A simple real-word error detection and correction using local word bigram and trigram

Pratip Samanta
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute, Kolkata
pratipsamanta@gmail.com

Bidyut B. Chaudhuri
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute, Kolkata
bbcisical@gmail.com

Abstract

Spelling error is broadly classified in two categories namely non word error and real word error. In this paper a localized real word error detection and correction method is proposed where the scores of bigrams generated by immediate left and right neighbour of the candidate word and the trigram of these three words are combined. A single character position error model is assumed so that if a word W is erroneous then the correct word belongs to the set of real words S generated by single character edit operation on W. The above combined score is calculated also on all members of S. These words are ranked in the decreasing order of the score. By observing the rank and using a rule based approach, the error decision and correction candidates are simultaneously selected. The approach gives comparable accuracy with other existing approaches but is computationally attractive. Since only left and right neighbor are involved, multiple errors in a sentence can also be detected (if the error occurs in every alternate words).

Keywords: Real word error, Local context.

1. Introduction

Word error is a major hindrance to the real world applications of Natural Language Processing. In textual documents, word-error can be of two types. One is non-word error which has no meaning and other is real word error which is meaningful but not the intended word in the context of the sentence. Of these, non-word has been widely studied and algorithms to detect and suggest correction word for the error have been proposed. These algorithms are generally termed as spell-checker, which are integrated in various word-processing software like Microsoft Word[1], LibreOffice Writer[2], Ispell[3], Aspell[4] etc. For error occurring at two positions of a word, the commercial spell checkers work fairly well. Some studies on spell checking approaches are found in [1-5], that include English and non-English language like Bangla.

However, the problem of real-word error is a more complex one. Usually, such error disturbs the syntax and semantics of the whole sentence, which requires human-being to detect it. However, an automatic syntactic/semantic analysis of a 'correct' sentence itself is a difficult

1 Microsoft Word is a word processor developed by Microsoft.
2 LibreOffice Writer a free open-source word processor.
3 Ispell is spell-checker for Unix.
4 Aspell is a free spell-checker for GNU software system.

task and the analysis of an 'erroneous' sentence is almost impossible in most cases. Any word-error can be represented in terms of insertion, deletion or substitution of one or more character. If we consider 'space' as one character, the problem can become more complex. For example, the word 'within' can become 'with' and 'in' if a 'space' is inserted wrongly after 'h'. Conversely, 'with' and 'in' can be merged to 'within' if a 'space' is unintentionally missed. This can be regarded as 'Split error' or 'Prune-on error'. Exploring further, 'these' can be split into 'the' and 'se'. This is an example of mixed case where the first part is real-word error and the second part is non word error, making them more difficult to correct. Real-word errors are also found in dyslexic text written by person having Dyslexia. Moreover, not only human-beings, these errors can occur due to 'Auto Correction' feature of some word processing software [6]. Sometimes by man and machine together, when user chooses a wrong word from list of suggestion against a flagged error by word processing software [7] .

To the best of our knowledge, the problem of real-word error is still at the research and development stage where instead of going at the full sentence level, anomaly is searched at the word bigram or trigram level. The first work in this direction was due to Mays et al. [8] who considered word trigram i.e. Second order Markov process for language modelling. If a word (W) in the sentence is unintended (i.e. erroneous) , then the correct word is assumed to come from the members of confusion set of real word C(W) of W generated by single edit operation. In this model, the observed word W is assumed to be correct with probability or degree of belief α. Hence any member of C(W) is equally likely to be a correction candidate with constant probability (1- α) / n where n is the cardinality of C(W). The member for which the sentence probability is maximum is the correction word.
In this paper we present a simpler method to deal with the real word errors based on bigram and trigram model. The method tries to detect an error by noting bigrams and trigram constituted by immediate left and right neighbour of candidate word and then generate some suggestions according to ranks/score calculated for the correction set of words. Here we use BYU[5] corpus of bigram and trigram corpus while test our method on text from Project Gutenberg[6].
This paper is organized as follows. Section 2 covers overview of related work. In section 3 we present our method. Section 4 highlights evaluation and experimental results. Concluding remarks are given in section 5.

2. Related Work

Apart from Mays et al. [8], several other methods have been proposed to handle real word spelling error problem. They are mainly based on either semantic information or machine learning and statistical method.
Among them, Golding and Schabes [9] introduced a hybrid approach called 'Tribayes' combining Trigram and Bayes' method. Trigram method uses part-of-speech trigrams to encode the context whereas Bayes' is a feature-based method. They use two types of features : context word and collocations. Their method worked better than MS-Word on a predefined confusion set. Later Golding with Roth [10] proposed a Winnow-based method for real word detection and correction. They modified the previous method [9] by applying a winnow multiplicative algorithm combining variants of winnow and weighted majority voting and achieved better accuracy. However , they used a small data set in their experiment. Word-net was considered as first lexical resources for real word error by Hirst and St-Onge

5 Details of Brigham Young University corpus can be found at http://corpus.byu.edu/.
6 Project Gutenberg is a collection of free electronic books, or ebooks. Details is available at http://www.gutenberg.org

[11]. They used a robust database of 1987-89 Wall Street Journal corpus as test data. Following them [11], Hirst and Budanitsky [6] made a study of the problem on same corpus of Wall Street Journal. Their method identifies tokens that are semantically unrelated to their context and was not restricted to checking words from predefined confusion set. They achieved Recall of 23%-50% and Precision of 18%-25%. In another Word-net based approach, Peddler [12] showed that semantic association can be useful in detecting real-word error using some confusion sets especially in case of Dyslexic text [13]. She achieved recall and precision of correction 40% and 81%, respectively for Dyslexic text. But most of these approaches consider that writers make spelling error by writing words which are semantically closer to what they intended to write. But this may not be generally true for Real-word error. W. O. Hearn et al. [7] analysed the advantages and limitations of Mays' [8] method and present a new evaluation to compare with Hirst and Budanitsky [6]. They showed that optimizing over sentences gives better result than the variants of the algorithm which optimize over fixed length of windows on WJS corpus data.
Statistically based approaches are highly dependent on corpora--its size and correctness. Results vary with its size and existence of words. Our approach is based on the notion that words used less frequently are less likely to be an instance of real-word error.

3. Proposed Method

Our proposed algorithm initially chooses a confusion set for each candidate word using Levenshtein distance [14] equal to one from the dictionary words. Then it calculate the ranks of the elements of the confusion set. Based on that it detects an error and suggests some words against the detected error. Both detection and suggestion are computed simultaneously, which is an advantage of this algorithm.

3.1. Confusion Set by Levenshtein Distance

The confusion set for a test word (W) is a set of words from the lexicon which can generate W by single edit operation. As stated, we use Levenshtein Distance, also known as as minimum edit distance, which is the minimum number of edit operations required to transform one word into another. An edit operation is either an insertion, deletion or a substitution of a character in the word. In our proposed model, for simplicity, we consider single error in the word. The confusion set may be represented as

$$C\left(W^i\right)=\left\{W^i_1,W^i_2,\ldots,W^i_j,\ldots,W^i_{k_i}\right\}$$

where W^i is the i-th word in the test sentence and k_i is number of elements in the $C\left(W^i\right)$. To generate this set we use a list of approximately 110,000 English words[7]. For convenience we rename W^i as W^i_0 and define

$$C'\left(W^i\right)=\left\{W^i_0,W^i_1,W^i_2,\ldots,W^i_j,\ldots,W^i_{k_i}\right\}$$

3.2. Forming N-gram model

Now we consider the sets of left bigram, right bigram and trigram for each member of $C'\left(W^i\right)$. We try to form them by taking left word, right word and both of them (for

7 http://www-01.sil.org/linguistics/wordlists/english/wordlist/wordsEn.txt

trigram). By this, the number of each type of Bigrams as well as Trigrams generated for W^i is k_i+1 . Thus for W^i , the following Bigrams and Trigram will be generated.

Left Bigram : $W^{i-1}W_j^i$
Right Bigram : $W_j^i W^{i+1}$
Trigram : $W^{i-1}W_j^i W^{i+1}$
where $0 \le j \le k_i$
Next we count the occurrence of these bigrams and trigrams from the BYU n-gram corpus of English.

3.3. Estimating N-Gram Probabilities

One of the ways to calculate probability of the sentence in N-gram model is using Markov chain rule. According to Markov assumption, probability of some future event (next word) depends only on a limited history of preceding events (previous words). For example in a bigram language model for a sentence of m words $W^1, W^2, \ldots, W^m$ it can be calculated as

$$P\left(W^1, W^2, \ldots, W^m\right) = P\left(W^1 | Ь\right) P\left(W^2 | W^1\right) P\left(W^3 | W^2\right) \ldots P\left(W^m | W^{m-1}\right) P\left(Ь | W^m\right)$$

where Ь denotes blank.
In our model we do not calculate the sentence probability. We take a weak assumption that occurrence of any event (word) depends on its previous and next events (words) only and independent of other events (words) in the sentence. By Maximum Likelihood Estimation we get the bigram and trigram probabilities as

$$P_1\left(W_j^i | W^{i-1}\right) = \frac{count\left(W^{i-1}W_j^i\right)}{\sum_{r=0}^{k_i} count\left(W^{i-1}W_r^i\right)} \quad (1)$$

$$P_2\left(W_j^i | W^{i+1}\right) = \frac{count\left(W^{i+1}W_j^i\right)}{\sum_{r=0}^{k_i} count\left(W_r^i W^{i+1}\right)} \quad (2)$$

$$P_3\left(W_j^i | W^{i-1}, W^{i+1}\right) = \frac{count\left(W^{i-1}W_j^i W^{i+1}\right)}{\sum_{r=0}^{k_i} count\left(W^{i-1}W_r^i W^{i+1}\right)} \quad (3)$$

By equation (1) , we calculate P_1 of each element of confusion set for each word using left bigram count. The denominator here represents the summation of all bigrams consisting of the previous word and one word from the confusion set. We get the counts directly from the BYU corpus. In the same way we compute P_2 using right bigram count in BYU corpus by equation (2). We compute it for every elements in respective confusion set so that the following condition is satisfied :

$$\sum_{r=0}^{k_i} P_1\left(W_r^i | W^{i-1}\right) = 1$$
$$\sum_{r=0}^{k_i} P_2\left(W_r^i | W^{i+1}\right) = 1$$

For equation 3, we use the trigram count of BYU corpus. The denominator here represents the summation of all trigram consisting of the previous word, one word from the confusion set and the next word. We get numerator value as before. We do it for every elements in confusion set so that it implies :

$$\sum_{r=0}^{k_i} P_3\left(W_r^i | W^{i-1}, W^{i+1}\right) = 1$$

We combine the probability estimates of equations (1), (2), (3) into a score of evidence that a W_j^i may be correct alternative to W^i. The score can be obtained by simple addition as follows. The values obtained from equation (1), (2) and (3) can be combined to get the final score $Score\left(W_j^i\right)$ by adding up. We use both the bigrams and trigram to be less dependent on a particular bigram or trigram.

$$Score\left(W_j^i\right) = P_1\left(W_j^i | W^{i-1}\right) + P_2\left(W_j^i | W^{i+1}\right) + P_3\left(W_j^i | W^{i-1}, W^{i+1}\right) \quad (4)$$

Note that 0 <= $Score\left(W_j^i\right)$ <= 3. Later on we noted that simple addition does not lead to best results. Hence we go for a weighted combinations score.

3.4. Weighted combination score

Higher and lower order n-gram models have different strengths and weaknesses . High-order n-grams are sensitive to more context, but have sparse counts. On the other hand, low-order n-grams consider only very limited context, but have robust counts. In order to follow the principle of Interpolation we put a weighting scheme on score generated by trigram. Combining them like equation (4) :

$$Score\left(W_j^i\right) = \lambda_1 P_1\left(W_j^i | W^{i-1}\right) + \lambda_2 P_2\left(W_j^i | W^{i+1}\right) + \lambda_3 P_3\left(W_j^i | W^{i-1}, W^{i+1}\right) \quad (5)$$

The values of λ_1, λ_2 and λ_3 can be computed by optimizing the accuracy on the training set. Let $\lambda_1 + \lambda_2 + \lambda_3 = 1$. We noted by trial and error that the results on the training set are best if $\lambda_3 = 2\lambda_2 = 2\lambda_1$. Then $\lambda_1 = \lambda_2 = 0.25$ we get $\lambda_3 = 0.5$. Also, W_j^i is limited by

$$0 \le Score\left(W_j^i\right) <= 1$$

3.5. Error detection & choice of suggestions

To confirm a word as a real-word error, we set some rules. At first, we arrange the score for members of confusion set in a descending order. Also a Stemming[8] method described later with an example, is used in our error detection. In addition to the above, we have used the

8 According to Wikipedia, 'In linguistics morphology and information retrieval, Stemming is process for reducing inflected (or sometimes derived) words to their stem, base or root form-generally a written word form'.

apriori belief that the observed test word is not a real word error. In their experiment Mays et al. obtained optimum value of this belief as 0.99 [8] which is used in our case as well. In other words, we believe that the test word can be a real word error in 1% cases. This value is used in normalizing the score in the real-word error detection algorithm described below.

Let W^i be a test word in the sentence. If W^i has a suffix part it can be stemmed into a root word say W_s^i . But if W^i is a root word, it cannot be stemmed. Thus, W_s^i may or may not exist, depending on the nature of W^i . Now, depending on the scores we make the decisions described in pseudo-code as follows.

Begin
if $Score(W^i) = 0$
    if W_s^i exists
        if $Score(W_s^i) = 0$
            declare W^i as real-word error
        else
            W^i is correct
        end if
    else
        declare W^i as real-word error
    end if
else
    if W_s^i exists
        if $Score(W_s^i) <$ 0.01 * Score of Top-ranked element of confusion set
            declare W^i as real-word error
        else
            W^i is correct
        end if
    else
        W^i is correct
    end if
end if
End

The above rules are now illustrated by an example. In a stream of text "... *new lodger made his appearance ink my modest bachelor quarters, but I was not ...*" , the word 'ink' is actually a real-word error.
In our approach, the system starts processing one word after another. While processing the word 'his', we have the confusion set { his, him, this, is, has }. For each of these words we calculate the score the score and arrange it in decreasing order of magnitude, as shown in Table 1.

Rank	Confusion Word	Score
1	his	0.3153
2	him	0.1177
3	this	0.0619

4	is	0.0044
5	has	4.9629E-4

Table 1: Confusion set for word “his” with score

Since 'his' is on top of the list, the system infers that it is a correct word.
In case of the word '*appearance*', we get the following results and it is also declared as a correct word.

Rank	Confusion Word	Score
1	appearance	0.2256
2	appearances	0.0243

Table 2: Confusion set for word “*appearance*” with scores

Now, for the word 'ink' we get the confusion set { in, sink, ink }

Rank	Confusion Word	Score
1	in	0.4998
2	sink	1.2672E-4
3	ink	0

Table 3: Confusion set for word “ink” with score

However, there is no count of the bigrams 'appearance ink' and 'ink my' as well as no count of the trigram 'appearance ink my' in the BYU corpus. So, the score of 'ink' is 0 and hence it is declared as real-word error. Since 'in' tops the score, the system considers it as the correct suggestion.
But score zero may not always mean that the word is an error. Sometimes a bigram/trigram score may be zero because it is absent in the particular corpus. For example, consider the word 'quarters'. From BYU corpus we get

$$Count(bachelor\ quarters) = 0$$
$$Count(quarters\ but) = 0$$
$$Count(bachelor\ quarters\ but) = 0$$

So we shall get zero score for the word 'quarters' and the system would declare 'quarters' as a wrdong word. But in reality 'quarters' is a correct word. So, the system will make an error. In order to reduce such incorrect decisions we do a kind of suffix stripping or stemming. In this case if we strip the plurality suffix -s, we get 'quarter'. Now, the bigrams and trigram generated by 'quarter' are not null in the corpus and hence the score is also non-zero, as shown in Table 4. Thus we include the stemmed word in the confusion set if the score for the test word is zero.

Rank	Confusion Word	Score
1	quarter	0.25
2	quarters	0

Table 4: Confusion set for word "*quarter*" with score

Some of the frequent elements we consider for stemming are given below :

{d, n, r, s, y, ed, es, ly, ies}

Now we have a word (W^i) decided as real-word error or not along with scores of the members of its confusion set. If decided as real-word error, we rank the members of the confusion set in descending order as suggestions for correction.

4. Experimental results and discussion

In order to evaluate our approach we collected test data from Project Gutenberg. We chose Project Gutenberg because it contains simple text files only, especially with no pictures i.e. only stream of text. Our data consists of around 100 files (approximately 25000 words) with headings removed.

We simulated real-word error synthetically and subject this erroneous document to our error detection and correction system. To make such a corrupted document, one in every 20 words is chosen. Suppose this current word is W. Then W is converted into a set of strings by one edit operation (insertion, deletion, substitution) at one character position. If W contains n characters then n substitutions, n deletions and n+1 additions will create 3n+1 strings. From all the generated strings we find those which are valid words. One of these valid words is chosen at random and W is replaced by this word. In this way we introduce 100/20 = 5% real-word error in the corpus. Here we have considered real-word error generated by single operation like substitution, deletion or insertion.

While typing people make single position character mistake in between 60%-80% of the erroneous cases [2]. A small portion of that becomes real-word error. Out of the rest 20%-40% two or more position mistakes, the chance of getting real-word error is even smaller. So, single portion mistyping based model can take care of a very high percentage of real-word errors. We give this qualitative statement because we did not find any robust statistics of the real-word errors presented in the published literature.

The performance of our approach can be evaluated from three aspects. The first one is to compute Precision and Recall of real-word error detection. Let n_1 be the number of total errors, n_2 be the be number of total detection and n_3 be the number of correct detection. Then,

$$Precision = \frac{n_3}{n_2}$$

$$Recall = \frac{n_3}{n_1}$$

While precision gives how precisely the system detects the error, we do not get an estimate on relative number of errors made by the system. The number of errors made by the system is

$(n_2 - n_3)$. However, n_2 can theoretically be equal to the total number of words in the test corpus (say N). So, we may normalize $(n_2 - n_3)$ with respect to N and represent it in percent as

$$Percent\,of\ erroneous\ detection = \frac{(n_2 - n_3)}{N} * 100\ \%$$

Table 5 shows Precision, Recall and percentage of erroneous detection. It is significantly better than [6] though test database is different.

Detected Real-word Error		Erroneous Detection
Precision	Recall	
71%-79%	81%-88%	1%-2%

Table 5: Evaluation results of Detection by our approach

If words in the sentence are real-word error which have been detected by the system, then the relative ranks of correct suggestion generated by the system is shown in chart 1.

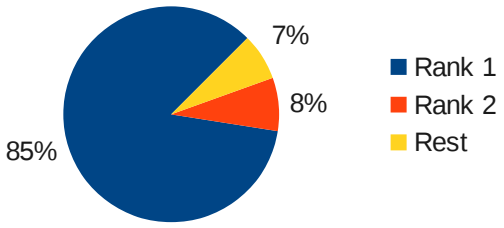

Chart 1 : Evaluation results of ranks of suggestions

It is noted that top-ranked suggestion is correct in 85% cases. Also, the correct suggestion lie in the top two ranks in 85+8 = 93 % cases. This shows that our proposed method can work very well by substitutions from our ranked list.

5. Conclusion

A simple but effective real-word error detection and correction approach is proposed here that employs only two bigrams and one trigram around the test word in a sentence. Since it works in a small neighbourhood around the test word, possibility of detecting and correcting more than one real-word error exist. The overall performance of the system on a moderate test set is quite satisfactory and comparable with those of state art correction systems. Evaluation of this method on global databases like Wall Street Journal corpus is a future scope of the work. The n-gram database used here is not huge, hence many valid bigrams and trigrams are not found in it, thus making the system less accurate. We tried to reduce such error by employing the stemming based method. This system may be further strengthened by using Word-net, which is our plan for future work. Test of this approach for Indian language text is another scope of future study.

6. Acknowledgement

Authors would like to thank Supriya Das and Purnendu Banerjee for useful discussion.

7. References

[1] F. J. Damerau. A technique for computer detection and correction of spelling errors, communication of ACM, 7(3), 171-176, 1964.
[2] Karen Kukich. Techniques for automatically correcting words in text, ACM Computing Surveys, 24 (4), page 377 - 439, 1992.
[3] B. B. Chaudhuri. Reversed word dictionary and phonetically similar word grouping based spell-checker to Bangla text, Proc. LESAL Workshop, Mumbai, 2001.
[4] Joseph J. Pollock and Antonio Zamora. Automatic spelling correction in scientific and scholarly text, Communication ACM, 27(4):358–368, 1984.
[5] Peterson James. Computer Programs for Detecting and Correcting Spelling Errors, Computing Practices, Communications of the ACM, 1980.
[6] G. Hirst and A. Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. Natural Language Engineering, 11(1):87–111, March 2005.
[7] L. A. Wilcox-O'Hearn, G. Hirst, and A. Budanitsky. Real-word spelling correction with trigrams: A reconsideration of the mays, damerau, and mercer model. In Proceedings of CICLing-2008 (LNCS 4919, Springer-Verlag), pages 605–616, Haifa, February 2008.
[8] E. Mays, F. J. Damerau and R. L. Mercer. Context based spelling correction. Information Processing and Management, 27(5):517–522, 1991 .
[9] A. R. Golding and Y. Schabes. Combining Trigram-based and Feature-based Methods for Context sensitive Spelling Correction. Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics,71-78, 1996.
[10] A. R. Golding and D. Roth. A winnow-based approach to context-sensitive spelling correction. Machine Learning, 34(1-3):107–130, 1999.
[11] G. Hirst and D. St-Onge. WordNet: An electronic lexical database, chapter Lexical chains as representations of context for the detection and correction of malapropisms. Pages 305–332, The MIT Press, Cambridge, MA, 1998.
[12] J. Pedler. Using semantic associations for the detection of real-word spelling errors. In Proceedings from The Corpus Linguistics Conference Series,vol. 1,no. 1,Corpus Linguistics, 2005.
[13] J. Pedler. Computer Correction of Real-word Spelling Errors in Dyslexic Text. PhD. Thesis, Birkbeck, London University , 2007.
[14] Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, No. 10,707-10, 1966.