COMP90049 Project1 Report

1. Introduction

String matching is becoming more and more important for many growing areas, such as information retrieval; and we often focus on the problem of string matching that allows errors, called approximate string matching (Navarro, 2001). There are amounts of methods described in papers, such as phonetic matching (Zobel et al. 1996) and edit distance (Navarro, 2001).

The purpose of this report is to implement some different approximate string matching algorithms to find a canonical form for each token with a document which comes from the social media platform Twitter, and then evaluate the performances of each algorithms.

Approximate string matching algorithms in this report include Global Edit Distance, Local Edit Distance, N-Gram, Soundex, and Editex; and the dataset is a sub-sample of actual Twitter data from the Workshop on Noisy User-generated Text (Baldwin et al. 2015), including 3 documents: *misspell.txt* (*Misspell*), *dict.txt (Dictionary)* and *correct.txt (Correct)*.

2. Related Work

A nearest neighbor search procedure is presented for automatic spelling correction using a trigram similarity measure (Angell et al. 1983); and for a higher optimized method, a spelling correction system is proposed to automatically correct spelling mistakes by using trigram and Bayesian approach with Brown corpus as a training set (Sharma et al. 2015). Also, a paper presents a ternary search tree data structure for search engine to select the best choice among all possible corrections for a misspelled token (Martins et al. 2004). As for the algorithm which correspond to what we learned in class, A language-independent spell-checker is proposed to improve the effectiveness of spelling correction by using revised N-Gram algorithm (Ahmed et al. 2009); and an efficient context sensitive spelling correction system is presented, which is combining approaches: edit distance and neighbor co-occurrence statistics computed from a specific corpus (Schierle et al. 2008).

3. Methods

The basic idea of this experiment has 3 steps:

(1) to check whether each token in *(misspell.txt)* is in *(dict.txt)*. If the token is in the dictionary, return the token directly, and continue the step 3; but if it is not in the dictionary, continue the step 2.

(2) to predict the best matches for each token in *(misspell.txt)* by different algorithms, based on the reference *(dict.txt)*.

(3) to evaluate the predictions by using precision and recall, based on the canonical form *(correct.txt)*.

In the next section, we will briefly introduce the algorithms and evaluation metrics we will use in this paper.

3.1 Global Edit Distance (GED)

Global Edit Distance, which measures the similarity of two strings, counts the minimum number of operations required to transform one string into the other. The weight of operations we will use in this paper is (match, replace, insertion, deletion) = (0, 1, 1, 1), also called Levenshtein distance (Schulz et al. 2002).

3.2 Local Edit Distance (LED)

Local Edit Distance, compares segments of all possible lengths and optimizes the similarity measure, also called Smith-Waterman algorithm (Smith et al. 1981).

3.3 N-Gram

N-Gram measures the similarity of two strings by calculating n-gram distance. In this paper, we compared 4 situations when N equals to 1,2,3,4 respectively.

3.4 Soundex

Soundex is a phonetic algorithm which can match different strings with similar sound, which have the same representation of 4-character code (Zobel et al. 1996).

3.5 Editex

Editex algorithm combines phonetic algorithms and edit distance to measure the similarity of two strings (Zobel et al. 1996).

3.6 Evaluation Metrics

We will use two evaluation metrics, which are

precision and recall.

(1) Precision:

Precision will be used to calculate the fraction of the number of returned correct predictions among the total number of returned predictions.

(2) Recall:

Recall will be used to calculate the fraction of the number of returned correct predictions among the total correct matches.

4 Results and Discussion

In this section, we will show the results of evaluation and analysis.

4.1 Dataset

The analysis for dataset is shown as follows (Table 1).

Dictionary Size	370099
Misspell and *Correct* Size	10322
Correct not in *Dictionary*	1778
Misspell in *Dictionary*	8376
Exact token matches	7805
Exact token mismatches	571
Misspell not in *Dictionary*	1946

Table 1: Analysis of dataset

From Table 1, we can easily know that the *Dictionary* is not very suitable for this task. One reason is because that there are only 1946 *Misspell* tokens not in *Dictionary*, other 8376 tokens will be directly returned as a result, which means that only 18.85% of *Misspell* is useful in this task. The other is because there are 1778 correct tokens which is not in the *Dictionary*, which means the highest recall in theory for this task is 82.77%, and the lowest recall is 75.62% (the fraction of correct tokens in *Dictionary* among the number of tokens in *Correct*), which means the recalls in different algorithms will look similar.

After we analyze the dataset, we will evaluate the results divided into two parts. One is to evaluate the precision and recall for the whole system, and the other is to evaluate these only for 1946 *Misspell* tokens which are not in the *Dictionary*.

4.2 Results

The results of whole system and only for 1946 tokens are as follows (Table 2 and Table 3).

	Predictions	Correct Predictions
GED	40,717	7,987
LED	1,202,213	8,289
1-Gram	14,373	7,892
2-Gram	11,549	7,938
3-Gram	11,448	7,936
4-Gram	11,386	7,935
Soundex	42,572	8,166
Editex	24,819	7,908

Table 2: Result for the Whole System

	Predictions	Correct Predictions
GED	32,341	182
LED	1,193,837	484
1-Gram	5,998	87
2-Gram	3,173	133
3-Gram	3,072	131
4-Gram	3,010	130
Soundex	34,196	361
Editex	16,443	103

Table 3: Result for 1946 Misspell Tokens

After calculation, the result of evaluation is shown as follows (Table 4).

	System Precision	System Recall	Precision (1946)	Recall (1946)
GED	19.62%	77.38%	0.56%	9.35%
LED	0.69%	80.30%	0.04%	24.87%
1-Gram	54.90%	76.46%	1.45%	4.47%
2-Gram	68.73%	76.90%	4.19%	6.83%
3-Gram	69.36%	76.88%	4.26%	6.73%
4-Gram	69.69%	76.87%	4.32%	6.68%
Soundex	19.18%	79.11%	1.05%	18.55%
Editex	31.96%	76.61%	0.63%	5.29%

Table 4: Result of Precision and Recall

The two figures (Figure 1 and Figure 2) are shown as follows to make it easier to distinguish, and we use 3-Gram to represent the evaluation of N-Gram.

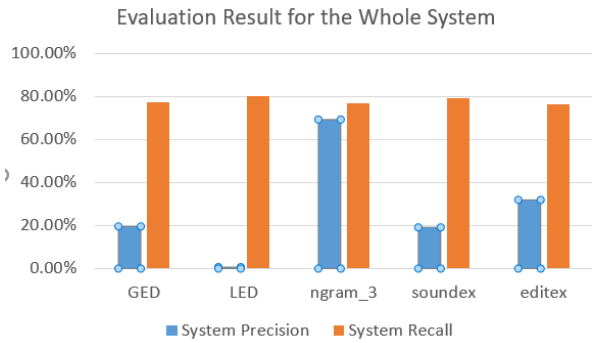

Figure 1: Evaluation Result for the Whole System

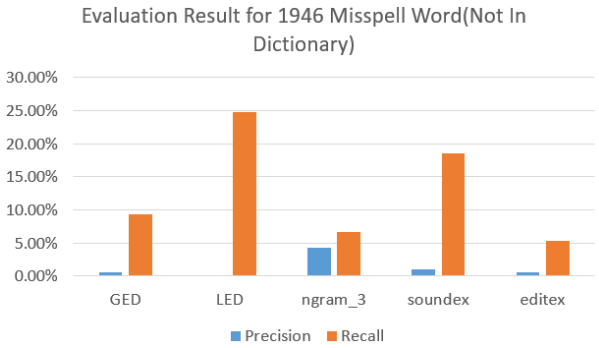

Figure 2: Evaluation Result for 1946 Misspell Tokens

N-Gram looks great for this task from Figure 1, but we should know that this is because of the disadvantages of the *Dictionary*.

We can easily see from Figure 2 that all these algorithms are not very effective for these tokens from Twitter.

4.3 Discussion

In this section, we will discuss the reason why these algorithms don't work well.

4.3.1 Noisy User-generated Text

It is worth to say that the tokens we use are from the actual data from Twitter platform, therefore, there are lots of noisy factors which can easily affect spelling correction.

4.3.1.1 Abbreviations

When we look at the results of our predictions and dataset, we find there are lots of abbreviations, some examples are as follows (Table 5).

Misspell	Correct
ight	alright
n	and
appt	appointment
ppl	people

Table 5: Examples of Abbreviations

These abbreviations really affect the effectiveness of algorithms. For example, for the misspell token "bc", the canonical form is "because". The predictions for different algorithms are as follows (Table 6). None of them get the correct prediction.

Almost all abbreviations are totally different from their canonical form, therefore, as for all algorithms, they may return lots of predictions and couldn't find the correct form.

Algorithms	Predictions
GED	abc ac b ba bac bb bcd bcf bch bd be bec bf bg bi by bk bl bm bn bo boc bp br bs bt bu bv bx bz c cc dc ec fc ic kc lc mc oc pc rc sc tc ubc uc vc wc xc
LED	bcd bcf bch bchs
3-Gram	abc bac bcd bcf bch bec boc ubc
Soundx	bbs bch bchs bk bks bps bs bsh
Editex	bbs bs

Table 6: Predictions for the string "bc"

4.3.1.2 Homophones

Some Homophone examples are as follows (Table 7).

Misspell	Correct
2	to / too
u	you
4	for

Table 7: Examples of Homophones

Homophones also seriously affect the predictions. For example, "4" means for in canonical form, but in edit distance, N-Gram and Soundex methods, it is not applicable. Also, as for "2", it can mean "to" and "too" based on the context of original Twitter. Therefore, it is difficult to predict this kind of word for all algorithms.

4.3.1.3 Cyber Word

Some examples are as follows (Table 8).

Misspell	Correct
hahaha	hahaha

exo	exo
ozil	ozil

Table 8: Examples of Cyber Word

The string “hahaha” means people laugh; the string “exo” is the name of a popular South-Korean band; and “ozil” is the name of a famous German football player. This kind of cyber words is not in the dictionary, which will definitely return wrong predictions, because these words are constantly changing, and new words are not added to the dictionary.

4.3.2 Algorithms

Table 4 shows that LED and Soundex methods have much higher recall. The reason for Soundex is that the there are lots of same 4-character code in *Dictionary*, which leads to many words returned. That increases the number of correct predictions, but also greatly decreases the precision; The reason for LED is that LED is only looking for the similar substring in *Dictionary*, which will lead to amounts of results. For example, the result for “ozil” is as follows (table 9).

GED	oil
LED	alguazil amazilia amizilis azilian azilut azobenzil benzil benzilic bezil bezils blowzily boozily brazil brazilein brazilette braziletto brazilian brazilianite brazilians brazilin brazilins brazilite brazils brazilwood breezily calabazilla courtzilite cozily crazily dizzily dozily frazil frenzily frizzily frowzily fuzil fuzils fuzzily gauzily glazily gozill hazily jazzily kizil kizilbash lazily manzil mazily motazilite muzzily obrazil oozily ozias ritzily sleazily swaziland tzotzil unhazily wheezily woozily wurtzilite
3-Gram	oil
Soundex	ouzel oxheal oxyl
Editex	axil gozill ixil oil oozier oozily ouzel oxyl zill

Table 9: Predictions for “ozil”

We have expected that Editex will have the best performances, which combines the advantages of GED and Soundex, but it didn’t. We think the reason is that too many noisy factors affect the performance.

Surprisingly, N-Gram works much better than other algorithms. For N = 1, 1-Gram means two strings have the same characters but differently order, whose performance is less than the situation for $N \geqslant 2$, which means 1-Gram is not suitable for this task. For $N \geqslant 2$, the performances are almost the same.

5 Conclusion

In summary, this report implements 5 classic and traditional algorithms to find the canonical form for each misspell token in Twitter Data and assess the effectiveness of these different algorithms.

We observed that noisy factors, which are abbreviations, homophones, and cyber words, greatly affect the effectiveness of traditional algorithms; and the dictionary is not suitable for this task.

For future work, we will add some common abbreviations, homophones, and cyber words into dictionary, and then assess the effectiveness of different algorithms.

6. References

Navarro, G. (2001). A guided tour to approximate string matching. ACM computing surveys (CSUR), 33(1), 31-88.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China.

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. Z¨urich, Switzerland. pp. 166–173.

Angell, R. C., Freund, G. E., & Willett, P. (1983). Automatic spelling correction using a trigram similarity measure. Information Processing & Management, 19(4), 255-261.

Sharma, Sumit and Gupta, Swadha. (2015). A Correction Model for Real-word Errors. Procedia Computer Science. 70. 99-106. 10.1016/j.procs.2015.10.047.

Martins, B., & Silva, M. J. (2004, October). Spelling correction for search engine queries. In International Conference on Natural Language Processing (in Spain) (pp. 372-383). Springer, Berlin, Heidelberg.

Ahmed, F., Luca, E. W. D., & Nürnberger, A. (2009). Revised n-gram based automatic spelling correction tool to improve retrieval effectiveness. Polibits, (40), 39-48.

Schierle, M., Schulz, S., & Ackermann, M. (2008). From spelling correction to text cleaning–using context information. In Data Analysis, Machine Learning and Applications (pp. 397-404). Springer, Berlin, Heidelberg.

Schulz, Klaus U.; Mihov, Stoyan (2002). "Fast string correction with Levenshtein automata". International Journal of Document Analysis and Recognition. 5 (1): 67–85.

Smith, Temple F. & Waterman, Michael S. (1981). "Identification of Common Molecular Subsequences" (PDF). Journal of Molecular Biology. 147 (1): 195–197.

Holmes, D., & McCabe, M. C. (2002, April). Improving precision and recall for soundex retrieval. In Proceedings. International Conference on Information Technology: Coding and Computing (pp. 22-26). IEEE.