COMP90049 Project1 Report: Lexical Normalization Based on Tweets

Ye Yang

1. Introduction

With the popularity of online social media in recent years, mining data from social media becomes an efficient approach to help enterprises understand the opinions on their products and services from the potential customers. Sentiment analysis, a machine learning application based on natural language processing (NLP) techniques, is used to extract people's opinions and feelings from the content of their social media. The quality of corpus is one of the key factors to the success of such algorithms (Gupta & Joshi, 2017).

A great deal of research has been conducted to adjust standard NLP pipeline to be able to handle the stream of non-canonical language, like tweets which contain a lot of non-standard and misspell words etc., by using lexical normalisation. For example, Supranovich and Patsepnia (2015) proposed a CRF model combined with the lexical normalisation approach for English tweets and obtained an attracting result with a score above 0.8 on Precision, Recall and F-score respectively.

In this report, three typical approximate string matching approaches, including Global Edit Distance, N-grams and Soundex, are implemented based on Python to build a text normalisation system for non-canonical lexical auto-correction.

2. Dataset

The dataset established by Baldwin et al. (2015) contains a misspell lexical list, a dictionary of standard English words and a correct token list which came from tweets (see Table 1). Different approximate string matching approaches are used to find the best match words, which respect to a collection from the dictionary, for each token in the misspell set and then the prediction results are evaluated by comparing with the corresponding canonical token in the correct set.

Corpus	Features
misspell	10322 tokens (non-canonical)
dictionary	370099 words (standard English)
correct	10322 tokens (canonical)

Table 1: The Corpus Dataset

3. Methodology

The possible causes for misspelling problem could be divided into two main categories corresponding to typing and pronunciation respectively.

Intuitively, the first one could be associated with careless mistyping which are caused by adjacent keys on keyboard, alphabet sequential errors or missing letters etc. (Flor & Futagi, 2012). In addition, the corpus from tweets would also contain some non-English vocabularies which might be similar but not exactly the same as original English words. In these cases, the length of a misspelled word would be similar to that of the corresponding canonical form. Therefore, the Global Edit Distance and N-grams would be two competitive approaches for this task.

3.1. Global Edit Distance

In this project, the edit distance library is provided by Haapala (2014) and the Levenshtein parameter for GED is (m, i, d, r) = (0, 1, 1, 1). A lexical result list is generated to collect all the candidate tokens with the minimum edit distance from the dictionary corpus.

Usually, the proportion of mistyping words are expected to be low in the whole corpus. Hence, for most proper spelling tokens, GED algorithm is predicted to work well. However, since the dataset contains numerous informal languages and the volume of the dictionary is very large and made up of standard English vocabularies, the result collection for one misspell token could contain many candidates if non-alphabet symbol occurs. For example, as for the tokens containing numbers like 'no1234567', the effectiveness is predicted to be low. In addition, the algorithm would fail in the case of non-canonical abbreviation tokens, such as 'plz' for 'please' and 'u' for 'you' etc. Therefore, in terms of the system performance for the whole corpus, the Recall is expected to be high, whereas the Precision would be much lower.

3.2. N-grams

The N-grams library is provided by Poulter (2017) and the parameter is N = 2 with padding character '$' before and after the target token. In addition, the library uses similarity (between 0 to 1) instead of distance to measure the difference between two tokens. According to the documentation, the definition of similarity in this library is as below:

$$\text{similarity} = \frac{a^e - d^e}{a^e}$$

where *a* is the total n-grams
d is the different n-grams
e is the warp (float in 1.0 ... 3.0) used to increase the similarity of shorter string pairs and its default value is 1.0

Since the misspell corpus would contain numbers or other non-English symbols, it should be careful when the minimum similarity equals 0 (e.g. the misspell token is '123'), and in this case, it just returns an empty candidate list instead of the entire dictionary corpus. Considering the features of N-grams algorithm, it is expected to have a good performance on both proper typing and careless mistyping tokens. However, it may not work well on similar languages (e.g. German or French) as well as non-canonical abbreviation expressions due to the limitation of the dictionary corpus. Owing to the short length of retrieved collection for this algorithm, the system Precision is expected to be higher than GED, while the Recall would be similar.

3.3. Soundex

Another possible reason for misspelling would be associated with phonetics. For instance, it is very common that people sometimes forget how to spell a word, but they should remember its pronunciation in general and they would try to spell it in this way. Although it is not always possible to make a perfect prediction of word spelling based on its sound, the phonetics matching approach is still a considerable potential solution for misspelling correction.

This project invokes a Soundex library called jellyfish developed by Turk (2019), which implements the standard American Soundex Coding System (see Table 2).

Code	Alphabet
0	a, e, i, o, u, y, h, w
1	b, f, p, v
2	c, g, j, k, q, s, x, z
3	d, t
4	l
5	m, n
6	r

Table 2: The American Soundex Code Table

It should note that the Soundex algorithm is only suitable for alphabetical string, which means that it does not apply to the tokens with numbers or other symbols. Therefore, it is necessary to check whether a token is composed of pure letters or not, and if not, just return an empty candidate list for the token. Obviously, the returned candidate set would be very large, especially for short words or the tokens with common prefixes, which would result high Recall and low Precision scores.

4. Evaluation Metrics

This report adopts three main evaluation indexes, including Precision, Recall and F-score, for system evaluation and result analysis. Precision and Recall usually present an inverse relationship, which means that, by reducing one of them as a cost, it is possible to enhance the other. Since the two indicators would be very different in this task due to the features of different approximate string matching approaches, the F-score, a combination of these two indexes, is introduced as a more reliable indicator. The definitions of these three evaluation indexes are shown as below:

$$\text{Presicion} = \frac{\textit{number of correct results retrieved}}{\textit{total number of results retrieved}}$$

$$\text{Recall} = \frac{\textit{number of correct results retrieved}}{\textit{total number of possible correct results}}$$

$$\text{F} - \text{score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. Result Analysis

The outcomes of system Precision, Recall and F-score for the three approximate string matching approaches applied in this project are shown in Figure 1 and Table 3.

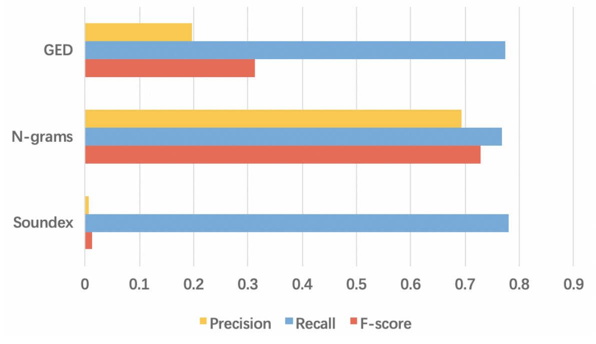

Figure 1: The Results for Three Approaches

	Precision	Recall	F-score
GED	0.1962	0.7738	0.3130
N-grams	0.6932	0.7688	0.7291
Soundex	0.0067	0.7807	0.0132

Table 3: The Results for Three Approaches

5.1. GED Result Analysis

From the Table 3, it can be seen that the GED approach obtains a good score on Recall, while the Precision performs less than satisfactory. Some result samples are shown in the following Table 4 ("✔" for correct, "✘" for wrong).

Misspell	Prediction	Correct	Result
wait	wait	wait	✔
today	today	today	✔
tommorow	tommyrot	tomorrow	✘
	tomorrow		✔
posible	posable	possible	✘
	possible		✔
neighbour	neighbour	neighbor	✘
favourite	favourite	favorite	✘
dont	dont	don't	✘
u	u	you	✘
no	no	know	✘
betetr	betear	better	✘
125th	airth	125th	✘
	… (>150)		...
	wroth		✘

Table 4: The Sample Results for GED

It can be seen that, for the proper spelling words (e.g. "wait" and "today") and the cases of missing letters or small order errors (e.g. "tommorow" and "posible"), GED method performs very well. However, it fails when the tokens contain some numbers, such as "125th". The algorithm returns a great number of candidates (without any correct ones) in this case, which would result a low Precision. The spelling rules (e.g. "neighbour" and "favourite") and non-canonical abbreviation (e.g. "u" for "you" and "no" for "know") would also affect the Precision and Recall. In addition, in the case of "betetr", the algorithm returns a really infrequent word "betear", whereas the correct form is a very common word "better". A potential solution for these cases might be using an expended dictionary combined with weight values evaluated by frequency.

5.2. N-grams Result Analysis

As for the result of N-grams approach, all the three evaluation indexes achieve about 0.7 to 0.8, which illustrates that this algorithm performs well on such a non-canonical misspelling correction task. Some sample outcomes are given in Table 5.

Misspell	Prediction	Correct	Result
scanner	scanner	scanner	✔
swearin	swearing	swearing	✔
ill	ill	i'll	✘
andrew	andrew	andrew	✔
wrked	warked	worked	✘
	worked		✔
neighbour	neighbour	neighbor	✘
aegyo	aegyrite	aegyo	✘
viru	virtu	virus	✘
	virus		✔
u	u	you	✘
4	-	for	✘
t6	t	t6	✘

Table 5: The Sample Results for N-grams

These sample results indicate that N-grams works very well on both accurate spelling or careless mistyping cases. However, in terms of some exotic vocabularies, like the "aegyo" which comes from Korea, it would fail. In addition, for the same reason of dictionary limitation, it has poor performance on numbers, abbreviation or words based on different spelling rules. It is worth noting that the N-grams approach returns candidate sets with very limited length in general, which in part results a relatively high Precision score.

5.3. Soundex Result Analysis

As predicted, Soundex method obtains a high Recall but a very low Precision score as well as F-score. Table 6 provides some sample outcomes.

Misspell	Prediction	Correct	Result
practice	parasite	practice	✘
	practice		✔
	… (>500)		…
	preceding		✘
125th	-	125th	✘
at	at	at	✔
	… (>50)		...
	auto		✘

Table 6: The Sample Results for Soundex

The Soundex algorithm returns a very large candidate collection for each token due to the features of its coding rules. This results an extremely low Precision as well as F-score.

6. Conclusions

Overall, this report implements a misspelling correction system for lexical normalisation based on three main approximate string matching algorithms. After comparing the performance of these three approaches based on a non-canonical corpus from tweets, the N-grams approach is proved to be a competitive potential solution for lexical normalisation task with a 0.73 F-score. In order to further enhance its performance, a combination of two or more fuzzy string matching algorithms would be a valuable attempt in the future.

7. References

Baldwin, T., de Marneffe, M. C., Han, B., Kim, Y. B., Ritter, A., & Xu, W. (2015). Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text* (pp. 126-135).

Chrupała, G. (2014). Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Vol. 2, pp. 680-686).

Flor, M., & Futagi, Y. (2012, June). On using context for automatic correction of non-word misspellings in student essays. In *Proceedings of the seventh workshop on building educational applications Using NLP* (pp. 105-115). Association for Computational Linguistics.

Gupta, I., & Joshi, N. (2017, December). Tweet normalization: A knowledge based approach. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)* (pp. 157-162). IEEE.

Haapala, A. (2014). Python-Levenshtein. Retrieved from https://github.com/ztane/python-Levenshtein.

Poulter, G. (2017). Python-ngram. Retrieved from https://github.com/gpoulter/python-ngram.

Supranovich, D., & Patsepnia, V. (2015). Ihs_rd: Lexical normalization for english tweets. In *Proceedings of the Workshop on Noisy User-generated Text* (pp. 78-81).

Turk, J. (2019). Jellyfish. Retrieved from https://github.com/jamesturk/jellyfish.