Spell Checker and Corrector Based on Word Segmentation, Levenshtein Distance and Grammar Analysis

Chen Chen chen680@usc.edu
Yawei Huang yaweihua@usc.edu
Sichi Zhang sichizha@usc.edu
Shuo Zhao zhaos@usc.edu

Abstract

In this paper, we present a system to detect and correct misspelled word based on edit distance, word frequency and grammar structure. In the first phase, all misspelled words are selected based on the combination of dictionary look-up and word segmentation. The second phase gives top four suggestions based on edit distance. The system give scores to all suggestions and select most suitable word. An experiment is conducted to evaluate the proposed approach by comparing the output text with those manually selected. The results show that the proposed approach achieved an accuracy of more than 90% in detecting misspelled words and more than 70% in correcting misspelled words.

Keywords: Spell Checker, Word Segmentation, Edit Distance, Grammar Analysis

1 Introduction

This project is targeted to highlight and suggest how to correct English word spelling errors and sentences grammar issues when people compose articles. It is common that people may not pay attention of spelling and grammar, and forget to check after finished. Since the word spelling may also affect the sentence grammar, we will combine the spelling check with grammar verification after the entire sentence is complete, and highlight the incorrect word or sentence segment with correction suggestions.

The spelling and grammar rules are sophisticated for linguist. It is hard to write grammar rules. We are curious about how Google does spelling and grammar correction well and quickly. How to prepare the data is also a huge problem.

2 Related Work

There are several existing technologies that have been developed from different approaches. The most common one is based on Edit Distance to check against a dictionary. If the word does not exist, the similar ones would be suggested as correction. Also the misspelling correction can be achieved by statistical method such as Hidden Markov Model[1].

3 Data Collection and Annotation

(This section of work and report are contributed by Sichi Zhang)

To collect data for development, we focus on two aspects: one is every individual misspelled word, and second is the misspelled word in a sentence.

3.1 Individual Misspelled Word

When considering the first situation, we select the Birkbeck spelling error corpus[2] as dataset source, because it includes both misspelled words and correct words from free writing and spelling test. The data is from schoolchildren, university students and adult literacy students, which could cover a large range of misspelled words. Besides, we also manually engineer some misspelled words by randomly insert, remove and switch characters in a word.

Since the Birkbeck corpus only includes the misspelled and correct words pairs with other information, we need to extract the data, change the format and label every word with TRUE or FALSE to indict correct or misspelled word respectively. Also because the corpus considers the American word spelling is wrong, we need to

manually inspect each word to see if it's correct in American spelling and correct the label.

For correct word dataset, we used Apple system words file that cover 236K unique words, and Brown words corpus that include 60K words with its usage frequency.

The size of development dataset is around 2K words, and test dataset is about 8K words.

3.2 Misspelled Word in a Sentence

Besides of each individual word, we also need to consider the context and grammar structure in every sentence when correcting the misspelled words. In this case, we select Wikipedia[5] as data source and select sentences that lengths are between 10 to 50 words. Then we manually engineer each sentence to include 2 - 4 misspelled words and grammar errors.

To generate the language model, we decide to use Gutenberg 2003 CD set[4] that contain about 600 books data.

The development set includes 10 sentences about 30 misspelled words, and the test set has 100 sentences, around 350 incorrect words.

4 Technical approach

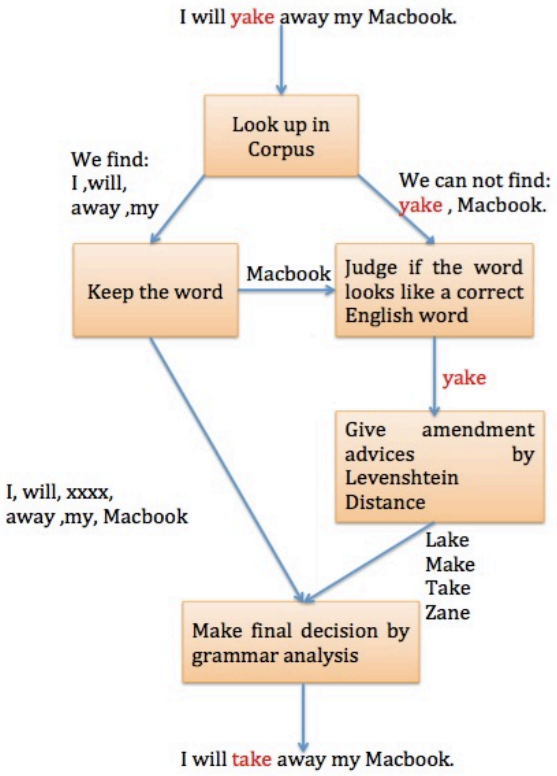

Figure 1: System workflow

Figure 1 shows working flow of project. Our work contains building segment-based spell checker and grammar-based corrector.

4.1 Find if a word is correct

(This section of work and report are contributed by Chen Chen)

There is no vocabulary can contain all words so merely looking up words may result in claim correct words wrong. Therefore, we add the part to judge if the word looks like a correct English when we cannot find the word in the vocabulary.

We decide to judge if a word looks like a correct English word by judge if it mainly consists of frequent letter sequences. A regular English word should contain some typical letter sequence like *tion, ment or ful*. So we assume that a word has a high probability to be correct if a word consists of many typical letter sequences. The drawback of this method is that it may approve wrong word consisting of many typical sequences. However, our system focuses on typo so we assume nobody will intentionally input a wrong word purely with frequent sequence like *fultion*.

To judge if a word mainly consists of frequent sequences, we have following steps:

1) Calculate the statistical data on frequency of all 2-letter sequences, 3-letter sequences, 4-letter sequences and 5-letter sequences.

2) Transfer the frequency statistics to the rank of frequency. For example, *tion* ranks 1st among all 4-letter sequences and *ing* ranks 5th among all 3-letter sequences.

3) Define rank(abc) is the frequency rank of letter sequence abc.

4) Divide the input word into letter sequences in order to get the lowest product of rank, the state transition equation of dynamic programing is:

$$LPOR(exampleword) = \min \begin{cases} LPOR(examplewo) * rank(rd) \\ LPOR(examplew) * rank(ord) \\ LPOR(example) * rank(word) \\ LPOR(exampl) * rank(eword) \end{cases}$$

(LPOR stands for Lowest Product of Rank)

5) Now we have the lowest product of rank of the word, we calculate its final score by following equation:

$$FinalScore(exampleword) = \frac{LowestProductOfRank(exampleword)}{min\begin{cases} rank(exam) \\ rank(ple) \\ rank(word) \end{cases}}$$

6) We divide the Lowest Product of Rank by min{ rank(exam),rank(ple),rank(word) } in order to detect wrong words contains good frequent letter sequences. For example, the wrong word "*cotribution*" may have a good score since it contains "*tion*". The word with typo contains only 1 or 2 bad sequence. Therefore, we want to rule out the best sequence when evaluating if the word is correct.

7) Group the words by their length and set the threshold for word of different length. If the final score of the word is lower than the threshold, we claim it correct.

We also considered following word segmentation method and actually implemented some algorithm, but it turned out none of them has higher accuracy than Lowest Product of Rank algorithm above.

1) Highest product of sequence frequency
We considered this algorithm but did not implement this one because it tended to match the shorter sequence. For example, the frequency of "*ti*" and the frequency of "on" are at least as high as frequency of "*tion*". So Highest Product of Frequency will segment the word into "*ti, on*" rather than "*tion*" since n*n>n.

2) Highest product of sequence probability
We actually expected this algorithm be the best at first time. So we implemented this method and it turned out it was harder to set a threshold since it can not split wrong words and correct words as good as lowest product of rank.

4.2 Provide Modification Suggestions

(This section of work and report are contributed by Shuo Zhao)

Around 80% to 95% of the mistakes are happened within 4 types below (Take word "spell" for example):

1) one letter wrong (speel)
2) one letter omitted (spll)
3) one letter inserted (speell)
4) two adjacent letters transposed (sepll)

It came naturally that we can use edit distance[3] to generate our suggestions. However, it is not that possible that a word is misspelled on too many letters. Thus we select all valid words that have edit distance less than 3.

The next step is to evaluate the results and give 4 best suggestions. We are using Naive Bayesian formula here:

To calculate the probability a misspelled word w is correct words, it should be:

$$P(s|w) = \frac{P(w|s)P(s)}{P(w)}$$

Assuming P(w) for all words are the same, P(w|s) is the edit distance between two words, P(s) is the general frequency of s, we can choose top 4 suggestion based on score ranking and return them to grammar and content checker.

However, for word with edit distance of 1 and 2, they should be treated evenly because people are more likely to make only slight mistakes. That

motivates us to modify the formula by giving a suitable boost to word with distance 1. The result shows nearly 10% rise because of the new formula.

4.3 Select Correct Modified Word Using Grammar Analysis

(This section of work and report are contributed by Yawei Huang)

Firstly, we used the data collected in the gutenberg CD set[4] in August 2003 as the training data and mac system words as vocabulary to learn a 3-gram language model. The corpus initially contains 600 of the best Ebooks in txt format. We have to extract the contents except the directory, copyright parts then generate them into one txt file. The vocabulary contains normal words plus tense one word per line.

Secondly, we used SRILM[6] to build a statistical language models. The basic approach is to generate and manipulates 3-gram counts, and estimates 3-gram language models from corpus. The program first builds an internal 3-gram count set, either by reading counts from a file, or by scanning text input. Following that, the resulting counts can be output back to a file or used for building a 3-gram language model in ARPA format.

Finally, we generated the suggestion sentences based on the suggestion word for each wrong word and gave the permutations. For example, the initial sentence has 3 wrong words. We gave 4 suggestions for each word from spelling part. Then we generate total 34 sentences as the suggestion list. The input for kenlm[7] is the suggestion list plus the language model. We estimated the score for each sentences from text using modified Kneser-Ney smoothing[8] without pruning. The max score sentences are the grammar suggestion we give as final output. The workflow is as follows in Figure 2.

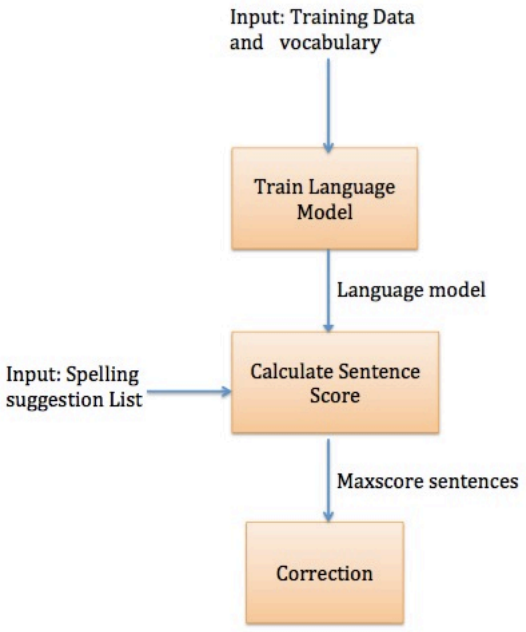

Figure 2: Spelling correction based on contents and grammar workflow

5 Result & Analysis

5.1 Result of misspell checking

1) Misspell checking merely by looking up in vocabulary

We manually created testing dataset that contains 50% misspelled word and 50% correct words, check if our spell corrector checked false word as expected. The summarized result is shown in Table 1. The result on words of different length is shown in Table 2. The system precision is 0.977, recall is 0.880 and F-score is 0.926.

Predict \Actual	True	False	Total
True	3392	79	3471
False	460	3773	4233
Total	3852	3852	7704

Table 1: Confusion matrix of baseline approach

Word Length	4	5	6	7
F-score	0.919	0.924	0.928	0.933

Word Length	8	9	10	11
F-score	0.930	0.933	0.922	0.928

Word Length	12	13	14	15
F-score	0.927	0.920	0.924	0.917

Word Length	16	17
F-score	0.915	0.915

Table 2: F-score of different word length based on word dictionary

For checking accuracy, the system gained a pretty great F-score. However, the system is more likely to predict the write spelling word into wrong one, that is because that the dictionary only contains around 60,000 words and is old, new words or rare words are not showing up in the dictionary results in a false answer.

2) Misspell checking by combined method:
Table 3 shows the F-score of misspell checking with combined method. If we cannot find a word in the vocabulary, we will judge if it looks like a correct English word. If both parts claim the word wrong, we will judge it wrong. Otherwise we claim that correct. The result is shown in Table 3.

Word Length	4	5	6	7
F-score	0.913	0.924	0.923	0.928

Word Length	8	9	10	11
F-score	0.930	0.936	0.930	0.931

Word Length	12	13	14	15
F-score	0.929	0.924	0.924	0.920

Word Length	16	17
F-score	0.918	0.919

Table 3: F-score of different word length based on both dictionary and word segmentation

Compared with Table 2, we don't find obvious improvement on F-score especially when the word is short. The reason is we add bias when set threshold on word correctness judging. It's not so bad if we judge correct word wrong but it's awful if we judge wrong word correct.

Moreover, what word judging part do, is to select minor correct word even it does not occurs in the vocabulary. The amount of words influenced by this part should be small. It's enough if we can approve some correct word that denied by Microsoft Word or Evernote. We do approve "facebook","macbook", "Livermorium", "instagram", "Lampard", "Aquaman" etc, even Microsoft Judge them wrong.

5.2 Result of misspell correcting

1) Correction result without grammar

To create test dataset, we intentionally modified correct word into wrong one. We have tried multiple values when boosting word with edit distance one, and the accuracy of correct wrong words are shown in Table 4.

W(d=1)/W(d=2)	1:0	1:1	1:5	1:10
Accuracy	0.566	0.404	0.548	0.591

W(d=1)/W(d=2)	1:100	1:1000	0:1
Accuracy	0.637	0.646	0.204

Table 4: Correction accuracy between different weights of edit distance 1 and 2

2) The result of correcting with referencing grammar

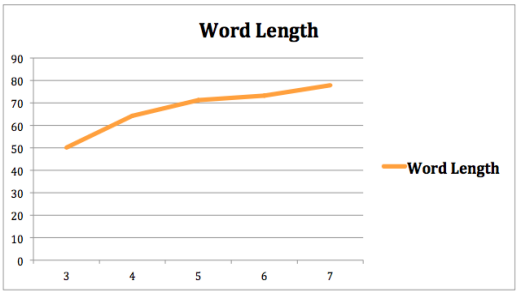

Figure 3: Modification result among different word length

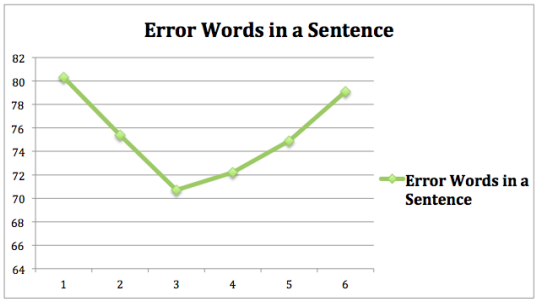

Figure 4: Modification result with grammar reference

Figure 3 shows the longer the word is, the better performance our system has. The reason is longer word results in less randomness when we provide correcting advises using edit Distance.

Figure 4 shows the accuracy decreases when the number of wrong words growing between 1 and 3. However, the accuracy increases when the number of wrong words is more than 3. This weird phenomenon results from the balance of 2 factors, the length of sentence and the number of wrong words. Increase on the number wrong words means the raise of difficulties on grammar analysis, therefore, resulting in the descending of accuracy. However, it also means the raise of the length of sentence, which benefit the grammar analysis. We conclude that those 2 factors reach a balance when the number of wrong words is 3.

Compared the result from Figure 3 and 4 with Table 4, our system that adds grammar analysis obviously has a better accuracy.

The correcting process gets good results on single letter wrong, inserted and two adjacent letters transposed. But not well on single letter omitted. Also correcting short words is hard. The reason is that shorter words are always have high frequency and has great impact on final scores.

6 Conclusion

We develop a system containing 2 relatively new methods. First, we consider word structure when checking the spell mistake. Second, we consider both word frequency and grammar analysis when correcting spell mistake. It turns out our corrector can find more than 90% mistakes and accurately correct more than 70% of them. Our performance is better in long word and long sentence. The goals set before we started project basically are achieved.

7 Future Work

Increase the volume of dictionary to lift the checking recall and f-score. Improve the scoring model by:

1) Take keyboard distance into consideration

2) Give different weight to each type of mistakes

3) Consider words with similar pronunciation

Reference

[1] Li, Yanen, Huizhong Duan, and ChengXiang Zhai. "Cloudspeller: Spelling correction for search queries by using a unified hidden markov model with web-scale resources." Spelling Alteration for Web Search Workshop. 2011.

[2] Birkbeck spelling error corpus http://ota.ox.ac.uk/headers/0643.xml

[3] Mitton, Roger. "Spellchecking by computer." Journal of the Simplified Spelling Society 20.1 (1996): 4-11.

[4] Gutenberg 2003 CD set http://www.gutenberg.org/wiki/Gutenberg:The_CD_and_DVD_Project

[5] Wikipedia Dump http://dumps.wikimedia.org/enwiki/20141208/

[6] Stolcke, Andreas, et al. "SRILM at sixteen: Update and outlook." Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop. 2011.

[7] kenlm: https://kheafield.com/code/kenlm/

[8] Heafield, Kenneth, et al. "Scalable Modified Kneser-Ney Language Model Estimation." ACL (2). 2013.

[9] Spell checking by computer http://www.dcs.bbk.ac.uk/~roger/spellchecking.html