COMP90049 Project 1: Lexical Normalisation of Twitter Data

792320 Sirui Yan

1 Introduction

The goal of this report is to implement lexical normalisation of Twitter data. This report will assess and analyze the performance of some basic approximate string matching methods and discuss how to optimize the method to improve the performance.

The dataset we use is actual tweets posted to Twitter and is provided by Bo Han (2011). It includes 549 tweets, 8841 case-folded tokens derived from these tweets and a reference dictionary. This report uses labelled tokens to evaluate the performance of lexical normalisation.

2 Evaluation Metrics

The performance of lexical normalisation will be evaluated in the following three aspects:

- Precision: proportion of correct canonical form among attempted responses
- Recall: proportion of tokens with correct canonical form
- Run time: time it takes in total to normalise 8841 tokens

3 Original Methods

Three original approximate string matching methods are used to normalise tokens from tweets.

3.1 Global Edit Distance

We use the Levenshtein parameter as global edit distance parameter, which is related to the number of edits required for conversion. The library is from Apache Software (2016) and the result is shown in Table 1.

Precision	45.05%
Recall	74.12%
Run time	7.19 min

Table 1: Result of Levenshtein Distance

3.2 Local Edit Distance

Local edit distance finds the best substring match and we use the library from Debatty (2017). The result is shown in Table 2.

Precision	42.09%
Recall	74.65%
Run time	15.29 min

Table 2: Result of Local Edit Distance

3.3 Phonetics

Both global edit distance and local edit distance string matching in terms of spelling, so I select soundex, which matches stirng in terms of phonetics. Soundex library I use is from Apache Software (2017). The result is shown in Table 3.

Precision	1.03%
Recall	74.85%
Run time	10.07 min

Table 3: Result of Soundex

3.4 Evaluation

All methods have similar recall, but they perform well in normalising different types of tokens, which will be further discussed in next section. Levenshtein distance and local edit distance have similar performance in precision, while soundex has a rather low recall because many tokens have same soundex encoding. Levenshtein distance performs best in terms of run time.

4 Analysis

4.1 Token in the dictionary

If the token is in the reference dictionary, precision and recall of Levenshtein distance and local edit distance are 100% because they can find the only canonical form, which is the same as

the token. Although soundex can find the right canonical form, it may provide multiple canonical forms for one token since many tokens in the dictionary can have the same soundex encoding. Thus, its recall is 100% but precision is not 100%.

4.2 Token not in the dictionary

If the token is not exactly the same as at least one of the token in the dictionary, all methods would find tokens in the dictionary that are most similar to the token as it canonical form. I notice there are following typical circumstances:

4.2.1 Misspelling

Tokens that are misspelled such as "nd"(and), "hve" (have), "blck" (black), "rememberr" (remember), "appearal"(apparel), "culd"(could), "expierince"(experience), "tlk"(talk), "possesion"(possession), "coult"(could), "serios" (serious), and "anniversery"(anniversary) are out of the dictionary.

Misspelled letters will not affect Soundex encoding and edit distance on a large scale most of the time. Thus, all three methods have a good performance in terms of recall, but Soundex has a much lower precision.

4.2.2 Similar Pronunciation

I notice that people often shorten the length of words while keeping similar pronunciation on Twitter. Using "-in" or "-n" instead of "-ing" is one typical example of this condition. For example, "trippn" for tripping, "puttn" for putting, "wantin" for "wanting". Other example includes "skool" for school, "nxt" for next, "tuu" for to, "tomoroe" for tomorrow, "somewre" for somewhere, etc.

Soundex performs well in finding the canonical form in this case because the encoding of the token doesn't change. The other two methods perform well when the token does not have many changes in spelling such as "thinkin" for thinking, but may be not able to find the right canonical form when there are many changes in spellling such as "sumtin" for something.

4.2.3 Plurals or Tenses

Plural none such as calls, heads, folks, members, plates and tokens with different tenses such as coming, putting, plays, gets and liked cause the drop of precision and recall for all three methods because the dictionary only contains singular form.

4.2.4 Repetition of letters

People always use repetition of letters in a word to express their strong feelings in tweets, such as "yesss", "damnnn", "sweeeet", "ohhhh" and "sooooo". These words are not the same as tokens in the dictionary but they are if repetition of letters are removed.

Soundex is guaranteed to find the canonical form because duplicates of sound encode are removed in the translating process and repetition of letters does not change its encoding. The other two methods can find the canonical form if the letter does not repeat many times.

4.2.5 Acronym

Acronyms are frequently used in text-based communication such as tweets but are not in the dictionary, which is one cause of the decline of precision and recall for all three methods. Acronyms in the token file includes "rt"(retweet), "omg"(oh my god), "lol"(laughing out loud), "lmao"(laughing my arse off), "idk"(I don't know), "imo"(in my opinion), "btw"(by the way), etc. Some of them appear frequently in tweets: "rt" appears 105 times, "lol" appears 83 times, "lmao" appears 18 times and "smh" appears 17 times.

4.2.6 Short Form or Variant Spelling

Short form or variant spelling also leads to the decrease of precision and recall for all methods. Short form or variant spelling of one word in the file includes "ppl"(people), "pic"(picture), "lil"(little), "congrats"(congratulations), "msg"(message), "prolly"(probably), "fav"(favorite), "u"(you), "c"(see), "y"(why), "2"(to), "4"(for), etc. "u" appears most often with more than 150 times in the token file, "ppl", "2", "cuz" and "pic" appears 20, 16, 12, 9 times respectively.

There is also the short form of two words such as "i'm" and "can't". They are also widely in tweets. "i'm" appears 70 times, "don't" appears 36 times, "ain't" appears 17 times, and "can't" appears 12 times.

4.2.7 Informal Words

Twitter is an informal social platform, so people always use spoken English such as "shit", "haha", "nigga", "wanna", "gotta", "gonna" and "online" in tweets. These words are not in the dictionary and reduce the precision and recall for all three methods.

4.2.8 Proper Noun

Proper noun such as "chicago", "farmington", "hbo", "spongebob", "espncricinfo", and "em-

inem" can lower precision and recall for all methods because they are not in the dictionary.

5 Modification to original methods

Levenshtein distance has a good performance in finding canonical form of tokens in dictionary and most mispelled words among these three approximate string matching methods in terms of precision, recall and run time. Soundex performs well in terms of precision in finding shortened tokens with similar pronunciation and words with repetitive letters as we discussed in 4.2.2 and 4.2.4. Thus, we can find the canonical form using Levenshtein distance first. If we cannot find a token within a threshold by Levenshtein distance, we can find canonical form using soundex. To improve the recall of soundex, we can combine it with Levenshtein distance by finding the token with similar pronunciation and low Levenshtein distance as predict canonical form.

We can also improve precision and recall by adding frequently-used acronym, informal words and short form to the dictionary. Recall and precision are expected to increase due to the increasing probability of finding the right canonical form.

The short form of two words can be added in the dictionary directly while the short form of one word can't because their canonical forms are not the same as the token. For short form or variant spelling of one word, we can first put short form of these tokens in the dictionary and then create a new dictionary with the short form and the canonical form. If the token we find as canonical form is in the new dictionary, we can convert it to its real canonical form.

To summarise, I made following modifications to original method according to the analysis:

- Combining Levenshtein distance with soundex
- Optimizing the dictionary by adding frequently-used acronym, informal words, short form
- Creating a new dictionary for short form and variant spelling

The result of modified lexial normalisation method is shown in Table 4.

We can see from the result that both precision and recall improve compared with original methods, which is consistent with our analysis. Run time is only a little slower than Levenshtein distance and much faster than local edit distance and soundex.

Precision	60.82%
Recall	82.81%
Run time	7.91 min

Table 4: Results of improved method

6 Improvements

To further optimizing lexical normalisation method, the following improvements can be considered:

- Optimizing the dictionary by adding frequently-used tokens on Twitter and removing uncommon tokens on Twitter
- Processing plurals and words with different tenses
- Processing proper noun
- Considering contexts in tweets

Dictionary plays an important role in the lexical normalisation process. A dictionary adjusted to Twitter data would be helpful in raising precision and recall.

One way to processing plurals and tenses is converting plurals and tokens with different tenses into singulars before normalisation and convert it back after. It is notable that local edit distance can find the corresponding singular, so using local edit distance to find the singular and then convert it to plural or different tenses might be an alternative way to solve this problem.

We can consider adding proper noun into the dictionary. However, there are a huge amount of proper noun and their frequencies of occurrence are not too high. Treating token with the first letter capitalized but are not the first word in the sentence as a proper noun before tweets being processed may be a more effective way to solve this problem.

The canonical of one token might be different in different contexts. For example, "2" may mean literally in "2yr" or "to" in "ride back 2 school", we need contexts to distinguish its canonical form.

7 Conclusions

This report evaluates the performance of Levensten distance, local edit distance and soundex in

lexical normalisation of Twitter data and analyse reasons that may cause the drop of recall and precision. Three modifications are made to improve the performance according to the analysis and are proved to be feasible.

References

Timothy Baldwin Bo Han. 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, USA.

Thibault Debatty. 2017. https://github.com/tdebatty/java-string-similarity.

Apache Software. 2016. Apache lucene. https://lucene.apache.org/core/.

Apache Software. 2017. Apache commons. https://commons.apache.org.