Normalisation of SMS Text

Gurpreet Singh Khanuja, Sachin Yadav
{gurpreet,sachinky}@iitk.ac.in
Advisor : Dr. Amitabh Mukerjee
Department of Computer Science and Engineering,
IIT Kanpur, India

April 18, 2013

---

1 Abstract

In our project we focus on normalizing the SMS Text. At first we find the OOV(Out of Vocabulary) Words in our text, and then try to identify and normalize the ill-formed words out of those Out of Vocabulary words. For this we first generate a set of correcting candidates known as confusion set on the basis of morphophonemic similarity, and then we will use dependency graph to figure out all ill-formed words. Then the similarity of words and the context of the text is exploited to select the most probable correction candidate for a given ill-formed word.

2 Introduction

Text messages now a days is the most widely used form of communication. As many users read or write text SMS while driving or walking, it has led to many safety concerns as many accidents have occurred in past when people use mobiles for texting friends or relatives while driving or walking. For people those who are physically handicapped or with visual impairments, it's also diffcult for them to use this form of communication. Thus normalising the SMS text(which is noisy and short)can help in Text-To-Speech Conversion to promote safe travel and ease of use for all the users. Also it would help in language translation from one to another.

Due to presence of adhoc abbreviations, typos, phonetic substitution, ungramatical structures and emoticons in the SMS, it becomes difficult for text processing tools to process them. So, to produce a more cannonical lexical rendering of these messages, we have to find some way to preprocess them. Thus all the noisiness in SMS like adhoc abbreviations, typos, phonetic substitution and unconvetional spellings or other causes of lexical deviation would be marked as ill-formed words.

This has a lot of similarity with spell checking and thus it's a challenging task. The main point for which it differs from spell checking is that the noisiness in SMS is often intentional, and mainly for saving the characters due to upper bound of 140 or 160 characters.lThus the SMS normalizing task is beyond spell checking and our goal would be to restore all ill-formed words to their cannonical lexical form in the standard English.

3 Previous Work

Noisy Channel Model The Noisy Channel Model was the first approach used primarily for text normalization(Shannon, 1948). This model is used in spell checkers, question answering, speech recognision and machine translation. This module consist of two components : language model P(S) and an error model P(T|S). So if the ill-formed text is T and its corresponding standard form is S, the appraoch aims to find argmaxP(S|T) by computing argmaxP(T|S)P(S). The error model is characterised by product of operation probabilities on string edits and also by pronounciation information. It is hard to approximate accurately P(T|S) and this method does take into account the context around the token which can be helpful in resolving ambiguities.

Statiscal Machine Translation Statiscal Machine Translation is a machine translational paradigm which generate translations on the basis of statistical models. Also the analysis of some bilingual text corpus has been done to derive its parameters. But unfortunately this method requires a lot of critical training data which is not easily available and also to sufficiently cover all ill-formed words and context-appropriate solutions, the construction of an annotated corpus is quite a labor intensive task. For the lexical normalisation part, harnessing of SMT is quite tough.

Automatic Speech Recognition Based on the analysis that it is very hard to capture lexical creativity observed in SMS using the phrase-based translation, Kobus et al. (2008b) proposed an Automatic Speech Recognition(ASR) metaphor which could be used to handle the SMS normalisation. Typically the best word sequence is discovered by the ASR system within the lattice of weighted phonetic sequences. Using the phoneme-to-grapheme dictionary, this system tries to convert it normal text SMS in phone lattice, and then further convert it into the word-based lattice. The most probable word sequence is finally chosen by applying the language model on the word lattice and using the best-path algorithm.

4 Our Implementation

Our implementation involves the following steps:

- Separation of OOV(Out of Vocabulary) Words.
- Confusion Set Generation.
- Detection of ill-formed words.
- Candidate Selection.

4.1 For example

hav bin reali happy since tlkin 2 u dnt no y tho
is converted to
have been really happy since talking to you dont know why though

4.2 Separation of OOV(Out of Vocabulary) Words

OOV(Out of Vocabulary) words are those words which are not present in the dictionary. For our implementation, we have used the **PyEnchant** library of python which use the standard 'en_US' dictionary for separating the OOV words from IV(In Vocabulary) words.

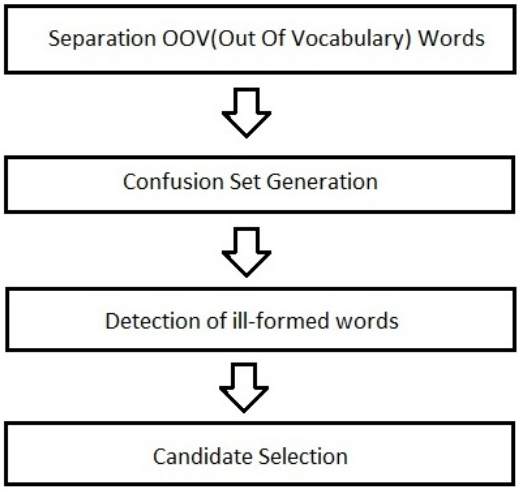

Figure 1: Our Approach

4.3 Confusion Set Generation

In this step, we first create a set of correction candidates called **Confusion Set** for the given OOV word which we selected in step one on the basis of morphophonemic similarity and lexical similarity. Here we have used the **Double Metaphone Algorithm** (Phonemic Matching) [Philips, 2000] and **Lexical Matching** (Levenshtein Distance) to generate the confusion set. We have used **difflib** python module for calculating the Levenshtein Distance.

4.3.1 Double Metaphone Algorithm

The Double Metaphone phonetic encoding algorithm can return both the primary as well as the secondry code for a given string and thus is called 'Double'. So it gives us a rough approximation of how an English Words sounds, and also gives the same primary or secondry codes for the words which sounds same by reducing them to 12 consonants sounds. So it can be used to do phonemic matching on the basis of how the word is pronounced. For Example :

The primary and secondry codes for 'ur' is a tuple : ('AR',")
The primary and secondry codes for 'your' is a tuple : ('AR',")

Also from our knowledge we know that both of these words sounds similar.

4.3.2 Levenshtein Distance

Levenshtein Distance between any two strings is defined as the minimum number of edits which we can do to transform one string into another and all the edit operations that can be done are insertion, deletion and updation.

For Example : Levenshtein Distance between 'kitten' and 'kitchen' is 2.

kitten - kitcen (substitution of 't' for 'c')
kitcen - kitchen (insertion of 'h' between 'c' and 'e')

4.4 Identifying the ill-formed words

In this step, we detect whether a target word is ill-formed word or not,relative to its confusion set.We extract the depencency of the text using the dependency parser.To predict whether a given target word is ill-formed, we form an exemplar for each of its confusion candidates.Then we extract its dependency,together with their relative positions in the form of (word1,word2,position) tuples.Given the dependency-based features, a linear kernel SVM classifier (Fan et al., 2008) is trained on clean Twitter data, i.e. the subset of Twitter messages without OOV words.
Due to lack of clean twitter dataset,we have skipped this part.

4.5 Candidate selection

The final selection is based on the lexical edit distance,phonemic edit distance,prefix substring,suffix substring and LCS(longest common substring).The suffix and prefix features take into care the cases where leading and trailing characters are skipped for example walkin,talkin etc.

4.6 Results

We use three factors for the analysis as shown in the table and compare the three models Noisy channel,Bo han's Lexical normalization and our model.As we can clearly see that detection of ill formed words from the OOV words play an important role and results are far much better.Due to unavailibility of clean dataset, we are not able to do that step.Using clean data for candidate selection may also improve the results.

Evaluation	NC	Model with ill-formed word detection	Our model
Precision	0.465	0.756	0.470
Recall	0.464	0.754	0.460
F score	0.464	0.756	0.465

Table:The results for first and second approach have been taken from [Bo Han 2011].
(NC = noisy channel model [Cook and Stevenson,2009]).

Precision=Number of normalised Words Correctly/Total Number of ill formed words.
FScore= 2.(Precision.Recall)/(Precision+Recall)

4.7 Conclusion

The proposed approach produce very good results and the accuracy of the confusion set is very high.It has a benefit that it does not use machine translation which require both clean and ill dataset, obtaining which is very labourous process.It depends on very simple morphophonemic tools and use the context very appropriately.

4.8 Acknowledgement

We thank Prof. Amitabha Mukerjee for his valuable support throughout the project, guiding us from time to time and looking into the project when it was needed. We have used open source available libraries like PyEnchant library, Double Metaphone Algorithm, difflib Python Module.

4.9 References

- Bo Han and Timothy Baldwin 2011
  Lexical Normalization of SMS text :Makn sense a #twitter
- Lawrence Philips. 2000. The double metaphone search algorithm. C/C++ Users Journal, 18:3843.
- AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normal- ization. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguis- tics, pages 3340, Sydney, Australia.
- Joseph Kaufmann and Jugal Kalita. 2010. Syntactic nor- malization of Twitter messages. In International Con- ference on Natural Language Processing, Kharagpur, India
- https://github.com/dracos/double-metaphone
- http://pythonhosted.org/pyenchant
- http://docs.python.org/2/library/difflib.html
- http://ww2.cs.mu.oz.au/ hanb/emnlp.tgz