Evaluation of Statistical Text Normalisation Techniques for Twitter

Phavanh Sosamphan[1], Veronica Liesaputra[1], Sira Yongchareon[2] and Mahsa Mohaghegh[1]
[1]*Department of Computer Science, Unitec Institute of Technology, New Zealand*
[2]*Department of Information Technology & Software Engineering, AUT, New Zealand*
phavanh.ss@gmail.com, {vliesaputra, mmohaghegh}@unitec.ac.nz, sira.yongchareon@aut.ac.nz

Keywords: Text Mining, Social Media, Text Normalisation, Twitter, Statistical Language Models, Lexical Normalisation

Abstract: One of the major challenges in the era of big data use is how to 'clean' the vast amount of data, particularly from micro-blog websites like Twitter. Twitter messages, called tweets, are commonly written in ill-forms, including abbreviations, repeated characters, and misspelled words. These 'noisy tweets' require text normalisation techniques to detect and convert them into more accurate English sentences. There are several existing techniques proposed to solve these issues, however each technique possess some limitations and therefore cannot achieve good overall results. This paper aims to evaluate individual existing statistical normalisation methods and their possible combinations in order to find the best combination that can efficiently clean noisy tweets at the character-level, which contains abbreviations, repeated letters and misspelled words. Tested on our Twitter sample dataset, the best combination can achieve 88% accuracy in the Bilingual Evaluation Understudy (BLEU) score and 7% Word Error Rate (WER) score, both of which are considered better than the baseline model.

1 INTRODUCTION

More than 80% of online data, especially from Twitter, is unstructured and written in ill-formed English in such a way that users may not understand it very well (Akerkar, 2013). Compared to the size of other online texts (such as comments on Facebook), tweets are much smaller (< 140 characters). Because of the restriction on the maximum number of characters that can be sent in a tweet, tweets are commonly written in shorthand and composed hastily with no corrections.

Data cleaning has been a longstanding issue. However, the resultant state-of-the-art methods and approaches are still missing the mark concerning an effective solution for cleaning Web data (Han et al., 2013). The major difficulty is how to enhance the accuracy, effectiveness and efficiency of the data cleaning algorithms all at the same time. The more accurate techniques usually require a larger amount of time to clean the data.

Existing work on text normalisation is usually designed to address a specific problem in noisy texts. For example, Out-of-Vocabulary (OOV) words and abbreviations have been the focus of most attempts. Even though there are many methods that could tackle these problems, some noisy texts still cannot be identified and normalised. This is mainly due to misspelling and no contextual features being accessible for extraction.

In order to produce a correct English sentence that does not contain misspellings, repeated letters, abbreviations or OOV words, we studied and evaluated each of the existing methods and their all possible combinations to find the best one that can efficiently correct a tweet at the character level. Our evaluation is based on the two following criteria: accuracy and run-time efficiency. Accuracy can be measured using the BLEU (Bilingual Evaluation Understudy) score and the WER (Word Error Rate) values. Run-time efficiency is measured by the time spent in cleaning noisy tweets.

2 BACKGROUND

Noisy texts can be caused by the use of acronyms, abbreviations, poor spelling and punctuation, idiomatic expressions and specific jargon. In this

paper, we discuss existing normalisation techniques for the noises at the character-level.

Misspellings. Spelling corrector, developed by Norvig (2012), defines the conditional probability of a given word by finding the dictionary entries with the smallest edit distance from the query term. It achieved 90% accuracy at the processing speed of at least 10 words per second.

Abbreviations. Li and Liu (2012) extended the work of Pennell and Liu (2011) and proposed a two-stage approach using the Machine Translation (MT) model. Abbreviations were firstly translated to phonetic sequences which then translated back to In-Vocabulary (IV) words by using a dictionary to eliminate words that were not in the dictionary and kept N-best candidates. It received 83.11% accuracy in the top-20 coverage.

Repeated characters. Saloot, Idris, and Mahmud (2014) eliminated repeated letters from Malay tweets by basing them on patterns setup. Extra letters were eliminated when a token was detected as a non-standard word by tagging with IV words and a normalised token label. After a token with repeated letters was converted to word patterns, the regular expression module was used as a pattern finder to determine whether a token fitted into the patterns. Then repeated letters were deleted based on the match pattern.

OOV words. Gouws et al. (2011) constructed an unsupervised exception dictionary by using automatically paired OOV words and IV words. Through similarity functions, OOV words were identified based on the list of IV words and created the output as a word mesh that contains the most likely clean candidates for each word. Then the model grouped them as a set of confusion lattices for decoding clean output by using an n-gram language model from SRI-LM (Stolcke, 2002). This approach reduced around 20% in the WER score over existing state-of-the art approaches, such as a Naïve baseline and IBM-baseline.

3 APPROACH

Based on the existing research on character-level problems, we have found challenges as well as opportunities in normalising non-standard words. First, we have observed that high levels of annotated training data are required. Furthermore, majority of the existing methods are specially designed to handle a specific normalisation problem.

The goal of this research is to find the best normalization combination in order to 'normalise' an ill-formed tweet to its most likely correct English representation with the highest accuracy. We consider four noisy problems, which cause noisy tweets, including repeated characters, abbreviations, OOV words and misspelling words.

Data Preparation Before we try to normalise the tweets, five basic steps of data preparation are deployed. First, we replace all HTML entities to standard English characters. For instance, "&" is converted to "&" and "<" is converted to "<". All tweets are then encoded into to UTF-8 format. The third step is the removal of emoticons, URLs and unnecessary punctuations. ":", ".", ",","?", however, @usernames and #tags are not removed. Concatenated or run-together words are then split into individual words. For example, "RainyDay" is converted to "Rainy Day". Finally, the tweets are tokenised into word-level.

Dictionary Two types of dictionaries are used to normalise noisy tweets: "en_US" dictionary from Aspell library (Atkinson, 2004), and abbreviation dictionary taken from reliable online sources such as www.urbandictionary.com, www.noslang.com, www.abbreviations.com & www.internetslang.com.

Abbreviations are expanded by using a simple string replace method. Each word is converted to lower case before trying to find it in our abbreviation dictionary. If it encounters the matching one, the abbreviation is replaced with its expanded version, i.e. "u r sooooo gorgeuos 2nite" → "you are sooooo gorgeuos tonight".

Repeated Characters are then normalised by first removing repeated letters found in a word until only two letters remain ("u r sooooo gorgeuos 2nite" to "u r soo gorgeuos 2nite"). Next, it utilises J. McCallum (2014) spell corrector to correct the word. Hence, "soo" is corrected to "so".

Misspelled Words are corrected at the last stage of our approach. In this step, we utilise the Enchant dictionary (Perkins, 2014) as well as the two dictionaries mentioned above. Spelling correction is not necessary if the given word is present in the dictionary, and the word is returned. But if the word is not found, it will return a word in the dictionary with the smallest edit distance. For instance, "gorgeuos" is corrected to "gorgeous".

4 EXPERIMENTS

To evaluate the performance of each of the techniques and their combinations, we conducted two types of experiments. The first experiment contained two stages. Re-implementing existing normalisation methods represented the first stage in finding the best cleaning techniques for solving each problem (misspelled words, abbreviations, and repeated characters). The second stage involved combining techniques used to address each problem in a different order. This experiment was intended to help us find the best combination of techniques that could solve all problems. The second experiment used the same tweets dataset to prove that our best combination of normalisation techniques found in the first stage is better than the baseline model, in terms of accuracy and time efficiency, at normalising a noisy tweet into a clean and readable sentence. All techniques were implemented in Python and NLTK framework. The baseline model used is Text Cleanser by Gouws et al. (2011).

Dataset. The experiments used a dataset of 1200 tweets containing messages from popular celebrities in the entertainment area and the replies from their fans. The dataset contains 489 abbreviations, 152 words with repeated characters, and 375 misspelled words.

For our evaluation setup, we formed the datasets for each of our tests by manually normalising those 1200 tweets and creating four reference datasets. In the first reference dataset, we corrected all the abbreviations from the original tweets. For instance, if the original tweet was "That viedo is fuuunnnnyy LOL", in the first reference dataset (Ref_AB) the tweet became "That viedo is fuuunnnnyy laugh out loud." In the second dataset (Ref_RC), we corrected only the repeated characters. Thus, the tweet became "That viedo is funny LOL". In the third dataset (Ref_MSW), we corrected only the misspelled words, i.e. "That video is fuuunnnnyy LOL". In the last dataset (Ref_All), we corrected all of those cases, i.e. "That video is funny laugh out loud." To sum up, the first three reference datasets were used for evaluating each technique that was used to address each problem and the fourth reference dataset was used to evaluate the combined models against the baseline model, and our combination model against the baseline model.

Evaluation Metrics. BLEU and WER metrics are widely used as evaluation metrics for finding a normalisation method's accuracy. We use the iBLEU developed by Madnani (2011) and Gouws et al. (2011) WER evaluator. The efficiency of a technique is evaluated by the time that is required by a normalisation technique to perform a data cleaning procedure. Furthermore, a paired t-test was used to examine whether there is a statistically significant difference between the performances of each technique at the 95% confidence level.

4.1 Results from Individual Method

A comparison of the techniques that solve the same noisy problem on the same tweet dataset is required to find the best technique. The results of the first experiment are presented according to the type of noisy problems they are trying to solve and they are explained in detail as follows.

4.1.1 Detecting Abbreviations

Two techniques for normalising abbreviations are compared: DAB1 and DAB2. Similar to DAB2, DAB1 expands abbreviations by performing dictionary look-up. However, DAB1 did not convert each word to lowercase prior to the look up. Ref_AB is used as the reference dataset in the BLEU and WER score calculations.

Both techniques achieve more than 90% in the BLEU score, less than 4% WER value and spent only 30 seconds. However, both are not able to resolve abbreviations that require context at sentence-level, which is out of the scope of this research. For example, "ur" in "I love that ur in" is currently resolved to "your" instead of "you are". Although our abbreviation dictionary has defined "ur" with two separate meanings, neither technique can select the right meaning of the given sentence. There is no significant difference in performance between DAB1 and DAB2, but DAB1 yields lower accuracy as it cannot detect an abbreviation that contains the upper case letter (i.e. "Wld") due to our dictionary merely having the reference abbreviations that have the lower case letters (i.e. "luv").

4.1.2 Removing Repeated Characters

Three variants of Perkins (2014) techniques for removing repeated characters are evaluated: RRC1, RRC2 and RRC3. RRC2 is Perkins (2014) original approach, where repeated letters in a word are removed 1 letter at a time. Each time a letter is deleted, the system performs a WordNet lookup. RRC2 will stop removing repeated characters if WordNet recognises the word. Instead of using

WordNet, Enchant dictionary is used in RRC3. In RRC1, repeated letters in a word are removed until only two letters remain and J. McCallum (2014) spell corrector is used to correct the resulted word.

Although our best combination model seems crude compared to the other two methods, on our Ref_RC dataset, we found that RRC1 (83.65%) is significantly better than RRC2 (79.76%) and RRC3 (80.13%) in terms of BLEU score. RRC1 (25 sec) and RRC2 (1 min) are also significantly faster than RRC3 (22 mins). However, there is no significant difference in the WER score among the three methods (9%-11%). We have also noted that Enchant dictionary contains more words than WordNet as such in some cases RRC3 performs better than RRC2.

4.1.3 Correcting Misspelled Words

To find out the best technique for spelling correction, we have compared SC1 with Norvig (2012) spelling corrector (SC2) and TextBlob (SC3) Python's library for correcting misspelled words.

On our Ref_MW dataset, SC1 (79.88% BLEU and 12.40% WER) is significantly better than SC2 (68.47% BLEU and 17.97% WER) and SC3 (69.39% BLEU and 16.68% WER) in terms of BLEU and WER scores. SC1 utilizes edit distance method to replace the misspelled word with the word in the Aspell or Enchant dictionary that has the lowest edit distance to the misspelled word. Hence, it can handle words written in a plural form, e.g. "skills" will not be resolved to "skill".

There is no significant difference between SC2 and SC3 in both BLEU and WER scores, but the run-time performance is significantly different. While SC2 spends only 4 minutes correcting misspelled words, SC3 spends 21 minutes. Given a large amount of input, SC3 will take considerably longer to process the whole input.

4.2 Results from Combination of Techniques

From the previous section, we know that the best techniques for resolving abbreviations, misspelling and repeated characters are DAB2, SC1 and RC1 respectively. Next, we set up an experiment to identify the best combination of DAB, SC and RRC cleaning techniques and the best order to execute those techniques. Thus, we know which type of problems should be addressed first and which ones should be addressed last.

We have set up and evaluated a total of 108 combinations using the BLEU, WER, and time criteria. Ref_All is the reference dataset used for calculating the BLEU and WER score for each combination of techniques. The results of this experiment are organised according to the execution order of each technique. As such, there are six groups of combinations.

4.2.1 RRC → DAB → SC

Overall, this group performs well with promising results in the BLEU, WER and time. The outstanding combination is RRC1 → DAB2 → SC1. This combination achieves the highest BLEU score (88.51%), the lowest WER value (7.14%), and spends only 2 minutes and 55 seconds. On the other hand, some techniques (i.e. RRC3 → DAB1 → SC3 and RRC3 → DAB2 → SC3) spend a long period of time normalising noisy tweets, due to fact that they combine the techniques (RRC3 and SC3) that individually consume a lot of time cleaning repeated letters and misspelled words. Furthermore, RRC2 → DAB1 → SC2 and RRC2 → DAB1 → SC3 are the combinations that achieve the lowest BLEU score (~73%) and the highest WER values (>14%).

4.2.2 RRC → SC → DAB

The outstanding combination in this group is RRC1 → SC1 → DAB2, which achieves the highest BLEU score (84.41%), the lowest WER value (9.60%), and spends only 2 minutes and 55 seconds. However, RRC1 → SC1 → DAB2 still achieves low accuracy when compared with RRC1 → DAB2 → SC1. In RRC1 → SC1 → DAB2, SC1 considers some of the abbreviations as misspelled words. For example, SC1 corrects "deze" to "daze" instead of "these", "whatchu" to "watch" instead of "what are you". Some techniques (i.e. RRC3 → SC3 → DAB1 and RRC3 → SC3 → DAB2) are very slow in processing normalisation and their accuracy is still not high enough.

4.2.3 DAB → RRC → SC

The outstanding combination is DAB2 → RRC1 → SC1, which achieves the highest BLEU score (88.55%) and the lowest WER (7.10%) and spends only 2 minutes and 55 seconds. DAB1 → RRC2 → SC3, DAB1 → RRC2 → SC3, DAB1 → RRC3 → SC2 and DAB1 → RRC3 → SC3 are the combinations that achieve the lowest BLEU score (~74%) and the highest WER values (>14%). On the other hand, the combination that spends the longest

time processing normalisation in this group is DAB2 → RRC3 → SC3 (43minutes and 30seconds).

When comparing this group with the previous ones, we observe that handling abbreviations as a first step can ensure that all given abbreviations have been checked and replaced by their formal format. For example, "deze" to "these" and "whatchu" to "what are you". Then, the elimination of repeated characters has made sure that words containing repeated characters are addressed thoroughly. Consequently, when these types of noisy tweets are cleaned, they have been returned to the normalised tweets dataset without any spelling correction from SC1. These first two steps have not only increased the capacity of the spell corrector to deal with only incorrect words, but also can normalise the final output with the highest accuracy. Hence, DAB2 → RRC1 → SC1 is better than RRC1 → SC1 → DAB2 and RRC1 → DAB2 → SC1.

4.2.4 DAB → SC → RRC

DAB2 → SC1 → RRC1 performs better than other combinations in this group, with an 83.50% in the BLEU score and 9.69% in the WER value. DAB1 → SC2 → RRC3, DAB1 → SC3 → RRC2 and DAB1 → SC3 → RRC3 are the combinations that achieve the lowest BLEU score (~69%) and the highest WER values (>16%). In addition, the DAB2 → SC3 → RRC3 technique spends nearly 44 minutes normalising noisy tweets.

However, the performance of DAB2 → SC1 → RRC1 is not good enough in comparison to DAB2 → RRC1 → SC1. After expanding abbreviations into their formal form, some words containing repeated characters initially have been corrected by SC1 before sending to RRC1. For example, "sooo" will be corrected to "soon" instead of removing repeated letters to get "so". Hence, most of the words that contain repeated characters have been transformed into another word, which causes a change in the final output of normalised tweets as well as having an effect on the accuracy.

4.2.5 SC → RRC → DAB

The outstanding combination is SC1 → RRC1 → DAB2, which achieves the highest BLEU score (86.94%) and the lowest WER value (8.30%) and spends 2 minutes and 55 seconds processing normalisation.

However, we are surprised by the results of this combination when we compare it with DAB2 → SC1 → RRC1. We find that the ill-formed words identified as abbreviations are not treated by SC1 at the first stage; only misspelled words are corrected. Thus, abbreviations and repeated letters are normalised by the proper techniques. The normalised results contradict with the fourth combination group, especially, the normalised result from the DAB2 → SC1 → RRC1 technique. It indicates that the original format of some words containing repeated letters has been changed by SC1.

While SC1 → RRC1 → DAB2 is the best combination in the group, SC3 → RRC2 → DAB1 and SC3 → RRC3 → DAB1 are the techniques that achieve the lowest BLEU score (~70%) and the highest WER values (>16%). On the other hand, the combinations of SC3 → RRC3 → DAB1 and SC3 → RRC3 → DAB2, which spent the longest run-time in normalising than any other combination, perform better than SC3 → RRC2 → DAB1 on both BLEU score and WER value.

4.2.6 SC → DAB → RRC

The outstanding combination of this group is SC1 → DAB2 → RRC1. This combination achieves 87.90% in the BLEU score and 7.40% in the WER. Based on the normalised result generated from SC1 → DAB2 → RRC1, we find that both abbreviations and repeated characters are not identified as misspelled words and corrected by SC1 at the first stage. Thus, noisy words that have been ignored by SC1 are detected and normalised by DAB2 and then RRC1 as a latter technique.

SC3 → DAB1 → RRC2 and SC3 → DAB1 → RRC3 are the combinations that achieve the lowest BLEU score (~71%) and the highest WER values (>15%). In addition, both SC3 → DAB1 → RRC3 and SC3 → DAB2 → RRC3 have spent nearly 44 minutes normalising noisy tweets.

4.3 Comparison to Baseline Model

Based on our experiment results presented in Section 4.2 and shown in Table 1, we can see that the best normalisation techniques and their order is DAB2 → RRC1 → SC1. To examine how well the best combination found can detect and convert a noisy tweet into an accurate English sentence, we compare its performance with the performance of Text Cleanser (TC) developed by Gouws et al. (2011). We chose TC as the baseline model because it claimed that the system can handle all the types of noisy words that we are trying to normalise, which they consider as OOV words, and because the system is open source.

Table 1: The group of the best combinations.

Model	BLEU (%)	WER (%)
RRC1 → DAB2 → SC1	88.51%	7.14%
RRC1 → SC1 → DAB2	84.41%	9.60%
DAB2 → RRC1 → SC1	**88.55%**	**7.10%**
DAB2 → SC1 → RRC1	83.50%	9.69%
SC1 → RRC1→ DAB2	86.94%	8.30%
SC1 → DAB2 → RRC1	87.90%	7.40%

Table 2: Comparison between the best combination model and the baseline model.

Model	BLEU (%)	WER (%)	Time
Baseline (TC)	63	38.22	3mins
Our best combination model	88.55	7.10	2mins55sec

As can be seen in Table 2, our best combination performs better than TC in terms of achieving higher accuracy on our Ref_All dataset. Our model achieves 88.55% of the BLEU score and 7.10% in the WER score, while TC achieves 63% in the BLEU score and 38.22% in the WER score. In terms of time efficiency, although both models spend a short amount of time on normalisation in the different operating systems, our best combination model is faster than the baseline model. The paired-t-test showed that the best combination's BLEU and WER values are statistically significant when compared with TC.

TC was unable to resolve some abbreviations and misspelled words, and incorrectly replaced an already correct word with another word. For example, "worst" to "wrest" "deez" to "diaz", and "conections" to "conditions". A run-on word such as "im" being replaced with "i am" – is another problem that could not be detected and handled.

Despite its positive side, our best combination found cannot correctly normalise tweets when there is a white space in a given word (e.g. "I' m"). The best combination recognises it as a noisy word due to its absence in the dictionary lookup and reference text. The "I' m" is transformed into "I' million" instead of "I' m". The best combination's tokenising algorithm treats a white space as a token separator. Hence, the "I" is recognised by dictionary lookup while the "m" is not. The "m" is recognised as an abbreviated term which means "million" according to our abbreviation dictionary. Therefore, this minor issue is another factor that reduces the accuracy of our combination model's performance.

5 CONCLUSION

It has been established that data cleaning is a crucial part of text pre-processing. Therefore, a noisy tweet needs to be normalised to a cleaned sentence to provide high quality data. Three main issues in noisy tweets have been considered in text normalisation. Existing techniques have been evaluated with the same dataset in order to identify and select the best combined techniques to deal with abbreviations, repeated characters, and misspelled words. Based on our experiments, our best combination not only provides the highest score of the BLEU score and the lowest WER, but also generates sentences with minimum efficiency; thus the cleaned texts can be effectively used in sentiment analysis and other NLP applications.

REFERENCES

Akerkar, R. (2013). *Big data computing*. Florida, USA: CRC Press.

Atkinson, K. (2004). GNU Aspell. Retrieved from http://aspell.net/

Gouws, S., Hovy, D., & Metzler, D. (2011). Unsupervised mining of lexical variants from noisy text. *Proc. of the First workshop on Unsupervised Learning in NLP* (pp. 82-90).

Han, B., Cook, P., & Baldwin, T. (2013). Lexical normalization for social media text. *ACM Trans. Intell. Syst. Technol., 4*(1), 1-27.

Li, C., & Liu, Y. (2012). Normalization of Text Messages Using Character-and Phone-based Machine Translation Approaches. In *INTERSPEECH* (pp. 2330-2333).

Madnani, N. (2011). iBLEU: Interactively debugging and scoring statistical machine translation systems. *Proc. for the ICSC Conf. on* (pp. 213-214).

McCallum, J. (2014). Python 3 Spelling Corrector. From https://pypi.python.org/pypi/autocorrect/0.1.0

Norvig, P. (2012). How to Write a Spelling Corrector. From http://norvig.com/spell-correct.html

Perkins, J. (2014). *Python 3 Text Processing with NLTK 3 Cookbook*. Birmingham, UK: Packt Publishing.

Saloot, M. A., Idris, N., & Mahmud, R. (2014). An architecture for Malay Tweet normalization. *Information Processing & Management, 50*(5), 621-633.

Stolcke, A. (2002). SRILM-an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing* (pp. 257-286). INTERSPEECH.