COMP90049 Project 1 Report:

*very tweetz. such non-canonickal. Amayze*

1 Introduction

The aim of this project is to study different spelling correction methods. Three documents are given as datasets, including misspell, dictionary and correction [1]. The short messages which are extracted from social media Twitter are documented in file misspell as tokens, meanwhile there is the corresponding canonical form of each short message in file correction. The methodology that we deploy in the program will compare each token with the the words or short messages in file dictionary, to predict the candidates for the token.

However, the main problem of this project is that the total number of tokens in the file misspell 10322, according to the calculation of length of file. To predict the best matches for numerous tokens, it is inevitable to look through every single short message in dictionary collection. The workload would be huge, and usually takes around 5-9 hours to get the results for one parameter changing test. Therefore, the pre-process will be applied to produce a shortcut dictionary which contains candidate(s) for each token, then the original token in the file misspell can look up the best match in a faster way.

2 Evaluation

To differentiate the best methodology, effectiveness and efficiency are important. However, as calculation time is beyond hours due to large collection of tokens, the difference of actual calculation time for each method is subtle. Efficiency of time is not taken into consideration.

2.1 Recall

The definition of recall in this project, is that calculate the number of correct results returned out of total number of possible correct results which is the number of tokens in the case each token has one corresponding correct form. This selected because it shows how the method works in this system. With higher Recall, the method can be more likely to pick up best match for the token accurately.

2.2 Precision

The second evaluation standard is precision which is calculated by the number of correct results returned out of total number of results returned. Precision is a problem of efficiency, because a system with higher precision normally returns less predictions for one token to the users, of course accuracy needs to be guaranteed. In the condition of that, less choice means this method works efficiently in the system.

3 Methodology

It is necessary to calculate the material similarity of tokens and potential candidate strings when do spelling correction [2][3]. The neighbors with higher similarity would be possible canonical words. In this project, edit distance and N-gram are selected because the data set is based on the text extracted from Twitter, which has less relation with word pronunciation.

3.1 Global Edit Distance

Firstly, each token will be separated into single characters for the future calculation. According to Levenshtein distance [4], global edit distance(GED) contains four parameters to calculate the similarity between each pair of strings, which are 0 for match, +1 for replacement one single character, +1 for delete or insert one character. The lower score the string gains, the more similar with the token. Thus the strings with lowest scores will be returned as predictions after comparison.

3.1.1Modified Global Edit Distance

To improve the efficiency of the program, some modifications are applied to the parameters. Considering the most internet users are prone to simplify the expressions on the social media. For example, people usually use “rockin” instead of “rocking”. Also there is another circumstance, that is people are likely to omit some characters when they are tying, such as “mesge” in the misspell file with corresponding correct word “message”. Above all, deletion and replacement weigh less than match and insertion when comparing. In the assumption, +1 is given to match, equally -1 is given to insertion. Score -5 is given to deletion and replacement. The strings with highest scores will be returned.

3.1.2 Results

The results are shown below:

|  |  |  |
| --- | --- | --- |
|  | GED | Modified GED |
| Correct strings returned | 7987 | 7926 |
| Total prediction | 40717 | 14155 |
| Recall | 0.7738 | 0.7678 |
| Precision | 0.1962 | 0.5599 |

Table 1: Results returned of GED and Modified GED

Figure 1: The comparison of efficiency

After modification, there is a significantly drop in prediction numbers. The precision nearly increases three times than the original GED, which means the custom parameters do contribute to the efficiency by increasing the threshold of candidates. However, there is a slight drop of recall for modified GED. Around 60 correct candidates are missed, this is because insertion becomes more important at the arbitration than deletion and replacement to predict a correct word. For example, for the word “youu” in misspell file, it gains more scores comparing with a long word “youngun” because the step of insertion. But in the original GED, with equal weight for insertion, deletion and replacement, it will get “you”, “yound”, “youl”, “your”, “yous”, and preditions exactly contain the correct word “you”.

Most tokens can still find the best match when they are not extremely longer than the correct strings as insertion is not the most steps for correction.

3.2 N-gram

Another similarity measurement is based on the numbers of shared n-grams, i.e. substrings of length n [5]. The tokens are divided by the substrings with equal length, then the sum of total numbers of substrings without the number of same substrings that the pair of strings share would be the similarity for them. The function of n-gram is shown:

[4]

where: s is token, t is string compared, G is the number of substrings.

It is obvious that the smaller distance, the closer the match is.

3.2.1 2-gram versus 3-gram

Setting n as 1 will cause numerous candidates for each token, which is time consuming. Through the calculation of number of tokens, it is shown there are 9691 tokens with more than one characters. For tokens with more than two characters, there are 7806 out of 10322 in total. To make sure most tokens will get the predictions with correct one in it, and more efficiency to be guaranteed, parameter n =2 and 3 are selected.

3.2.2 Results

|  |  |  |
| --- | --- | --- |
|  | 2-gram | 3-gram |
| Correct strings returned | 7926 | 7945 |
| Total prediction | 32657 | 1358579 |
| Recall | 0.7678 | 0.7697 |
| Precision | 0.2427 | 0.0058 |

Table 2: Results returned of n-gram

Figure 2: Comparison between GED and n-gram

As it is shown above, the precision of 3-gram is nearly zero, that is because when the threshold for substrings is 3 which means for tokens with less 3 characters will get far more predictions than the circumstance when n = 2. For example, for token “rt” gets only “rt” in 2-gram, but gets 453 candidates in 3-gram. This would absolutely lead to lower precision. Compared with the modified GED, 2-gram is still not efficient enough due to the redundant predictions returned from single-character tokens.

4 Improvement

Based on the analysis above, 2-gram is effective on predictions but still not precise enough. To reduce the number of candidates to get higher precision, an assumption that modified GED is deployed as the first step to sift a list of candidates for the token, then 2-gram is used on the candidate list to reduce the total number so as to get more precise predictions.

|  |  |
| --- | --- |
| Correct string returned | 7919 |
| Total prediction | 12497 |
| Recall | 0.7672 |
| Precision | 0.6337 |

Table 3: Results returned for the combination of modified GED and 2-gram

Figure 3: Comparison of different methodologies

With nearly no difference in recall ratio among theses methods, the improvement that combines modified GED (m, i, d, r) = (1, -1, -5, -5) and 2-gram is successful to provide more precise predictions to users. This is because when 2-gram gives the secondary treatment to the candidates, more redundant strings are omitted.

5 Conclusion

This project has attempted different GED and n-grams to obtain higher precision for each tokens based on accuracy. Despite the runtime for the method, GED with modified parameters is undoubtedly effective. Furthermore, the combination of 2-gram and improved GED gives a better performance with 323% increasing on precision. Thus the improved method can be effective for lexical normalization without considering time consumed.

References

[1] Baldwin, Timothy, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu (2015) Shared Tasks of the 2015 Workshop on Noisy User-generated Text: Twitter Lexical Normalization and Named Entity Recognition. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text*, Beijing, China, pp. 126–135.

[2] Angell, R. C., Freund, G. E., & Willett, P. (1983). Automatic spelling correction using a trigram similarity measure. *Information Processing & Management,19*(4), 255-261.

[3] Ristad, E. and Yianilos, P. (1998). Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5), pp.522-532.

[4] Zobel, Justin and Philip Dart. (1996). Phonetic String Matching: Lessons from Informa- tion Retrieval. In *Proceedings of the Eighteenth International ACM SIGIR Conference on Research and Development in Information Retrieval*. Zu ̈rich, Switzerland. pp. 166–173.

[5] Kondrak, G. (2005). N-Gram Similarity and Distance. *String Processing and Information Retrieval Lecture Notes in Computer Science,*115-126.