suggests that looking for many (on the normalized side) to one (on the SMS side) might be good enough to capture most pairings. Finally, statistical MT tools incorporate mechanisms to model the possible mismatch in word order between source and target, which are virtually nonexisting when it comes to translating SMS.

This metaphor is, nonetheless, the one resorted to in Aw *et al.* (2006), which uses a statistical phrase-based MT tool to convert English SMS texts into standardized English. This system incorporates some of the peculiarities of the SMS translation task, which simplifies the construction of both the phrase-table and the decoding (search) algorithm. Using this system, Aw *et al.* (2006) reported a 0.81 Bilingual Evaluation Understudy (BLEU) (Papineni *et al.* 2001) score on a set of 5,000 English SMSs.

Normalization as translation is certainly a natural idea; it is also simple to implement, given sufficient training data. Using conventional phrase-based systems, it becomes possible to model (context-dependent) one-to-many relationships that are out of reach of the spell-checking approach. We feel that it still overlooks some aspects of the task, notably the fact that the lexical creativity attested in SMS messages can hardly be captured in a static phrase table, where correspondences between SMS 'phrases' and normalized phrases are learned by rote, rather than modeled.

2.3 The 'speech recognition' metaphor

The SMS language is often described as being closer to oral productions than to regular written texts. If we do not completely agree with this view (see the discussion in Fairon *et al.* 2006), we nonetheless feel that a third metaphor is worth considering, namely, the 'automatic speech recognition' (ASR) metaphor. This analogy stems from the fact that for a significant fraction of tokens, the spelling of SMS forms tends to be an approximation of the word's phonemic representation rather than of its normative spelling.

In the speech recognition metaphor, an SMS message is thus primarily viewed as an alphabetic/syllabic approximation of a phonetic form. Given a suitable mechanism for converting the SMS stream into a phone lattice, the problem of SMS normalization becomes very similar to that of speech recognition, that is, the decoding of a word sequence in a (weighted) phone lattice. It actually becomes a much simpler problem, as (i) the acoustic ambiguity of speech input is typically much higher than the phonemic indeterminacy of SMS messages, and (ii) some segmentation information is already available in the SMS text, which is in sharp contrast with (continuous) speech recognition, where word boundaries have to be uncovered.

Based on this general principle, we proposed in Kobus, Yvon, and Damnati (2008b) an ASR-like normalization system, which has three additional merits over the other approaches: using a phonemic approximation provides the system with the ability to correct some (unintentional) typos; adopting an ASR-like architecture provides us with a 'natural' framework for resegmenting agglutinated word forms; finally, in the larger context of SMS-to-speech applications, which is one of our targeted applications, the computation of a phonemic representation of the message can prove extremely valuable.

2.4 Combining systems

Each of these metaphors has its own merits and provides a partial solution to the normalization problem, as discussed in Kobus *et al.* (2008a). This suggests that they could be combined, and indeed, a combination of the MT and the ASR-like normalization systems allowed to improve significantly over each best individual result. On the basis of the observation that the MT-based system was at pain recovering long agglutinated chunks, the idea is to automatically detect these problematic segments and to process them *in isolation* through the phoneme-based model. The resulting output, taking the form of local word lattices containing various decompositions, is recombined with the remaining portion(s) of the message, yielding a global word lattice that is finally reevaluated by the language model.

In the work reported here, we consider a more principled way to combine these various metaphors in one integrated architecture, which extensively uses the formalism of finite-state transducers. As we will discuss, our system can be configured so as to instantiate either of the metaphors and to use them in combination.

3 Architecture of the normalization system

In a nutshell, the normalization system is composed of two layers of nondeterministic weighted mappings realized by finite-state transducers. The first layer of transduction nondeterministically maps sequences of letters to sequences of letters and produces a weighted finite-state automaton on a letter alphabet: it plays the same role as the acoustic models in speech recognition applications. The second layer maps sequences of letters to weighted sequences of words, using statistical language modeling tools.

3.1 Principles

Before describing the architecture of our normalization system, let us introduce some notations: A denotes the set of alphabetic symbols (letters, numbers, and conventional punctuation marks). Further, A is complemented with additional symbols, namely, '<' (start message), '>' (end message), and '␣' (token separator).

This extended symbol set is denoted B. An SMS message is thus a sequence in $< \textvisiblespace(A + \textvisiblespace)^{\star} >$. We will also use two additional alphabets: Z is a finite set of phonetic symbols, and W is a finite set of words, including conventional units for 'start of message' and 'end of message'.

The following sections assume a basic knowledge of weighted finite-state transducer technologies. We refer the reader to the abundant literature on these issues, notably to Roche and Schabes (1997), Mohri (1997a, 1997b), and Mohri, Pereira, and Riley (2000), which have reported many useful results concerning (weighted) finite-state transducers and their applications to NLP problems. Given two transducers T_1 and T_2, representing rational relations $R(T_1)$ and $R(T_2)$, we will denote $T_1^{\star}$ the transducer computing the closure of $R(T_1)$, $T_1 T_2$ (resp. $T_1 + T_2$, $T_1 \circ T_2$), the transducer computing the concatenation (resp. the union, the composition) of $R(T_1)$ and $R(T_2)$; $\pi_1(T_1)$ and $\pi_2(T_1)$ denote the automata recognizing the domain and codomain of $R(T_1)$.

As explained above, the normalization system is composed of two layers of nondeterministic weighted mappings realized by finite-state transducers. The first one realizes a rational transduction G over $B^{\star} \times B^{\star}$ that nondeterministically maps an input into all the possible ways in which it can be normalized: this is the *graphemic model*. The second layer, realized by the weighted automaton S, represents a mapping from $B^{\star}$ to $\mathbb{R}$, which evaluates the validity of these hypotheses, using a local syntactic model, here an n-gram language model: this will be referred to as the *syntactic model*. The normalized version $n(x)$ of a message x (represented as a finite-state automaton over $B^{\star}$) is thus computed as $n(x) = \text{bestpath}(\pi_2(x \circ G \circ S))$. These two layers are described in detail in the following sections. To be complete, note that our system also includes some simple pre- and postprocessing scripts that are merely in charge of converting character strings to and from the canonical automaton representing that string. A complete picture of this architecture is given in Figure 1.

3.2 Generating normalizations: The graphemic model

The first layer of normalization is itself composed of several simpler machines, which are aimed at capturing three types of non-conventional spellings.

- Purely consonantic spellings: at present, these are simply taken care of through a dictionary look-up.
- Typos, or simple cases of shortenings, which are modeled through a simple *error model*.
- Phonetic or rebus spellings: these are modeled through nondeterministic grapheme-to-phoneme rules.

The finite-state transducers implementing these models are described in the following sections.

3.2.1 Exception dictionaries

The first action is to look up tokens of the SMS message in an exception dictionary E: when a form is found in E, it is replaced by the standardized counterpart(s).

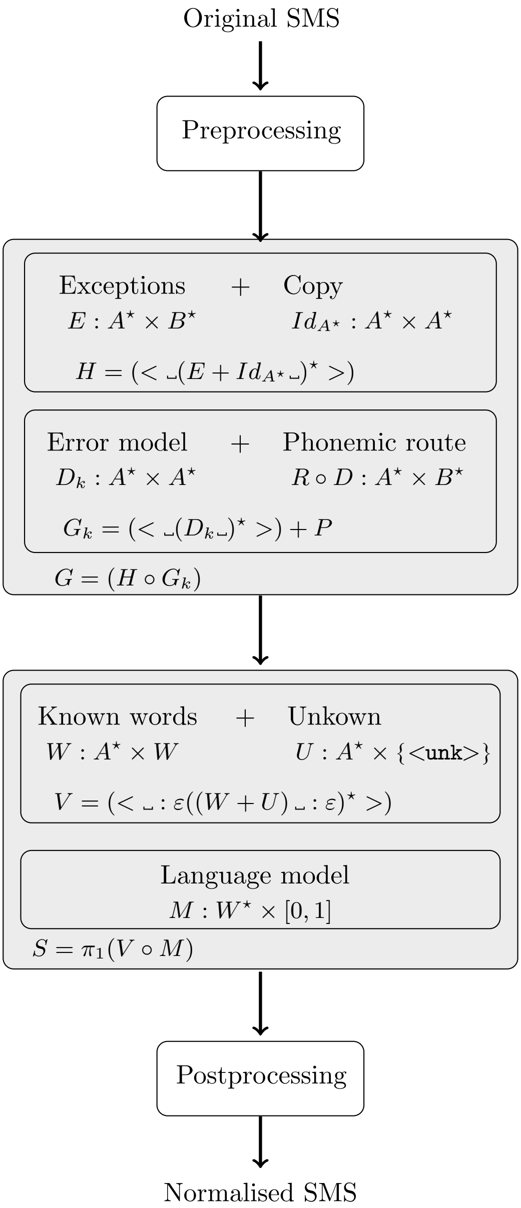
Original SMS
Preprocessing
Exceptions + Copy
$E : A^\star \times B^\star$
$Id_{A^\star} : A^\star \times A^\star$
$H = (< \llcorner(E + Id_{A^\star}\llcorner)^\star >)$
Error model + Phonemic route
$D_k : A^\star \times A^\star$
$R \circ D : A^\star \times B^\star$
$G_k = (< \llcorner(D_k\llcorner)^\star >) + P$
$G = (H \circ G_k)$
Known words + Unkown
$W : A^\star \times W$
$U : A^\star \times \{<\texttt{unk}>\}$
$V = (< \llcorner : \varepsilon((W + U)\llcorner : \varepsilon)^\star >)$
Language model
$M : W^\star \times [0, 1]$
$S = \pi_1(V \circ M)$
Postprocessing
Normalised SMS

Fig. 1. The complete normalization system.

This replacement is optional (or nondeterministic) meaning that the output stream contains both the standardized and the nonstandardized forms. All other tokens are simply copied to the output.

Let E denote the finite-state transducer mapping exceptions to their standardized form(s); each individual token in the input message is thus rewritten through composition with

$$E + Id_{A^\star}$$

where $Id_{A^\star}$ is the identity relationship over $A^\star$. The transducer computing this transformation on complete sentences is therefore defined as[5]

$$H = (< \llcorner((E + Id_{A^\star})\llcorner)^\star >)$$

It maps any sequence of $\llcorner$-separated tokens in $A^\star$, completed with adequate initial and final symbols, to sequences of orthographic symbols, where exceptions have been nondeterministically replaced by their expanded correlate(s).

3.2.2 Error-correction model

To account for spelling errors, typos, and the replacement of accented letters with their unaccented counterpart, we extend the grapheme-to-grapheme transduction as follows: each token in the original message is not only copied but also associated with all its orthographic neighbors (with respect to the traditional edit distance) at bounded distance. The computation of these neighbors can also be computed through a weighted finite-state machine. Let us denote D_k the finite-state transducer realizing the following weighted relation:

$$\forall u, v \in A^\star, D_k(u, v) = l \text{ if and only if } l = d(u, v) \leq k$$

where $d()$ is the usual string edit distance. The extended graphemic route is computed as

$$G_k = (< \llcorner(D_k\llcorner)^\star >)$$

Furthermore, G_k is parameterized by a specific choice of k: for $k = 0$, Δ_k is computing the identity relationship, and no error model is actually used.

3.2.3 A phonetic route

The 'phonetic route' also maps strings in $B^\star$ to strings in $B^\star$, albeit in a less direct manner, by way of a nondeterministic projection of orthographic strings into phonetic strings and back.

Let R denote the finite-state transducer computing of (ordered) composition of the set of nondeterministic grapheme-to-phoneme rules described in Section 4.2: R maps strings in $A^\star$ to strings in $Z^\star$. The mapping back to orthographic strings is performed through an inverted phonetic dictionary D, represented as a finite-state transducer D, mapping sequences of phones to sequence of letters.

The complete phonetic route is thus computed as

$$P = (< \llcorner((R \circ D)\llcorner)^\star >)$$

Crucially, D is designed so as to allow for the insertion of *novel* token separators: starting with a canonical representation of a finite string-to-string relationship, we add transitions mapping ε (on the input tape) onto $\llcorner$ (on the output tape) between

[5] In the rational expressions above, an isolated symbol such as $<$ represents the identity transducer for that symbol.

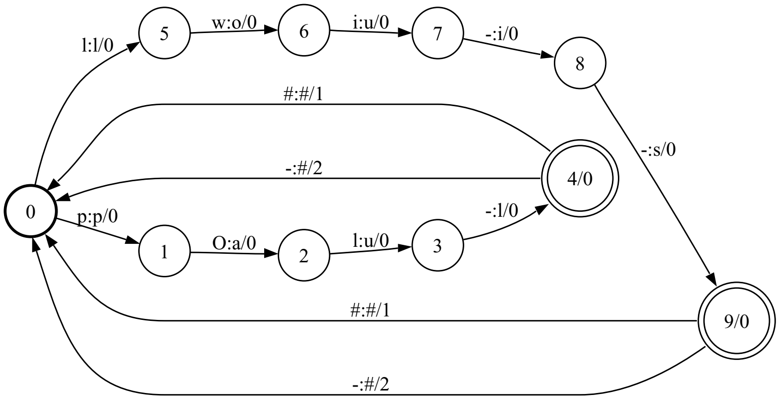

Fig. 2. A very small finite-state transducer representing a phonetic dictionary: *The path from 0 to 9 maps the input phonetic string /lwi/ to the orthographic string 'louis'. Upon reaching the final node 9, two transitions loop back to state* 0, *the initial state: the first one copies an actual token separator ('#' stands for '␣' in the graph) and has a unit cost; the second spontaneously emits the token separator (the input symbol is* ε*, represented as '-' on the graph) and incurs a cost of 2.*

any final state and the initial state, thus making the resegmentation of input tokens possible. This mechanism is illustrated in Figure 2.

We performed experiments both with and without this phonemic module: in the latter case, the rule set is empty and P maps $B^\star$ to the empty set.

3.2.4 The complete graphemic model

Both routes can be used in combination, in which case the first step of rewriting is performed by a transducer G realizing the union of the orthographic (G_k) and phonemic (P) routes.

A final step consists of 'standardizing' these transducers, i.e., to make sure that for each state q, the weights of all edges leaving q plus the final weight of q sum to one. This operation ensures that the total weight associated with the transductions y of a message x through G can be interpreted as the joint probability $\Pr_G(x, y)$ in the model computed by G.

3.2.5 Training the graphemic transducer

As outlined in our previous work (Kobus *et al.* 2008a), all orthographic errors or phonetic mappings are not equally likely. It is, for instance, expected that the replacement of an accented letter with the non-accented counterpart will be very frequent, as will be any substitution of characters mapped onto the same keypad.

To better model graphemic variants, we also derived trained versions of G, where the weights are set so as to maximize the likelihood of the training corpus. To this end, for each specific instantiation of G, we computed the subset of the training corpus that could be computed by G and performed maximum-likelihood estimation of the transducer weights based on this subcorpus.[6]

The training problem is formalized as follows: given a (supposedly i.i.d) sample of pairs $\{(x_i, y_i), i = 1, \ldots, N\}$ composed of an SMS message and its standardized version, we search for the set of weights that maximizes $\sum_i w(x_i, y_i)$. The actual paths through G being hidden, this problem is readily solved though a straightforward adaption of the popular expectation-maximization (EM) algorithm for general finite-state transducers (see e.g. Eisner 2002 or the very complete presentation in Jansche 2003). As is common with EM, convergence is reached in a few iterations, and we arbitrarily stopped training after five iterations of this procedure.

3.3 Syntactic model

The second main component of the standardization system is a language model component, aimed at providing each output orthographic sequence with a ‘syntactic’ likelihood score. This operation is performed by a string-to-weight transducer S over $(B^\star \times [0,1])$ built as follows.

Based on the normalizations available in the SMS corpus, we build a conventional *open-vocabulary* n-gram language model. The lexicon simply comprises all the words having at least o occurrences in the training corpus. This language model is expressed as a weighted finite-state automaton M, mapping sequences of words from $(W \cup \{\texttt{<unk>}\})^\star$, where <unk> represents the ‘OOV’ word, to numbers in $[0, 1]$.

We then compile the word list W as a costless finite-state transducer over $A^\star \times W^\star$. Strings that do not belong to this list are mapped, through the transducer U, to the special ‘word’ token <unk>. Unknown words incur an extra penalty u, which compensates for the high probability the language model assigns to the <unk> token: the occurrence of an unknown word is a quite likely event; however, each individual unknown word should be given a small probability.

These finite-state machines are finally assembled as follows:

$$S = \pi_1((< (\textvisiblespace : \varepsilon)((W + U)(\textvisiblespace : \varepsilon))^\star >) \circ M)$$

The reason for taking (through π_1) the domain of this relationship is to keep track of the original orthographic sequences associated with unknown words, which would otherwise be lost through the mapping realized by U.

Overall, S’s exact behavior is governed by several parameters:

- n, the language model order;
- o, the unigram frequency cutoff used for building the vocabulary;
- u, the extra penalty for OOV words;
- f, the language model scaling factor, which is required to make the scores associated through G and S comparable;
- w, a sentence-length penalty, which plays the role of a prior over sentence lengths.

[6] Depending on the shape of G, the training corpus for EM contained between a third and a half of the total training corpus. This means that many useful training instances are not used during this step, suggesting that we could probably improve our system by finding ways to remedy this situation.

Table 1. *Statistics of the training corpora*
Statistics on the original messages are computed after preprocessing (for example, punctuation removal); the length of a message is the number of tokens; % types unk. (resp. tokens unk.) is the percentage of types (resp. tokens) that do not occur in a large reference dictionary containing several hundreds of thousands of forms.

	Train	Dev	Test
Number of messages	23,798	2,931	2,976
Original SMS			
Average length	21.2	21.0	21.1
Number of types	37,537	9,390	9,777
Number of tokens	505,105	61,693	62,700
% types unk.	73.15%	63.0%	63.2%
% tokens unk.	29.67%	29.0%	29.4%
Normalized messages			
Average length	23.1	23.2	22.8
Number of types	21,224	6,301	6,557
Number of tokens	548,703	66,772	67,958
% types unk.	44.21%	34.5%	35.2%
% tokens unk.	9.28%	9.31%	9.56%

4 Implementation

The implementation of this normalization system requires resources, taking the form of training corpora, dictionaries and pronunciation rules, and software components. These more practical aspects of our work are detailed in this section.

4.1 Base corpora

In the current study, we only use a corpus of SMSs collected in Belgium by the Catholic University of Louvain, which totals about 30, 000 messages (Fairon and Paumier 2006). This corpus contains, for each message, a reference normalization that has been produced and validated by human annotators. This raw corpus was additionally lowercased and stripped of any punctuation sign.

This database was randomly split in a training set (about 24, 000 messages), a development set (about 3,000 messages), and a test set (about 3, 000 messages). The training set was used to train both the transducers and the language models; the development set was used to adjust the values of the various numerical parameters, and the test set was used to evaluate the quality of our normalization system.

Some statistics regarding these subcorpora are given in Table 1. As is readily observed from these figures, the non-normalized messages tend to be on average slightly shorter than their normalized counterparts. More strikingly, the large difference between the number of types in both types of messages is a sign of the great lexical variability that characterizes the SMS texts.

Table 2. *An excerpt of the exception dictionary*

SMS	Standard	
2day	today	‘loan’ sms word + rebus
2m1	demain	rebus: read ‘2’ as /də/ and ‘1’ as /ẽ/
2mand	demander	rebus: read ‘2’ as /də/ and ‘d’ as /de/
2ri1	de rien	rebus + agglutination
5pa	sympa	rebus + abbreviation (of *sympathique*)
@2m1	à demain	logogram + rebus + agglutination
chuis	je suis	phonetic spelling (assimilation of /ʒ/ to /ʃ/)
cmt	comment	consonantic spelling

4.2 Resources

As for linguistic resources, namely, dictionaries and grapheme-to-phoneme rules, we mostly resorted to generic resources that were already available on site, without trying to fine-tune these resources for our application.

4.2.1 Exception dictionary

Our system incorporates a dictionary mapping the most common abbreviations to their complete orthographic expansion. It contains pairs such as *btw* for *by the way*, as well as instances of ‘consonantic’ spellings (see Table 2). In the majority of our experiments, we use a small dictionary containing 229 entries, collected on various sites dedicated to the SMS language. This is much smaller than the dictionaries used in Guimier de Neef *et al.* (2007) and Kobus *et al.* (2008a).

In contrastive experiments, this dictionary was completed with automatically extracted mappings (see below), yielding a medium-sized (about 800 entries) dictionary and a large (2,500 entries) dictionary. Once compiled as a finite-state transducer, this larger dictionary contains approximately 20,000 states.

4.2.2 Grapheme-to-phoneme-to-grapheme mappings

Our system, in some settings, normalizes SMS through a phonetic route, which necessitates two supplementary linguistic resources. The first is a set of manually designed, *nondeterministic*, letter-to-phone rules, which convert the graphemic portions of the input message into a phonemic string. As illustrated in the short excerpt reproduced in Table 3, these rules notably encode the possibility for each symbol to stand for its spelling (e.g., ‘*u*’ for /ju/ or ‘*r*’ for /ər/).

These rules have been manually ordered so as to avoid potential conflicts, meaning that they fire in such a way that the application of rule R_i does not destroy the context necessary for rule R_j (with $j > i$). Each of these rules R_i is compiled as a finite-state transducer R_i (Kaplan and Kay 1994; Mohri and Sproat 1996), and the grapheme-to-phoneme conversion is performed through the composition $P = \bigcirc_i R_i$.

Our system currently comprises about 160 letter-to-phone rules, which deal with approximately 60 orthographic symbols, that is, on average less than 3 rules for each

Table 3. *Some nondeterministic grapheme-to-phoneme rules*
The first rule states that the trigram 'aim' rewrites both as /ɛm/ and as /ẽ/, in all possible left and right contexts.

source	→	target	/	left	—	right
aim	→	(ɛm+ẽ)	/	*	—	*
ain	→	(ɛn+ẽ)	/	*	—	*
ai	→	(ə+ɛ)	/	f	—	s(a+i+o)
ai	→	ɛ	/	*	—	*
a	→	a	/	*	—	*
b	→	(be+b)	/	*	—	*
2	→	d(ə+œ)	/	*	—	*

symbol, a very small number compared with what would be required to convert French spelled words into an accurate phonemic representation (see Divay and Vitale 1997; Yvon *et al.* 1998; Béchet 2001). This means that most of these rules are pretty general (i.e., that they apply in all contexts) and yield compact transducer representations. Overall, P contains about 5,000 states and 80,000 transitions.

4.2.3 Dictionaries

In addition to the exception dictionary, our system also requires a fixed list of words, which represents the vocabulary of the language model and constitutes the interface between the graphemic transducer G and the syntactic transducer S. This list simply collects all the orthographic word forms that occur in the training set and contains approximately 21,000 word forms. To complete the phonemic route, we also required phonetic pronunciations for those words: for most of them,[7] a pronunciation was readily available in our pronunciation dictionary (Boula de Mareüil *et al.* 2000); for the remaining ones, we simply considered that they could not be normalized through the phonemic route. The pronunciation dictionary, once turned into a finite-state transducer D, contains about 12,000 states and a double number of transitions.

4.2.4 Language models

The main language models used in our experiments are conventional n-gram language models (Jelinek 1990) estimated using the training corpus with the open-source Stanford Research Institute Language Modeling (SRILM) toolkit (Stolcke 2002), using a Kneser–Ney smoothing scheme (Chen and Goodman 1996). We trained language models for $n = 0$ to $n = 3$; the largest language language model contains 21,227 unigrams, 158,785 bigrams, and 330,932 trigrams. Using this model, the perplexity on the development corpus was found to be about 92.5.

A contrastive experiment tries to take advantage of much larger, albeit less adapted, training material. We used transcriptions from phone conversations, totaling 1.5 million tokens. This corpus is used to train a larger language model, which was

[7] About 80% of the forms were found in the dictionary.

then linearly interpolated with the SMS language model. As expected, this combined language model yields an improved perplexity on the development corpus, which goes down to seventy-six.

4.3 Implementation issues

Our system makes extensive use of the AT&T Finite-State Machine (FSM) toolkit (Mohri *et al.* 2000), with which all the transducer manipulations are performed. The Grammar (GRM) Library (Allauzen, Mohri, and Roark 2005) is used to compile grapheme-to-phoneme rewrite rules as finite-state transducers. We used in-house scripts to produce the transducer representation from our various linguistic resources and our own implementation of the EM algorithm for finite-state transducers, which is simplified here by the acyclic nature of the transducer we train.

For the sake of flexibility, most of our experiments were conducted using a faithful implementation of $G \circ S$: the normalization of each message mainly involves the computation of two composition operations, plus one search for the most likely path. The average runtime for processing one message was about 0.5 s on a conventional desktop machine (including preprocessing). Should speed become an issue, there are several ways in which these runtimes could be improved. An obvious idea would be to precompute $G \circ S$ and to load it only once in memory; additionally, this transducer could be pruned and/or locally determinized.

4.4 Using machine translation tools

4.4.1 SMS normalization as machine translation

For comparison purposes, we also developed a normalization system based on open-source, public-domain packages for statistical MT. Giza++ (Och and Ney 2003) is used to induce (Brown *et al.* 1990) an automatic alignment of SMS tokens with their normalized counterparts. Moses (Koehn *et al.* 2007) is used to learn the various parameters of a phrase-based ‘translation’ model for the SMS language, to optimize the weight combination on the development corpus using minimum error rate training (Och 2003) and to perform the translation using a multi-stack search algorithm. The SRILM toolkit (Stolcke 2002) is again used to estimate a 3-gram statistical language model, smoothed with Kneser–Ney back-off.

Given the simplified setting for translation provided by the absence of reordering between source and target languages, during both training and decoding we consider only ‘monotonic’ alignments between the source and the target. The maximum phrase length is set to seven.

4.4.2 Learning exceptions

This architecture was also used to automatically acquire new entries for our exception dictionaries. In fact, the translation model represented in the form of a phrase table stores weighted associations between SMS ‘phrases’ and their normalized counterpart. From the 85,128 entries in this table, we focused on pairs that (i) contained a single SMS token and (ii) for which the association was deemed significant, so as to filter some of the noise present in these tables. Varying the threshold for selecting such mappings, we collected a medium-size (about 600) and large (approximately 2,500) list of pairs, which were added to the exception dictionary. A rapid check of these learned exceptions indicates that they mostly account for 'unusual' abbreviations, such as '*ti*' or '*tit*' for '*petit*' (*small*) and '*tjs*' or '*tj*' for '*toujours*' (*always*), whose received consonantic spelling is '*tjrs*'. A significant portion of these new exceptions corresponds to agglutinated forms, the most common pattern being the blending of the first-person subject clitic '*je*' with the main verb: '*jveu*' or '*jveux*' for '*je veux*' (*I want*), '*jcrois*' or '*jcroi*' for '*je crois*' (*I believe*), and so on.

Using these novel exceptions, it is possible to devise a system that loosely emulates the behavior of a conventional statistical MT system: it suffices to build G in such a way that (i) no error model is used, which is achieved by taking the edit distance parameter $k = 0$; (ii) the phonemic route is not used; and (iii) the graphemic model induced by G is optimized on the training corpus. As a result, messages are solely normalized by accessing the phrases stored in the exception dictionary.

5 Experiments

5.1 Experimental protocol

The tuning of the various parameters (language model scaling factor, word insertion penalty, unknown word penalty) was performed on a subset of the development corpus, using a brute-force approach, yielding reasonable (albeit probably slightly suboptimal) values. All the numbers reported below are based on a single run on the test set.

As far as the evaluation metrics is concerned, contrary to Aw *et al.* (2006) and Guimier de Neff *et al.* (2007), who assessed their system with the BLEU metric (Papineni *et al.* 2001) (or rather by a variant of this score) by analogy with the MT task, we decided to measure the performance of our normalization tool with the word error rate (WER) and sentence error rate (SER) metrics. This choice is motivated by the fact that the outcome of the normalization process is – notwithstanding a couple of arbitrary normalization decisions – almost deterministic and does not warrant the use of BLEU, which is more appropriate to evaluate tasks with multiple references. Additionally, we feel that error rates are easier to interpret than BLEU values.

Significance analysis of WER differences was performed using a matched-pairs test, as proposed in Gillick and Cox (1989).

5.2 Main results

5.2.1 Baseline

Our systems were tested in systematic contrastive experiments along five dimensions:

- the use of a 'correction' model, parameterized by the maximum edit distance between an SMS word and its correct counterpart ($k = 1, 2$, $k = 0$ means no correction);

Table 4. *Baseline performance*

	Corr	Sub	Del	Ins	Err	S.Err
Copy	63.9	27.5	8.6	0.1	36.3	86.3
Moses	83.8	10.9	5.3	0.9	17.2	73.3

Table 5. *The twenty most common confusions of the baseline (copy) system.*

Frequency	SMS	Normalized	Frequency	SMS	Normalized
563	a	à	388	pr	pour
354	c	est	349	ca	ça
308	pa	pas	290	g	ai
184	ke	que	165	t	t'
118	j	j'	117	tt	tout
116	j	je	110	d	de
89	etre	être	86	tantot	tantôt
85	ds	dans	85	la	là
85	ms	mais	73	l	l'
70	vs	vous	69	po	pas

- the use of a 'phonemic route' (yes or no);
- the use of training (yes or no);
- the use of a small or medium exception dictionary;
- the language model order.

These performances are contrasted with respect to our two baseline systems: one that simply recopies the SMS as the proposed normalization and one that uses the MT approach. The performance of these baseline systems are reported in Table 4.[8]

The high percentage of substitution errors reflects the strong deviation from the conventional spelling (see Table 5 for the most frequent confusions). The high percentage of deletions is due to the frequency of blending (cf. also the frequency of the alignment of '*g*' with '*ai*', which occurs every time '*g*' is used for '*j'ai*', incurring one deletion and one substitution). These blended forms remain difficult to handle for the phrase-based translation system and continue to account for a substantial number of errors.

5.2.2 *The benefits of training*

In this section, we report the results of our experiments conducted with the *small* exception dictionary. The main figures are reported in Table 6.

The conclusions that can be drawn from this series of experiments are the following:

[8] The performance reported in the current paper are systematically much worse than those reported in Kobus *et al.* (2008a). For instance, in this earlier work, the WER of the Moses system is close to 12.3%, five points below the value reported here. As far as we can see, these differences are mainly due to the fact that our previous experiments used a second (proprietary) corpus, containing much shorter (and thus simpler) messages. As a result,

Table 6. *Performance of untrained (wot) and trained (wt) systems*
A system relying on a trained (wt) correction model (with k = 1) and phomemic model ('y'), plus a 3-gram language model yields a WER of 17.8%. Some parameter combinations were not evaluated, hence the missing numbers in this table.

	$k = 1, n$		$k = 2, n$		$k = 0, y$		$k = 1, y$		$k = 2, y$	
LM order	wot	wt	wot	wt	wot	wt	wot	wt	wot	wt
0	81.4	–	90.1	–	56.3	–	80.9	–	90.8	–
1	38.2	–	42.5	–	33.5	–	39.1	–	42.1	–
2	32.3	25.1	36.6	27.2	23.6	18.3	27.3	18.2	29.2	20.9
3	31.6	24.9	35.6	26.0	22.9	18.0	25.8	17.8	27.7	19.8

- Unsurprisingly, using a larger language model is always rewarded, even though the difference between bigram and trigram is small; this is obvious when reading the table column-wise.
- Using an error model with $k = 2$ is always worse than using a more constrained model, no matter whether a phonemic route or training is used or not. This suggests that the resulting search space is too large to be efficiently evaluated by our current language model component.
- Training provides a very large gain for all our systems, between six and ten points.[9]
- The combination of an error model (with $k = 1$) and a phonemic model yields the best system, corresponding to a WER of 17.8%, only a half point worse than the MT system. While our previous work (Kobus *et al.* 2008a) found the MT system much better than our untrained ASR-based one, this result confirms the intuition that the difference between the two mainly comes from the optimization of the translation model. Adding this kind of optimization through training into the ASR-like system makes both systems more comparable.

These results are comforted by additional experiments conducted with large-scale resources (larger exception dictionaries and a larger language model). The corresponding performance are reported in Table 7.

These experiments show that using an automatically extracted exception dictionary has a very positive impact on performance, yielding an improvement of two to three points on the WER for our best systems. The performance then seem to plateau, as more exceptions are integrated into the system. Perhaps surprisingly, these results also tend to indicate that when larger exception dictionaries are available, the benefits of the correction model are less clear, since the best system for both the large and the huge exception dictionary does not integrate any correction model.[10]

our training corpus was much larger, containing approximately 33,000 messages instead of 24,000; and our test corpus was also somewhat simpler.

[9] These gains are probably overly optimistic, as the untrained systems may be suboptimal, because of the effect of the standardization operation (see Section 3.2.4). The per state normalization it yields has rather unpredictable effects on the derived probability distribution, hence the poor starting point it provides.

[10] The difference in deletion rates between both systems suggests that additional tuning might tighten the difference between both architectures.

Table 7. *The benefits of larger resources*
A system employing training (wt), a phonemic model (y), no correction model ($k = 0$), and a huge exception dictionary achieves a 14.2% WER. The numbers in boldface indicate that these WERs are statistically different from the Moses baseline at the 1% confidence level according to a matched-pairs test.

	Corr	Sub	Del	Ins	Err	S.Err
Small						
$k = 0$, y, wt	83.6	12.9	3.5	1.6	18.0	77.4
$k = 1$, y, wt	82.8	11.8	5.4	0.5	17.8	77.2
Large						
$k = 1$, n, wt	81.7	11.9	6.3	0.5	18.7	76.9
$k = 0$, y, wt	87.1	10.2	2.7	1.4	**14.3**	72.8
$k = 1$, y, wt	85.4	10.1	4.6	0.6	**15.3**	75.0
Huge						
$k = 0$, n, wt,	83.5	11.9	4.6	0.7	17.2	73.7
$k = 0$, y, wt	87.3	10.0	2.7	1.5	**14.2**	72.9
$k = 0$, y, wt	85.3	10.0	4.7	0.7	**15.4**	75.3
Huge + large LM						
$k = 0$, y, wt	86.4	10.3	3.3	0.9	**14.5**	74.8
$k = 1$, y, wt	83.9	10.9	5.2	0.5	16.6	77.4

This has the additional benefits of slightly reducing the size of P, thus accelerating the normalization procedure.

Finally, building larger language models does not seem to be of much help, probably because of the poor resemblance of the additional textual data we could use with real SMS messages.

5.2.3 *Issues solved, issues unresolved*

Figure 3 displays the distribution of the errors within the test set. As clearly reflected by this graph, the errors are very unevenly distributed: 10% of the messages contribute for about 40% of the errors, and 50% of the messages account of about 90% of the errors. At the other end of the scale, about one third of the test messages are completely error free.

The few messages that incur a large number of errors are mainly composed of a small number of long agglutinated segments, as in the following message:

als cva ê lordi2tpapa sinotatt fini,

Each unsegmented form in the output causes a large number of errors: here *lordi2tpapa*, which cryptically stands for *l' ordi de ton papa* (*the computer of your dad*), yields five errors and *sinotatt*, which stands for *sinon t' as tout* (*other than that, you 've got everything*), another four errors. One reason why these errors remain difficult to correct is our using an n-gram language model, which assigns a higher probability to shorter sentences. In such pathological cases, the length difference between the normalized message and the original SMS yields a difference

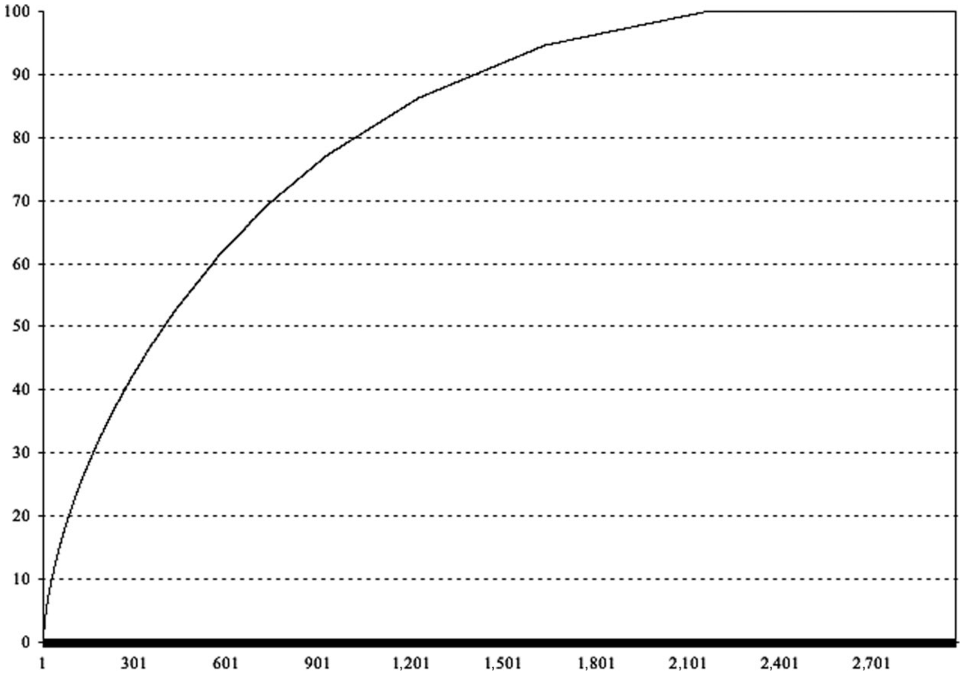

Fig. 3. Distribution of errors.

in likelihood that cannot be compensated by the other components of the system. It should be noted that the more common forms of blending are correctly handled: the compression ratio (measured in tokens) between the SMSs and their normalized counterparts is about 0.92; this ratio is close to 0.99 when the measurement is performed using our normalization hypotheses.

Apart from that, the most common sources of errors are due to short and very ambiguous tokens such as '*c*', which can stand for '*c' est*' (*it is*), '*sais*' (*I/you know*), or '*ce*' (*this*) (see Table 8). As all these forms are very common, the current language model is at pain deciding the right output. This is illustrated in the following example, where this letter stands for two different words:

SMS[11]	salut cmt va j **c** pa pq y fau envoyé 1 msg a **c** num m jle fè kan meme on vera b1
HYP	salut comment va je *c' est* pas pourquoi il faut envoyer 1 message à *c' est* numéro mais je le fais quand même on verra bien
REF	salut comment va je *sais* pas pourquoi il faut envoyer 1 message à *ce* numèro mais je le fais quand même on verra bien

[11] SMS denotes the original message, HYP the system's output, and REF the true reference. Incidentally, this example demonstrates the ability of the system to correctly decipher many words, as the *only* errors are those incurred by the letter '*c*'.

Table 8. *The twenty most common confusions of our best system*

Frequency	SMS	Normalized frequency	SMS	Normalized	
79	je	j’	70	d’	de
68	1	un	57	sais	est
52	bisou	bisous	50	des	de
47	biz	bizz	35	désolée	désolé
34	le	l’	33	a	à
32	as	a	31	fait	fais
28	je	ai	28	te	t’
27	c’	c	26	1	une
26	biz	bizzz	25	bon	bonne
25	s’	c’	25	vas	va

The same kind of confusion is observed with ‘*d*’, (*de*, *des*, *d’*), ‘*l*’ (*le*, *l’*, *elle*), and the like. This last example illustrates a quite unexpected ambiguity that occurs between elided and non-elided forms of articles or prepositions. Take the case of ‘*de*’ and ‘*d*’: in standard French, the latter stands for an elided form of the former, to be used whenever the following word starts with a vowel; their set of contexts are entirely disjoint. Yet, these kinds of confusions account for a significant number of errors: this is because an isolated ‘*d*’ is much more likely to be pronounced /d/ than /də/, yielding a score difference between both hypotheses that the language model fails to compensate.[12]

5.2.4 *What we can do, and what we cannot*

An interesting facet of our architecture is its *reversibility*. Further, G represents a weighted rational relationship over strings of letters and is thus bidirectional: it has been used here to transduce SMS into normalized message, but the reverse would also be possible. Using our existing resources, building a French-to-SMS generation system would simply amount to training a language model on the SMS side of the messages.

Our architecture also provides a mechanism for aligning on a per-character basis SMS messages and their normalized counterparts. Assume that both a message x and its normalization $n(x)$ are known; then an alignment can be recovered as bestpath($x \circ G \circ n(x)$), granted that the normalization $n(x)$ exists in $x \circ G$. This is basically the proposal developed in Beaufort, Roekhaut, and Fairon (2008), which however uses a manually designed transducer.

From the perspective of linguistic studies, a very useful information would be to project the weights learned during training back to the grapheme-to-phoneme rules, in order to quantitatively assess the contribution of each individual rule. Our current implementation of the EM algorithm does not allow direct retrieval of this information. As explained in Eisner (2002), it would be possible to perform this

[12] These errors would be fairly easy to correct using a simple rule-based postprocessing.

computation, by keeping track, for each transition in G, of the original transitions in the various subcomponents of G.

Finally, contrarily to the system presented in Kobus *et al.* (2008b), this architecture does not allow easy recovery of a phonetic transcription of messages, as would be required for SMS-to-speech applications.

6 Conclusion and perspectives

Main results In the current paper, we have proposed a generic software architecture for the SMS normalization task, on the basis of weighted finite-state transducers. Our architecture draws inspiration from several related NLP problems, notably the spell-checking problem and the speech-decoding problem, and accordingly reuses as much as possible existing tools or modules. Following our initial proposal (Kobus *et al.* 2008b), our normalization system consists of two main layers of weighted transduction: the first rewrites strings of characters to strings of characters, using an error model and a phonemic model; the second layer rewrites strings of characters to strings of words, whose likelihood is evaluated by a statistical language model. This architecture has several virtues: it allows for an efficient implementation of the complete system, using weighted finite-state transducers which can be optimized online. It also provides a *reversible* device, which can be used to generate compressed versions of regular texts.

As suggested in Kobus *et al.* (2008a), we found that optimizing through learning the first layer of transduction brings a large improvement over the non-optimized version. By comparison, the gain provided by the error model is modest and vanishes when large exception dictionaries are used. Increasing the size of the exception dictionary proves to bring limited gains, suggesting that adding more exceptions to the existing list is probably not worth the effort.

Following Aw *et al.* (2006), we also found that using off-the-shell statistical MT software allows us to achieve very satisfactory performance; this strategy nonetheless fails to match the performance of our best system, because of their lack of an adequate strategy for processing the very frequent unknown forms. A useful by-product of the SMT approach is the word-based alignment between SMS messages and their normalized counterparts, from which additional exceptions can be automatically extracted.

From a more general standpoint, the architecture developed for normalizing SMS texts is pretty general and could be used to handle many applicative contexts involving noisy inputs such as blogs, mails, optical character recognition (OCR) output. The integration of a training loop in the grapheme to grapheme layer makes it fairly trivial to adapt this architecture to novel situations. Another very practical lesson that has been learned in the course of the current work is the efficiency of the phrase-based statistical MT machinery as a general mechanism for learning linguistic mappings between variable length sequences, a quite common problem in NLP applications. Even if this might be considered an overkill for many tasks, in which the number of symbols and the order of symbols is preserved through transduction (e.g., part-of-speech tagging), these systems provide very good baselines and should probably be considered before trying anything fancier.

Perspectives. As it stands, our statistical normalization system achieves a WER that is probably good enough for SMS text mining purposes. It also provides a useful tool for quantitatively analyzing the various mechanisms involved in SMS spelling. The problem nonetheless remains far from being solved: our best system still makes at least one error on more than 70% of the test messages. Even if these errors are very unevenly distributed, this might be too much for high-quality SMS-to-speech applications.

There are a number of obvious useful improvements we might consider, such as using more accurate grapheme-to-phoneme rules, larger dictionaries, or a discriminative training procedure. Among this 'small' changes, the most desirable one concerns the language model, which, because of the relative paucity of the training corpus, is currently poorly estimated. If throwing in more training data would certainly help, it seems that the peculiarities of the genre/domain under study imposes the use of text corpora that match well our SMS messages. Such resources are not easily found. Our error analysis however indicates that a large number of residual errors stems from long unknown forms that corresponds to non-segmented text chunks. As explained earlier, we have, in our previous work, proposed and assessed an *ad hoc* two-pass strategy that was able to solve part of these issues. An alternative approach to this problem, more in line with the global architecture of our current system, would be to use a better unknown word model, as is sometimes done in open-vocabulary speech recognition (Bazzi and Glass 2000). In the system presented here, every unknown form incurs a fixed penalty; more complex models can easily be designed, which should assign distinct probabilities to unknown word based on their 'morphology', thereby duly penalizing these abnormally long forms.

Another line of improvement concerns the processing of case and punctuation information: at present, both informations are lost during the preprocessing stage. There are several reasons why we might like to reconsider this preprocessing strategy. Firstly, case and punctuation might be required for further preprocessing steps (e.g., SMS-to-voice applications): if this were the only reason, we could easily resort to existing punctuation and case restoration techniques (Lita *et al.* 2003). Secondly, and more importantly, there seems to be some information in the use of case and punctuation marks that should be taken into account. For instance, some writers of the SMS systematically use uppercasing to trigger the rebus reading of some letters. Likewise, modeling the distribution of punctuation signs might enhance the normalization of messages for which this information is available. This is especially crucial with this compound punctuation marks known as *smileys*, without which the intended meaning of some messages cannot be disambiguated.

Our roadmap finally includes the portability of this architecture to other languages. SMS corpora exist or are being collected in many languages,[13] and it might be of interest to adapt and evaluate our normalization system on these databases. As we

[13] See `http://www.sms4science.org`.

have shown in the current paper, this would only be a matter of having access to a set approximative pronunciation rules and to a pronunciation dictionary, two resources that are in fact available for many languages.

References

Allauzen, C., Mohri, M., and Roark, B. 2005. The design principles and algorithms of a weighted grammar library. *International Journal of Foundations of Computer Science* **16**(3): 403–21.

Anis, J. 2001. *Parlez-vous texto? Guide des nouveaux langages du réseau*. Paris, France: Éditions du Cherche Midi.

Aw, A., Zhang, M., Xiao, J., and Su, J. 2006. A phrase-based statistical model for SMS text normalization. In N. Calzolari, C. Cardie, and P. Isabelle (eds.) *Proceedings of COLING/Association for Computational Linguistics*, pp. 33–40. Sydney, Australia: the Association for Computational Linguistics.

Bazzi, I., and Glass, J. 2000. Modelling OOV words for robust speech recognition. In *Proceedings of the Internatinal Conference on Spoken Language Processing (ICSLP)*, pp. 401–4. Beijing, China: International Speech Communication Association (ISCA).

Beaufort, R., Roekhaut, S., and Fairon, C. 2008. Définition d'un système d'alignement SMS/français standard à l'aide d'un filtre de composition. In S. Heiden, and B. Pincemin (eds.) *Actes des Journées Internationales de l'Analyse des Données Textuelles (JADT)*, pp. 55–166. Lyon: Presses Universitaires de Lyon.

Béchet, F. 2001. LIA_PHON: Un système complet de phonétisation de textes. *Traitement Automatique des Langues* **42**(1): 47–67.

Boula de Mareüil, P., d'Alessandro, C., Yvon, F., Aubergé, V., Vaissière, J., and Amelot, A. 2000. A French phonetic lexicon with variants for speech and language processing. In *Proceedings of the 2nd Language Resources Engineering Conference (LREC)*, vol. I, pp. 273–6, Athens, Greece.

Brill, E., and Moore, R. C. 2000. An improved error model for noisy channel spelling correction. In C.-N. Huang, and K. Vijay-Shankar, K. (eds.) *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 286–93. Hong Kong: the Association for Computational Linguistics.

Brown, P. F., Cocke, J., Pietra, S. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. 1990. A statistical approach to machine translation. *Computational Linguistics* **16**(2): 79–85.

Chen, S. F., and Goodman, J. T. 1996. An empirical study of smoothing techniques for language modeling. In A. Joshi, and M. Palmer (eds.) *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 310–18. Santa Cruz, NM: the Association for Computational Linguistics.

Choudhury, M., Saraf, R., Jain, V., Sarkar, S., and Basu, A. 2007. Investigation and modeling of the structure of texting language. In C. Knoblock, D. Lopresti, S. Roy, and L. Venkata Subramaniam (eds.) *Proceedings of the IJCAI Workshop on 'Analytics for Noisy Unstructured Text Data'*, pp. 63–70. Hyderabad, India: International Association for Pattern Recognition.

Church, K. W., and Gale, W. 1991. Probability scoring for spelling correction. *Statistics and Computing* **1**: 91–103.

Clark, A. 2003. Pre-processing very noisy text. In *Proceedings of Workshop on Shallow Processing of Large Corpora*, Lancaster, UK.

Crystal, D. 2001. *Language and the Internet*. Cambridge, UK: Cambridge University Press.

Divay, M., and Vitale, A. J. 1997. Algorithm for grapheme-to-phoneme translation for French and English: applications. *Computational Linguistics* **23**(4): 495–524.

Eisner, J. 2002. Parameter estimation for probabilistic finite-state transducers. In E. Charniak, and D. Lin (eds.) *Proceedings of the 40th Annual Meeting of the Association for*

Computational Linguistics, pp. 1–8. Philadelphia, PA: the Association for Computational Linguistics.

Fairon, C., Klein, J. R., and Paumier, S. 2006. *Le langage SMS*. Louvain, Belgium: UCL Presses.

Fairon, C., and Paumier, S. 2006. A translated corpus of 30,000 French SMS. In *Proceedings of LREC 2006*, Genoa, Italy.

Gillick, L., and Cox, S. 1989. Some statistical issues in the comparison of speech recognition algorithm. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 532–35, Glasgow, UK.

Golding, A. R., and Roth, D. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine Learning* **34**: 107–30.

Golding, A. R., and Schabes, Y. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In A. Joshi, and M. Palmer (eds.) *the Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 71–8. Santa Cruz, CA: the Association for Computational Linguistics.

Guimier de Neef, E., Debeurme, A., and Park, J. 2007. TILT correcteur de SMS: évaluation et bilan quantitatif. In *Actes de la Conférence sur le Traitement Automatique des Langues (TALN'07)*, pp. 123–32, Toulouse, France.

Jansche, M. 2003. *Inference of string mappings for language technology*. PhD thesis, Ohio State University.

Jelinek, F. 1990. Self-organized language modeling for speech recognition. In Waibel, A., and Lee, K.-F. (eds.), *Readings in Speech Recognition*, pp. 450–506, San Mateo, CA: Morgan-Kaufman.

Kaplan, R. M., and Kay, M. 1994. Regular models of phonological rule systems. *Computational Linguistics* **20**(3): 331–78. (First appeared as a paper presented to the Winter Meeting of the Linguistic Society of America, New York, 1981.).

Kobus, C., Yvon, F., and Damnati, G. 2008a. Normalizing SMS: are two metaphors better than one? In D. Scott, and H. Uszkoreit (eds.) *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pp. 441–8. Manchester, UK: the Association for Computational Linguistics.

Kobus, C., Yvon, F., and Damnati, G. 2008b. Transcrire les SMS comme on reconnaît la parole. In *Actes de la Conférence sur le Traitement Automatique des Langues (TALN'08)*, pp. 128–38, Avignon, France.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Demonstration Session*, Prague, Czech Republic.

Koehn, P., Och, F. J., and Marcu, D. 2003. Statistical phrase-based translation. In M. Hearst, and M. Ostendorf (eds.) *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistic*, pp. 127–33. Edmondton, AB, Canada: the Association for Computational Linguistics.

Kukich, K. 1992. Techniques for automatically correcting words in text. *Computing Surveys* **24**(4): 377–439.

Lita, L. V., Ittycheriah, A., Roukos, S., and Kambhatla, N. 2003. tRuEcasIng. In E. W. Hinrichs, and D. Roth (eds.) *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pp. 152–9. Sapporo, Japan: the Association for Computational Linguistics.

Mikheev, A. 2000. Document centered approach to text normalization. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 136–43. New York: Association for Computing Machinery.

Mitton, R. 1996. Spellchecking by computer. *Journal of the Simplified Spelling Society* **20**(1): 4–11.

Mohri, M. 1997a. On the use of sequential transducers in natural language processing. In Roche, E., and Schabes, Y. (eds.), *Finite State Natural Language Processing*, Cambridge, MA: MIT Press.

Mohri, M. 1997b. Transducers in language and speech. *Computational Linguistics* **23**(2): 269–311.

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

Mohri, M., and Sproat, R. W. 1996. An efficient compiler for weighted rewrite rules. In A. Joshi, and M. Palmer (eds.) *Proceedings of the annual Meeting of the Association for Computational Linguistics*, pp. 231–8. Santa Cruz, CA: the Association for Computational Linguistics.

Och, F. J. 2003. Minimum error rate training in statistical machine translation. In E. W. Hinrichs, and D. Roth (eds.) *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 160–7. Sapporo, Japan: the Association for Computational Linguistics.

Och, F. J., and Ney, H. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* **29**(1): 19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. 2001. Bleu: a method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center.

Roche, E., and Schabes, Y. 1997. Introduction to finite-state devices in natural language processing. In Roche, E., and Schabes, Y. (eds.), *Finite State Natural Language Processing*, pp. 1–66, Cambridge, MA: MIT Press.

Simard, M., and Deslauriers, A. 2001. Real-time automatic insertion of accents in French text. *Journal of Natural Language Engineering* **7**(2): 143–65.

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

Stolcke, A. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Langage Processing (ICSLP),* vol. 2, pp. 901–4, Denver, CO.

Toutanova, K., and Moore, R. 2002. Pronunciation modeling for improved spelling correction. In E. Charniak, and D. Lin (eds.) *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 144–51. Philadelphia, PA: the Association for Computational Linguistics.

Véronis, J., and Guimier de Neef, E. 2006. Le traitement des nouvelles formes de communication écrite. In Sabah, G. (ed.), *Compréhension automatique des langues et interaction*, pp. 227–48. Paris: Hermès Science.

Yvon, F., de Mareüil, P. B., d'Alessandro, C., Aubergé, V., Bagein, M., Bailly, G., Béchet, F., Foukia, S., Goldman, J.-P., Keller, E., O'Shaughnessy, D., Pagel, V., Sannier, F., Véronis, J., and Zellner, B. 1998. Objective evaluation of grapheme-to-phoneme conversion for text-to-speech synthesis in French. *Computer, Speech and Language* **12**(4): 393–410.

Zhu, C., Tang, J., Li, H., Ng, H. T., and Zhao, T. 2007. A unified tagging approach to text normalization. In A. Zaenen, and A. van den Bosch (eds.) *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 688–95. Prague, Czech Republic: the Association for Computational Linguistics.