COMP90049 Report: Analysis of Spelling Correction Methods

Anonymous

1 Introduction

This report examines a diverse set of spelling correction techniques and compares their respective performances on a particular dataset from UrbanDictionary. Spelling correction can be treated as an approximate string matching problem.

2 Dataset

The data consisted of misspelled words, the correct spellings of the respective words, and a "dictionary" of correctly spelled words. These words were identified from UrbanDictionary by Saphra and Lopez (2016). The words are formed from Latin characters but are not necessarily English words.

This dataset has several deficiencies, which could produce false negatives or false positives in the results. For example, "guilford" (a "correct" entry) doesn't appear in dictionary but "guildford" (a "misspelled" entry) does. The following table collates these shortcomings:

misspelled entries in dictionary	*171*
correct entries **not** in dictionary	*108*
correct entries same as misspelled ones	*9*

Table 1: Deficiencies in the provided dataset

Further calculation revealed that 275 (out of 716) words are in some way "deficient". This is close to 38.4% of the data. Since this is such a high percentage, precautions should be taken before making any critical conclusions from this study.

3 Methodology

Three different approximate string matching approaches were used for this problem. This was done in order to get a broader scope of results and to appreciate the respective strengths and weaknesses of each technique.

3.1 Global Edit Distance

The Needleman-Wunsch Algorithm was used to calculate the Global Edit Distance (GED) between the misspelled word and a word in the given dictionary. It returns the maximum possible "score" which represents the GED between the two words. This score is determined by the chosen parameters. This dynamic-programming algorithm was selected in order to improve the run time complexity of the otherwise lengthy process. Regardless, this program took the longest to execute (i.e., approximately 7 hours for only 716 words).

A simple `Python` script was used to execute the algorithm, calculating the GED between each word in the given file of misspelled words and each word in the dictionary, returning a list of dictionary words with the largest GED from each respective misspelled word.

3.1.1 Parameters

After careful consideration, the parameters $(m, i, d, r) = (+1, -1, -1, -1)$ were chosen over the Levenshtein distance parameters $(m, i, d, r) = (0, -1, -1, -1)$. With these parameters, the difference between a match and a non-match character is further exaggerated. This could be useful when searching for "similar" strings.

3.2 Neighbourhood Search

3.2.1 Implementation with agrep

This search algorithm was implemented with a `bash` script which utilised the UNIX command-line utility `agrep` due to its impressive efficiency. The code used the `-#` flag in the `agrep` command in order to define the neighbourhood limit, `k`. `agrep` was executed on each misspelled word with `k`-values starting from `0` and incremented by `1`, until at least one match from the dictionary (a neighbour) was found.

All the matches for that specific misspelled word were then outputted to standard output.

This allowed for each misspelled entry to have at least one result without requiring preprocessing to determine the smallest value of `k` needed to guarantee a match.

3.3 Phonetic Matching

3.3.1 Implementation with Soundex

The phonetic matching algorithm was implemented with a `Python` program, which utilized Soundex to encode each word in the dictionary into a four-character code. This encoding is meant to be shared amongst "like-sounding" words. The program maps each encoding to the list of words that translate to it. Then, the corresponding list for each given misspelled word is returned. It is possible for an empty list to be returned. Given that all the dictionary information is cached, the overall time complexity is very efficient.

3.3.2 Truncation

It was decided that the truncation of Soundex codes to four characters should be kept in order to retain the simplicity of exact indexing. Having to account for "similar" codes introduces another approximate string matching problem where additional techniques (eg. edit distance) would be required. Even though it may give a better solution, this could introduce unnecessary complexity instead.

4 Evaluation Metrics

In the initial stages of the project, all methods returned a single answer chosen at random from the pool of appropriate solutions. This was repeated 10 times and the accuracy for each run was calculated. (see Table 2).

#	GED	`agrep`	Soundex
1	85	110	2
2	80	105	3
3	89	115	3
4	83	107	4
5	84	118	6
6	79	106	4
7	87	102	4
8	83	109	7
9	84	110	2
10	85	109	4
Average	83.9	109.1	3.9

Table 2: Accuracies of each method over 10 runs

Due to the low accuracy results (only 0.5% - 15.2% of the total words) and consistent presence of random factor(s), it was difficult to reliably compare the respective algorithms based on accuracy.

Thus, precision and recall were more appropriate evaluation metrics since they are designed to work on lists of results rather than just one. The algorithms were then amended accordingly. This reduced the unaccountable variation in results.

4.1 Precision

Precision is determined by calculating

$$\frac{|correct|}{|attempted|} \tag{1}$$

where *correct* is the correct words the method obtained and *attempted* is all of the obtained words.

Determining the precision is useful because even though a technique may return the correct answer, that solution is useless if many wrong answers are returned as well. It is difficult to discern the actual desired solution from such a large pool.

4.2 Recall

Recall is determined by calculating

$$\frac{|correct|}{|total|} \tag{2}$$

where *correct* is the correct words the method obtained and *total* is the misspelled words.

Recall is a method of verifying if a certain method will return the right answer at all. If the precision of a procedure is low but it has a high recall, perhaps additional post processing could help sieve through the results and determine the most relevant one.

5 Results

A `Python` script was written to calculate the precision and recall of each approximate string matching method.

Method	Precision	Recall
GED	0.0481378262	0.2653631285
`agrep`	0.0457670043	0.3533519553
Soundex	0.0029336621	0.5949720670

Table 3: Results for each method, approximated to 10 decimal places

6 Analysis

In this study, Soundex obtained the highest recall but the lowest precision out of the three methods. These results seem to indicate high numbers of false positives for Soundex. This may be a consequence of truncating the Soundex codes.

The method with the highest precision is the Global Edit Distance. Interestingly, it also acquired the lowest recall. This implies that ...

6.1 GED vs `agrep`

GED has better precision but not that much better, only approximately 5.18% better, with respect to agrep's precision. However, the recall of `agrep` is 33.16% better (wrt to GED recall), which could significantly affect any dataset. If there exists some algorithm that could help rank the relevance of all the results, it could be quite likely to get the "correct" answer, rendering the difference in precision negligible.

6.2 GED vs Soundex

Eventhough the recall value of Soundex is 2.24 times better than GED, the precision of GED is approximately 16.4 times better than Soundex. This infers that although Soundex would most likely have the correct answer in its list of solutions, if selection of the final answer is random, there is a much lower chance of getting the actual answer than if GED was used. Due to this, unless a sufficiently effective algorithm that could help rank the relevance of all the results is available, the Soundex algorithm is most likely useless for real world application.

6.3 `agrep` vs Soundex

`agrep` is also significantly better at precision than Soundex, 15.6 times better. Soundex's recall value is only 1.68 times better than `agrep`. That would most likely make the most difference in a large dataset, maybe with some extra information that could provide some context to the problem.

7 Conclusions