Comp90049 Knowledge Technology

Project 1 Lexical Normalisation of Twitter Data Report

Tiange Wang

Uni_ID: 903588

3/9/2017

Introduction

There are large amounts of tweets which have spelling non-standard issues on the Internet. Thus, the system aims to conduct the spelling correction to find the best-match word for the token in each line of the file 'labelled-token.txt' which is provided by Bo and Timothy (2011) through two spelling correction algorithms, referred to the dictionary file 'Dict.txt' which is also managed by Bo and Timothy (2011). The 'labelled-token' file is a list of separated tweets, including some understandable words, acronyms and mentioned twitter accounts, which should be marked as 'IV', 'OOV' and 'NO' types in the process of correction. The two spelling correction algorithms I applied are EditDistance and 2-Gram. The pros and cons of them will be discussed in the following sections.

Description of the system's behaviour

The system is implemented in Java and contains one Main function, two algorithm internal packages and one record package which store the token, type and best match in a line.

1. Main function

Before proceeding the spelling correction algorithms, it is necessary to separate the data in the 'labelled-token.txt' since there are many redundant and meaningless data in tweets. For instance, the token '@dedasje' is used to mention someone’s social media account, but it is unnecessary to find the best match word for it as it is just a name. Therefore, the system applied the ‘judgeLetter’ function to judge whether the token has other characters besides letters and return true if the token is only consisted by letters. The ‘calcRecordED/calcRecordNG’ function is to mark the type ‘NO’ or ‘IV’ or ‘OOV’ for each token. Thereinto, ‘NO’ means ‘not a normalisation candidate’, ‘IV’ implies ‘in vocabulary’ and ‘OOV’ indicates ‘out of vocabulary’. The tokens with ‘IV’ or ‘OOV’ will be utilised to find the best match words by the correction algorithms.

2. Global EditDistance algorithm

The system utilises the hash function (hashmap and hashset) to accelerate the speed of searching.

The global EditDistance algorithm referred to the EditDistance code from Prince of Songkla University (Fivedots.coe.psu.ac.th, 2017), which used ‘d’ for ‘delete’, ‘ins’ for ‘insert’, ‘r’ for ‘replace’ and ‘m’ for ‘match’, and assigned 1,1,1,0 respectively (Ristad and Yianilos, 1998). First, it takes the token and one word from the Dict.txt to divide to single letters and constitute the members of a matrix. Then it compares the distance[j][k] with the threshold (val in the program). If the value of distance[j][k] is smaller than the minimum of the threshold, then the token will match the word in the Dict.txt (Ristad and Yianilos, 1998).

In the evaluate function, it calls the algorithm and related functions to finish the matching process. Then the function outputs all the best match words with the tokens and types to a file called ‘labelled-token-ED.txt’ and compares with the correct words in the ‘labelled-token.txt’ to calculate the precision of the algorithm.

3. N-Gram algorithm (2-Gram algorithm)

I applied 2-Gram algorithm here to compare the precision with the EditDistance algorithm. The process of the matching is similar to ED algorithm, which separates the token and the word from the dictionary into single letters and calculates the distance. Specifically, the algorithm also requires calculating the intersection as the formula is ‘distance = length of token + length of word – 2* length of intersection’. Likewise, if the value of the distance is smaller than the minimum of the threshold, the two words will be matched.

After matching the best word for each token, the function will create a file named ‘labelled-token-NG.txt’ and store the result in it.

Evaluation

I have tested the 8841 tokens in the file ‘labelled-token.txt’. In the aspects of the time complexity and the effectiveness, the result showed that there exist considerable discrepancies in these two different algorithms.

1. Global Edit Distance

The running time of it is about six mins. Meanwhile, the system shows when applied the EditDistance algorithm, the precision reached 71.8%, including 6346 correct matched words and 2495 wrong matched words. The algorithm illustrated the relatively high performance. Nevertheless, in order to figure out the reason of correction failure, I opened the file ‘labelled-token-ED.txt’ since the correction results are stored in it and ‘labelled-token.txt’, and selected two sets of failed token records randomly.

1.1 ‘effin OOV elfin’ vs ‘effin OOV fucking’.

The former is the result of spelling correction by the system and the latter is the correct record in the ‘labelled-token.txt’. The matrixes of ‘effin’ and ‘elfin’, ‘effin’ and ‘fucking’ are shown below:

	ε	e	f	f	i	n
ε	0	1	2	3	4	5
e	1	0	1	2	3	4
l	2	2	1	1	2	3
f	3	3	3	1	2	3
i	4	4	4	3	1	2
n	5	5	5	5	3	1

Figure 1: The first matrix.

	ε	e	f	f	i	n
ε	0	1	2	3	4	5
f	1	1	1	2	3	4
u	2	2	2	2	3	4
c	3	3	3	3	3	4
k	4	4	4	4	4	4
i	5	5	5	5	4	5
n	6	6	6	6	6	4
g	7	7	7	7	7	6

Figure 2: The second matrix.

It is evident that the best match for the token ‘effin’ is ‘elfin’ since the global edit distance is 1. However, in the ‘labelled-token.txt’, the correct word is fucking although the global edit distance is 6.

1.2 ‘huyy OOV huey’ vs ‘huyy OOV huyy’.

The former is the result of spelling correction by the system and the latter is the correct record in the ‘labelled-token.txt’. The ‘OOV’ means ‘not in the dictionary’. Nonetheless, the ‘huyy’ in the ‘labelled-token.txt’ does not show any correction result.

Based on examples above, due to the difference of the dictionaries, some tokens did not obtain the correct word. The true words ‘fucking’ and ‘huyy’ are even not included in the existing ‘Dict.txt’. Apart from the dictionary reasons, the Global Edit Distance also has some disadvantages. For example, the efficiency of the algorithm is ordinary because it takes six mins to run the algorithm which is tripled than the time cost of the 2-Gram algorithm. Moreover, Jiang et al. (2013) also demonstrated that the time complexity is suboptimal and they had to utilise concurrent algorithm to increase the efficiency of it.

2.N-Gram (2-Gram)

The running time of it is only about two mins, which is beyond my expectation. The system shows when applied the EditDistance algorithm, the precision reached 71.0%, including 6275 correct matched words and 2566 wrong matched words. The accuracy of it demonstrated a high performance. However, after open the result file ‘labelled-token-NG.txt’, the reason of the ordinary precision is apparent. In my opinion, this is related to the shortcoming of N-Gram, that is, two approximate words could possibly gain the same value of the N-Gram distance. For instance, the set ‘cart’ and ‘crat’ could obtain the same distance (six) as the set ‘cart’ and ‘coat’. Once the system selects the wrong word from qualified words, the precision would decrease.

On the other hand, the advantage of N-Gram is distinct. The running time of N-Gram was fast, and it maintained good precision which exceeded the baseline (Fivez, Suster and Daelemans, 2017). In addition, Yeh et al. (2013) illustrated on the aspects of spelling error detection and correction, the N-Gram algorithm was excellent and had better performance than other algorithms.

Conclusion

According to the result of the test, apart from the circumstances that the tweet is a correct word or has existed in the dictionary, it is difficult to find the best match for each tweet when only applied the Global Edit Distance and N-Gram algorithm because of the low precision.

During the process of test, there are many challenges, such as filtering the non-letter tokens (data cleaning), the long waiting time after modifying the code and fixing the bug as the running time of algorithms are long. Meanwhile, in the actual Twitter world, tweets may contain multiple languages. Hence, besides the English, the correction work for French, Spanish and Chinese words should also be taken into consideration.

Lastly, the test also indicated that the precision is related to not only the selection of the algorithm but also the dictionary the system utilised. Using concurrent algorithms is an efficient method to diminish the running time (Jiang et al., 2013). Thus, in this case, in my opinion, integrating existing dictionaries and expanding the scope of the dictionary are also a way to advance the accuracy of spelling correction.

Reference

1. 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 Compu- tational Linguistics*, Portland, USA. pp. 368–378.

2. Fivedots.coe.psu.ac.th. (2017). [online] Available at: http://fivedots.coe.psu.ac.th/Software.coe/ContestAlgs/Code/11.%20Dynamic%20Prog/EditDistance.java [Accessed 4 Sep. 2017].

3. Fivez, P., Suster, S., & Daelemans, W. (2017). Unsupervised Context-Sensitive Spelling Correction of Clinical Free-Text with Word and Character N-Gram Embeddings. *BioNLP 2017*, pp.143-148.

4. Jiang, Y., Deng, D., Wang, J., Li, G., & Feng, J. (2013, March). Efficient parallel partition-based algorithms for similarity search and join with edit distance constraints. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM. pp. 341-348.

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

6. Yeh, J. F., Li, S. F., Wu, M. R., Chen, W. Y., & Su, M. C. (2013). Chinese word spelling correction based on n-gram ranked inverted index list. In *Sixth International Joint Conference on Natural Language Processing*, pp.43-48.