Investigation and Modeling of the Structure of Texting Language*

Monojit Choudhury[1], **Rahul Saraf**[2], **Vijit Jain**[1], **Sudeshna Sarkar**[1], **Anupam Basu**[1]
[1]Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, India
monojit@cse.iitkgp.ernet.in
[2]Department of Computer Engineering
Malaviya National Institute of Technology, Jaipur, India
rahul.shillong@gmail.com

Abstract

Language usage over computer mediated discourses, like chats, emails and SMS texts, significantly differs from the standard form of the language. An urge towards shorter message length facilitating faster typing and the need for semantic clarity, shape the structure of this non-standard form known as the *texting language*. In this work we formally investigate the nature and type of compressions used in SMS texts, and based on the findings develop a word level model for the texting language. For every word in the standard language, we construct a Hidden Markov Model that succinctly represent all possible variations of that word in the texting language along with their associated observation probabilities. The structure of the HMM is novel and arrived at through linguistic analysis of the SMS data. The model parameters have been estimated from a word-aligned SMS and standard English parallel corpus, through machine learning techniques. Preliminary evaluation shows that the word-model can be used for decoding texting language words to their standard counterparts with more than 80% accuracy.

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 messages and semantic unambiguity - shape the structure of this compressed non-standard form, called the *NetSpeak* [Crystal, 2001] or the *texting language* (*http://en.wikipedia.org/wiki/Texting_language*).

The objective of the current work is to study as well as formally model the characteristics of the texting language (TL), and based on the compression model so obtained, construct a decoder from a TL word to its standard form. Ideally, we would like to construct a sentence level decoder for TL, such that given an input sentence in the TL, the decoder should be able to generate the corresponding standard form. This is illustrated through the following example.

Input:	*btw wenz ur flt 2moro*
Output:	*By the way, when is your flight tomorrow?*

However, the scope of the current work is limited to word level analysis and modeling, which forms the first step towards a sentence level decoder.

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 hypothetical noisy channel. Hidden Markov Models (HMM) [Rabiner, 1989] have been used to characterize the stochastic properties of the noisy channel. In order to learn the characteristics of the noisy channel (or equivalently the TL), we gathered 1000 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 20000 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 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 [Crystal, 2001; Herring, 2001]. Moreover, the proposed model makes no language specific assumptions and therefore, can be suitably trained for the TL of other languages.

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, emails, ASR transcripted data, and call center data. 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

*The work has been partially supported by Media Lab Asia research funding.

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.). As we shall see shortly, 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.

The structure of the TL and its socio-cultural effects have been a subject matter of diachronic and socio-linguistic studies for the past two decades (see [Crystal, 2001; Herring, 2001] and references therein). Researchers have studied the social communication patterns [Grinter and Eldridge, 2001] as well as the linguistic structures [Nishimura, 2003; Palfreyman and al Khalil, 2003] used in TL. However, we do not know of any work on decoding of TL to the standard form, except for a very recent work on chat text normalization [Xia *et al.*, 2006]. On the other hand, the compression model described here can provide deeper insights into the phenomenon of language change over the texting medium, that is of significant interest to researchers in diachronic linguistics.

The rest of the paper is organized as follows. Section 2 presents a noisy channel formulation of the problem; section 3 describes the creation of training data and enumerates some of the observations made on the data. On the basis of the observations, in section 4 we describe the structure of the HMM for the words of the standard language. Section 5 discusses the learning algorithm for estimation and generalization of the HMM parameters. The evaluation results of the model is presented in Section 6. Section 7 concludes the paper by summarizing the contributions and listing out possible improvement schemes.

2 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 specified in terms of the noisy channel model as follows.

$$\delta(T) = \underset{S}{argmax}\ Pr(T|S)Pr(S) \qquad (1)$$

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

$$\delta(T) = \underset{S}{argmax}\ \Big[\prod_{i=1}^{l} Pr(t_i|s_i)\Big] Pr(S) \qquad (2)$$

Thus, the decoding model δ for a TL sentence can be abstracted out into two distinct levels: (1) the *word model*, $Pr(t_i|s_i)$ and (2) the *language model*, $Pr(S)$. This is a very common technique used in several NLP systems such as speech recognition [Jelinek, 1997], machine translation [Brown *et al.*, 1993] and spell checking [Kernighan *et al.*, 1990]. Since language model is the common part of all such NLP systems and well studied in the literature, we shall not elaborate on it; rather the objective of this work is to develop an appropriate word model for TL.

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. For instance in the input-output pair of sentences cited in the example in section 1, if we assume that a word for word translation has taken place, then the corresponding alignment is (ϵ represents 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 like the IBM Models [Brown *et al.*, 1993] 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 negative marker ("are not" to "aint" or "would not" to "wudnt").

3 Creation of Training Data

The formulation of the compression characteristics $Pr(t|s)$, where t is a token of the texting language and s a word in the standard form, demands the information about the variations of a word over the texting medium and their probabilities. In order to gather realistic TL data we created a 20000 word aligned corpus of SMS texts and standard English. In this section we report the process of acquiring this data and some observations made on it.

3.1 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 apparantly it features the maximum amount of compression and thus, is the most challenging one to model. The reason for this extreme brevity is presumably an effect of the small number of keys (9 to be precise) used for typing the message over cell phones, which necessitates multiple key presses for entering a single character (eg. 4 key presses are required to type in the characters 's' or 'z').

The website *http://www.treasuremytext.com* hosts a huge number of SMS texts in a large number of languages, which have been donated by anonymous users. The messages are not indexed by languages and many of them are written using more than one language. A large number of messages have been downloaded from the site, from which around 1000 English SMS texts are 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 have been manually translated to their standard English form. 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 20000 tokens (words) in the translated standard text out of which around 2000 are distinct.

3.2 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; though during each recursive call the conditions of 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 hypothetical *pivots* at the beginning and the end of the texts.
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 hypothetical 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 the region are computed using the Soundex algorithm [Odell and Russell, 1918].
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%.

[1]T9 stands for *Text on 9 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.

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 1: The variations of the word "tomorrow" and their occurrence frequencies observed in the SMS corpus

3.3 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 2000 distinct English words, only 234 occur at least 10 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 corresonding 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 1 shows the variation characteristics of the word "tomorrow".

3.4 Observations

On an average there are 83 characters in the SMS text for every 100 characters in the corresponding standard English text. We also observe the following common techniques for compression of the words.

- **Deletion of characters:** The commonly observed patterns include deletion of vowels (as in "msg" for "message"), deletion of repeted character (as in "tomorow" for "tomorrow") and truncation (as in "tom" for "tomorrow").

- **Phonetic substitution:** Examples are “2” for “to” or “too”, “lyk” for “like”, “rite” for “right” etc.
- **Abbreviation:** Some frequently used abbreviations are “tb” for “text back”, “lol” for “laughs out loud” etc.
- **Dialectal and informal usage:** Often multiple words are combined into a single token following certain dialectal conventions. For example, “gonna” is used for “going to”, “aint” is used for “are not”, etc.
- **Deletion of words:** Function words (e.g. articles) and pronouns are commonly deleted. “I am reading the book” for example may be typed as “readin bk”.

It is interesting to note that 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.

4 Word Model

The purpose of the word model is to capture the intentional compressions applied by the user to shorten the length of the word as well as the unintentional errors made during tying. There are several works related to *automatic speech recognition* in general (see [Jelinek, 1997] and references therein) and *grapheme to phoneme conversion* [Taylor, 2005] in particular that utilize HMM to model the variations in pronunciation and spellings. Inspired by such works, here we model each word w in the standard language as an HMM. Let $w = g_1 g_2 \dots g_l$, where g_i represents a character or *grapheme* of the standard language. Let $\widetilde{w} = p_1 p_2 \dots p_k$ be the pronunciation of w, where p_i represents a *phoneme* of the standard language[2]. For example, if $w =$ ‘$today$’, then $\widetilde{w} =$ ‘$/T/\ /AH/\ /D/\ /AY/$’.

Suppose the word $w = g_1 g_2 \dots g_l$ is typed without any compression or error. This situation can be modeled using a left-to-right HMM [Rabiner, 1989] 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 transtion to G_{i+1} and from G_l there is a transtion to S_{l+1}. The HMM so defined has a probability of 1 for the obsrevation sequence $\$g_1 g_2 \dots g_l\$$, i.e. the word, and 0 for all other sequences. We describe below how this basic HMM is modified to capture the word model of TL. The process has been graphically illustrated in Fig. 1 for the word “today”.

4.1 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*. As we observe that the most common compression technique employed is deletion of characters, we allow the null observation, ϵ, for each graphemic state. In order to take care of tying errors, we also allow each graphemic states to emit all other characters including numbers and punctuation marks, which we collectively represent by the wildcard symbol ‘@’. Fig. 1(a) shows the graphemic path for the word “today”. We denote the different observation probabilities associated with the graphemic state G_i by $G_i(g_i)$, $G_i(\epsilon)$ and $G_i(@)$.

4.2 Phonemic path

We define k *phonemic states* P_1 to P_k, corresponding to the phonemes p_1 to p_k in the pronunciation $\widetilde{w}$ of w. As shown in Fig. 1(b), the phonemic states are linked from left to right, which forms the *phonemic path*. Each phonemic state P_i

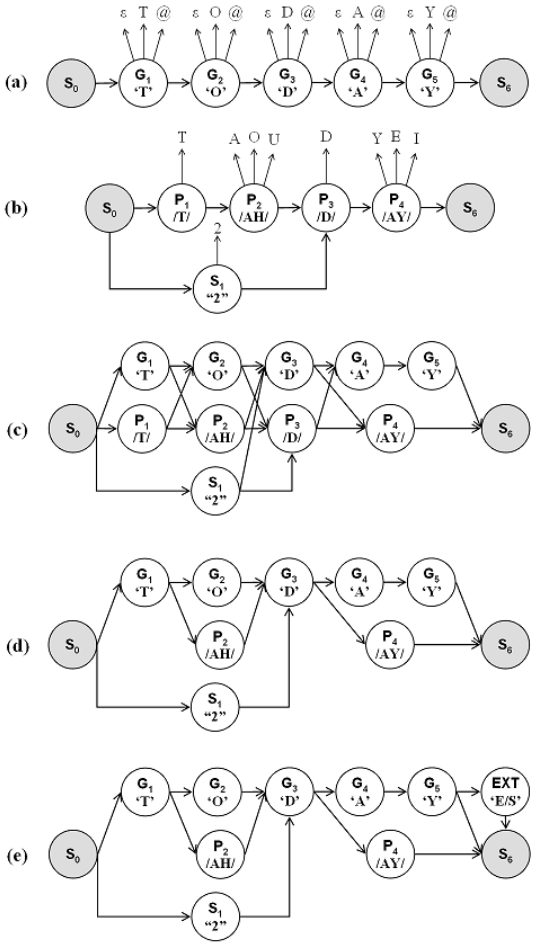

Figure 1: Construction of the word HMM illustrated for the word “today”. The shaded nodes S_0 and S_6 represent the start and the end states respectively. (a) graphemic path, (b) phonemic path, (c) crosslinkages, (d) after state minimization, (e) after inclusion of the *extended state* EXT. For clarity of presentation, the emissions have been ommitted for c, d and e.

[2]The phonemes 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*.

emits a set of characters that might be used for representing the phoneme p_i. For example, the phoneme /AH/ may be represented by 'u' (as in "gud" for "good") or 'o' (as in "today"). The purpose of the phonemic path is to capture the phonetic substitutions used in TL. Although the path through P_is can capture the substitution of a single phoneme by a single character, it cannot capture the substitution of a syllable or string of phonemes 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 phonemic 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 phonemic 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 1 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. Also note that we do not allow null or wildcard emissions for the phonemic and syllabic states, because any deletion or typo in the phonemic path can be equivalently modeled by a deletion or typo in the graphemic path.

4.3 Crosslinkages and state minimization

While typing a word, people often switch between the graphemic and phonemic domains. For example, in "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 phonemic paths. A transition is added from every graphemic state G_i to a state P_j, if and only if in w the grapheme g_{i+1} is the surface realization of the phoneme p_j. Similar transitions are introduced from the graphemic states to syllabic states, and phonemic and syllabic states to graphemic states. Fig. 1(c) shows the HMM for "today" after addition of the crosslinkages.

At this stage we observe that the HMM so constructed have some redundant phonemic 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 compression model. We minimize the HMM by identifying and merging such redundant phonemic states and appropriately redefining the transition edges. The minimized HMM for "today" is shown in Fig. 1(d).

4.4 Extended state

Initial experimentation showed that the word model described by the aforementioned HMM had a poor performance because, the HMM failed to recognize the correct word when a word final 'e' or 's' was appended, as in "rite" for "right" and "anyways" for "anyway". This however is a common feature of TL. In order to capture both this, we introduce an *extra state* EXT between G_l and S_{l+1}, which emits 'e' and 's'. However, the transition between G_l and S_{l+1} is also preserved as the extra character need not be appended always. Fig. 1(e) shows the HMM after addition of the extra state.

4.5 Construction of word HMM

Given a word w, the structure of the HMM for w is constructed as follows. The graphemic path is constructed first. The pronunciation $\widetilde{w}$ for w is searched from a pronouncing dictionary[3], using which the phonemic path is constructed. The emission set for each phoneme is obtained from a phoneme-to-character mapping file that has been constructed manually. Similarly, a list of mapping from characters to possible phoneme strings that the character might substitute for, has been prepared manually. This list is used to identify the possible syllabic states. Crosslinking is done through the alignment of w with $\widetilde{w}$, from which the appropriate linking sites are identified. This is followed by state minimization and introduction of the extra state.

5 Learning

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. 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. By definition, the initial state distribution is given by

$$\pi(X) = \begin{cases} 1, \text{ if } X = S_0 \\ \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

The parameters A and B are learnt from the training data. The learning proceeds in three steps: (1) estimation of the model parameters for the 234 words present in the training data, (2) generalization of the probability values from the learnt models, and (3) computation of the HMM parameters for the unseen and infrequent words from the generalized model parameters.

5.1 Supervised estimation

We construct the HMMs for the words w_1 to w_{234} present in the training set. For each word w_j, we define an initial model λ_j^0 as follows. The initial state distribution π_j^0 is defined according to Eq. 3. The emission probabilities for each graphemic state G_i are defined as $G_i(g_i) = 0.7$, $G_i(\epsilon) = 0.2$, $G_i(@) = 0.1$. The emission probabilities of phonemic, syllabic and extended states are assigned a uniform distribution. The transition probabilities for the edges from the start state is defined uniformly. For the graphemic states, the transition to the next graphemic state is assigned twice the probability of the transition to a phonemic or syllabic state. Similarly, for a phonemic state, the transition probability to another phonemic state is set at a higher value than to other states. Note that by construction there can be at most three transitions from a state.

[3]We use the CMU Pronouncing Dictionary

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 [Rabiner, 1989]. We keep a count with every transition and emission in the HMM, which is increased by f_i^j (the frequency of the variant v_i^j) if and only if the particular transition or emission is used in the Viterbi path of v_i^j. Once the process is completed for all the variations of w_j, we reestimate the model parameters of the HMM based on the final counts associated with each transition and emission. *Add-α smoothing* [Jurafsky and Martin, 2000] is used to handle 0 counts. We shall denote this reestimated model for w_j by λ_j^*. Thus, at the end of the supervised estimation, we have 234 HMM models λ_1^* to λ_{234}^*.

5.2 Generalization

Due to paucity of training data it is not possible to learn the HMM parameters for every word of the standard language. Therefore, we extrapolate from the HMM models λ_i^* learnt from the training set to generate the model parameters of unseen and infrequent words through generalization of the probability values. The motivation behind the choice of the generalization parameters comes from the observations such as the probability of deletion of a character depends on the position of the character in the word (e.g. deletion probability is higher towards the end of the word), the length of the word (longer words feature more deletion), whether the character is a vowel or consonant (vowels are deletd more frequently) and so on. The following parameters are estimated during generalization.

- The null and wildcard emission probabilities for the graphemic states are learnt as a function of (1) the position of the state from the start state, (2) whether the grapheme associated with the state is a vowel or consonant, and (3) the length of the word.
- The emission probabilities of the phonemic states depend only on the particular phoneme ρ represented by the state and the observed character p. Thus, the values for $Pr(p|\rho)$ are obtained by averaging over all phonemic states representing ρ.
- The ratios of the transition probabilities from a graphemic state to another graphemic state and a graphemic state to any other state (phonemic or syllabic) is learnt depending on whether the grapheme represented by the graphemic state is a vowel or a consonant. Similar ratios are learnt for the phonemic states and the syllabic states.
- The ratios of transition probabilities from the start state to the graphemic and phonemic states are also estimated irrespective of other factors.

We refrain from any further discussions on the findings of the generalization step owing to space limitations, even though they reveal several interesting facts about the structure of the TL.

5.3 Model for unseen words

Given a word w, if w is frequently observed in the training data, that is 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, we construct the HMM for w as described in Sec. 4.5 and compute the model parameters from the generalized probabilities and their ratios learnt from the training data.

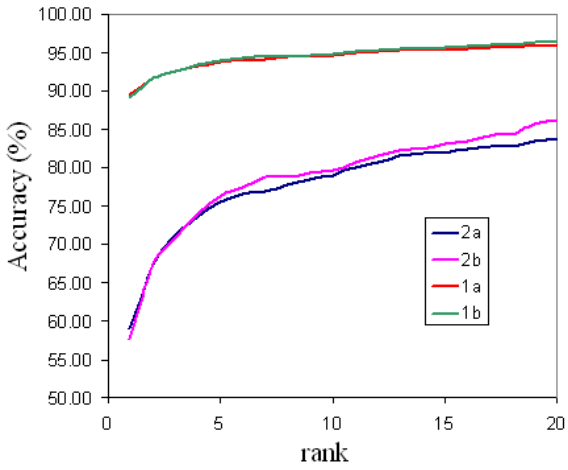

Figure 2: Evaluation results for the word model. 1 and 2 stand for the experiments with and without the undistorted data, whereas a and b stand for the model without and with the extended state.

6 Evaluation

We evaluate the word model on 1228 distinct tokens obtained from the SMS corpus that are not present in the training set. Since the test tokens are chosen from the word-aligned parallel corpus, we also know the actual translation(s) for every word. Also note that as there can be multiple translations for a given token (for example "bin" might come from "been" or "being"), we assume the output of the decoder to be correct if it matches any of the observed translations in the empirical data.

The evaluation procedure is as follows. For each token t in the test set, we compute the probabilities $Pr(t|w_i)$ for every word w_i in the lexicon. We use a lexicon of 12000 most frequent English words. The probabilities are computed using the Viterbi algorithm, which has been modified to handle the null emissions efficiently. We sort the words w_i in descending order of the probabilities, and thus obtain a ranked list of suggestions. The higher the probability, the smaller the rank. The results are summarized in Fig. 2. The x-axis plots the rank of a suggested translation; in the y-axis we plot the word level accuracy $A(r)$, which is defined as the percentage of words in the test set where the correct translation is within the top r suggestions by the system.

We report the results for two different HMM models: a) when the extended state is not present, and b) when it is present. Initially, we observe very little differences between the performance of the two models. The accuracy for both the models start at 89% for $r = 1$ and reaches upto 97% for $r = 20$ (the curves 1a and 1b in Fig. 2). However, on in-

TL token (SL token)	Decoder output ($-log(P(s\|t))$)	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
thx (thanks)	the (6.67), tax (6.89), thanks (7.53), tucks (8.36), takes (8.36)	3
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)	–

Table 2: Suggestions generated by the decoder for some TL tokens. – in the last column means the correct suggestion did not feature in top 100 suggestions of the decoder.

spection it was found that more than 74% of the tokens in the test set were devoid of any distortion. In order to design a stricter evaluation strategy, we removed all the undistorted tokens from the test data, and conducted the same experiments over the remaining 319 tokens. We observe that the accuracy of the model without the extended state (depicted as 2a in Fig. 2) rises from 59% ($r = 1$) to 84% ($r = 20$), whereas the accuracy with the extended state (depicted as 2b in Fig. 2) rises from 58% ($r = 1$) to 86% ($r = 20$). Thus, the inclusion of the extended state in the HMM boosts up the overall performance by around 2% for unseen and distorted inputs.

Even though the improvement in accuracy due to the extended state is quite small, this modification cannot be neglected altogether. This is because by construction, the word HMM for a word of length n returns a probability of 0 for any word whose length is greater than n. In the SMS data, we observe cases where an extra 's' or 'z' is appended to the words. In some cases this leads to an increase in the length of the token in TL than its SL counterpart. Some examples of extra 's' or 'z' observed in the data are: "bdays" for "birthday", "datz" for "that", "anyways" for "anyway". Extra 'e' is usually appended for phonological reasons, for example "nite" for "night", "grate" for "great", etc. Although the state extension is not absolutely necessary for handling such cases, we find that the modification helps in improving the rank of the suggestions in the aforementioned cases.

Table 2 shows some tokens of TL, there gold standard translations in SL and the first five suggestions of the decoder with the extended state. The negative logarithm of the conditional probabilities of the suggestions for the TL token are given in parenthesis for comparision, along with the rank of the correct suggestion.

7 Conclusion

In this paper we described an HMM-based conversion model between TL and the standard language. The model has been used to construct a decoder from English SMS texts to their standard English forms with an accuracy of 89% at the word level. The decoder can be used for automatic correction as well as information extraction and retrieval from noisy English documents such as emails, blogs, wikis and chatlogs that are written in TL.

The novelty of the current work resides in the construction of the word HMMs reflecting the compression techniques used by human users and learning of the associated model parameters from extremely sparse data. Nevertheless, the structure of the current HMM can be improved in several ways, such as (1) addition of self-loops to graphemic states to capture emphasis as in "soooo" for "so" and (2) modeling of transposition errors as in "aks" for "ask". The decoder can be improved by incorporating language model and modules to handle abbreviations, deletion and fusion of words, etc. Similarly, a 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, like "late" and "test", can significantly boost up the performance of the parameter estimation module.

References

[Brown *et al.*, 1993] P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, 1993.

[Crystal, 2001] D. Crystal. *Language and the Internet*. CUP, Cambridge, UK, 2001.

[Grinter and Eldridge, 2001] R. Grinter and M. Eldridge. y do tngrs luv 2 txt msg. In *Proceedings of the Seventh European Conference on Computer-Supported Cooperative Work*, pages 219–238, Bonn, Germany, 2001. Kluwer Academic Publishers.

[Herring, 2001] S. C. Herring. Computer-mediated discourse. In D. Tannen, D. Schiffrin, and H. Hamilton, editors, *Handbook of Discourse Analysis*, pages 612–634, Oxford, 2001. Blackwell.

[Jelinek, 1997] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA, 1997.

[Jurafsky and Martin, 2000] D. Jurafsky and J. H. Martin. *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2000.

[Kernighan *et al.*, 1990] M. D. Kernighan, K. W. Church, and W. A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of COLING*, pages 205–210, NJ, USA, 1990. ACL.

[Nishimura, 2003] Y. Nishimura. A funky language for teenzz to use: Representing gulf arabic in instant messaging. *Journal of Computer Mediated Communication*, 9(1), 2003.

[Odell and Russell, 1918] M. Odell and R. C. Russell. The soundex coding system. *US Patents*, 1261167, 1918.

[Palfreyman and al Khalil, 2003] D. Palfreyman and M. al Khalil. Linguistic innovations and interactional features of casual online communication in japanese. *Journal of Computer Mediated Communication*, 9(1), 2003.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Taylor, 2005] P. Taylor. Hidden markov models for grapheme to phoneme conversion. In *Proceedings of 9th European Conference on Speech Communication and Technology - Interspeech*, pages 1973–1976, 2005.

[Xia *et al.*, 2006] Y. Xia, K-F. Wong, and W. Li. A phonetic-based approach to chinese chat text normalization. In *Proceedings of the COLING-ACL'06*. ACL, 2006.