COMP90049 Knowledge Technologies

Project 1: Waht kinda typoz do poeple mak?

1 Introduction

This report aims to implement several prediction systems which based on Levenshtein distance, N-gram and Phonetic algorithm, in order to find correct forms for misspell words. The initial system will rely on single models among above algorithms. Through knowledge gained from analysis of these implementations, a improved boost system will be discussed at the last part.

2 Dataset

Dataset includes three essential documents. *Dict* is a list of approximately 370K English entries, which comprises the dictionary for approximate string search methods. The format of this file is one entry per line, in alphabetical order. *Wiki_misspell* is a list of 4453 tokens that have been identified as common errors made by Wikipedia editors. The format of this file is one misspelling per line, in alphabetical order. *Wiki_correct* is a list of the truly intended spellings of the corresponding misspelled tokens from Wiki_misspell.

3 Evaluation Metrics

Throughout this paper, the following terms will be used to evaluate each methods (Blum & Langley, 1997):

Accuracy: It can measure the effectivity of a system. It shows how much proportion are correct against the system that predict only one word, or how much proportion are correct at least one words for a multi-prediction system.

Precision:, Precision can measure the average correct proportion against the system that predict more than one word. To further measure the performance of the different methods, Precision at N (P@N) was defined, which is calculated by matching predicted words and the correct word, and determining whether or not the matching correct word was contained in the top N of predicted word lists.

Recall: It shows the proportion of words with a correct response.

Runtime: It is also a significant indicator of models. Out of consideration for the practicality of models, we might need to balance the efficiency and the accuracy.

Unless specified otherwise, all testing was done on the wiki dataset for increasing running speed and to avoid over-fitting.

4 Hypothesis

4.1 Duplicate letters

Duplicate keystrokes during typing may result in misspelled words. Especially, some words which includes multiple consecutive letters.

4.2 Similar structure/pronunciation

Some words with similar structures may be misspelled. For example, "abilities" and "abilties" is a common typographical error in this type. Meanwhile, the words have similar pronunciation may a cause of spelling mistakes.

5 Methodology

5.1 Global Edit Distance

GED is a classical similarity algorithm between two strings, which referred to as the source and the target. The distance is the number of Insertions (i), Deletions (d) or Replace(r) required to transform source into target. Table 1 summarises the running result of original GED for wiki dataset:

Accuracy	0.664
Recall	0.790
P@3	0.272
P@5	0.263
Runtime	509 s

Table 1: Results of original GED for wiki dataset

5.1.1 Evaluation

Depending on the result above, this system is relatively accurate at returning multiple predictions for each misspell words. Especially, it has a particular increasing between P@3 and P@5. In order to return only one prediction, the evaluation system ranks these candidates by N-gram distance, and then the most similar item will be returned (Yujian & Bo, 2007).

5.1.2 Implement

The original GED function used the edit distance library from Zante (2016), which used the Levenshtein parameters of (m, i, d, r) = (0, 1, 1, 1) and written in C++ with a python wrapper for speed.

5.2 N-gram Algorithm

In the fields of computational linguistics and probability, N-gram is another approximate matching algorithm, which comparing contiguous sequence of n items from given strings (Kondrak, 2015). Figure 1 is an example of N-gram algorithm. The similar structure of words may be a remarkable aspect of misspelling. Hence, the correct form might be matched with misspelled words through N-gram algorithm.

2-grams of crat: **#c, cr, ra, at, t**
2-grams of arts: **#a, ar, rt, ts, s**
2-Gram Distance between crat and arts:
$|G2(crat)| + |G2(arts)| - 2 \times |G2(crat) \cap G2(arts)|$
$= 5 + 5 - 2 \times 0 = 10$

Figure 1: Example of N-gram algorithm

5.2.1 Optimisation method

Due to the fixed subsequence of each word, programme can establish the whole dictionary by Tire tree at the initialisation, and then search every misspelled words at this dictionary. Through a series of experiments, this artifice can boost the efficiency.

5.2.2 Evaluation

The results for N-gram algorithm with different parameters are shown below at Table 2.

N-gram	Accuracy	P@3	P@5	Time
2	0.728	0.283	0.274	337 s
3	0.745	0.279	0.271	326 s
4	0.712	0.267	0.261	301 s

Table 2. Result of N-gram

5.2.3 Implement

Due to some functions of python-ngram library which not in the range of this experiment, this program uses a modified version of python-ngram library from Gpoulter (2017). Moreover, the Ngram library is modified to compatible with multi-processes, in order to save the runtime.

5.3 Phonetic Algorithm

Phonetically similar segments are two or more sounds which share phonetic features and are frequently found as variants of a single phonological unit in a language ("Phonetically Similar Segment", 2018). Therefore, the words that have similar phonetic forms may have higher similarity.

Four single methods are used to measure the similarity of two comparison strings initially. Subsequently, a boost algorithm, which combine these methods, will replace these four single methods. Figure 2 is an illustration of four different methods.

```
Metaphone('Jellyfish')  --->  'JLFX'
Soundex('Jellyfish')  --->  'J412'
Nysiis('Jellyfish')  --->  'JALYF'
Match_rating_codex('Jellyfish')  --->  'JLLFSH'
```

Figure 2. Four methods of phonetic

5.3.1 Boost methods

The main idea of Boost is to assemble weak classifiers into a strong classifier. Considering the limited accuracy of a single method, these four methods (Metaphone, Soundex, Nysiis, Match_rating_codex) are combined together in interest of higher accuracy. Through endowing methods the corresponding weight, the accuracy of the matching is significantly improved (Kondrak, 2003).

5.3.2 Evaluation

The individual results of these four method can be seen in Table 3. Depending on the performance of these method, corresponding weights are endowed in the boost function, which has an obvious better performance than these single four methods.

MP	Soundex	Nysiis	MRC	Accuracy
1	0	0	0	0.560
0	1	0	0	0.007

0	0	1	0	0.373
0	0	0	1	0.392
0.4	0.1	0.25	0.25	0.700
0.3	0.2	0.3	0.2	0.730
0.4	0.1	0.3	0.2	0.765

Table 3. Result of phonetic methods

5.3.3 Implement

The program uses a phonetic library named Jellyfish (Turk, 2018), which provides many packaged functions of string similarity. According to above experiments, the boost function was endued different weight parameters.

6 Discussion

According to results of the above experiments, many misspelled words caused by duplicate keystrokes can be found in the misspell dataset, which can support the first hypothesis of the report.

On the other hand, the similar structure of words is also a significant factor of typographical errors. Editors generally cannot find out this type of error without the assistance of computer or very careful review. Especially, it will be more difficult if these words both in the dictionary. A enormous number of errors of this type found in experiments can support hypothesis 2.

Typos	Misspelled word	Correct word
H1	ninetenth	nineteenth
	infalibility	infallibility
	millenium	millennium
H2	abilities	abilties
	consiciousness	consciousness
	humerous	humorous

Table 4. Typical words of each hypothesis

7 Improvements

7.1 Multi-processes

For fully using CPU resource, this code can be modified to multi-process version. After practice testing, the multi-process version can effectively improve the efficiency.Considering the serialisation of the output, this program uses Partial Function and Map function ensuring the output can be compared with the correct words list.

7.2 Trie tree optimisation

The original methods, including GED, N-gram or Phonetic, both need to compare each word of misspelled list with each word of the dictionary, which will spend a huge amount of runtime. Against this problem, a elegant data structure, Trie tree, is able to avoid numerous unnecessary comparisons. Figure 3 is an illustration of Trie. According to same rules of Trie, we can establish a pre-suffix tree by the selected function (GED, N-gram or Phonetic). Consequently, nodes of Trie can be traversed by the rule that merely traverse the nodes which in the range from D - N to D + N (D is similarity distance and N is threshold).

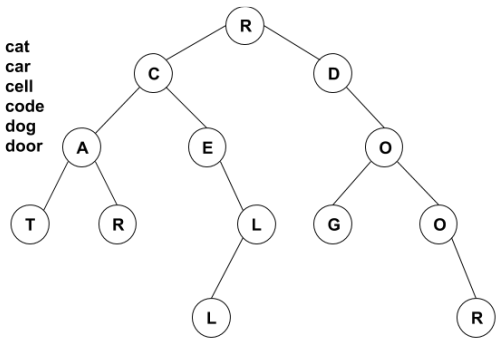

Figure 3. Trie tree

Through configuring the range of error counter, this algorithm can update the result in the search process. The experimental results show that the nodes traversed by one query will not exceed 5% to 8% of all nodes. Proper caching and reducing the constant N can make the algorithm more efficient.

8 Conclusions

The report carefully compares and analysis three different string similarity algorithms. Modified Phonetic method (Table 3) reaches 76.5% of accuracy in the wiki dataset.

Moreover, the N-gram algorithm with Trie is able to complete matching process in 400 s.

The next step to improve the accuracy would be using Hidden Markov model to predict correct words or building a integrated method based on Trie tree.

References

Blum, A., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*(1-2), 245-271. doi: 10.1016/s0004-3702(97)00063-5

Gpoulter. (2018). gpoulter/python-ngram. Retrieved from https://github.com/gpoulter/python-ngram

Kondrak, G. (2003). Phonetic Alignment and Similarity. *Computers And The Humanities*, *37*(3), 273-291. doi: 10.1023/a:1025071200644

Kondrak, G. (2015). N-Gram Similarity and Distance. *International Symposium On String Processing And Information Retrieval*, 115-126.

Phonetically Similar Segment. (2018). Retrieved from https://glossary.sil.org/term/phonetically-similar-segment

Turk, J. (2018). jamesturk/jellyfish. Retrieved from https://github.com/jamesturk/jellyfish

Wikipedia:Lists of common misspellings. (2018). Retrieved from https://en.wikipedia.org/w/index.php?title=Wikipedia:Lists of common misspellings& oldid=813410985.

Yujian, L., & Bo, L. (2007). A Normalized Levenshtein Distance Metric. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, *29*(6), 1091-1095. doi: 10.1109/tpami.2007.1078

Ztane. (2018). ztane/python-Levenshtein. Retrieved from https://github.com/ztane/python-Levenshtein