COMP90049 Knowledge Technology Project 1 Report

Anonymous

1 Introduction

1.1 Problem

In this project, we need to implement various spelling correction algorithms to correct a list of misspelled words. Then we need to analyse the effectiveness of these algorithms.

1.2 Dataset

We are provided with one text file that contains 716 misspelled words, and the text file contains the corresponding correct words and one dictionary text file from UrbanDictionary [Saphra et al. 2016].

2 Methodology

I have applied three spelling correction algorithms. They are Global Edit Distance (with Levenshtein Distance), N-grams and Neighbourhood Search.

2.1 Global Edit Distance (Levenshtein Distance)

For each entry in the misspelled.txt, compute the Levenshtein distance ([m, i, d, r] = [0, 1, 1, 1]) between it and every entry in the dictionary. Record all entries with computed distance less than the defined scope, that is two and three in this case. Finally, compare these potential words with the correct word listed in the correct.txt and check if any of them matches the correct word. If there is a correct response, mark the misspelled word as corrected.

2.2 N-gram Distance

For each entry in the misspelled.txt and dictionary.txt, split the words into n-grams, # character was added at both the begining and end. The n in this method is defined to be two. Then compute the N-gram Distance between misspelled words and every entry in the dictionary.txt. Record all entries with distance less than the defined threshold, that is two and three in this case. Finally, compare these potential words with the correct word listed in the correct.txt and check if any of them matches the correct word. If there is a correct response, mark the misspelled word as corrected.

2.3 Neighbourhood Search

For each entry in the misspelled.txt, generate all variations of the word within defined number of edits. The algorithm is from Peter Norvig [2007]. The allowed edits to the word are deletion, transposition, replacement and insertion. The threshold of edits is two and three. Then record all entries that appear in the dictionary.txt. Compare these potential words with the correct word and check if any of them matches the correct word. If there is a match, mark the misspelled word as corrected.

3 Evaluation Metrics

I have used the following evaluation metrics to evaluate the result of different methodologies.

Recall The proportion of words with a correct response in the given solution set.

Precision The proportion of words corrected comparing to the total attempts generated in different methodologies.

The runtime (in seconds) for each different algorithm has also been recorded. They can show the efficiency of these algorithms. However, runtime will not be directly considered while evaluating the effect of these methodologies.

4 Results

4.1 Global Edit Distance (Levenshtein Distance)

The used library is editdistance that is written in Python by Hinoyuki Tanaka (2013). The result is listed in the table 1 below:

From the result, Global edit distance has quite high recall. This method is the most effecitve one among all methods I implemented.

Metrics	Scope = 2	Scope = 3
Words Corrected	504	556
Recall	70.39%	77.65%
Precision	0.351%	0.346%
Runtime	1040.58s	1079.97s

Table 1: Result of Global Edit Distance

Improving the scope from two to three will result in correcting 52 more words, the recall increases around 7%. This is not a bad result, especially that the runtime does not rise as the scope increases. However, the precisions are very low. This implies global edit distance has generated large amount of attempts.

After increasing the scope, the precision only drops by 0.005%. This indicates that increasing the scope does not generate huge number of additional attempts.

4.1.1 Example

Misspelled	Correct	Candidate(s)
b4	before	ab, aba, etc.
gawgus	gorgeous	gangue etc.
tiddies	titties	✓
genious	genius	✓
thru	through	✓

Table 2: Examples of Global Edit Distance

4.1.2 Analysis

Firstly, the global edit distance is not so effective when the misspelled words is very different with the correct words in lengths. Example **b4** and **gawgus** can prove this. The huge difference in lengths between **before**, **gorgeous** and **b4**, **gawgus** makes global edit distance hard to find the correct words within such small scope.

On the other hand, global edit distance is relatively more useful with words having similar lengths. Such as, **tiddies** and **titties**, **genious** and **genius**.

By observing the result, it is found that most misspelled words corrected have similar lengths as their correct words. This result enhances the statement above, similar lengths between misspelled words and correct words makes the Levenshtein distance between them relatively smaller. Hence, the global edit distance method is easier to identify the correct words.

Also, by increasing the scope. Global edit distance can now identify words more words with larger differences in lengths between misspelled and correct words. For instance, **thru** and its correct word **through**.

4.2 N-gram Distance

Metrics	Scope=2	Scope=3
Words Corrected	47	153
Recall	6.56%	21.37%
precision	10.24%	5.73%
Runtime	968.39s	857.26s

Table 3: Result of N-gram Distance

As the table states, the recall of N-gram distance are not high in general. This indicates N-gram distance method does not find the corrected words for most misspelled words. The precision is quite high, which states that the number of attempts generated is relatively small comparing to global edit distance, some of the misspelled words even do not have any attempts generated. However, the efficiency is similar to global edit distance.

After increasing the scope from two to three, the number of words corrected and the recall has tripled. The precision has dropped, which implies more attempts are generated.

4.2.1 Example

Misspelled	Correct	Candidate(s)
addidas	adidas	✓
proabably	probably	✓
quen	queen	✓
servise	service	✓
beutiful	beautiful	✓
camoflauge	camouflage	N/A

Table 4: Examples of N-gram Distance

4.3 Analysis

From the result, it is found that N-gram distance is most effective when the misspelled words and correct words are made of similar letters. As the examples in the table illustrates, **addidas** is made of the same letters as its correct form **adidas**, but with one more letter **d**. Similarly, **proabably** is also made of same letters as **probably**, with one more letter **a**.

Besides the situations when misspelled words have more letters than their correct form, n-gram can also be effective for missing letters. For instance, **quen** has one less **e** than **queen**.

In addition, n-grams can also be relatively effective when there is only small difference between misspelled form and correct form. Such as **servise** and **service** in the examples.

Nevertheless, when the misspelled words and correct words are made with very different letters, it is ineffective. This is since the n-gram distance for words made of similar letters is smaller. This can be proven by the result of increasing the scope. After the scope increases, far more attempts with higher n-gram distance are generated. Like the word **beutiful** is now corrected to **beautiful**, where the letter **a** does not appear in the misspelled form.

It is also relatively inefficient when the length of strings are long. One example is the misspelled word **camoflauge** does not generate any attempt in N-gram distance.

4.4 Neighbourhood Search

Metrics	Edits = 2	Edits = 3
Words Corrected	267	285
Recall	37.29%	39.80%
Precision	3.05%	2.72%
Runtime	55.49s	4482.37s

Table 5: Result of Neighbourhood Search

By observing the results, neighbourhood search results in a relatively low recall. It is better than the N-gram approach, but worse than global edit distance approach. The precision results display the fact that neighbourhood search has generated relatively large number of attempts. The efficiency when number of edits is equal to two is very high, only took the program to complete within one minute. However, efficiency drops exponentially after increasing the number of edits. Both the recall and precision do not change massively after the number of edits increased.

4.5 Example

Misspelled	Correct	Candidate(s)
psychadelic	psychedelic	✓
raido	radio	✓

Table 6: Examples of Neighbourhood Search

4.6 Analysis

From the results, similar conclusion as the global edit distance is obtained, where misspelled words and the correct words with similar lengths are likely to be corrected by neighbourhood search. For instance, **psychadelic** and **psychedelic**. In addition, since this neighbourhood search algorithm has transposition as one edit, it can easily correct misspelled with wrong order between letters, like **raido** and **radio**.

5 Conclusions

In conclusion, by comparing three different spelling correction algorithms, global edit distance achieves the best results out of them. Neighbourhood search achieves a relatively good result and with absolutely higher effeciency. N-gram distance has limited effectiveness comparing to the other two methods.

6 References

Graham Poulter (2012). python-ngram. http://github.com/gpoulter/python-ngram.

Hiroyuki Tanaka (2013). editdistance. https://www.github.com/aflc/editdistance.

Naomi Saphra and Adam Lopez (2016) Evaluating Informal-Domain Word Representations with UrbanDictionary. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, Berlin, Germany. pp. 94-98.

Peter Norvig (2007). How to Write a Spelling Corrector. http://norvig.com/spell-correct.html.

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