Spelling Correction Methods Evaluation

1 Introduction

Spelling correction is an old NLP task. From traditional non-neural methods (Hall and Dowling, 1980a; Gadd, 1988; Brill and Moore, 2000), to neural methods presents in recent years (Han and Baldwin, 2011; Eger et al., 2016; Silfverberg et al., 2016). The performance of spelling correction methods has been better and better and relatively mature.

In this paper, we investigate some old but really effective and impactful algorithms' performance on spelling correction task. Use a newly UrbanDictionary[1] dataset. The algorithm been evaluated includes Soundex, N-gram, Edit-distance and Editex.

2 Methods

In this section, we simply introduce the algorithms we evaluated in our paper.

Soundex was developed by Odell and Russell, and patented in 1918 (Hall and Dowling, 1980b). Uses codes based on the sound of each letter to translate a word into an at most 4 character's string. We call it Soundex code in this paper. Figure 1 shows the letter translation table and Algorithm 1 describes the 4 steps processing.

Algorithm 1 Soundex

```
Input: One string
Output: Soundex code of the string
Except for first character, translate string characters according to table.
Eliminate any adjacent repetitions of codes.
Eliminate all occurrences of code 0.
Return the first four characters.
```

N-Gram methods are string distance methods based on n-gram counts, where a n-gram of string s is any substring of s of some fixed length. This algorithm defines similarity of words by calculating n-gram distance, which was proposed by (Ukkonen, 1992) defines as:

$$|N_s| + |N_t| - 2|N_s N_t| \tag{1}$$

where N_s is the set of n-gram in string s.

aehiouwy	-->	0 (vowels)
bpfv	-->	1 (labials)
cgjkqsxz	-->	2 (misc: velars, fricatives, etc.)
dt	-->	3 (dentals)
l	-->	4 (lateral)
mn	-->	5 (nasals)
r	-->	6 (rhotic)

Figure 1: Soundex letter translation table

Edit distance presented by (Levenshtein, 1965), also know as Levenshtein distance, which is defined as the minimum number of elementary edit operations needed to transform one string into another. Algorithm 2 gives the main part of one kind of edit distance (global edit distance) with particular distance calculation strategy.

Algorithm 2 Global edit distance

```
Input: Two strings $s$ and $t$
Output: Edit distance between $s$ and $t$
$ls = length(s)$, $lt = length(t)$
Initialize $edit[0][0] = 0$
Initialize $edit[i][0] = i$, $edit[0][j] = j$
for $i = 0$ to $ls$ do
    for $j = 0$ to $lt$ do
        $edit[i][j] = \min($
        $edit[i-1]j + 1,$
        $edit[i][j-1] + 1,$
        $edit[i-1][j-1] + equal(s[i-1], t[j-1]))$
    end for
end for
return edit[ls][lt]
```

[1] A crowd-compiled dictionary of informal words and slang with over 7 million entries. http://urbandictionary.com

aeiouy --> 1	dt --> 4	fpv--> 7
bp --> 2	lr --> 5	sxz--> 8
ckq--> 3	gj--> 6	csz --> 9

Figure 2: Editex letter groups

Item	#Word
Testset size	716
Dictionary size	393954
Misspelled in Dictionary	175
Correct not in Dictionary	122

Table 1: Dataset statistics (# stands for the number of words)

Editex presented by (Zobel and Dart, 1996), is a method combines phonetic matching and edit distance. In this method, alphabets also be grouped similar with Soundex, shows in Figure 2 . Then, edit distance will be calculated with the grouping info, the distance between two letters in the same group defines 1 while the different groups defines 2. This scheme makes distance between similar phonetic words closer.[2]

3 Experiments

3.1 Data

For our experiments we use a particular dataset: a sub-sample of actual data posted to UrbanDictionary, in which a number of headwords taken from have been automatically identified as being misspelled (Saphra, 2016). This dataset simulates the spelling error data in the real world as much as possible so that is suitable for our task. Table 1 shows the statistics of our dataset.

3.2 Setting

Soundex Calculate the Soundex code for every word and then matched with global edit distance.

N-Gram We evaluate the N-Gram algorithm for n in range 1 to 9. For a particular n, we first pad (n-1) "♯" in the front and end of every word. This guarantee the building of n-gram set. For example, 5-gram set for word "he" defined as:

$$\{\sharp\sharp\sharp\sharp h, \sharp\sharp\sharp he, \sharp\sharp he\sharp, \sharp he\sharp\sharp, he\sharp\sharp\sharp, e\sharp\sharp\sharp\sharp\} \tag{2}$$

Edit Distance There are two kinds of edit distance algorithm, local edit distance (LED) and global edit distance (GED). In this paper, we implement both of them and evaluate them. For global edit distance, we have two distance calculation schemes : 1) (+1) for indel and mismatch and nothing to do for match; 2) (+1) for indel and mismatch and (-1) for match. The different between them will discuss in Section 4. For local distance algorithm, we use (-1) for indel and mismatch and (+1) for match, and always assign 0 if 0 is better.

Editex We calculate the Editex follows (Zobel and Dart, 1996) setting.

N	Predicted	Right	Precision	Recall
1	7150	183	2.56	25.56
2	1484	151	10.19	21.09
3	1429	149	10.43	20.81
4	1426	148	10.38	20.67

Table 2: N-gram algorithm results

Scheme	Predicted	Right	Precision	Recall
GED-1	5528	253	4.57	35.34
GED-2	2497	204	8.16	28.49
LED	727774	133	0.02	18.58

Table 3: Edit distance algorithm results

4 Results & Evaluation

4.1 Evaluation Method

For all experiments in this paper, we keep all parallel best results as matched corrections for a misspelling. For example, the Soundex code for word "adn" is "a35", while for "attain", "attainabilities", "adamas", "atom" etc. are the same. No matter how many words in dictionary have the same code with "adn", all of them seem equally. Although this may result in a bad performance on precision when the predicted set is large.[3]

4.2 N-Gram Algorithm Evaluation

We employ the original results to evaluate since the purpose of this paper is to illustrate the characteristics of different string matching algorithms through a spelling correction task, rather than find the best algorithm with parameters to correct the misspelled datasets.[4] We use recall

[2] Because of the space limitation, we can not describe detailed algorithm, it is illustrated in our code.

[3] We also do more experiments on parameter optimization. For example, randomly selecting three best matches as predicted words, which may increase precision for Soundex and local edit distance and impact recall for all algorithm. We can not list them here due to space limitation.

[4] The dataset we used contains only 716 words, even find the best algorithm with parameters, it might be some kind of overfitting of the particular datasets.

Method	Predicted	Right	Precision	Recall
Soundex	495146	436	0.09	60.89
Local Edit Distance	727774	133	0.02	18.58
Global Edit Distance	2497	204	8.16	28.49
N-Gram (N=2)	1484	151	10.18	21.09
Editex	2830	230	8.13	32.12

Table 4: Full Scale results

and precision as measured of algorithm performance follows (Raghavan et al., 1989).

The results of n-gram algorithm shows in Table 2, in which we see that when $N = 1$, the number of predicted words is large and recall is the highest. This illustrates that around 25.56% of the misspelled words are character shifted without deletions and insertions for the best matches for $N = 1$ means two words have the same consist of characters but maybe different order. However, the precision of $N = 1$ is really low which makes it not a good parameter for the algorithm. $N = 2$ and $N = 3$ gives more balanced and significant results. For $N \geq 4$, the results are all the same. The reason might be that padding $n - 1$ “♯” strategy for n-gram result in the related similarity tend to be stable when n is large.

4.3 Edit Distance Algorithm Evaluation

Table 3 shows the results of edit distance algorithm. Which indicates that although the first two lines are both global edit distance algorithm, different distance calculate strategy leads to divergent results. Both strategies with (+1) for indels and mismatches and the difference is that first strategy do nothing for matches while the second (-1). This difference always makes the first strategy matches more result. For example, calculate the edit distance between “aginst” with “against” and “agist”. The first strategy's result is the all equals 1 because “against” add one character and “agist” delete one character compare with “aginst”. But the second strategy calculate -5 for “against” and -4 for “agist”, which means “against” have more similar elements (characters) with “against”. We could not say which one is always better in practice because it depends on the application scenarios.

As for results of local edit distance algorithm. The predicted words is explosive large but both precision and recall are not satisfactory. Which indicates that this algorithm is not suitable for the task. The reason might be that misspelled words are not tend to be a part of the corresponding corrections or reverse. Nevertheless, tt does not means local edit distance is a bad algorithm. It is very powerful in other tasks such as gene alignments (Altschul and Erickson, 1986).

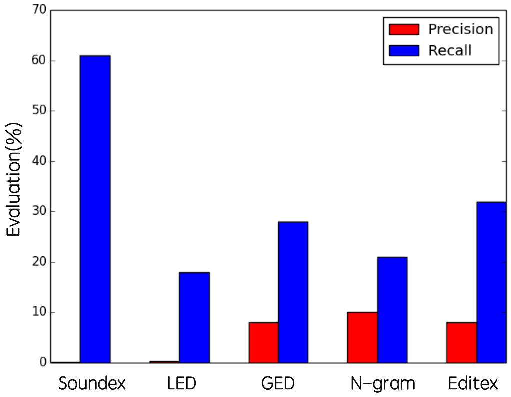

Figure 3: Evaluation visualization

4.4 Full Scale Evaluation and Demonstration

Table 4 shows a full scale evaluation and Figure 3 is the visualization of it. Table 5 demonstrates output of the best match of two words for each algorithm.

Table 4 indicates that both Soundex and local edit distance predicted plenty of best matches. The reason for Soundex is that for words with a fixed first letter, the number of different Soundex code is only 1000 (3 digitals from 0-9). This will lead to many words have the same Soundex code. The first two lines of Table 5 demonstrates the Soundex's results, from which we find the pronunciation of the matched set is somewhat similar but the spelling of them various, which will leads to a number of meaningless predicts.

Although the precision of Soundex is extremely low, it seems much better than local edit distance for both precision and recall, which also demonstrates in Table 5. Obviously, “actually” and any string with a substring “actually” are treated the same. This is the trait

Method	Misspelled	Correct	Matched set
Soundex	accually ahain	actually again	akal, axile, azalea, asylees, auxiliar, ... awin, annoy, aani, aoyama, anne, ayme, anay, ...
Local Edit Distance	accually ahain	actually again	actually, tactually, unactually, contactually, ... disenchain, rechain, toolchain, toolchains, ...
Global Edit Distance	accually ahain	actually again	actually chain, amain, arain, again, ghain, alain, hain
N-Gram (N=2)	accually ahain	actually again	actually, ally ain
Editex	actually ahain	actually again	usually, actually, annually, casually, chally, ... amain, attain, arain, again, alain, hain

Table 5: Output demonstration for all algorithms.

of local edit distance, part matching.

As for global edit distance matching, it is such a pretty good algorithm for this spelling correction task that get an acceptable precision and recall. Based on the analysis of Soundex's high recall and global edit distance's high precision and recall, it is not hard to think of computing edit distance based on the Soundex results. Editex algorithm makes use of Soundex information and combine it with edit distance algorithm.

We find that Editex gets almost the same precision with global edit distance but higher recall, which might because Editex combines the advantage of Soundex with edit distance. Form the last two rows of Table 5, we find that Editex matches words with small edit distance and also considers pronunciation.

5 Conclusion

We have investigated several classical approximate string matching algorithms with a particular spelling correction task and evaluated by recall and pricision. We find that Editex algorithm achieves better performance then others. Besides, we also illustrate the characteristics of each method include n-gram, local edit distance, global edit distance with different parameters, Soundex and Editex in the paper.

References

S. F. Altschul and B. W. Erickson. 1986. Locally optimal subalignments using nonlinear similarity functions. *Bulletin of Mathematical Biology*, 48(5-6):633–660.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proc. Meeting of the Association for Computarional Linguistics*, pages 286–293.

Steffen Eger, Tim Vor Der Brck, and Alexander Mehler. 2016. A comparison of four character-level string-to-string translation models for (ocr) spelling error correction. *Prague Bulletin of Mathematical Linguistics*, 105(1):77–99.

T. N. Gadd. 1988. fisching fore werds: phonetic retrieval of written text in information systems. *Program Electronic Library & Information Systems*, 22(3):222–237.

Patrick A. V. Hall and Geoff R. Dowling. 1980a. Approximate string matching. *ACM Comput. Surv.*

Patrick A. V. Hall and Geoff R. Dowling. 1980b. Approximate string matching. *ACM Comput. Surv.*, 12:381–402.

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 Computational Linguistics: Human Language Technologies - Volume 1*, pages 368–378.

V. I Levenshtein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Dokl.akad.nauk Sssr*, 10(1).

Vijay Raghavan, Gwang S. Jung, and Peter Bollmann. 1989. A critical investigation of recall and precision as measures of retrieval system performance. 7:205–229, 07.

Naomi Saphra. 2016. Evaluating informal-domain word representations with urbandictionary. In *RepEval@ACL*.

Miikka Silfverberg, Pekka Kauppinen, and Krister Lindn. 2016. Data-driven spelling correction using weighted finite-state methods. In *Sigfsm Workshop on Statistical Nlp and Weighted Automata*, pages 51–59.

Esko Ukkonen. 1992. Approximate string matching with q-grams and maximal matches. *Theor. Comput. Sci.*, 92:191–211.

Justin Zobel and Philip W. Dart. 1996. Phonetic string matching: Lessons from information retrieval. In *SIGIR*.