NORMALIZATION OF TEXT MESSAGES FOR TEXT-TO-SPEECH

Deana L. Pennell and Yang Liu

The University of Texas at Dallas
Computer Science Department
800 West Campbell Road
Richardson, TX 75080
{deana,yangl}@hlt.utdallas.edu

ABSTRACT

This paper describes a normalization system for text messages to allow them to be read by a TTS engine. To address the large number of texting abbreviations, we use a statistical classifier to learn when to delete a character. The features we use are based on character context, function, and position in the word and containing syllable. To ensure that our system is robust to different abbreviations for a word, we generate multiple abbreviation hypotheses for each word based on the classifier's prediction. We then reverse the mappings to enable prediction of English words from the abbreviations. Our results show that this approach is feasible and warrants further exploration.

Index Terms— text normalization, text-to-speech, text messages, abbreviation

1. INTRODUCTION

Text messaging (or SMS) is a rapidly growing form of communication offered by cell phone service providers. This popularity has led to safety concerns because users read and write text messages while driving. SMS use is also difficult for users with visual impairments or physical handicaps. We believe a text-to-speech (TTS) system for cell phones can decrease these problems to promote safe travel and ease of use for all users. This paper describes our preliminary work to determine the feasibility of creating such a system.

The usual first step in a TTS system is text normalization. For SMS messages, this is complicated by the large number of abbreviations used to shorten message length. There is limited previous work on this problem. [1] used a machine translation approach for text message normalization; however, a large annotated corpus is required for such a supervised learning method since the learning is performed at the word level. Although we are in the process of collecting and annotating such a corpus, the process is time consuming and the corpus is not yet large enough to be useful.

In this paper, we propose a framework to be used as a starting point for a text message normalization system. A list of possible abbreviations for English words is generated and then reversed to create a look-up table from text message lingo to proper English. Our system can thereby recognize abbreviations that have not appeared in training, but may appear in the future. We create the mapping table using a statistical classifier to determine whether a character can be removed based on contextual information.

Our results are satisfying when tested using different data sets and metrics, showing this is a reasonable preliminary step toward a normalization system. Improvements to this work using additional features and language models are in progress.

2. RELATED WORK

Text normalization is an important first step for any text-to-speech (TTS) system. Regardless of the size of the training corpus, there will always be tokens that do not appear and have unknown pronunciations. For any unknown token, there are three generally accepted processing methods. First, the system may pronounce each letter, such as in "SMS". Second, the token may be read as if it is a word, such as "NATO". Third, it may be expanded into a dictionary word as when "Corp." is pronounced "Corporation". The system may also combine the previous three methods, such as for "WinNT". Text normalization has been widely studied in many formal domains. See [2] for an overview of text normalization and its associated problems.

Text message normalization presents many difficulties that are not encountered in other domains. This domain is constantly evolving; New abbreviations appear frequently and inconsistently. One user may abbreviate a single word in multiple ways, or occasionally choose to not abbreviate. Abbreviations may also contain numbers and symbols, which is very uncommon in formal text. Using a typical spell-checking algorithm to normalize text messages is mostly ineffective. This is probably because most spelling systems focus on single typographic errors using edit distance, such as [3], or a combination of this approach and pronunciation modeling, such as [4], but do not account for the characteristics of text messages.

Some prior research to normalize text messages works toward translating text messages from one language to another. [1] viewed texting lingo as another language with its own words and grammar. They used a statistical method to translate from the text message language to English, producing grammatically correct sentences. [5] combines a machine translation approach with a second phase mapping the abbreviation characters to corresponding phonemes. The phonemes are viewed as the output of an ASR system and decoded into English words using a non-deterministic phonemic transducer. A translation system using consensus translations to be used for instant messages and chat rooms (where texting abbreviations are commonly used) is described in [6].

The work in [7] describes a supervised noisy channel model using hidden Markov models for text message normalization. [8] modifies this work, making it unsupervised by dividing abbreviations into eleven categories and creating probabilistic models for the most common types. The models are combined, and the English word with the highest probability is chosen as the standard form of the texting word. Most similar to our method is the work in [9], which also views abbreviation of Chinese business names as a binary tagging problem. Abbreviations are formed by using conditional random fields to tag characters for deletion.

3. DATA

We are in the process of creating a large text message corpus for this project. We anonymously gathered approximately 20,000 sentences from volunteers using a Telit GM862-GPS cell phone modem. The modem interacts with a computer program through an RS-232 serial communication bus on a development board. A few users also sent text files containing hundreds of their sent messages.

Since this process is costly and slow, we rapidly enlarged the corpus by using public data from the social networking site *Twitter* [http://twitter.com]. *Twitter* allows users to share 140 character "status messages" (20 characters fewer than allowed in an SMS message). In fact, users may update their status messages by sending a text message to the website. To ensure that the messages we gather from *Twitter* strongly resemble our target domain, we collect only status messages sent by SMS. To maintain anonymity, we do not collect information about the sender of the status update. Currently, we have collected almost a million sentences from Twitter status messages updated by text message. Many of the sentences in our corpus contain no abbreviations at all, or very few. Some examples of highly abbreviated messages in our corpus are listed below.

- `itll b good to see u`
- `Sry 4got`
- `txt me if u want 2 game 2nght`
- `Wntry shwrs, then fair.`
- `Ur doin gd hunni, plz continue wen u av time, tho i probs wnt b here wen u upload ur nxt chap`

Humans who speak English natively have little difficulty deciphering these messages. A TTS system, on the other hand, does not have the background knowledge necessary to infer meanings and pronunciations from these abbreviations. Notice that the writers are not consistent. The first message contains the word "see" instead of the letter "c", although other words are replaced with their character analogues. The fourth message abbreviates the first two words, but not the final two words. For this reason, it is important that our system be robust to multiple spellings of the same word. We are therefore annotating the corpus using two methods.

First, we ask human annotators to list the standard English form for abbreviations found in the corpus. We have collected some annotations of this form using Amazon's "Mechanical Turk".[1] Currently this task is still in progress. From this data, we will form a parallel corpus, which we plan to release when complete.

Because some abbreviations for a given word may not appear in our corpus, we also asked annotators to translate English words to texting lingo. Sentences in our corpus with five or more tokens were chosen for this process. We then eliminated sentences containing out-of-vocabulary words of at least three characters as compared to the CMU lexicon[2]. For each unique word, we presented the annotator with both the word and a sentence in which it appeared for context. We asked the annotator to list all possible reasonable annotations for the given word. It was left up to the individual annotator to decide the meaning of "reasonable". Each word has three distinct annotators, and each annotator has completed a minimum of 250 words. Currently, 1000 words have been annotated in this fashion, and we continue to collect this type of annotation.

[1] This tool allows researchers to upload human intelligence tasks such as annotation, called HITs. Turk users complete the tasks for a small price. This tool can be found at https://www.mturk.com/mturk/welcome

[2] Available at http://www.speech.cs.cmu.edu/cgi-bin/cmudict

4. APPROACH

In this work, we focus on abbreviations involving only deletion of characters from the English form, without addition or substitution of characters. Abbreviations based on phonemes, such as "2moro" for "tomorrow", typically have the same pronunciation as the standard form and do not pose as many problems for TTS systems. On the other hand, an abbreviation such as "tmr", also for "tomorrow", is difficult for a TTS program to decipher.

Our preliminary approach to text message normalization is similar to both the Chinese abbreviation task of [9] and the transliteration work described in [10]. First, we "translate" valid English words to possible text message abbreviations using a statistical approach to tag characters for deletion, thereby creating a mapping from English to the text message domain. We then reverse the mapping so that English words may be predicted from the text message input. We create a set of handcrafted rules for word abbreviation, which are used to design features for the classifier.

4.1. Handcrafted rules to "translate" English to texting lingo

We designed the rules listed below after examining abbreviations of the deletion-based abbreviations found in our corpus.

Remove all instances of double consonants from the original word. For the purposes of this experiment, consonants include 'b', 'c', 'd', 'f', 'g', 'k', 'l', 'm', 'n', 'p', 'r', 's', 't' and 'z'. These consonants are sometimes doubled in English words. An example of this type of change is using "spel" for "spell".

Remove silent 'e', initial 'h' and ending 'g'. We remove any 'e' at the end of a word that is not preceded by a vowel, for instance changing "able" to "abl". The previous character is checked because we do not remove the 'e' from the end of all words (e.g., "coffee"). We consider 'a', 'e', 'i', 'o', and 'u' to be vowels, while 'y' is sometimes considered a vowel (e.g., in words such as "syllable"). The 'h' is dropped from the beginning of words, and the 'g' is dropped from the "ing" ending.

Remove all "unnecessary" vowels. What is an unnecessary vowel? By examining abbreviations, it appears that typically vowels are removed only when they are internal to the syllable. Therefore, the first action taken is to deconstruct the word into its constituent syllables. To accomplish this task, we use the online dictionary found at http://www.dictionary.com to gather syllable information. This dictionary has limitations when accessing words with prefixes or suffixes, which are not syllabified in the same format as the base words on their website. We plan to implement the automatic syllabification of words as described in [11] to address this issue. To remove these internal vowels, we search each syllable for any number of vowels surrounded by consonants. An example using this rule is replacing "text" with "txt".

Special cases. Some combinations of letters are so common that people abbreviate them even though the vowel is not internal to the syllable. For instance, the suffix -er that ends a noun such as "teacher" is often abbreviated to just -r. We created a list of these special cases and abbreviate them during this stage.

Prefix/Suffix clipping.[3] This type of abbreviation is marked by the use of only the first syllable of a word, or the removal of the first syllable of a word. We implement these rules for all multisyllabic words whose first syllable does not correspond to a common morphological prefix, such as "un-". We also ensure that, in two syllable words, the second syllable does not correspond to a common morphological suffix, such as "ing".

[3] Names for these types of abbreviations are taken from categories in [8]

4.2. Automatic abbreviation creation using a maxent classifier

Statistical approaches generally perform better than rule based approaches. Since we only consider abbreviations made by deletion, we treat our problem as a binary classification task. For each character, a decision is made whether or not to remove the character when forming an abbreviation using a maximum entropy model.

To train the model, we used the 1000 English words annotated with abbreviations from the annotations described in Section 3. We use abbreviations entered by at least two of the three annotators for a word. Since annotators were instructed to list *all* reasonable annotations, and not just those made up of deletions, we align the word and abbreviation using the edit distance algorithm. The penalties for insertions and substitutions are set to be one and the penalty for deletions is zero. Any pair with an edit distance greater than zero therefore contains at least one insertion or substitution and is discarded. This produces 4773 characters from 864 words (727 unique) to serve as training instances. The alignment information between a word and its abbreviation also provides the class labels for each character for the classification task. A character is given a positive label if it appears in the abbreviation, and a negative label if it has been removed. We create the following list of features loosely based on the rules in Section 4.1:

1. Contextual Features
   (a) The character itself, c_i.
   (b) Up to two previous characters, c_{i-1}, c_{i-2}.
   (c) Up to two following characters, c_{i+1}, c_{i+2}.
   (d) The two bigrams that include the current character, $c_{i-1}c_i$, c_ic_{i+1}.
   (e) The three trigrams containing the current character, $c_{i-2}c_{i-1}c_i$, $c_{i-1}c_ic_{i+1}$, $c_ic_{i+1}c_{i+2}$.
2. Function Features
   (a) Whether the character serves as a vowel.
   (b) Combination of features 1a and 2a.
3. Syllabic Features
   (a) The characters making up the syllable in which the character appears.
   (b) The character's position in its containing syllable.
   (c) Combination of features 3a and 3b.
   (d) Multiple features describing the position of the syllable in the word (first/last/mid)[4]

We hypothesize that the rules for removal of specific letters, double letters and special cases are captured implicitly by the contextual features. Feature 2b was included to help locate the silent 'e' character, since we do not include any pronunciation features. We assume any word-final 'e' character is silent when preceded by a consonant and mark the 'e' as a consonant in Feature 2a.

To find syllable boundaries, we again used the free online dictionary available at http://www.dictionary.com. For feature 3b, we determine a consonant's position within a syllable by whether it falls to the left or right of the syllable's sonorant vowel(s). Those on the left are assigned "beginning", while those on the right are assigned "end". Vowels with consonants on both sides are assigned the position "middle". Those with no consonants to the left are assigned "beginning" and those with none on the right are assigned "end". If a syllable consists of only vowels, they are assigned the position "beginning". We consider a silent 'e' to be a consonant, since it does not function as a vowel in the syllable. Figure 1 shows the positions of characters in the word "symmetry".

s	y	m	/	m	e	/	t	r	y
|	|	|		|	|		|	|	|
B	M	E		B	E		B	B	E

Fig. 1. An example of syllable positions assigned to characters. The beginning, middle and end positions of a syllable are noted by 'B', 'M' and 'E', respectively.

Feature 3d was intended to help generate prefix- and suffix-clipping sub-class of deletion-based abbreviations. We were able to correctly generate only one of the 26 clipping abbreviations in the data set. Including these features hurt our performance on the subsequence category (our intended set for comparison), so it is not included in the experiments described below.

We use the LeZhang implementation of the maxent algorithm[5] to classify each character as keep (positive) or delete (negative). We call the posterior probability for the positive class $p(c)$ for a test character. Maxent classifies it as a positive instance if $p(c) > 0.5$, and negative otherwise. By combining the decisions for each character, we obtain the 1-best abbreviation for a given word. As we mentioned above, text message users are inconsistent and abbreviate a single word in multiple ways. To ensure that our system is robust to these variations, we want to generate multiple abbreviations for each word. We use the posterior probabilities from maxent to define a score s_a for each abbreviation a of an N-character word w:

$$S_a = \prod_{i=1}^{N} sc(c_i), \tag{1}$$

where

$$sc(c_i) = \begin{cases} p(c_i) & c_i \in a \\ 1 - p(c_i) & \text{otherwise} \end{cases} \tag{2}$$

and c_i is the i^{th} character of word w. The Viterbi decoding algorithm is used to generate an ordered list of abbreviations for each word based on this score. [9] found that word length and abbreviation length are highly correlated. For this reason, the abbreviations are re-ranked using a length prior for the probability of using an M-character abbreviation for an N-character word, $P(M|N)$. These probabilities (and the related $P(N|M)$ used below) are gathered from the parallel corpus provided by [7]. We use the top $(N-1)$ abbreviations for an N-character word after re-ranking.

4.3. Mapping abbreviations to English words

We create a mapping from the abbreviations to the English words from which they were derived. The English word is also mapped to itself, to account for the case where the user chooses not to abbreviate the word. It is possible for one abbreviation to map to multiple English words. Each word in the list is given a score combining its unigram probability estimated from the English Gigaword corpus, the length prior $P(N|M)$, and the score generated above. The words are ordered by their scores and the top word is chosen as the translation. When context information is available, we can use a language model to resolve ambiguity and determine the most likely English words.

[4] Unused in the final experiments.

[5] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

	Accuracy (%)
1-best	58.46
in 2-best	69.23
in 3-best	75.38
Position of correct word in list:	1.3 mean, 3 max

Table 1. Accuracy when translating from texting lingo to English.

5. RESULTS

We conduct two different evaluations for our approach. First we evaluate classifier accuracy by performing 10-fold cross validation on the training data. Always choosing the positive class yields a baseline accuracy of 74.7%. Our classifier has an 88.2% 10-fold cross validation accuracy. It is impossible to receive a score of 100% because our training data contains words with multiple abbreviations. A character deleted in one abbreviation but not the other thus appears with identical features but conflicting class information. The highest possible score for this data set is 95.7%.

Second, we consider system accuracy when returning the English form of an abbreviation. The look-up table is formed from all single-word entries of length at least two from the CMU lexicon containing only alphabetical characters and also having an entry at dictionary.com. We manually enter the words "I" and "a" ignoring other single characters to prevent, for instance, the abbreviation "v" (usually for "very") from being mapped to the single character word 'v'. This yields a large look-up table with 340,008 unique abbreviations having an average of 2.77 words per abbreviation (176 max).

It is unreasonable to use the complete 308-term test set from [7] and [8] for our deletion-based system because it consists mostly of abbreviations containing substitutions. [8] annotated this dataset with eleven categories and presents results specific to some categories. We evaluate our performance using the 65 instance "subsequence abbreviation" test set in this work.[6].

Table 1 shows our system performance on this data. Our 1-best accuracy compares favorably to the 56.9% 1-best accuracy achieved by [8] using their type-specific model, although the very small data set means this is not statistically significant. To determine the extent to which this might be increased by improving our re-ranking scheme, we also consider the "in n-best" accuracy. This metric considers the system correct if the correct standard form is in the top n words for the abbreviation. Our system accuracy is bounded by the percentage of correct pairs present in the look-up table. Our system reaches its maximum accuracy at $n = 3$ despite the large size of the look-up table. We expect that obtaining higher coverage through new features and better re-ranking before creation of the look-up table will therefore yield a similar increase in 1-best accuracy.

6. CONCLUSIONS AND FUTURE WORK

Text-to-speech for text messages would be a beneficial tool in many situations where a user is unable to focus his or her attention on the cell phone screen. Toward that goal, we have shown our preliminary steps toward the creation of a normalization framework for text messaging lingo so that it may be more easily read by current TTS systems. Our results show that this method of normalizing text messages is worth pursuing.

The test set used in this work contains no contextual information, but consists only of a list of word/abbreviation pairs. For this reason we could only consider the unigram probabilities of the words when computing scores. Thus, a particular abbreviation will *always* be translated to the same English word by the current system. In practice, abbreviations appear as part of sentences. We can use n-gram probabilities to compute new scores using the abbreviation's forward and backward contexts, potentially changing the word prediction. We plan to implement this using n-gram probabilities gathered from our corpus and evaluate our normalization system using text message sentences.

Our method is convenient because it requires very little training data to produce a usable model. With the addition of language models based on a corpus of text messages and new features to capture more categories of abbreviations, we feel that our normalization framework will be capable of producing text that can be read by a TTS system. We plan to investigate the tradeoff between memory use (e.g., look-up table size) and processing time.

[6]Thanks to Paul Cook for providing us his category annotations. Due to the ambiguity of his categories, many abbreviations in other categories are also deletion-based. For a comparison with their work, we followed their defined categories in our experiments.

7. REFERENCES

[1] A. Aw, M. Zhang, J. Xian, and J. Su, "A phrase-based statistical model for sms text normalization," in *COLING/ACL*, Sydney, Australia, 2006, pp. 33–40.

[2] R. Sproat, A. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Computer Speech and Language*, vol. 15, no. 3, pp. 287–333, 2001.

[3] K. Kukich, "Technique for automatically correcting words in text," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377–439, 1992.

[4] K. Toutanova and R. C. Moore, "Pronunciation modeling for improved spelling correction," in *ACL*, 2001, pp. 144–151.

[5] C. Kobus, F. Yvon, and G. Damnati, "Normalizing sms: Are two metaphors better than one?," in *22nd International Conference on Computational Linguistics*, Manchester, UK, 2008, pp. 441–448.

[6] S. Bangalore, V. Murdock, and G. Riccardi, "Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system," in *19th International Conference on Computational linguistics*, 2002, pp. 1–7.

[7] M. Choudhury, R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu, "Investigation and modeling of the structure of texting language," *International Journal of Document Analysis and Recognition*, vol. 10, pp. 157–174, 2007.

[8] P. Cook and S. Stevenson, "An unsupervised model for text message normalization," in *NAACL HLT Workshop on Computational Approaches to Linguistic Creativity*, Boulder, CO, 2009.

[9] D. Yang, Y. Pan, and S. Furui, "Automatic chinese abbreviation generation using conditional random field," in *NAACL-HLT 2009*, 2009, pp. 273–276.

[10] U. Hermjakob, K. Knight, and H. Daumé III, "Name translation in statistical machine translation: Learning when to transliterate," in *ACL*, Columbus, OH, 2008.

[11] S. Bartlett, G. Kondrak, and C. Cherry, "Automatic syllabification with structured svms for letter-to-phoneme conversion," in *ACL*, Columbus, OH, 2008.