COMP90049 Project 1 Report

Anonymous

1. Introduction

People often make spelling mistakes when typing words on the computer for reasons of typing too fast or wrong-pressed the key on the laptop etc. The aim of this report is to figure out whether the reason for people misspell words is typing letters with a wrong sequence or not. By implementing a prediction system with different algorithms and different parameters, predict the misspell word and calculate the prediction accuracy in order to prove the hypothesis mentioned above.

This paper is structured as follows. Section 2 briefly introduces the dataset, section 3 lists some evaluating methods for the spelling predicting system. Section 4 introduces the methodology for system implementation and some of the experiment data. Finally, section 5 analyzes and concludes the cause of typos.

2. Dataset

The dictionary file[1] has approximately 370K English words and the misspell file[2] has 4453 tokens (words) which contain common errors made by Wikipedia editors. The correct file is a list of truly intended spellings of the corresponding misspelled tokens (words) from the misspell file.

3. Evaluating Metrics

In this paper, the following indicator is used to access the performance of the spelling prediction system in order to analyze the reasons for typing errors.

● Precision: For systems that predict more than one token (word), the fraction of correct responses among attempted responses.

$$Precision = \frac{\text{Correct prediction number}}{\text{Times of prediction}}$$

● Recall: For systems that predict more than one token (word), the proportion of tokens (words) with a correct response.

$$Recall = \frac{\text{Correct prediction number}}{\text{total number of words}}$$

● Accuracy: For systems that predict only one token (word), accuracy equals precision as well as recall.

For method as Global Edit Distance, total distance between the predicting word and all words is provided for analyzing.

4. Methodology

The abbreviation for the following paper is declared below (see Table 1).

NOMENCLATURE	
GED	Global Edit Distance
LD	Levenshtein Distance
NG @ i	N-Gram, N = i
P @ i	Take i prediction(s) for misspell word
Pre	Precision
Re	Recall
NoPW	Number of Predicted Words

Table 1: The nomenclature table

4.1. Global Edit Distance

The package for GED I uses is imported from PyPI [3] (Python Package Index), named 'editdistance', which is a fast implementation of Levenshtein distance. The distance for [match, insertion, deletion, replacement] is [0, 1, 1, 1]. Levenshtein distance ("Levenshtein distance", 2018) between word A and B is defined below:

$$lev_{A,B}(i,j) = \begin{cases} \max(i,j) & if\, min(i,j) = 0, \\ \min\begin{cases} lev_{A,B}(i-1,j)+1 \\ lev_{A,B}(i,j-1)+1 \\ lev_{A,B}(i-1,j-1)+1_{(A_i \neq B_j)} \end{cases} & otherwise. \end{cases}$$

Considering the misspell and dictionary files given are all at a lower case, we can compare them directly without transforming from/to capital letters.

First, I calculated the Levenshtein distance between each misspelling word and the dictionary, then found out the minimum Levenshtein distance for each word and finally returns the prediction word(s) with the minimum global edit distance. The results are shown below (see Table 2 and Figure 3).

[1]https://github.com/dwyl/english-words

[2]https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings

[3] https://pypi.org/project/editdistance/

GED	P @ 1	P @ 2	P @ 3	P @ ∞
Pre	55.49 %	47.01 %	42.29 %	26.27 %
Re	55.49 %	67.48 %	72.18 %	79.72 %

Table 2: The prediction accuracy for GED

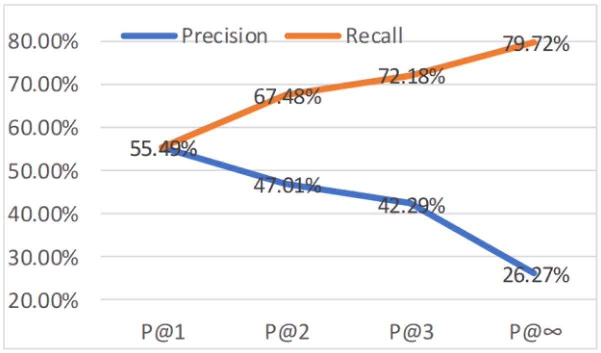

Figure 1: The prediction accuracy for GED

As can be seen above, the accuracy for predicting system is 55.49% when we keep one word for prediction. As the number of candidates grows, the predicting precision decreases while the predicting recall increases. Here, Levenshtein distance with 2 or 3 words keeping has the best prediction result. Let's take a look at the global edit distance for each word when taking 3 words for prediction below (see Table 3).

Distance	1	2	3	4	5
NoPW	3445	631	46	7	2

Table 3: LD between each misspell word & correct word

The table shows that most of the misspelled words have only 1 edit-distance, so the misspell operation could be (1) double-pressed one letter (2) miss-pressed one letter (3) pressed a wrong letter (4) sequence of letters is reversed etc. Sample errors and predictions are given below (see Table 4).

	Misspell	Predictions	Correct
Case (1)	abbout	abbot, abbott, about	about
Case (2)	aberation	aberration, aeration	aberration
Case (3)	presense	prepense, presence, pretense	presence
Case (4)	repeteadly	repeatedly, reputedly	repeatedly

Table 4: Sample misspell errors and predictions

For further study, I use N-Gram for modifying the predicting system.

4.2. N-Gram

The package for n-gram I use is imported from PyPI, named 'ngram'[4]. This package can compare substring similarity, so we found the highest similarity between the misspelled word and the word in the dictionary.

A python snippet for 2 Gram algorithm and take at most 3 prediction words is shown below:

```
for w in dict:
    v = ngram.NGram.compare(misspell, w.strip(), N=2)
    if (v > highestV):
        prediction = w.strip()
        highestV = v
        predictionList = []
        predictionList.append(pre)
    elif (v = highestV):
        prediction = w.strip()
        highestV = v
        if (len(predictionList < 3)):
            predictionList.append(prediction)
```

First, I didn't set the limit (threshold) for word prediction, tested the precision rate and the recall rate with n-gram algorithm when N = 1, 2 and 3. The results are shown below (see Table 5 and Figure 2).

P@ ∞	NG @ 1	NG @ 2	NG @ 3
Pre	13.11 %	6.11 %	3.46 %
Re	51.61 %	65.48 %	64.74 %

Table 5: Accuracy for N-Gram algorithm

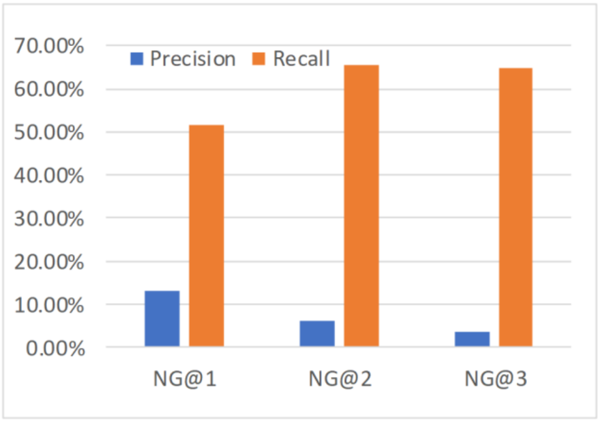

Figure 2: Accuracy for N-Gram algorithm

The data illustrates that the recall is the highest when N is 2, i.e. the bigram algorithm has the best accuracy in predicting words. However, the precision rate is too low – only about 6 per cent, so then I tried to limit the words for prediction with 1, 2, 3 and 4 respectively. (see Table 6 and

[4] https://pypi.org/project/ngram/

Figure 3):

N @ 2	P@1	P@2	P@3	P@4	P@∞
Pre	63.89%	59.70%	60.10%	60.06%	6.11%
Re	63.89%	66.49%	66.90%	66.90%	65.48%

Table 6: Accuracy for Bigram with 2 predictions

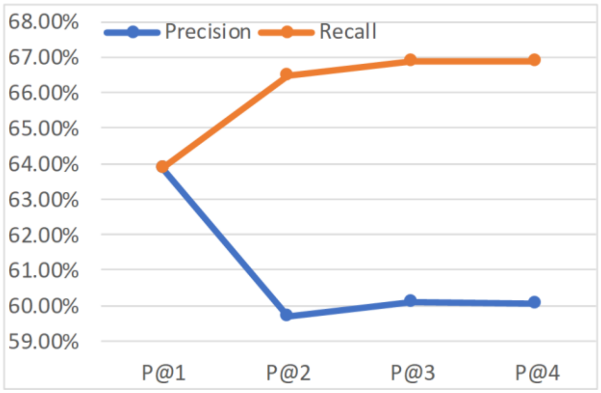

Figure 3: Accuracy for Bigram with 2 predictions

Based on multiple tests, the precision ratio grew rapidly to approximately 60% and the recall ratio slightly falls to about 67%. When taking 3 words as prediction, the prediction accuracy (recall)effects the best.

Then I looked at the prediction data for bigram. Take the misspell word "acheived" for example, bigram algorithm tokenizes the word into "$a ac ch he ei iv ve ed d$". As for the correct word "achieved" was tokenized into "$a ac ch hi ie ev ve ed d$". The similarity calculated between these tokens are only 50.00% for only one-letter sequence reversed. But for the prediction word "ached", the token "$a ac ch he ed d$" has a similarity rate of 66.67% compared to the correct word. That is to say the N-Gram algorithm with threshold is not good at testing words that have bad sequence.

So, I take out the words that have only one levenshtein distance, then use bigram algorithm for testing, the result is very low.

5. Conclusion

When the edit distance tested in GED is only one, the main reason for misspelling could be duplicated, typed a wrong letter, miss-typed a wrong letter or a bad sequence.

After taking out the words which have one edit distance and running in bigram algorithm with a threshold of 3, the accuracy is low.

That is to say, the main typo is not because of reversed sequence of letters, but mainly other three cases.

References

Levenshtein distance. (2018). Retrieved from https://en.wikipedia.org/wiki/Levenshtein_distance