Spelling Correction Method Evaluation

Abstract

This paper investages several spelling correction methods. The main goal is to compare and analysis the performance of spelling correction methods on a peculiar data set, illustrate the characteristic of different string matchind algorithm.

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 (Han and Baldwin, 2011)(Eger et al., 2016)(Silfverberg et al., 2016) presented in recent years. The performance of spelling correction methods have been better and better and become relatively mature.

In this paper, we investigates some old but really effective and impactful algorithms' performance on spelling correction task. Use a newly UrbanDictionary[1] dataset. The algorithm been evaluated Includ 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.

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 silimarity of words by calculating n-gram distance, which was proposed by (Ukkonen, 1992) defines as:

$$|N_s| + |N_t| - 2|N_s N_t| \quad (1)$$

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

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.

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. 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 defins 2. This scheme makes distance between similar phonetic words closer.

Type	Words number
Testset size	716
Dictionary size	393954
Misspelled in Dictionary	175
Correct not in Dictionary	122

Table 1: Dataset

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 errer 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.

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

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

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 gurantee the building of n-gram set. For example, 5-gram set for word "he" defined as:

$$\{\sharp\sharp\sharp\sharp h, \sharp\sharp\sharp he, \sharp\sharp he\sharp, \sharp he\sharp\sharp, he\sharp\sharp\sharp, e\sharp\sharp\sharp\sharp\} \quad (2)$$

Edit Distance There are two kinds of edit distance algorithm, local edit distance and global edit distance. In this paper, we implement both of them and evaluate them. For global edit distance, we have two distance calculate scheme: 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 **??**. 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) settings.

4 Results & Evaluation

For all experiments in this paper. We keep all parallel best results as the predicted correction. That is, for example, the soundex code for word "adn" is "a35", while for "attain", "attainabilities", "adamas", "atom" etc. the code are the same. No matter how many word in dictionary have the same code with "adn", all of them seem equally. Although we know this may result in a bad performance on precision if the predicted set is too large.

We use this evaluation method because the purpose of this paper is illustrate the characteristics of different string matching alrothrims by spelling correction experiments on a particular dataset, rather than find a best algorithm with parameters to correct the misspelled datasets.[2]

4.1 N-Gram Algorithm Evaluation

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 also the highest. This illustrates that around 25.56% of the misspelled words are character shifted without deletion and insertion, because 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 so it is not a good parameter for this algorithm. $N = 2$ and $N = 3$ gives more balanced and signicicant resuls. For $N >= 4$, the results are all the same. The reason might be that padding $n - 1\sharp$ strategy for n-gram result in the related similarity tend to be table when n is large.

4.2 Edit Distance Algorithm Evaluation

Table 3 shows the resuls of edit distance algorithm. We see that even both the first line and second line are results of global edit distance, different distance calculate strategy result in widely divergent results. Both strategy with (+1) for indel and mismatch. The difference is the first do nothing for match while the second (-1). Which always makes the first strategy matchs more result. For example, calculate the edit distance between "aginst" with "against" and "agist". This first strategy's result is the same 1 because "against" add one character and "agist" delete one character. But the second strategy calculate -5 for "against" and -4 for "agist", which means "against" is more similar with "against" because they have more similar elements (characters). Both strategy have its characteristic and we can not say which one is better, it depends on the application scenarious.

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 corresponing correction or in reverse. Nevertheless, we can not say local edit distance is a bad algorithm. It is very powerful in other tasks such as gene alignments.

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

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 4: Demostrate of different algorithm's spelling correction result.

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 5: All method results

4.3 Comprehensive Evaluation

Table 4 shows the two words' demostrate output of the best match for each algorithm and Table 5 shows the results of all evaluation.

From Table 5 we knows that toth soundex and local edit distance predicted too much 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 plenty of words have the same soundex code.The first two lines of Table 4 demostrate the soundex's result, from which we may find the pronounce of the matched set is somewhat similar with the misspelled word. But the spelling of them are various, which may lead to a number of meaningless predicet.

Although the precision of soundex is extremely low, it's result seems much better than local edit distance for both precision and recall, which also demostrates in Table 4. Obviously, both "actually" and any string with a substring is "actually" are treated the same. This is the trait of local edit distance, part matching.

As for global edit distance matching, this is a pretty good algorithm for spelling correction that get a acceptable precision and recall. It is not hard to think of computing edis distance based on the soundex results. This is a naive combination of two algorithm which is exceeded by editex.

We find that editex can get 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 4, we find that editex matched words with small edit distance and also consider pronounciation.

5 Conclusions

References

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. Lex-

ical 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):845ndash;848.

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*.