IJDAR (2007) 10:157–174
DOI 10.1007/s10032-007-0054-0

ORIGINAL PAPER

Investigation and modeling of the structure of texting language

Monojit Choudhury · Rahul Saraf · Vijit Jain · Animesh Mukherjee · Sudeshna Sarkar · Anupam Basu

Received: 20 March 2007 / Revised: 13 July 2007 / Accepted: 10 September 2007 / Published online: 24 October 2007

Abstract Language usage over computer mediated discourses, such as chats, emails and SMS texts, significantly differs from the standard form of the language and is referred to as texting language (TL). The presence of intentional misspellings significantly decrease the accuracy of existing spell checking techniques for TL words. In this work, we formally investigate the nature and type of compressions used in SMS texts, and develop a Hidden Markov Model based word-model for TL. The model parameters have been estimated through standard machine learning techniques from a word-aligned SMS and standard English parallel corpus. The accuracy of the model in correcting TL words is 57.7%, which is almost a threefold improvement over the performance of Aspell. The use of simple bigram language model results in a 35% reduction of the relative word level error rates.

M. Choudhury (✉) · A. Mukherjee · S. Sarkar · A. Basu
Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur, India
e-mail: monojit@cse.iitkgp.ernet.in

R. Saraf
Department of Computer Engineering,
Malaviya National Institute of Technology, Jaipur, India
e-mail: rahul.shillong@gmail.com

V. Jain
D.E. Shaw India Software Private Ltd,
Hyderabad, India
e-mail: jain.vijit@gmail.com

A. Mukherjee
e-mail: animeshm@cse.iitkgp.ernet.in

S. Sarkar
e-mail: sudeshna@cse.iitkgp.ernet.in

A. Basu
e-mail: anupam@cse.iitkgp.ernet.in

Keywords Texting language · SMS · Hidden Markov Model · Text correction · Spell checking

1 Introduction

While communicating over texting media, such as chats, emails and short messages over mobile phones (SMS), users have a tendency to use a non-standard form of the language that disregards grammar, punctuation and spelling rules. In order to type faster, especially while using a small keyboard like that of the mobile phones, users employ a large number of compression techniques to reduce the message length. Commonly used abbreviations, shorter phonetic substitutions, deletion of words and characters, are some of the popular methods for shortening the message length. Nevertheless, the characters and words cannot be deleted arbitrarily, as it may seriously hamper the understandability of the message. Thus, two opposing forces—shorter message length and semantic unambiguity—shape the structure of this compressed non-standard form, called the *NetSpeak* [16] or the *Texting Language* (TL) [46].

The objective of the current work is to study and formally model the characteristics of TL, and based on the compression model so obtained, construct a decoder from TL to the standard language. The decoder is expected to transform a sentence in TL, such as "*btw wenz ur flt 2moro*" to its standard English counterpart, i.e., "*By the way, when is your flight tomorrow?*" However, here we focus on the word level analysis and modeling of TL, which forms the first step towards a sentence level decoder. Compressions at the level of syntax or those grounded in semantic or pragmatic knowledge are beyond the scope of the current work.

We model the problem as a *noisy channel* process; we assume that the standard word is compressed or distorted to

the TL form, while being transmitted over the noisy channel. Hidden Markov Models (HMM) [39] have been used to characterize the stochastic properties of the channel. The conception and construction of the HMM is a novel as well as a significant contribution of the present work. In order to learn the characteristics of the noisy channel (or equivalently the TL), we have gathered 1,000 English SMS texts from http://www.treasuremytext.com. The website hosts a large number of SMS texts in several languages uploaded by anonymous donors. The SMS texts were manually translated to their standard English form and automatically aligned at the word level using a heuristic algorithm. The resulting 20,000 word parallel-corpus has been inspected to formulate the structure of the word level HMM. The HMM parameters are estimated from the training corpus using standard machine learning techniques. Even though only English SMS texts have been used here as the primary TL data, it is representative of the TL used over other computer mediated discourses as well [16,22].

A decoder from TL to the standard language has several practical applications including search engines and automatic correction tools for noisy text documents such as blogs, chatlogs and emails. Non-standard spellings and grammatical usage is very common over the web. Therefore, a search engine for noisy texts (say blogs, chatlogs or emails) must be immune to the several TL variants of a word. For example, given a query "translator", a search engine is expected to also return the webpages with words like "transl8or", "trnsl8r", "trnsltr", etc. This can be readily achieved through a word level decoder from the TL to standard language. Similarly, owing to a large number of spelling variations, search for proper nouns also calls for appropriate normalization. For instance, we find several variations for the name "Saurav Ganguly" in the web ("Saurabh Ganguly", "Sourav Ganguli", "Sourabh Ganguly", etc.). The proposed model for TL can also efficiently capture the variations in proper nouns. Yet another application of this technique could be in text-to-speech systems that read out webpages, emails and blogs to the visually challenged.

Apart from the aforementioned practical applications, the present study is also interesting from the perspective of diachronic linguistics. It has been claimed by several researchers in the past that languages change over time to optimize its function, i.e., communication (see, for example, [10,11,15,28] and references therein). Such accounts of language change are called *functional explanations* (see [11] for review), where optimal communication is often equated to the two opposing objectives of shorter message length (or ease of message generation) and robustness to noise (maximally distinct messages). In the field of diachronic linguistics, where evidence is rare and experimentation is impossible [1,30], the structure of TL provides an ideal case for studying language change in the functionalist framework [6,8,41]. As we shall see here, in TL shorter message lengths are achieved through a variety of ingenuous compression techniques.

The rest of the paper is organized as follows: Sect. 2 presents a brief overview of the linguistic and computational studies regarding the language usage over computer mediated discourses and automatic correction of the same. Sections 3 and 4 describe a noisy channel formulation of the problem and the creation of training data respectively. Section 5 discusses the compression characteristics of the TL, as has been observed from the corpus. On the basis of the observations, in Sect. 6 we propose an HMM-based representation of the words. Section 7 discusses automatic construction of the HMMs as well as the learning algorithm for estimation and generalization of the HMM parameters. The evaluation results of the model is presented in Sect. 8. Section 9 concludes the paper by summarizing the contributions and listing out possible improvement schemes.

2 Related work

The structure of the TL and its socio-cultural effects have been a subject matter of diachronic as well as socio-linguistic studies for the past two decades (see [16,22,41] and references therein). Some of the researchers, e.g. [19,32], have suggested that computer-mediated communication, particularly on-line, synchronous communication, challenges the generally assumed dichotomy between written and oral language. The orthograhic and syntactic structures used in TL have been extensively studied, not only for English [6–8,22], but also for several other languages, such as Arabic [37], German [2,18], Japanese [34] and Swedish [41].

2.1 Studies on SMS

Short messages over mobile phones or SMS are arguably the most distorted form of TL [18,41]. The misspelling in SMS texts are, in most of the cases, intentional, because, the typical mobile-phone keypad, that has only nine keys, requires pressing of a key multiple times for entering a character. For example, the characters 's' and 'z' require four key presses. The small display and the bounded message length (usually 160 characters) create further pressure towards reduction of message length.

The common trends of compression and/or distortion observed in SMS texts, as discussed in [41] for Swedish, are summarized below.

- **Spelling and punctuation**:
  - Deletion of space between words
  - Splitting of compound words
  - Deletion of punctuation marks

  - Use of all lower or all upper case
  - Phonetic substitutions for letter sequences or words
  - Typos
  - Deletion of vowels
- **Grammatical features**:
  - Deletion of pronouns (especially subject)
  - Deletion of auxilaries in verb phrases
  - Substitution of shorter words
  - Deletion of preposition, possessive pronouns
- **Abbreviations and other features**:
  - Conventional and unconventional abbreviations
  - Features from spoken languages
  - Use of dialectal forms and slangs
  - Code switching
  - Use of emoticons

2.2 Computational models

Despite the applications and aforementioned linguistic studies, there are very few computational models built exclusively for decoding or analyzing TL (see for example [4,5,40,42,47]) . Bangalore et al. [5] trained a hierarchical statistical translation model for general domain instant messaging language collected from Internet chat rooms. However, as discussed above, the structure of the chat language is very different from that of the SMS. More recently, Aw et al. [4] developed an English SMS text normalization technique based on standard statistical machine translation models. They formulate the problem of normalization as machine translation from TL to SL, train a phrase-based MT system on a parallel data of 5,000 SMS messages, and report a significant improvement over the baseline model in terms of BLEU score. Some basic analysis of the TL is also reported. Nevertheless, the method performs poorly for out-of-vocabulary (OOV) TL words, because no technique has been used to estimate the probability of an unseen token based on character level or pronunciation level similarities. The aim of the present work is to develop a very powerful word-model which can efficiently deal with unseen TL tokens and thereby, to address some of the limitations of the technique described in [4]. In other words, the current work is complementary in nature to [4], because while the latter tries to deal with the issues at the sentence level, the former focuses on word level compressions.

The problem of normalizing the TL words to their standard counterpart is in many ways similar to spelling correction. Spell checking is a well researched area in NLP and there are several approaches to spelling error detection and correction (see [26] for a dated, but relevant and extensive survey). Spelling errors are usually classified as *typographic* and *cognitive* errors. Typographic errors occur due to slip of finger (pressing a wrong key unknowingly), whereas cognitive errors arise when the user does not know the correct spelling and usually substitute the letters by their phonetic equivalents. For example, the word "latex" might be misspelt as "layex"—a typograhic error, or "latecks"—a cognitive error.

It has been documented that around 80% of the unintentional misspellings deviate from their correct spellings by at most one insertion, deletion, substitution or transposition [17]. Therefore, majority of the spell checking literature is devoted to small *edit-distance* [27] based error correction (see, e.g. [25,29,31]). Nevertheless, as is evident from the "latex"—"latecks" example, cognitive errors can significantly increase the edit-distance between the misspelling and the correct word.

Several techniques have been proposed to incorporate phonetic information in the spell checkers, of which the techniques based on metaphone encoding [35,36,38] and noisy channel models [12,20,44] are the most popular ones. In the metaphone encoding or similarity key based techniques, the letters or letter sequences in a word are substituted by metaphones, and the the correct spelling is searched for in the space of strings over metaphones, rather than the letters. Although these techniques perform better than nearest edit-distance based techniques, noisy channel models that take into account the pronunciation information are known to substantially boost up the performance of the spell checkers [44].

2.3 Pronunciation modeling for spell checking

Statistical spelling correction techniques can be viewed as instances of noisy channel model, where a word w while being transmitted over the channel gets transformed to w'. A spelling error refers to a situation when $w \neq w'$. The problem of finding the intended spelling w, given the observed spelling w' is equivalent to searching for w in the lexicon, for which

$$Pr(w|w') = \frac{Pr(w'|w)Pr(w)}{Pr(w')}$$

is maximized. Since $Pr(w')$ is independent of w, the problem boils down to maximizing $Pr(w'|w)Pr(w)$, where the distribution $Pr(w'|w)$ models the characteristics of the noisy channel (also known as the error model), whereas $Pr(w)$ is known as the language model.

Brill and Moore [12] proposed an improved error model for spelling correction, that takes into account substitution of a string of characters of arbitrary length (upto 5) by another string of arbitrary length. The model automatically learns the probability of substitutions $Pr(\alpha \rightarrow \beta|PSN)$ from a corpus of words and their misspellings. In this expression, α is a sequence in w, β the corresponding sequence in w' and PSN refers to the position of the string α in w. Given a partition of w, $\alpha_1\alpha_2, \ldots, \alpha_k$, and a corresponding partition for w',

$\beta_1\beta_2, \ldots, \beta_k$, the probability that w' has been generated from w is defined as

$$\prod_{i=1}^{k} Pr(\alpha_i \rightarrow \beta_i | PSN_i)$$

Finally, $Pr(w'|w)$ is estimated as the maximum over all partitions of w of the probability that w' is generated from w given that partition.

Toutanova and Moore [44] extended the Brill and Moore model by combining the pronunciation information. Let $Pr_{\text{LTR}}(w'|w)$ be the probability that w is mapped to w' by substitution of letter sequences as proposed in [12], and $Pr_{\text{PHL}}(w'|w)$ be the probability that w' has been generated by substituting the pronunciation (sequence of phones) of w by letter sequences. Then, according to [44], the scores of a word w are estimated using the following equation (higher the $Pr(w'|w)$, higher is the score $S(w'|w)$)

$$S(w'|w) = \log Pr_{\text{LTR}}(w'|w) + \lambda \log Pr_{\text{PHL}}(w'|w) \quad (1)$$

The quantity $Pr_{\text{PHL}}(w'|w)$ is estimated as follows:

$$Pr_{\text{PHL}}(w'|w) = \sum_{\widetilde{w}} \frac{1}{N} \max(Pr(\widetilde{w}|w) Pr_{PH}(\widetilde{w}'|\widetilde{w})) \quad (2)$$

N is the number of possible pronunciations of w, $\widetilde{w}$ and $\widetilde{w}'$ represent pronunciations of w and w', respectively, and $Pr_{PH}(x|y)$ is the probability of the phone sequence x being generated from the phone sequence y.

In the Toutanova and Moore model the probability distributions have been estimated from word/misspelling and word/pronunciation pairs. The pronunciation of the unseen words (misspellings) are generated stochastically using the pronunciation model, which is learnt from the word/pronunciation pairs. The pronunciation model was trained on around 100,000 pairs and the letter model was trained on approximately 7,500 pairs. The resulting system has been reported to significantly reduce the error rate over the Brill and Moore model.

Since words in TL feature a large number of phonetic substitutions, pronunciation modeling becomes extremely important in the perspective of the current work. However, for the following reasons, it seems inappropriate to directly employ the aforementioned models for the correction of TL.

- The Toutanova and Moore model assumes that w' is generated from w either through graphemic substitutions or through phonetic substitutions, and Eq. 1 is a log–linear combination of the probabilities of these two hypotheses. However, as we shall see, it is quite possible that parts of w are substituted phonetically, whereas other parts are substituted orthographically. These two types of stretches of w may be arbitrarily interleaved. Identification of the parts help us have a better estimate of $Pr(w'|w)$ than a simplified log–linear combination.
- There is insufficient data for TL that can be used to train these models.

In the subsequent sections, we propose a novel approach to combine the orthographic and phonetic information of a word using HMMs.

3 Noisy channel formulation

Let $S = s_1\ s_2, \ldots, s_l$ be a sentence in the standard form, where s_i represents the tokens including words, punctuations and abbreviations. When S is transmitted through the noisy channel, it is converted to $T = t_1\ t_2, \ldots, t_l$, where T is the corresponding sentence in TL and t_i is the transmitted form for the token s_i. The noisy channel is characterized by the conditional probability $Pr(T|S)$. In order to decode T to the corresponding standard form $\delta(T)$, where δ is the decoder model, we need to determine $Pr(S|T)$, which can be stated as follows:

$$\delta(T) = \underset{S}{\text{argmax}}\ Pr(T|S) Pr(S) \quad (3)$$

Under the assumption that the compressed form t_i of s_i depends only on s_i, and not the context, we obtain the following relation,

$$\delta(T) = \underset{S}{\text{argmax}} \left[\prod_{i=1}^{l} Pr(t_i|s_i) \right] Pr(S) \quad (4)$$

Thus, the decoding model $\delta(\cdot)$ for a TL sentence can be abstracted out into two distinct levels: (1) the *word error model*, $Pr(t|s)$ and (2) the *language model*, $Pr(S)$. Although, we report some initial experimentations with n-gram language models, the focus of the present work is the word error model. Note that the basic formulation of the word error model $Pr(t|s)$ is identical to the noisy channel model of spell checking discussed earlier, where t and s has been denoted by w' and w respectively.

At this point, it might be worthwhile to inspect the validity of the aforementioned assumption made regarding word-to-word transmission. Although it is quite reasonable to assume that for TL the compression applied on a word is independent of its context, a significant drawback of the assumption lies in the fact that often words are deleted or combined during typing (deletion of space or splitting of compounds as reported in [41]). For instance, in the source-TL sentence pair cited in Sect. 1, if we assume that a word for word translation has taken place, then the corresponding alignment is (ϵ stands for the null symbol, i.e. the deletion of a token)

	S	T
1	By	b
2	the	t
3	way	w
4	,	ϵ
5	$when$	wen
6	is	z
7	$your$	ur
8	$flight$	flt
9	$tomorrow$	$2moro$
10	?	ϵ

However, in this case T will not reflect the tokenization shown in the above alignment, since the delimiter (*white space*) is missing between the tokens 1 and 2, 2 and 3, and 5 and 6. To solve this issue, a more general formalism such as the IBM Models of translation [13] could have been used, where phrases can be aligned to phrases. However, that would be an overkill, because manual inspection of the TL data reveals that intentional deletion of space takes place only under two specific situations: (1) for commonly used abbreviations and (2) for certain grammatical categories like the auxiliaries ("when is" to "wenz" or "how are" to "howz") and the negative marker ("are not" to "aint" or "would not" to "wudnt").

Therefore, we believe that a word for word decoding scheme along with a dictionary of commonly used abbreviations and rules for handling the space omission for specific grammatical classes can achieve a reasonably good accuracy for sentence level translations. As described in [4], one might also resort to standard machine translation techniques for solving these issues. The scope of this work, however, is restricted to the design of a word level decoder that is motivated by the cognitive process behind the text compressions. We note that the use of acronyms, ellipses, and other syntactic and pragmatic compressions present interesting research challenges to SMS text normalization, which we intend to address in the future as an extension to this work.

4 Creation of training data

The estimation of the error model $Pr(t|s)$ demands the information about the variations of a word s over the texting medium and their probabilities. In order to gather realistic TL data we created a 20,000 word aligned corpus of SMS texts and standard English. The process of data collection, translation, cleaning and alignment are described below.

4.1 Data collection

There are a few websites that host a good amount of medium specific TL data. We chose to work with SMS data, because apparently it features the maximum amount of compression and thus, is the most challenging one to model.

The website http://www.treasuremytext.com hosts a huge number of SMS texts in several languages, which have been donated by anonymous users. The messages are not indexed by languages and many of them feature code switching. A large number of messages were downloaded from the site, from which around 1,000 English SMS texts were collected such that

(a) The texts are written only in English, and contain no word written in any other language.
(b) There is at least one misspelling in the message. This criterion is important because often messages are typed using the T9 mode[1] and therefore, does not reflect the TL pattern. In other words, it would be misleading to consider the texts entered using T9 mode as instances of TL.

The SMS texts were manually translated to their standard English forms. The proper nouns (names of people, places, etc.) have not been translated, but replaced by a tag $<NAME>$ in both the TL and standard data. There are around 20,000 tokens (words) in the translated standard text out of which around 2,000 are distinct. On an average there are 83 characters in the SMS text for every 100 characters in the corresponding standard English text.

4.2 Automatic alignment

The translated corpus has been automatically aligned at the word level using a heuristic algorithm. The algorithm searches for sites (tokens) in the original and translated texts, which can be aligned to each other with *high confidence*. We define these sites as *pivots*. The unaligned words between two consecutive pivots are then recursively aligned by searching for more pivots between them. During each recursive call the conditions for alignment are further relaxed. A sketch of the algorithm is provided below.

1. **Input:** SMS text $T = t_1\ t_2, \ldots, t_k$, translated standard English text $S = s_1\ s_2, \ldots, s_l$, where t_i and s_j are tokens delimited by white spaces.
2. We introduce two hypothetical start symbols t_0 and s_0 at the beginning of T and S, and two similar symbols t_{end} and s_{end} to mark the end of the sequences, respectively. t_0 is aligned to s_0 and t_{end} to s_{end}, which forms the two initial *pivots* at the beginning and the end of the texts.

[1] T9 stands for *Text on nine Keys*, which is a predictive texting method for mobile phones. The technology allows words to be entered by a single key press for each letter, as opposed to the standard method, in which selecting one letter often requires multiple key presses.

3. Each t_i is compared with $s_{i'}$ and other tokens around $s_{i'}$, where the value of i' is computed on the basis of i, l and k. If $t_i = s_j$ for some j within the local window around i', s_j is declared as the translation of t_i and this alignment gives rise to a new pivot. In case an exact match is not found, the alignment of t_i is defered.
4. Suppose there are $p + 2$ pivots including the two boundary pivots. These pivots segment T and S into $p + 1$ non-overlapping regions, where the tokens in a particular region of T align to the tokens in the corresponding region of S.
5. If a region has 0 or 1 token in either of T or S, they are trivially aligned.
6. If both the regions have more than one token, the character level similarity between all pairs (t_i, s_j) of tokens within a particular region are computed using the Soundex algorithm [35,36].
7. The best matches are aligned and declared as new pivots, and the process continues recursively, until all the tokens are aligned.

The accuracy of the alignment algorithm is around 80%.

There are two main assumptions behind the alignment algorithm; first, the ordering of words in TL and SL are identical, and second, there are tokens that can be aligned as pivots with high confidence during the initial few calls of the procedure. In fact, inspection of the TL data reveals that both of these assumptions hold good. The first one regarding the ordering of words is clearly true, because while typing over the texting medium, the sentence of SL is compressed rather than garbled. The second assumption regarding existence of undistorted or predictably distorted words holds good because, usually the frequently used words show very little variations and therefore, they can be readily aligned as pivots with high confidence. By the virtue of their high frequency, one or more of these words are present in almost all the SMSes. Table 1 shows the five most frequent tokens of the SL and their corresponding variations. It is evident from the table that a pattern such as "u" in TL can be aligned with "you" in SL with a high confidence. Some of the other frequent words, for which we have not observed any distortion are (frequencies are provided in parentheses): in (169), so (157), on (159), it (143), if (138).

Table 1 Variations of the five most frequent tokens of SL found in the corpus

Word	Freq.	Variations
I	660	i (660)
to	538	to (448), 2 (81), t (9)
you	523	u (469), you (54)
me	377	me (363), mi (14)
for	227	for (153), 4 (62), fr (12)

Note that there are several statistical alignment algorithms that are used in machine translation for word or phrase level alignment of parallel data (see [21] and references therein for examples of state-of-the-art alignment algorithms). These algorithms assume that the order of the words/phrases between the source and target languages can be different, which is seldom the case for TL. Moreover, the rich information in the form of orthographic similarity between the corresponding SL and TL tokens are not exploited by most of the standard alignment algorithms. In [4], the authors report a statistical alignment algorithm for SMS texts based on expectation maximization (EM), where they try to overcome both of the aforementioned limitations. The search space is restricted to monotone alignments (i.e., the phrases are assumed to be in the same order in SL and TL), whereas the initial alignments are generated based on othographic similarity between the words. It would be interesting to compare the performance of these standard algorithms on the TL data to that of the heuristic approach suggested here that makes maximal use of the orthographic and phonetic information to circumvent problems arising from data scarcity.

4.3 Extraction of word-variant pairs

From the aligned corpus, a list of the unique English words and their corresponding variations in the SMS texts were extracted along with the occurrence frequencies. The list was manually cleaned to remove noise introduced due to error in alignment. Out of the 2,000 distinct English words, only 234 occur at least ten times in the corpus, and thus, can provide useful information about the nature of the TL; the other words being quite infrequent cannot be used for statistical learning purpose.

The total number of variants corresponding to these 234 frequent words is 702. This extracted list of 234 words $\{w_1, w_2, \ldots, w_{234}\}$, and their corresponding variation characteristics $\{\langle v_1^i, f_1^i\rangle, \langle v_2^i, f_2^i\rangle \ldots \langle v_{m_i}^i, f_{m_i}^i\rangle\}$, where v_j^i and f_j^i denote the jth variation and its frequency for the word w_i respectively, constitutes our word level training corpus. Table 2 shows the variation characteristics of the words "back", "today" and "thanks". The largest number of variations has been observed for the word "tomorrow", which are listed in Table 3. The word aligned parallel corpus as well as the manually cleaned training data are available at [14].

5 Variation characteristics

The distortions observed in the SMS text can be broadly classified into two groups—*intentional* and *unintentional* errors.

Table 2 The variations of the words with their occurrence frequencies

back(136)	today (60)	thanks (26)
bk (62)	today (46)	thanks (12)
bac (31)	2day (7)	thanx (9)
back (27)	to (3)	thnx (1)
bak (13)	tday (3)	thx (1)
bec (1)	day (1)	tx (1)
bc (1)		fanx (1)
bakk (1)		thks (1)

Table 3 The variations of the word "tomorrow" with their occurrence frequencies

tomoz (25)	tomora (4)	2moro (9)	tom (2)
tomorow (3)	tmorro (1)	tomrw (5)	2mro (2)
tomoro (12)	morrow (1)	tomo (3)	tomm (1)
tomorrow (24)	tomra (2)	tomor (2)	moro (1)

Table 4 Examples of unintentional errors. Occurrence frequencies are shown in parentheses

Word	Misspelling	Type
love (129)	loe (1)	3
got (95)	git (1)	1
out (88)	ott (1)	2
work (48)	werk (1)	4
message (18)	massage (1)	4

The unintentional errors are caused by (1) pressing of the wrong key, (2) pressing of a key more than the desired number of times, (3) deletion of a character or (4) inadequate knowledge of spelling. Table 4 shows some of the unintentional errors in the SMS corpus. Given an SMS text, it is quite difficult to predict which of the errors are unintentional and which are intentional. This is because, the text is usually highly distorted. Nevertheless, a few thumb rules for detection of unintentional errors are as follows.

- The misspelling is incomprehensible (assuming that intended misspellings do not hamper understandability). E.g. "loe" for "love"
- The misspelling uses a variant that does not minimize the typing effort, but at the same time hampers understandability. E.g. "werk" for "work".
- The misspelling is an extremely rare variant. This is a commonly made and well established assumption in the spell checking literature [17].

Unlike unintentional errors, intentional errors are quite frequently observed in the SMS text. They can be classified into four categories: *character deletion*, *phonetic substitution*, *abbreviations* and non-standard usage. We describe underneath each of these categories in details.

5.1 Character deletion

There are three major types of intentional deletions

- **Vowel deletion:** Deletion of vowels from the words usually do not hamper the understandability of the word. Examples are "tlk" (9) for "talk" (37), "bk" (62) for "back" (136), "gd" (13) for "good" (74), and "msg" (12) for "message" (18). However, word initial vowels are immune to deletion, such as "abt" (5) for "about" (68).
- **Repeated consonant deletion:** Examples are "tomoro" (12) for "tomorrow" (95), "al" (5) for "all" (72), and "chtin" (2) for "chatting" (6).
- **Truncation of words:** Examples are "tomm" (1), "tom" (2) or "tomo" (3) for "tomorrow" (95), "mob" (9) for "mobile" (18), and "morn" (1) for "morning" (15),
  Among the above, vowel deletion accounts for majority of the distortions.

5.2 Phonetic substitution

Phonetic substitutions are the next common compression style, after vowel deletion. The various types of phonetic substitutions are enumerated below.

- **Substitution of phonetically equivalent vowels/ consonants:** Examples are "fone" (65) for "phone" (36), "nite" (27) for "night" (49), and "thanx" (9) for "thanks" (26)
- **Deletion of silent characters:** Examples are "no" (27) for "know" (78), "bk" (62) for "back" (136). Deletion of h is also a commonly observed phenomenon, e.g. "ave" (14) for "have" (146), and "im" (2) for "him" (22)
- **Substitution of single letters for sequences of phones:** Examples are "v" (7) for "we" (37), "lyk" (7) for "like" (58), and "u" (469) for "you" (523).
- **Substitution of numerals for phone sequences:** Examples are "2moro" (9) for "tomorrow" (95), "m8" (5) for "mate" (16), and "4" (62) for "for" (227).

5.3 Abbreviations

Abbreviations used in TL can be classified as standard and non-standard abbreviations. The standard abbreviations are those, which are understood by all the TL users. Examples include "tb" for "text back", "lol" for "laughs out loud", "bday" (7) for "birthday" (15), and "btw" for "by the way". The non-standard abbreviations, on the other hand, are known

and used within a community. For example, "kgp" for "Kharagpur" or "cnt" for "complex network". It is extremely difficult, even for human translators, to identify and decode the non-standard abbreviations.

5.4 Other non-standard usage

Several other non-standard forms are used in TL, some of which are discussed below.

- **Dialect variations and spoken forms:** Examples are "wanna" (35) for "want to", "aint" (25) for "are not", "did not" or "have not", and "sis" (3) for "sister" (6)
- **Alternative pronunciations:** Examples are "betta" (4) for "better" (17), and "propa" (3) for "proper" (4).
- **Character substitution:** "f" or "v" is used instead of "th" as in "fink" (7) for "think" (34), "sumfin" (3) for "something" (30), and "wiv" (7) for "with" (89)
- **Miscellaneous:** "b/c" (1) for "because" (72)

Often more than one of these compression techniques are used over a single word to achieve maximal reduction in length. The token "2mro", for example, is obtained from "tomorrow" through phonetic substitution as well as character deletion. Similarly, "cz" (3) has been obtained from "because" through the following steps:

- "because" → "cause" (non standard usage, transformation takes place in 62 out of 75 cases)
- "cause" → "coz" or "cuz" (phonetic substitution, transformation takes place in 38 out of 62 cases)
- "coz", "cuz" → "cz" (vowel deletion, transformation takes place in 3 out of 38 cases)

The rest of this article focuses on modeling of the unintentional errors as well as the first two kinds of intentional errors, and their combinations. Abbreviations and non-standard usages can be dealt with using dictionaries and rules, which is beyond the scope of the current work. Nevertheless, distortions due to alternative pronunciations and character substitution gets automatically captured in the model, as these can be represented in terms of the aforementioned compression and error patterns.

6 Conception of the word model

The purpose of the word model is to capture the intentional compressions employed by the user to shorten the length of the word as well as the unintentional errors made during typing. There are several works related to *automatic speech recognition* in general (see [23] and references therein) and *grapheme to phoneme conversion* [43] in particular that utilize HMM to capture the variations in pronunciation and spellings. Inspired by such works, here we model each word w in the standard language as an HMM (see [39] for a review).

Recall that the error model in [12] is based on the assumption that the user, while typing a word, mentally splits the word into fragments of letter sequences (the length of each sequence being restricted to five). Each of these fragments are then replaced by an equivalent letter sequence (possibly the same) independent of the structure and the fate of the other sequences. Irrespective of whether this assumption reflects the underlying cognitive process of typing or not, the above analogy provides an intuitive understanding and visualization of the error model involved. In order to explain and justify the word error model proposed here, we appeal to a similar thought experiment.

Let $w = g_1 g_2, \ldots, g_l$, where g_i represents a character or *grapheme* of the standard language. Let $\widetilde{w} = p_1 p_2, \ldots, p_k$ be the pronunciation of w, where p_i represents a *phone* of the standard language.[2] For example, if $w =$ "today", then $\widetilde{w} =$ "/T/ /AH/ /D/ /AY/".

Suppose, a user wants to reduce the length of the word w while typing it on a mobile keypad. She mentally splits the word into fragments, as in [12], and inspects each of the fragments independently. For each of the fragments, she might decide to type a phonetic or a graphemic equivalent of the fragment. If she chooses to type a graphemic equivalent, she inspects the fragment character by character and decides whether to delete or type the particular character under consideration. On the other hand, if she chooses to proceed with a phonetic equivalent, then she might decide to replace the whole fragment with a single character if possible (e.g., '2' for "/T/ /AH/"), or she might inspect the pronunciation of the fragment phone by phone and replace each of the phones by letters following some sound-to-letter rules. Note that for every key press she makes, there is also a possibility of unintentional error, that is pressing of an unintended key.

We encode the aforementioned typing process through the structure of an HMM. Each state in the HMM represents a possible key press. Transition between the states represent the possible choices after a key press (for example, whether to type the next fragment of the word graphemically or phonetically). Thus, the transition probabilities reflect how often the users choose to type a particular fragment in a specific manner, whereas the emission probabilities associated with a state reflect the uncertainty in the output (say due to unintentional errors), given a particular choice. We describe below the structure of the word HMM and its construction.

[2] The phones are represented here between two '/' following the convention used in the CMU Pronouncing Dictionary available from http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

6.1 The Basic HMM

Suppose the word $w = g_1 g_2, \ldots, g_l$ is typed without any compression or error. This situation can be modeled using a left-to-right HMM having $l + 2$ states as follows. There is a *start state* and an *end state* denoted by S_0 and S_{l+1}, respectively. The only observation associated with these two states is the special symbol \$, which marks the beginning and the end of a word. There are l intermediate states G_1 to G_l, where the only observation associated with state G_i is the grapheme g_i. There is a transition from S_0 to G_1. From each G_i there is a transition to G_{i+1} and from G_l there is a transition to S_{l+1}. The HMM so defined has a probability of 1 for the observation sequence $\$g_1 g_2, \ldots, g_l\$$, i.e. the word, and 0 for all other sequences.

As illustrated in Fig. 1 for the word "today", a series of modifications applied on this basic HMM finally leads to the word model of TL. This process has been detailed out in the subsequent sections.

6.2 Graphemic path

We define each state G_i as a *graphemic state* and the path from S_0 to S_{l+1} through the sequence of graphemic states as the *graphemic path*. At each graphemic state, the user might choose to type or delete the grapheme associated with that state. Suppose we want to model the deletion of the character g_i. There are two possibilities: (1) a new transition is added from G_{i-1} to G_{i+1}, and (2) we associate the emission of the null character ϵ with G_i.

The disadvantage of choice (1) is that in order to model the deletion of any arbitrary number of characters in the HMM, we need to add transitions from a state G_i to all states G_j, whenever $j > i$. Therefore, the number of transitions in the HMM becomes $O(l^2)$, and consequently, we have to estimate $O(l^2)$ transition probabilities, which calls for more training examples. For choice (2), in order to model deletion of an arbitrary number of characters, it suffices to add a null emission to every state. Thus, we need to estimate only l new emission probabilities. Therefore, we have chosen option (2), whereby every graphemic state is associated with a null emission.

It may be noted that existence of null emissions makes it difficult to estimate the observation probability of a string for a given HMM because, Viterbi algorithm cannot be used for HMMs with null emission. Nevertheless, as described in the next section, we modify the Viterbi algorithm suitably to deal with null emissions efficiently.

In order to take care of the unintentional errors, we also allow each of the graphemic states to emit all other characters including numbers and punctuation marks, which we collectively represent by the wildcard symbol '@'. We shall denote the different observation probabilities associated with the

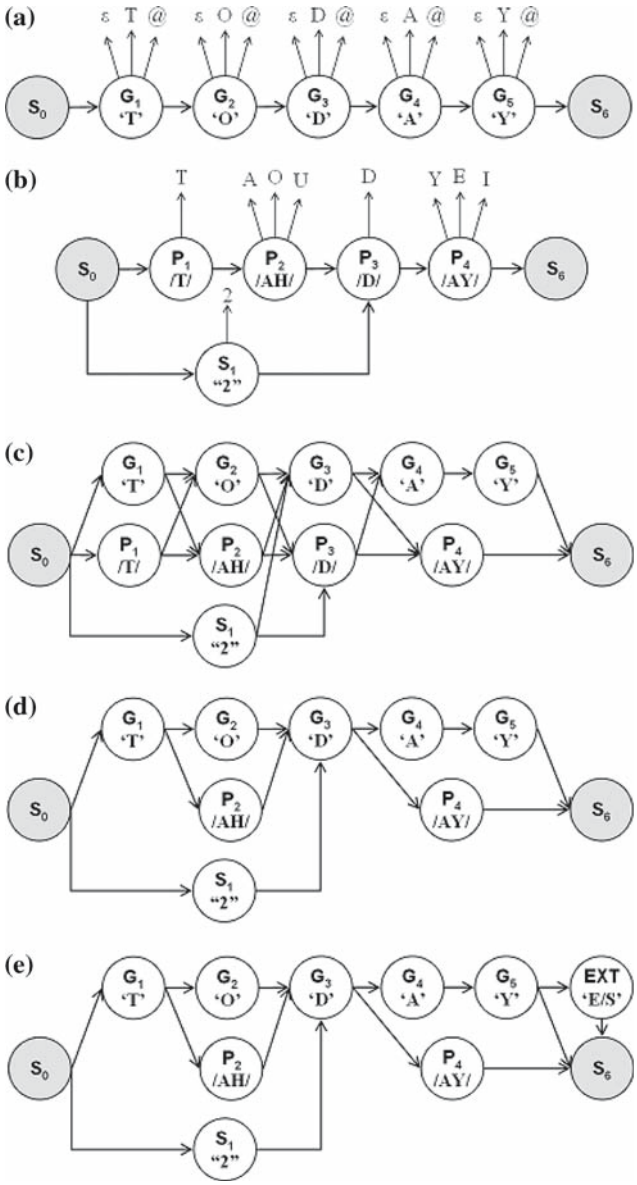

Fig. 1 Construction of the word HMM illustrated for the word "today". The shaded nodes S_0 and S_6 represent the start and the end states, respectively. **a** graphemic path, **b** phonetic path, **c** crosslinkages, **d** after state minimization, **e** after inclusion of the *extended state* EXT. For clarity of presentation, the emissions have been omitted for c, d and e

graphemic state G_i by $Pr(g_i|G_i)$, $Pr(\epsilon|G_i)$ and $Pr(@|G_i)$, where for all i,

$$Pr(g_i|G_i) + Pr(\epsilon|G_i) + Pr(@|G_i) = 1 \tag{5}$$

Note that the above representation precludes the possibility of assigning different emission probabilities for the different instantiations of the wildcard symbol, which is clearly an oversimplification. For instance, consider a state G_i, corresponding to the grapheme g_i = 'a'. The probability of observation of 'b' or 'c' in G_i is expected to be greater than that of 'x' or 'y', because unintentional errors due to pressing

of a single key either more or less than the desired number of times are more frequent than those arising from pressing of an altogether different key. Ideally, the emission probabilities of all the symbols for a given graphemic state should be estimated from the corpus. However, owing to the paucity of appropriate TL data, we have decided to merge all the emissions due to unintentional errors into a single wildcard emission @.

Figure 1a shows the structure of the HMM for the word "today", after the incorporation of the aforementioned modifications. The graphemic path, so modeled, captures the unintentional as well as the deletion errors (described in Sect. 5.1). It is interesting to note that the graphemic path for a word w returns a non-zero observation probability for any string that has a length smaller than or equal to $|w|$. However, for all strings of length greater than w it returns an observation probability of 0. Stated differently, the graphemic path, in a sense, computes the edit distance between an arbitrary string w' and w, where the cost of insertion is infinity, and that of substitution, deletion and perfect match are $-\log Pr(@|G_i)$, $-\log Pr(\epsilon|G_i)$ and $-\log Pr(g_i|G_i)$, respectively.

6.3 Phonetic path

We define k *phonetic states* P_1 to P_k, corresponding to the phones p_1 to p_k in the pronunciation $\widetilde{w}$ of w. As shown in Fig. 1b, there is transition from P_i to P_{i+1} for all i from 1 to $k-1$. There are also two transitions from S_0 to P_1 and P_k to S_6. Each phonetic state P_i emits a set of characters that might be used for representing the phone p_i. For example, the phone /AH/ may be represented by 'u' (as in "gud" for "good") or 'o' (as in "today"). We shall refer to this part of the HMM as the *phonetic path*. The purpose of the phonetic path is to model the intentional phonetic substitutions (as described in Sect. 5.2).

Although the path through P_is can capture the substitution of a single phone by a single character, it cannot capture the substitution of a syllable or string of phones by a single letter (as in "2day" for "today", where '2' stands for the phonemic string '/T/ /AH/'). Therefore, we introduce the concept of *syllabic states* represented by S_1, S_2, etc. Each syllabic state S_i, is a bypass between two phonetic states P_{i-1} and P_{j+1} $(i > j)$, such that the observation s_i associated with S_i is a shorter substitution for the phonetic string $p_i\, p_{i+1}, \ldots, p_j$. We introduce syllabic states for all possible substrings of $\widetilde{w}$ that have length greater than one and can be substituted by a single character.

Note that a syllabic state emits only one character and therefore, the observation probability associated with that character is always one. However, in a phonetic state, more than one observations are possible. We shall denote the probability of observation of a character g in a phonetic state P_i as $Pr(g|P_i)$. Thus,

$$\sum_g Pr(g|P_i) = 1 \tag{6}$$

in order to model the unintentional errors committed while typing a letter corresponding to a phonetic or syllabic state, ideally, we should allow null and wildcard emissions with these states. However, it turns out to be unnecessary, because any deletion or typo in the phonetic path can be equivalently modeled by a deletion or typo in the graphemic path. In fact, by inspecting a TL word, it is impossible to predict whether an unintentional error (such as those in Table 4) has been caused in the context of phonetic substitution or not.

6.4 Crosslinkages and state minimization

While typing a word, people often switch between the graphemic and phonemic domains. For example, in the case of "transl8in" for "translating", "transl" and "in" are in the "graphemic" domain, whereas the "8" represents a phonetic substitution. To capture this pheonomenon, we introduce crosslinkages between the graphemic and the phonetic paths. A transition is added from every graphemic state G_i to a state P_j, if and only if the grapheme g_{i+1} in w is the surface realization of the phone p_j. Similarly, a transition is added from a phonetic state P_i to a graphemic state G_j, if and only if p_{i+1} is the phonetic correspondent of g_j. Similar transitions are also introduced from the graphemic states to syllabic states, and vice versa. Figure 1c shows the HMM for "today" after addition of the crosslinkages.

The heuristics used for identification of crosslinking sites are as follows.

- The graphemic and phonetic states are scanned simultaneously from left to right. If g_i is a consonant (other than y and h), then the nearest phonetic state P_j is searched for, such that p_j is a consonant. If p_j is a possible pronunciation of g_i, then P_j is aligned to G_i.
- States corresponding to vowels are also aligned similarly.
- If G_i is aligned to P_j, and G_{i-1} and P_{j-1} are unaligned, then G_{i-1} is aligned to P_{j-1}.
- After alignment of the states, finding the crosslinking sites between phonetic and graphemic states is trivial.
- Let there be a transition from the state G_i to a state P_j. If there exists a syllabic state S_k, such that there is a transition from P_{j-1} to S_k, then a transition is introduced from G_i to S_k. Similarly, if there is a transition from P_i to G_j, and there is a transition from a syllabic state S_k to P_{i+1}, then a transition is introduced from S_k to G_j.

The HMM so constructed might have some redundant phonetic states, which emit only a single character that is

identical to the primary character emitted by the corresponding graphemic state. For example, P_1 in the HMM of "today" emits only the letter 't', which is also the primary observation for the graphemic state G_1. Therefore, P_1 can be merged with G_1 without affecting the word model. We minimize the HMM by identifying and merging such redundant phonetic states and appropriately redefining the transition edges. The minimized HMM for "today" is shown in Fig. 1d. The algorithm for minimization is straightforward; after the alignment of the phonetic and graphemic states using the procedure described above, the corresponding graphemic and phonetic states, say G_i and P_j, are inspected. If the only possible observation for P_j is g_i, then the state P_j is removed. A transition is added from P_{j-1} to G_i and from G_i to P_{j+1}. If there is a transition to/from P_j from/to a syllabic state S_k, then a transition is introduced to/from G_i from/to S_k.

At this point, it is worthwhile to compare the word model of TL proposed here with the Toutanova and Moore error model. Though not exactly, but in essence, the graphemic and phonetic paths in the TL word model capture the distributions $Pr_{\mathrm{LTR}}(w'|w)$ and $Pr_{\mathrm{PHL}}(w'|w)$ respectively (Eq. 1). The correspondence is not exact, because unlike the graphemic path, which models only letter-by-letter substitution or deletion, the probability $Pr_{\mathrm{LTR}}(w'|w)$ models the substitution of letter-strings by letter-strings. The phonetic path, however, also models the case of substitution of a string of letter by a single character. But we would like to emphasize the fact that while in Toutanova and Moore model the final score of a word is computed through a log–linear combination of the phonetic and graphemic error models, here the two models (i.e., the graphemic and phonetic paths) are combined through appropriate crosslinking of the states.

Thus, conceptually, the Toutanova and Moore model assumes that a word w' could have been generated from w, either by graphemic or by phonetic substitutions applied independently on the whole word, and a weighted sum of these two distinct hypotheses is used to score a word. On the other hand, we assume that w' has been generated through the transformation of certain fragments of w at the grapheme level, while certain other fragments of w at the phonetic level. Therefore, we estimate a single probability $Pr(w|w')$ of the transformation, during which we also identify the fragments that have been transformed at the grapheme level, and those that have been transformed at the phonetic level.

6.5 Extended state

Initial experimentation with the HMMs so constructed revealed that the model failed to suggest the correct word for several cases, where an 'e', 'z' or 's' was appended at the end of TL word (e.g., "rite" for "right" and "anyways" for "anyway"). In order to capture both this, we introduce an *extra state* EXT between G_l and S_{l+1}, which emits 'e', 'z' and 's'. However, the transition between G_l and S_{l+1} is also preserved, because the extra character need not be appended always. Figure 1e shows the HMM after addition of the extra state.

This completes the description of the structure of the HMM. In the next section, we describe automatic construction and parameter estimation of the HMMs.

7 Construction of the word model

There are two steps involved in the construction of the word model: (1) construction of the structure of the HMM, and (2) assignment of the transition and emission probabilities. The second step can be further subdivided into two steps: (2a) estimating the parameters from a training corpus for the frequently observed words, and (2b) generalization of the parameters for any given word. These three processes are described below.

7.1 Construction of the structure

Given a word $w = g_1 g_2, \ldots, g_l$, the structure of the HMM for w is automatically constructed as follows:

1. The initial state S_0 and the final state S_{l+1} are defined, both of which emit the symbol \$.
2. The l graphemic states, G_1 to G_l, are constructed. Transitions are defined from S_0 to G_1, G_i to G_{i+1} (for $1 \leq i < l$) and G_l to S_{l+1}.
3. Every state G_i is allowed to emit the three symbols: g_i, ϵ and @.
4. The pronunciation, $\widetilde{w} = p_1 p_2, \ldots, p_k$, for w is searched from a pronouncing dictionary.[3]
5. The k phonetic states, P_1 to P_k are constructed. Transitions are defined from S_0 to P_1, P_i to P_{i+1} (for $1 \leq i < k$) and P_k to S_{l+1}.
6. Every state P_i is allowed to emit a set of characters corresponding to the phone p_i. This emission set for p_i is obtained from a phone-to-character mapping file that has been constructed manually. These mappings could, as well, have been automatically learnt from the pronunciation dictionary following any standard machine learning technique (e.g., [20,44]). Nevertheless, at this stage, we only need to know the possible set of phones; the emission probabilities are estimated in a later stage, during the learning phase.
7. Similarly, a list of mapping from characters to possible phoneme strings that the character might substitute for, has been prepared manually. The list consists of entries for every letter in the alphabet, as well as the numerals,

[3] We have used the CMU Pronouncing Dictionary.

which are the possible substitutes. Furthermore, the list has been indexed by the string of phones that are being substituted, rather than the substitutes. Every substring of $\widetilde{w}$ is searched for in the list and if a substitute (say s) is found for the substring, say $p_i p_{i+1}, \ldots, p_j$, then a syllabic state S is constructed that can emit the character s. Transitions are introduced from P_{i-1} (or S_0 if $i = 1$) to S and S to P_{j+1} (or S_{l+1} if $j = k$).

8. The crosslinkages are introduced based on the heuristics described in Sect. 6.4.
9. The redundant phonetic states are removed using the state minimization procedure described in Sect. 6.4.
10. A state EXT is constructed that emits 's', 'z' and 'e'. Transitions are added from G_l to EXT and EXT to S_{l+1}

The structure of the HMM for a word described so far defines the possibilities, in the sense that the transitions and observations that are present have a probability greater than 0 and the ones not present in the model have probability equal to 0. The process of estimation of the parameters are described underneath.

7.2 Supervised estimation

In order to specify the HMM completely, we need to define the associated model parameters $\lambda \equiv (A, B, \pi)$, where A is the state transition probabilities, B is the observation probabilities and π is the initial state distribution. We shall denote the probability of transition from a state X to a state X' as $Pr(X \rightarrow X')$. The probabilities of all the outgoing transitions from a state must sum up to 1.

By the definition of the model, the initial state distribution is given by

$$\pi(X) = \begin{cases} 1, & \text{if } X = S_0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The parameters A and B are estimated from the training corpus using the EM algorithm. For this purpose, we construct the structure of the HMMs for the frequent words w_1 to w_{234} present in the training set. For each word w_j, we define an initial model λ_j^0, where the initial state distribution π_j^0 is defined according to Eq. 7. The emission probabilities for each graphemic state G_i are defined as follows:

$$Pr(g_i, G_i) = 0.7, \quad Pr(\epsilon, G_i) = 0.2, \quad Pr(@, G_i) = 0.1 \tag{8}$$

The probabilities have been so chosen to reflect the fact that unintentional errors are rarer than deletion, whereas faithfulness to the word form gets even a higher priority.

The emission probabilities of phonetic, syllabic and extended states are assigned a uniform distribution. Therefore, if there are n characters emitted by a phonetic state P_i, then for each character g, the emission probability $Pr(g, P_i)$ is $1/n$. Similarly,

$$Pr(\text{'s'}, EXT) = Pr(\text{'z'}, EXT) = Pr(\text{'e'}, EXT) = \frac{1}{3} \tag{9}$$

The transition probabilities for the edges from the start state are defined uniformly. This implies that to start with, we assume that phonetic and graphemic substitutions have the same probability. For the graphemic states, the transition to the next graphemic state is assigned twice the probability of the transition to a phonetic or syllabic state. Thus, if from the state G_i, there are transition to the states P_j and S_k, then

$$Pr(G_i \rightarrow G_{i+1}) = 0.5 \tag{10}$$

$$Pr(G_i \rightarrow P_j) = Pr(G_i \rightarrow S_k) = 0.25 \tag{11}$$

On the other hand, if there is no transition to a syllabic state, then

$$Pr(G_i \rightarrow G_{i+1}) = 0.67 \tag{12}$$

$$Pr(G_i \rightarrow P_j) = 0.33 \tag{13}$$

Similarly, for a phonetic state P_i, the transition probability to another state in the phonetic path (say S_k or P_{i+1}) is twice the probability of transition to a graphemic state G_j. The same principle is also followed for assigning the probabilities to the transitions from a syllabic state. These initial assignments of the transition probabilities are motivated by the fact that there is an inherent inertia in the user against switching between graphemic and phonetic substitutions. Note that by construction there can be at most three transitions from a state.

The EM algorithm proceeds as follows. Given λ_j^0, we find out the most likely sequences of states through the HMM for each observation (i.e. variation in the training set) v_i^j for w_j using the Viterbi algorithm [39]. This is called the Viterbi path of v_i^j in the HMM for w_j with parameter λ_j^0. We keep a count with every transition and emission in the HMM, which are initialized to 0. Then for every variant v_i^j, the count of the transitions and emissions in the Viterbi path of v_i^j is incremented by f_i^j (the frequency of the variant v_i^j). Let the final counts associated with emissions and transitions be $C(g, X)$ and $C(X \rightarrow Y)$, respectively. The model parameters of the HMM are re-estimated using the following equations.:

$$Pr(g, X) = \frac{C(g, X) + \alpha}{\sum_g (C(g, X) + \alpha)} \tag{14}$$

$$Pr(X \rightarrow Y) = \frac{C(X \rightarrow Y) + \alpha}{\sum_Y (C(X \rightarrow Y) + \alpha)} \tag{15}$$

We add the term $\alpha = 0.1$ to deal with 0 counts; this process is known as *Add-α smoothing* (see [24] for details). We shall denote this re-estimated model for w_j as λ_j^1. Using λ_j^1 as

the initial model, we re-estimate the parameters again following the aforementioned step, and arrive at λ_j^2. The process is continued till a stage, after which re-estimation does not change the model parameters any more, i.e., $\lambda_j^m \equiv \lambda_j^{m+1}$. This is the final HMM model, which we shall denote as λ_j. Thus, at the end of the supervised estimation, we have 234 HMM models λ_1 to λ_{234}.

There are two points worth mentioning. First, the paucity of training instances led to very fast convergence of the models. In fact, in many cases just one iteration was sufficient, and there were hardly any case that required more than two iterations. The second issue is regarding the modification of the Viterbi algorithm to deal with null emissions. In the standard form of the algorithm that uses dynamic programing, a matrix $V_{m\times n}$ is filled up columnwise, where the entry $v_{i,j}$ denotes the probability of being in the ith state having seen till the jth input symbol. The implicit assumption is that the Viterbi path constructed so far has j states, one state each for the j input symbols seen so far. However, if null emissions are allowed, the number of states in the Viterbi path can be much larger than the number of input symbols seen. To resolve this problem, we define a three-dimensional matrix $V_{m\times n\times r}$, where the entry $v_{i,j,k}$ denotes the probability of being in the ith state, having seen till the jth input symbol, and having traversed k states (i.e., the partial Viterbi path has exactly k states). The definition of $v_{i,j,k}$ is a simple extension of the standard recursive definition followed while filling up the two-dimensional Viterbi matrix, and therefore, omitted. Nonetheless, it must be emphasized that the acyclic structure of the HMM essentially bounds the value of r by $l+2$ (l being the length of the word w), making the extension possible.

7.3 Generalization

Due to the paucity of training data it is not possible to learn the HMM parameters for every word of the standard language. Therefore, in order to generate the model parameters of unseen and infrequent words, we extrapolate the HMM models λ_1 to λ_{234} learnt from the training set. In Sect. 5 we have attempted to enlist and categorize the different intentional and unintentional distortions observed in TL. We have identified the correlates of each of these error (or compression) types with the HMM parameters. The values of the parameters or the compression operations are learnt as functions of the features of the word and the characters.

The list of generalization parameters used for extrapolation and their physical significance are described below.

- $Pr(\epsilon, G_i)$ and $Pr(@, G_i)$ are learnt as functions of i, l (the length of the word), g_i (whether g_i is a consonant or vowel) and the fact whether $g_i = g_{i-1}$ or not. The choice of function parameters are motivated by the following facts:
  1. In the graphemic domain, deletion or substitution of the first character of a word is rare, but it is much more prevalent towards the end of a word. The positional information i is expected to reflect this dichotomy.
  2. Deletion/truncation is more frequently observed for long words, which is captured by l.
  3. Deletion of vowels is more common than that of consonants. The feature, whether g_i is a vowel or a consonant, has been introduced to capture this effect.
  4. Finally, we have also seen that in TL, usually the double letters (e.g., 'rr') are reduced to single letter ('r'). The feature, whether $g_i = g_{i-1}$, is expected to capture this phenomenon.
- The emission probability $Pr(g, P_i)$, associated with the phonetic state P_i, is learnt as a function of p_i. Thus, all the phonetic states corresponding to a phone has the same distribution of the emission probability. This is an oversimplification, because context of a phone also governs the emission probabilities. However, the paucity of training data makes it difficult to learn the contextual effects. Furthermore, we do not expect the phone-to-letter mapping probabilities to be different for TL than other sources of texts and therefore, is not a primary concern of the current work. One could as well use the standard phone-to-letter probabilities estimated from other sources as in [20,44].
- The ratios of path switching probabilities (i.e., $Pr(G_i \rightarrow G_{i+1})/Pr(G_i \rightarrow P_j)$, $Pr(G_i \rightarrow G_{i+1})/Pr(G_i \rightarrow S_k)$, $Pr(P_i \rightarrow P_{i+1})/Pr(P_i \rightarrow G_j)$, $Pr(P_i \rightarrow P_{i+1})/Pr(P_i \rightarrow S_k)$, etc.) are estimated as functions of l and g_i or p_i being consonant or vowel. The ratio $Pr(S_i \rightarrow P_j)/Pr(S_i \rightarrow G_k)$ is estimated irrespective of any other factor.
- The ratios of transition probabilities from the start state (i.e., $Pr(S_0 \rightarrow G_1)/Pr(S_0 \rightarrow P_1)$ and $Pr(S_0 \rightarrow G_1)/Pr(S_0 \rightarrow S_1)$, if there is a transition from S_0 to S_1) and the last graphemic state (i.e., $Pr(G_l \rightarrow S_{l+1})/Pr(G_l \rightarrow EXT)$) are also estimated as functions of the word length.

Since, we have only 234 instances (HMMs) to learn from, we cannot afford to learn any of the aforementioned parameters as continuous functions of l. Therefore, we split the words into three categories: short ($1 \leq l \leq 3$), medium ($4 \leq l \leq 6$) and long ($l \geq 7$), and learn the functions for each of these word categories separately. Similarly, the positional information i is further discretized as word-beginning, word-middle and word-end, where the value of i distinguishing for word-middle and word-end is chosen based on l.

These design choices help us restrict the domain of each of the functions, so that we can compute the values of the functions for each point in the domain by averaging over the corresponding values obtained from the 234 HMMs. For

example, if out of the 234 HMMs, in 60 the value of $Pr(S_0 \rightarrow G_1)/Pr(S_0 \rightarrow P_1)$ is 2, in another 40 it is 1, and in rest 134 HMMs, there is no transition from S_0 to P_1 (may be because P_1 has been merged with G_1 during the state-minimization phase), then the estimated value of the ratio is $(60 \times 2 + 40 \times 1)/100 = 1.6$

Given a word w, if w belongs to the set of 234 words that has been used for training the initial models, then we adopt the learnt model λ_k for w (where k is the index of w in the training set) without any change. However, if w is an infrequent or unseen word,[4] we construct the HMM for w as described in Sect. 7.1 and assign the emission and transition probabilities based on the functions learnt for the generalization parameters. Note that the ratios of transition probabilities along with the constraint that the sum of all the outgoing transition probabilities from a state is 1, allows us to compute the exact probabilities for each transition.

8 Evaluation

In order to evaluate the goodness of the word model of TL, we build a decoder from TL to SL as described in Eq. 4. Given a TL token t and a word w_i of SL, the term $Pr(t|w_i)$ can be estimated as follows: first, we construct the HMM for w_i; then we compute the Viterbi path in the HMM for t. If there exists such a path, then $Pr(t|w_i)$ is assigned the probability of the Viterbi path, else it is assigned a value of 0. However, as shown in Eq. 4, to build the decoder, we also need to estimate the value of $Pr(S)$, that is the model of SL. We run experiments for the following three assumptions of SL:

Uniform model: $Pr(S)$ is same for all S. The corresponding decoder δ_0 is defined as follows.

$$\delta_0(T) = \underset{S}{\text{argmax}} \prod_{i=1}^{l} Pr(t_i|s_i) \tag{16}$$

$$s_i = \underset{w}{\text{argmax}}\, Pr(t_i|w) \tag{17}$$

Unigram model: Under the unigram assumption, the probability of a sentence S is the product of the probabilities of the words, where the probability of a word, $Pr(w)$ is its normalized occurrence frequency. Thus, the corresponding decoder equation is

$$\delta_1(T) = \underset{S}{\text{argmax}} \prod_{i=1}^{l} [Pr(t_i|s_i)Pr(s_i)] \tag{18}$$

$$s_i = \underset{w}{\text{argmax}}\, [Pr(t_i|w)Pr(w)] \tag{19}$$

[4] Unseen word refers to a word that has not been observed in the corpus of TL, however the word is in the lexicon of SL.

Bigram model: Under the bigram assumption, the decoder equation is

$$\delta_2(T) = \underset{S}{\text{argmax}} \prod_{i=1}^{l} \left[Pr(t_i|s_i)Pr(s_i|s_{i-1})\right] \tag{20}$$

Note that for the uniform and unigram models, sentence level decoding is equivalent to a word by word decoding. Therefore, while testing these models, we shall use a list of words as test cases rather than sentences. However, for the bigram model, sentence level testing is necessary. The implementation of δ_0 and δ_1 is straightforward. We carry out an exhaustive search over all the words in the lexicon of SL to find out the best match. Nevertheless, exhaustive search is not possible in the case of the bigram model, and there we do a beam search [33], with a beam width of 20 words. Although it is possible to optimize the search process in several ways, that would only increase the search speed (which is not our concern here), and not the accuracy.

8.1 Data for SL model

In order to implement the decoders, we need a list of words in SL, over which the argmax search has to be carried out. For this purpose, we use a lexicon of 12,000 most frequent English words. For building δ_1, the unigram frequencies of the words are also required. The unigram frequencies were obtained from [45]. The bigram frequencies required for δ_2 have been obtained from [9]. It has been observed that the results of the decoder is poorer for the bigram model than the other two. This apparently absurd result is due to the fact that the bigram estimates obtained from a standard English corpus hardly reflects the word distribution of the SMS corpus. This led us to carry out experiments with a bigram model, where the bigram frequencies are estimated from the SL part of the SMS corpus that has been collected during this work. We shall denote this decoder model as δ_{2TL}, where the 2 in the subscript stands for bigram and the TL stands for Texting language corpus, from which the model has been estimated.

We also carried out some experiments on language model adaptation, where the bigram frequencies were estimated through a weighted linear interpolation of the bigram frequencies obtained from the TL corpus and a standard English corpus [9]. Best results are obtained when the weights assigned to the TL and Standard English models are 1 and 0 respectively. Therefore, we do not report the detailed results of language model adaptation.

8.2 Test data

We test the different decoder models on the following four data sets:

- *Test Set 1* (TS1) contains 702 distinct TL tokens, which have also been used for training the 234 basic word HMMs. Therefore, TS1 serves the purpose of model testing.
- *Test Set 2* (TS2) consists of 1,228 distinct TL tokens randomly selected from the SMS corpus, none of which were used during training.
- *Test Set 3* (TS3) is a subset of TS2, consisting of all the distorted tokens of the latter. There are 319 tokens in TS3.
- *Test Set 4* (TS4) has been constructed to test the bigram decoders. It contains 138 randomly selected sentences from the SMS corpus. Note that the sentences consist of both distorted and undistorted tokens, some of which have been used for training, while others are unseen.

Note that for all the test data sets, we also know the gold standards, as they are available from the manual tranlsations. All the test sets are available at [14].

8.3 Testing methodology

The word level decoder models (δ_0 and δ_1) are tested as follows: given a TL token t, the probability $Pr(w|t)$ is computed for each word w of SL following Eqs. 17 and 19. The words are sorted in decreasing order of $Pr(w|t)$. For ease of implementation, we use negative logarithms of the probabilities, which monotonically decrease with the probabilities. Consequently, we arrange the results (i.e. w) in increasing order of $-\log Pr(w|t)$. The *rank* of a translation (decoder's output) is its index in this sorted list. Thus, the rank 1 translation, which has the maximum value for $Pr(w|t)$, is the output of the decoder, in case only the single best translation is allowed. However, we evaluate the decoder based on the first 20 suggestions (i.e., form rank 1 to 100) because, incorporation of a more sophisticated language model is expected to further discriminate and elicit the correct suggestion among the top 100.

Note that the unseen TL tokens (i.e. OOV) are handled appropriately by the word-level decoder as long as the corresponding SL word is present in the dictionary. However, if the TL token is generated from a word w which is not present in the dictionary, then w will not be included in the translations suggested by the decoder. This is more often the case with names and other proper nouns, and is an extremely difficult problem to deal with.

Let the gold standard translations of t be $gold_1, gold_2$, etc. For example, for the token "bin", the gold standard translations are "been" and "being". If the output of the decoder matches any of these translations, we assume it to be correct. Thus, given a test data set, we can compute the ranked-recall $A(r)$ of a decoder as the ratio of "the number of tokens, for which at least one of the gold standard translations are within the top r suggested translations" to "the total number of tokens". Note that $A(r)$ is a monotonically increasing function of r.

The sentence level decoders, δ_2 and δ_{2TL}, are evaluated as follows: for every input sentence in TL, the output is a sentence in SL. Although, in order to analyze the incorrect results, we have tapped the list of suggestions for every token in the suggestion list separately, here we report the accuracy for only the final single best sentence obtained from the decoder. The accuracies are computed at the word level, i.e., the ratio of the number of words correctly translated to the total number of words.

8.4 Baseline experiments

In order to compare the proposed method with existing spell checkers, we use the suggestions of Aspell [3], when run in the default mode, as the baseline. The baseline tests are conducted on TS1. The baseline results are: $A(1) = 21.97\%$ and $A(20) = 62.12\%$. Aspell is an open source spell checker that uses the double metaphone encoding strategy [38] for correcting spellings. The technique used is very similar to that of Soundex [35]. Note that the baseline results are only indicative and not very rigorous, since the standard language dictionary used in the baseline experiment is not same as that used for testing our model.

It would be nice to compare the results with that of Brill and Moore model [12] or Toutanova and Moore model [44]. However, we could not do so for several reasons. Firstly, the decoder models proposed here are customized for TL and do not capture other common types of errors such as insertion and transposition. Therefore, it would be crude to compare the models on the data used in [44]. Secondly, it is difficult to train the other two models solely on the TL data (due to the size and the type of the data), which is important if we want to compare the models on the TL data. Nevertheless, with some modifications it may be possible to train the Brill and Moore, and Toutanova and Moore models on the TL data, which we deem as a part of our future work.

8.5 Results for word level decoders

Table 5 summarizes the results of the experiments for the word level decoders. Comparing experiment E1 (refer to the table) with E2, E4 or E5, we observe that the word level decoders proposed here reduces the error rate of Aspell by around 50% for rank 1 suggestions and 70% for rank 20. We also note that the surprisingly high accuracy of E3 on unseen data as compared to the results of model testing, i.e. E2, is due to the presence of a large number of undistorted tokens in TS2. The experiment E4, conducted on TS3 that contains only distorted tokens, is comparable to that of the model testing results. The fact that the accuracy of δ_0 for unseen data is

Table 5 The results of the experiments with the word level decoders

Exp. no.	Decoder	Test set	$A(1)(\%)$	$A(10)(\%)$	$A(20)(\%)$
E1	Aspell	TS1	22.0	56.0	62.1
E2	δ_0	TS1	66.6	86.5	90.7
E3	δ_0	TS2	89.1	94.8	96.4
E4	δ_0	TS3	57.7	78.7	85.9
E5	δ_1	TS3	59.9	84.3	88.7
E6	δ_0	TS4	86	93	96

marginally less than that for seen data, shows that the generalization process has successfully captured the structure of TL and the models are not overfitted. The increment in the accuracy due to the incorporation of the unigram language model has been nominal. This is presumably due to the fact that the unigram frequencies estimated from a standard English corpus do not reflect the characteristics of TL.

Table 6 shows some of the suggestions generated by δ_0 during experiment E2 and E4. Table 7 compares the first few suggestions generated by δ_0 with that of Aspell for some representative TL tokens (taken from E1 and E2). Since Aspell cannot handle tokens with numerals (e.g., "2day") it does not generate any suggestions. It is worthwhile to note that whereas the suggestions generated by Aspell is strongly correlated to the token in terms of edit distance, the same is not the case for our model.

8.6 Results for sentence level decoders

The fact that in around 90% of the cases the correct suggestion has a rank less than or equal to 20, we choose the beam width to be 20 while implementing the argmax search for the bigram models. The bigram models δ_2 and δ_{2TL} are tested on TS4. TS2 and TS3 have all unseen tokens while TS4, which has been used for testing δ_2 and δ_{2TL}, has both seen and unseen tokens. Therefore, in order to measure the improvement obtained through incorporation of the language model, we re-estimate the accuracy of δ_0 on TS4, which varies from $A(1) = 86\%$ to $A(20) = 96\%$ (Table 5).

The accuracy of δ_2 (where bigram probabilities are estimated from standard corpus) measured in terms of number of words correctly translated is 65%. This drastic fall in the accuracy is due to the fact that the bigram characteristics of the SMS text is largely different from that of standard English text. For δ_{2TL}, where the bigram characteristics are obtained from the TL data, the accuracy is 91%. Thus, the incorporation of the bigram model reduces the error rate of δ_0 (for rank 1 suggestions) by 35%. Some example input–output pairs for δ_{2TL} are shown in Table 8.

9 Conclusion

In this article, we have described an HMM-based word error model for TL. The model has been used to construct a decoder

Table 6 Suggestions generated by δ_0 for some seen and unseen (marked *) TL tokens

TL token (SL token)	Decoder output ($-\log(P(w\|t))$)	Rank
2day (today)	today (3.02), stay (11.46), away (13.13), play (13.14), clay (13.14)	1
fne (phone)	fine (3.52), phone (5.13), funny (6.26), fined (6.51), fines (6.72)	2
m8 (mate)	my (6.80), ms (6.86), mr (6.86), mate (8.06), me (8.94)	4
ant (cannot)	ant (0.20), aunt (3.57), ants (3.61), cannot (6.06), cant (6.55)	5
dem (them)	deem (3.52), deems (6.61), dec (6.74), dream (7.06), drum (7.15)	10
orite (alright)	write (7.02), omit (9.79), writes (10.22), writer (10.22), writers (10.69)	95
cuz (because)	crews (6.19), cut (6.74), cup (6.74), occurs (6.82), acres (6.82)	–
cin (seeing)	coin (3.52), chin (3.79), clean (5.95), coins (6.61), china (6.75)	–
chk* (check)	cheek (6.72), check (7.01), echo (9.57), chop (9.6), chin (9.6)	2
jkin* (joking)	join (6.78), joking (6.85), skin (7.09), joint (9.76), joins (9.86)	2

– in the last column means the correct suggestion did not feature in top 100 suggestions of the decoder

Table 7 Suggestions generated by δ_0 and Aspell in ascending order of rank

TL Token	Suggestions by δ_0	Suggestions by Aspell
nyt (night)	night, nut, not, net, next	nut, nt, ny, yt, nwt
lotz (lots)	clothes, lots, logs, loves, plots	lot, lodz, lutz, lots, lott
wkend (weekend)	weekend, weakened, weekends, widened, scandal	wk end, wk-end, wend, kent, kind

The gold standard suggestions are provided within parentheses

Table 8 Examples of translation by δ_{2TL}

Example 1	
Input:	would b gd 2 c u some time soon
Output:	would be good to see you some time soon
Gold Std:	would be good to see you some time soon
Example 2	
Input:	just wanted 2 say a big thanx 4 my bday card
Output:	just wanted to say a big thanks for my *today* card
Gold std:	just wanted to say a big thanks for my birthday card
Example 3	
Input:	me wel i fink bein at home makes me feel a lot more stressed den bein away from it
Output:	me well i think being at home makes me feel a lot more stressed *deny* being away from it
Gold Std:	me well i think being at home makes me feel a lot more stressed than being away from it

from English SMS texts to their standard English forms with a considerable accuracy. We summarize below the salient contributions of this work.

- A word-aligned parallel corpus for SMS and standard English text has been created, which can be used for studying the structure of TL. The corpus is freely downloadable from [14]. A manually cleaned list of word-variation pairs is also available at [14].
- A taxonomy has been proposed for the errors or distortion patterns observed at the word level in TL text.
- An HMM-based error model for the TL words, which attempts to capture the cognitive processes underlying the typing of TL, has been conceptualized. This error model is a novel and significant contribution of this work.
- We have devised algorithms for construction of the word level HMMs and subsequent training of the same from very sparse data. Although the training algorithm used here is a standard one, the proposed set of generalization parameters is an original idea.
- The decoders constructed on the basis of the word model perform much better than Aspell, and the promising accuracy at rank 20 suggests that use of good language models can substantially boost up the performance of sentence level decoder. The poor performance of the decoders with standard unigram and bigram models show that the word distribution characteristics of TL is substantially different from the standard language. This fact is further validated by the observation that the decoder performance is considerably enhanced when the bigram model is estimated from the SMS corpus itself.

Although the proposed word error model is tailor-made for the SMS error patterns, we believe that it can be appropriately modified to capture distortions over other types of TL. For instance, a commonly observed pattern in the chat texts is the repetition of characters to show emphasis (e.g., "sooooo" for "so"). This can be captured by incorporating self-loops in the graphemic states. Transposition errors (e.g., "aks" for "ask") are also common with texts typed over a computer keyboard. This can be dealt with appropriately by introducing backward edges in the graphemic states. The extension of the error model forms a part of our future work.

The learning technique proposed here can also be enhanced in several ways. For instance, syllable and word level analogical learning technique, where the HMM parameters of an unseen word, say "greatest", can be learnt from the HMMs of the known words having similar phonetic or graphemic structure, such as "late" and "test", can be useful during the generalization phase. Another important research direction, as pointed out earlier, can be the study of the syntactic and semantic properties of TL, which in turn can facilitate better language modeling and provide useful linguistic insights.

We have already discussed the possible applications of this work in search engines, text correction over web and other texting medium, and text-to-speech systems. We would like to further emphasize the fact that TL is not only a co-existing, infrequent and informal variant of the standard language; rather it is a process of language change, propelled by the technological and social changes, and has the potential to change the structure of the standard language altogether [6,16,41]. In this light, study of the structure of TL becomes an extremely important research program, where diachronic linguistics can provide valuable insights.

In this work, we have modeled the principle of economy through the structure of HMM. Nevertheless, functionalists also suggest the existence of an opposing force, i.e. distinctiveness, which shape the structure of language. The concept of distinctiveness has not been modeled explicitly in this work. Consequently, we find several counterintuitive suggestions obtained from the decoder. For instance, in Table 7, we observe that for the token "m8", "my", "ms" and "mr" is ranked lower than "mate". Similarly, "coin", "chin", etc. gets lower ranks than "seeing", which, in fact, does not feature in the top 100 suggestions for the token "cin".

Note that if "8" was an unintentional substitution error in "m8" or "cin" was obtained by deletion of some character between 'c' and 'n', then there are numerous possibilities, from which the tokens could have been generated. The force of distinctiveness (or unambiguity) would act against cases of such unintentional substitution or intentional deletion. Thus, appropriate modeling of the distinctiveness principle (i.e., the concept that user would provide just enough information to alleviate any confusion) can improve the performance of the decoder. Unlike the case of the principle of economy, which is a word level phenomenon, distinctiveness has to be measured globally, with respect to the whole lexicon. This is clearly a useful and stimulating direction of research.

References

1. Andersen, J.M.: Structural Aspects of Language Change. Longmans, London (1973)
2. Androutopoulos, J., Schimdt, G.: SMS-kommunikation: etnografische gattungsanaslyse am beispeil einer kleingruppe. Zeitschrift fü Angewandte Linguistik (2001)
3. Atkinson, K.: Gnu Aspell. http://aspell.sourceforge.net/ (2005)
4. Aw, A., Zhang, M., Xiao, J., Su, J.: A phrase-based statistical model for SMS text normalization. In: Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pp. 33–40. ACL, Sydney (2006)
5. Bangalore, S., Murdock, V., Riccardi, G.: Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system. In: Proceedings of COLING-2002 (2002)
6. Baron, N.S.: Computer mediated communication as a force in language change. Vis. Lang. **18**(2), 118–141 (1984)
7. Baron, N.S.: Letters by phone or speech by other means: the linguistics of e-mail. Lang. Comm. **18**, 133–170 (1998)
8. Baron, N.S.: Alphabet to e-mail: How written English evolved and where it's heading. Routledge, London (2000)
9. Bigram: Frequency list. http://www.clg.wlv.ac.uk/projects/style/corpus
10. Boersma, P.: Sound change in functional phonology. Technical Report (1997). URL Rutgers Optimality Archive. http://ruccs.rutgers.edu/roa.html
11. Boersma, P.: Functional Phonology: Formalizing the interactions between articulatory and perceptual drives. Uitgave van Holland Academic Graphics, Hague, Netherlands (1998)
12. Brill, E., Moore, R.C.: An improved error model for noisy spelling correction. In: Proceedings of the 38th Annual Meeting of the ACL, pp. 286–293. ACL (2000)
13. Brown, P.F., Pietra, S.A.D., Pietra, V.J.D., Mercer, R.L.: The mathematics of statistical machine translation: parameter estimation. Comput. Linguistics **19**(2), 263–312 (1993)
14. Choudhury, M.: Word-aligned SMS-standard English parallel corpus. http://www.cel.iitkgp.ernet.in/~monojit/sms.html
15. Choudhury, M., Basu, A., Sarkar, S.: A diachronic approach for schwa deletion in Indo-Aryan languages. In: Proceedings of the 7th Meeting of the ACL SIGPHON, pp. 20–26 (2004)
16. Crystal, D.: Language and the Internet. CUP, Cambridge (2001)
17. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM **7**(3), 171–176 (1964)
18. Döring, N.: Kurzm wird gesendet—abkürzungen und akronyme in der SMS-kommunikation. Muttersprache Vierteljahresschrift für deutsche Sprache **2** (2002)
19. Ferrara, K., Brunner, H., Whittemore, G.: Interactive written discourse as an emergent register. Writt. Commun. **8**, 8–34 (1990)
20. Fisher, W.M.: A statistical text-to-phone function using ngrams and rules. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 649–652. IEEE, New York (1999)
21. Fraser, A., Marcu, D.: Getting the structure right for word alignment: leaf. In: Proceedings of the 2007 Joint Conference on Empirical Methods in natural Language Processing and Computational natural Language Learning, pp. 51–60. ACL, Prague (2007)
22. Herring, S.C.: Computer-mediated discourse. In: Tannen, D., Schiffrin, D., Hamilton, H. (eds.) Handbook of Discourse Analysis, pp. 612–634. Blackwell, Oxford (2001)
23. Jelinek, F.: Statistical Methods for Speech Recognition. MIT Press, Cambridge (1997)
24. Jurafsky, D., Martin, J.H.: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, Englewood cliffs (2000)
25. Kernighan, M.D., Church, K.W., Gale, W.A.: A spelling correction program based on a noisy channel model. In: Proceedings of COLING, pp. 205–210. ACL, NJ (1990)
26. Kukich, K.: Technique for automatically correcting words in text. ACM Comput. Surv. **24**, 377–439 (1992)
27. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Phys Doklady **10**, 707–710 (1966)
28. Lindblom, B.: Systemic constraints and adaptive change in the formation of sound structure. In: Hurford, J.R., Studdert-Kennedy, M., K. C. (eds.) Approaches to the Evolution of Language: Social and Cognitive Bases. Cambridge University Press, Cambridge (1998)
29. Mayes, F., Damerau, F.J.: Context based spelling correction. Inform. Process. Manage. **27**(5), 517–522 (1991)
30. Meillet, A.: The comparative method in historical linguistics. Champion, Paris (1967)
31. Mihov, S., Schulz, K.U.: Fast approximate search in large dictionaries. Comput. Lingu. **30**(4), 451–477 (2004)
32. Murray, D.: Composition as conversation: the computer terminal as medium of communication. In: Odell, L., Goswami, D. (eds.) Writing in Nonacademic Settings, pp. 203–228. The Guilford Press, New York (1985)
33. Ney, H., Mergel, D., Noll, A., Paesler, A.: Data-driven search organisation for continuous speech recognition. IEEE Trans. Sig. Process **40**, 272–281 (1992)
34. Nishimura, Y.: Linguistic innovations and interactional features of casual online communication in Japanese. J. Comput. Med. Commun. **9**(1) (2003)
35. Odell, M.K., Russell, R.C.: U.S. patent number 1,261,167 (1918)
36. Odell, M.K., Russell, R.C.: U.S. patent number 1,435,663 (1922)
37. Palfreyman, D., al Khalil, M.: A funky language for teenzz to use: representing Gulf Arabic in instant messaging. J. Comput. Med. Commun. **9**(1) (2003)
38. Philips, L.: The double metaphone search algorithm. C/C++ Users J. (2000). http://www.ddj.com/dept/cpp/184401251
39. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE **77**(2), 257–286 (1989)
40. Ringlstetter, C., Schulz, K.U., S.M.: Orthographic errors in web ages: towards a cleaner web corpora. Comput. Lingu. **32**(3), 295–340 (2006)
41. af Segerstad, Y.H.: Use and adaptation of written language to the conditions of computer-mediated communication. Ph.D. thesis, Department of Linguistics, Göteborg University Sweden (2002)
42. Stubbe, A., Ringlstetter, C., Schulz, K.U.: Genre as noise—noise as genre. In: Proceedings of IJCAI-07 Workshop on Analytics for Noisy Unstructured Text Data (AND-07), pp. 9–16 (2007)
43. Taylor, P.: Hidden markov models for grapheme to phoneme conversion. In: Proceedings of 9th European Conference on Speech Communication and Technology—Interspeech, pp. 1973–1976 (2005)
44. Toutanova, K., Moore, R.C.: Pronunciation modeling for improved spelling correction. In: Proceedings of the 40th Annual Meeting of the ACL, pp. 144–151. ACL (2002)
45. Unigram: Frequency list. http://www.comp.lancs.ac.uk/ucrel/bncfreq/flists.html
46. Wikipedia: Texting language. http://en.wikipedia.org/wiki/Texting_language
47. Xia, Y., Wong, K.F., Li, W.: A phonetic-based approach to Chinese chat text normalization. In: Proceedings of the COLING-ACL'06, ACL (2006)