Lexical Normalisation of Tweeter Data

Rui Hu
The University of Melbourne

Abstract

Tweets, referring to the short messages sent by users over Twitter, offer abundant real-time data and information, however, misspellings, grammatical errors and malformed abbreviations present a significant hindrance to the utilisation of natural language processing (NLP) techniques. In this report, existing string-matching approaches and tools such as edit distance, n-gram and Peter Norvig's spelling corrector are fully implemented and experimented. Based on the experiment results, I have applied a system which uses Aspell to generate correction candidates and then ranks candidates based on words similarity, context, as well as empirical rules. My experiments on the system demonstrated that it led to decent improvements in both normalisation accuracy and effectiveness.

1. Introduction

Twitter is an online microblogging website where users can send and read messages to broadcast news and socialise with friends. As of 2013, there were more than 500 million registered users sending about 340 million messages per day. Meanwhile, due to the words constrains of tweets, which is up to 140 characters, users have created a novel syntax to make their messages as brief as possible. However, this brevity results in a lack of standardization of Tweeter data - e.g., *abt* "about" and *2day* "today" - which in turn hindering its utility for information extraction and text mining. In order to adapt NLP techniques to noisy text data, in other words, to convert noisy data into a more standard form of English, a number of attempts have been made. A particular approach of them is to use a classifier to detect ill-formed words and then select the most probable candidate by exploiting both word similarity and context (Han and Baldwin, 2011). For example, *se u 2morw!* would be normalised to *see you tomorrow!* It is notable that tweets will be first pre-processed into individual tokens of words or punctuations, following the syntax shown below:

raw-token\tTYPE\tnormalised-token\n

Where type is a flag indicating whether the token is a candidate for normalisation, it takes one of three values, namely IV "in-vocabulary", OOV "out-of-vocabulary" and NO "not a candidate". In this way, the further normalisation would be decently simplified and structured.

In this report, based on the text normalisation workflow proposed by Han and Baldwin, I would mainly focus on the task of how to make judgements between IV and OOV tokens, as well as how to normalise OOV tokens in an accurate and effective manner.

This report will proceed as follows: Existing normalisation techniques and tools are experimented in section 2. My implemented system is described in section 3 . Experiment on my system is depicted in section 4. Conclusions are given in section 5.

2. Existing Normalisation Techniques

In this section, three existing techniques and tools are described and experimented with. In order to contrast their performance, evaluation results are shown together with the description. Detailed evaluation strategy is explained in section 4.

2.1. Peter Norvig's Spelling Corrector

Peter Norvig's spelling corrector is a very efficient word correction tool that can achieve 80 - 90% accuracy at a processing speed of 10 words per second (Peter Norvig, 2011). The guiding techniques applied in this tool are edit distance and probability principles. Specifically, the tool attempts to find the correction *c*, out of all possible corrections, which maximises the probability of *c* given the original word w.

$$\max_c P(c|w)$$

Based on Bayes' theorem, the above expression can be modified to

$$\max_c P(w|c)\, P(c)$$

- which means the ranking score of each correction candidate is determined by both the probability of *c* and the probability of changing from *c* to *w*. Edit distance refers to the number of single character insertions, deletions and replacements to transform one string into another, which basically reflects *P(w|c)*. Thus, *P(c)* is calculated by counting the occurrence of each individual word in a large text corpus. Then possible corrections whose edit distance is 1 or 2 to the original word are generated. In this way, the most possible correction can be selected according to the value of P(*w*|*c*) and P(*c*). I implemented this tool to normalise tweets and the evaluation results are shown in Table 1.

	Peter Norvig
Types Accuracy	4461/4974 (0.90)
Normalisation Accuracy	4207/4974 (0.85)
Normalisation Recall	128/507 (0.25)
Normalisation Precision	128/712 (0.18)

Table 1: Peter Norvig's spelling corrector (corpus.train)

2.2. N-gram

A n-gram of string *s* is a continuous sequence of n items from *s*. String similarity can be measured by counting the number of n-grams that two strings have in common. Basically, for two given string s1 and s2, the n-gram distance can be calculated as:

$$|Gs1| + |Gs2| - 2*| Gs1 \cap Gs2|$$

Where *Gs1* is the set of n-grams in string s1.

Built on Peter Norvig's spelling corrector, I implement 2-gram method to rank corrections, therefore, the probability of each correction *c* is not only determined by P(*w*|*c*)P(*c*), but also the 2-grams distance between *c* and the original word *w*. Evaluation results are shown in Table 2.

	Peter Norvig + 2-grams
Types Accuracy	4542/4974 (0.91)
Normalisation Accuracy	4234/4974 (0.85)
Normalisation Recall	110/507 (0.22)
Normalisation Precision	110/682 (0.16)

Table 2: Peter Norvig + 2-grams (corpus.train)

2.3. Soundex and Aspell

Soundex is a phonemic method targeted to determine whether two strings have a similar pronunciation. Particularly, it applies soundex code by encoding the word's consonants. Basically, soundex code is made up by the first letter of the word and another three pronunciation digits, where the same codes mean the same pronunciation. Aspell is a combinatorial application of both edit distance and soundex techniques. Basically, it finds all words that have a sounds-like within one or two edit distances from the original word sounds-like (Atkinson, 2004). In order to take advantage of pronunciation as one of the ranking factors, I implement Aspell library to suggest probable corrections and soundex code is also applied as one of the probability measures. As a whole, the most probable candidate is selected based on P(*w*|*c*)P(*c*) and soundex code similarities. The experiment results are depicted in Table 3.

	Peter Norvig + Aspell +Soundex
Types Accuracy	4542/4974 (0.91)
Normalisation Accuracy	4402/4974 (0.89)
Normalisation Recall	119/507 (0.23)
Normalisation Precision	119/417 (0.29)

Table 3: Peter Norvig + Aspell +Soundex (corpus.train)

3. My Implementation

As it can be seen from the comparisons in section 2, there are several normalisation techniques that could be applied to normalise tweets. However, all the experiment results are barely satisfactory. Additionally, the lack of consideration of word context also results in a low accuracy rate. Hence, I decided to implement a system which not only combines the essence of each approach but also fully takes context into account. As a whole, the system works in a sequential workflow shown in Figure 1.

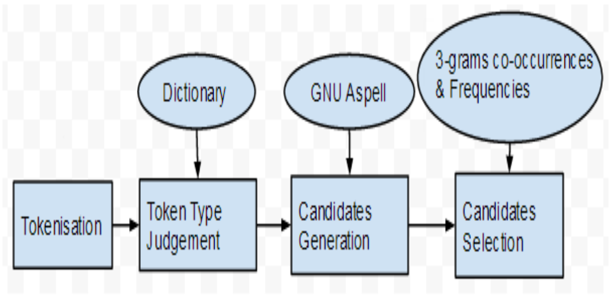


Figure 1: Workflow

3.1. Pre-process Tweets

Tweet pre-processing is also known as tokenisation, which splits the tweet into individual tokens of words or punctuations. Moreover, a header, consisting of a line with a single number, is used to specify how many tokens are in the tweet, for instance, "*see u 2morrow !*" would be tokenised to be as followings:

4
see
u
2morrow
!

3.2. Token Type Judgement

To make judgement among IV, OOV and NO, I rely on a dictionary. Specifically, if a token cannot be found in the dictionary, which means it drops into OOV category, I regard it as suspicious. If a token can be found in the dictionary, it is an IV token and if the token consists of punctuation or other non-word tokens, it would drop into NO. All IV and NO tokens are treated as unsuspicious tokens. In this way, during the evaluation, we rather take a loss in recall than inserting an incorrect word. However, this strategy would lead two problems: First if the dictionary is large enough, some tokens that are supposed to be normalised would be skipped, for example, "*take a pix*" should be normalised to "*take a picture*", but since *pix* was found the dictionary, it would not be normalised. Second it implies a new error probability of changing a correct word into an incorrect one when the dictionary is not large enough.

3.3. Candidates Generation

According to the experiment results in section 2, it can be realised that compared with other correction approaches, Aspell suggests correction candidates in a more meaningful and effective manner, which is due to it applies both edit distance and soundex techniques. Thus, Aspell library is incorporated into my system to form the candidates set for all the OOV tokens.

3.4. Candidates Selection

In this step, all the OOV tokens are corrected by the most probable correction from the candidates set obtained in section 3.3. The following depicts the strategy applied to measure candidate probabilities.

To achieve context normalisation, two statistical measures are taken advantage of: word frequencies and co-occurrence. The word frequency *f(w)* for a word *w* counts the frequency of its appearance within a given corpus. Co-occurrence is usually referring to a pair of words, which commonly appear in similar contexts, however, in order to make the context information more accurate, I would use three words as a group instead, which is also known as 3-grams. To illustrate, if a given line is *"I know that there are seventeen steps"*, system would sequentially read it as "*I know that*", "*know that three*", "*that three are*" and so on, in the meantime, appearance of each threesome group is captured. As a result, a list of *Cooc(w1,w2,w3)* is obtained, which provides a metric to rank candidates. For instance, if the original word is *w*, its left neighbour word is *lw*, right-adjacent word is *rw* and correction candidates are *c1*, *c2*, *c3* and *c4*. Then the candidate *c* whose *Cooc(lw,c,rw)* is ranked as the highest can be said as the most probable candidate. It is notable that a number of words in a text do not have two neighbours but at least one, in this case, candidates are ranked based on the *Cooc(lw, c)* or *Cooc(c, rw)*. In addition, some empirical rules were applied to tweet normalisation, such us in 90% cases, *u* would be normalised to *you*, especially when *u'* appears, and *coz* would be treated as *because*.

4. Experiments

4.1. Experiments Objectives

Experiments in Section 2 are mainly conducted to convey a better understanding of existing normalisation tools and approaches. The objective of experiments on my system is to see whether it achieves a better evaluation result than other methodologies.

4.2. Evaluation Strategies

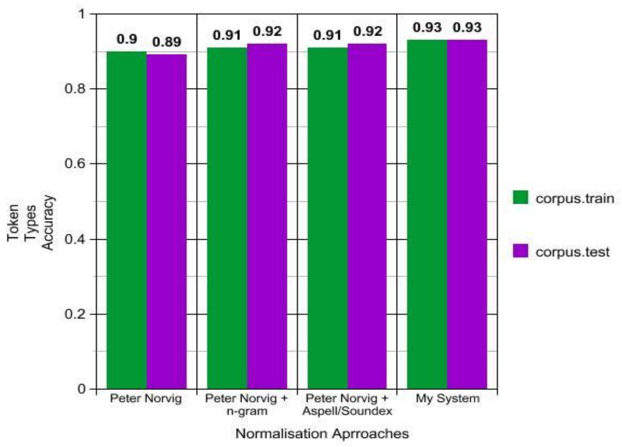
On the whole, evaluation of tweets normalisation is comprised of two strategies: token type judgement and candidate selection. In regard to judgement, I want to assess how well the system can identify OOV tokens and leave IV and NO words untouched. Hence, the judgement accuracy is evaluated by comparing the token types in result file and the correct file. For candidate selection, the same comparison method is applied, but apart from normalisation accuracy, recall and precision are evaluated as well.

4.3. Datasets

British National Corpus (BNC), which consists of 100 million British English words, is utilised to calculate word frequencies and co-occurrences including 2-gram and 3-gram. Two corpora of pre-processed tweets, attached with their correctly normalised corpora, are applied to evaluate the system performance, namely corpus.train and corpus.test. Moreover, token type judgement is relied on a dictionary (words.txt) comprising 1 million individual words.

4.4. Results and Analysis

Evaluation results are presented in Figure 2, and some conclusions can be drawn from the graphs.



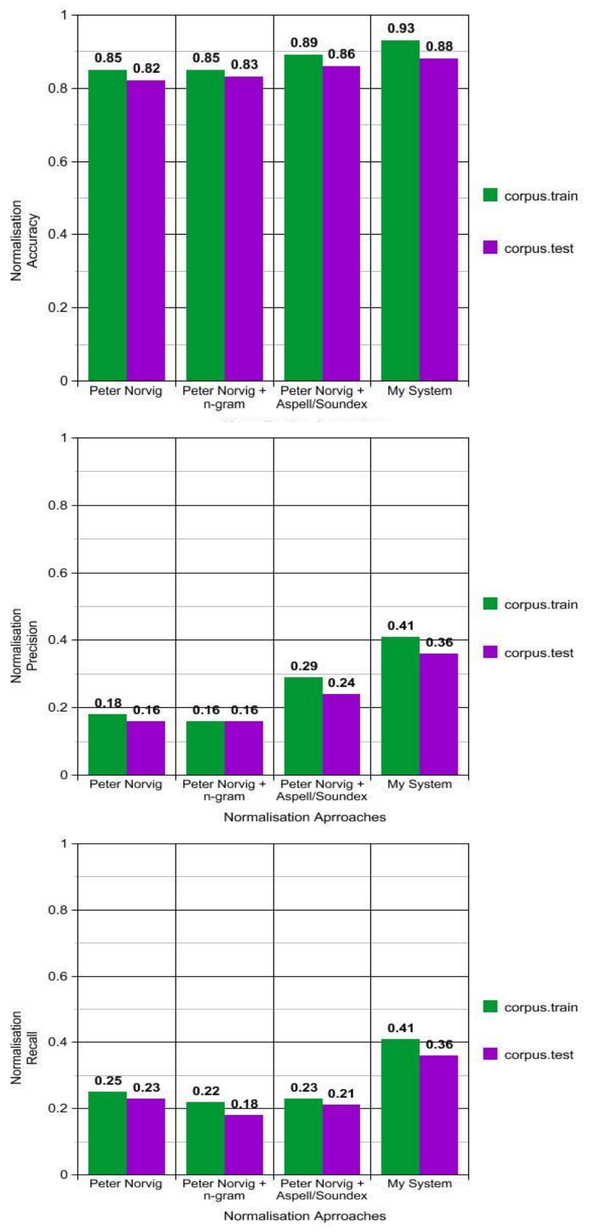


Figure 2: Evaluation Results

First, an overall performance increasing trend can be discovered from the graphs and it is obvious that my system achieved the best performance. Second, it shows that a 10% accuracy divergence exists between the token types generated by all systems and the given data, which is mainly due to the dictionary that the systems used. For instance, *u* was identified as an IV token since it could be found in my dictionary *word.txt*, whereas if another dictionary such as Common Spoken English Corpus is used, *u* might drop into OOV. Hence, to a great extent, the accuracy of token types depends on the applied dictionary.

5. **Conclusion**

In this report, I primarily analysed how to effectively and accurately categorise and normalise tweet tokens. I found that not all OOV tokens require normalisation whereas it is true that some IV tokens ought to be normalised. Therefore, a better classifier should be applied to more accurately detect which words need to be normalised. For normalisation, my system is build on Peter Norvig's Spelling Corrector, Aspell Library as well as 2 and 3 grams context matching methodology. Via the evaluation, it can be seen that my system achieved the best performance in terms of both token types and normalisation. Furthermore, I found that the more grams used to extract context information, the higher accuracy rates could be achieved. However, due to the space constrains (up to 140 characters) of tweets, a gram threshold exists when n-grams context matching strategy is applied on short messages, which could be estimated to be 5 or more.

References

Bo Han and Timothy Baldwin. 2011. Lexical Normalisation of Short Text Messages: Makn Sens a #twitter. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 368–378, Portland, Oregon.

Justin Zobel and Philip Dart. 1995. Finding Approximate Matches in Large Lexicons. *Software—Practice and Experience, VOL. 25(3), 331–345.*

Justin Zobel and Philip Dart. 1996. Phonetic String Matching: Lessons from Information Retrieval

Max Kaufmann. 2010. Syntactic Normalization of Twitter Messages.

Martin Schierle, Sascha Schulz and Markus Ackermann. 2007. From Spelling Correction to Text Cleaning--Using Context Information. In *Proceedings of the 31st Annual Conference of the Gesellschaft fur Klassifikation e.V., Albert-Ludwigs-Univesitat Freiburg, pp. 397-404.*

Mark Davies. 2011. N-grams data from the Corpus of Contemporary American English (COCA). Downloaded from http://www.ngrams.info on April 19, 2013.

Casey Whitelaw, Ben Hutchinson, Grace Y Chuang and Gerard Ellis. 2009. *In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 890–899,* Singapore.

Kevin Atkinson. 2004. Aspell User Manual. http://aspell.net. Retrieved on April 19, 2013.