A Toy System for Spell Correction

Abstract Spell correction is applied in many areas. Tasks of spell correction include spell detection and spell correction. As there are two kinds of errors—non-word error and real-word error, different methods are used to detect these errors. But for the correction task, the channel model is focused, in which two probabilities are calculated by language model and error model to get the best candidate word. In this report, different ways to get the transition probability *p(x|w)* have been discussed and their performance have been compared.

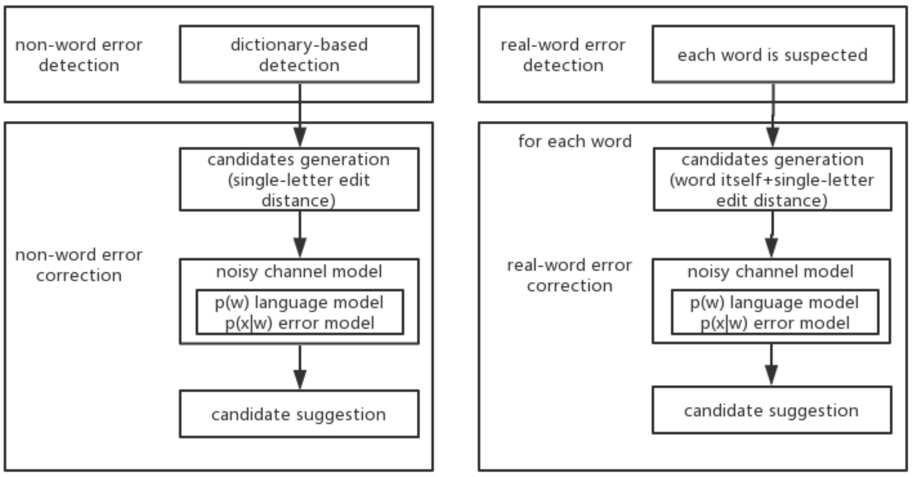
1 Introduction

Spell correction, which is also known as spell checker, has been widely used in word processing software, search engines and many editors. In a spell correction task, non-word error and real-word error need to be handled. While the former one refers to error words that are not included in the dictionary, the latter one is much more confusing, which means the word itself exists, but it is improper in the context. This system aims to detect and corrects errors above.

2 Model

2.1 Basic Model

For each type of error, the correction task can be divided into two steps. Spell errors should be detected at first, and then automatic correction is supposed to be achieved. Based on the features of these two kinds of errors, process of spell correction is below.


non-word error detection
dictionary-based detection
candidates generation (single-letter edit distance)
non-word error correction
noisy channel model
p(w) language model
p(x|w) error model
candidate suggestion
real-word error detection
each word is suspected
for each word
candidates generation (word itself+single-letter edit distance)
real-word error correction
noisy channel model
p(w) language model
p(x|w) error model
candidate suggestion

Pic 1. Experiment Process

i. Noisy Channel Model

The noisy channel model attempts to recover input signals from noisy output signals. In this task, noisy word is converted by original word after crossing the noisy channel. *w* and *x* represent the original and observation word, and then we need to find the word *w* which maximize the probability *p(w|x)*.

$$\hat{w} = argmax_{w\in V} p(w|x)$$
$$= argmax_{w\in V} \frac{p(x|w)p(w)}{p(x)}$$
$$= argmax_{w\in V} p(x|w)p(w) \quad (1)$$

In this case, to find the best original word *w*, probability $p(w)$ and $p(x|w)$ need to be calculated. $p(w)$ can be acquired through the language model, and $p(x|w)$ relies on confusion matrixes or error model.

ii. Language Model

To refer to the contextual information, bigram probability is calculated by a language model with Add-1 smoothing here. As sentences extracted from news articles need to be corrected in our task, corpus of ***reuters*** news in ***nltk*** module of python was used to count tokens. It is noticeable that corpus chosen must be similar to those aimed to be corrected, or the result will be terrible. For example, chat corpus is not proper for news sentences correction, because frequencies of key words of these two areas are quite different. In other words, the probability distribution is totally different.

What's more, to avoid problem that probabilities of some tokens are calculated as zero as the tokens don't exist in the corpus, Add-1 smoothing (Laplace smoothing) is used to steal probability mass to generalize better.

$$p_{Add-1}(w_i|w_{i-1}) = \frac{count(w_{i-1},w_i)+1}{count(w_{i-1})+V} \quad (2)$$

where V refers to the number of types in the corpus.

iii. Error Model

When it comes to the calculation of probability $p(x|w)$, massive misspelled word pairs are needed to get the confusion matrixes, and then transition probability $p(x|w)$ can be calculated. Peter Norvig[1] provides a file(count_1edit.txt) from which we can get the counts of single-edit errors. Then, the transition probability $p(x|w)$ can be estimated as product of the probability of a word *w* being wrong and the probability of that word having a specific kind of single-edit error. In expressions:

$$p(x|w) = p_{error}(w) * p_{edit-1}(x|w) \quad (3)$$

Where
$$p_{edit-1}(x|w) = \begin{cases} \frac{del[w_{i-1},w_i]}{count[w_{i-1},w_i]} & if\ deletion \\ \frac{ins[w_{i-1},w_i]}{count[w_{i-1}]} & if\ insertion \\ \frac{sub[w_i,x_i]}{count[w_i]} & if\ substitution \\ \frac{trans[w_i,w_{i+1}]}{count[w_i,w_{i+1}]} & if\ transposition \end{cases} \quad (4)$$

Here, the probability of no error is set to 0.9 artificially, so that probability a word being wrong is 0.1.

Besides, some scholars have provided other methods to implement the error model. According to Peter Norvig[2], a shortcut can be took when there is no data to build a good spelling error model. He defined a trivial, meaning that words of edit distance 1 are infinitely more probable than known words of edit distance 2. This priority has been reflected on the function of selecting candidates. In that case, calculation of probability can be avoided. Additionally, Mayes, Damerau etal[3]. used a confusion set of a string x to include x, along with all words w in the dictionary such that x can be derived from w by a single application of one of the four edit operations. Then they define the error model, for all w in the confusion set of x, as:

$$p(x|w) = \begin{cases} \alpha & if\ w = x \\ \frac{1-\alpha}{C-1} & otherwise \end{cases} \tag{5}$$

This a simple error model, where α is the prior on a typed word being correct, and the remaining probability mass is distributed evenly among all other words in the confusion set, which refers to candidate list in this report.

2.2 Improved Model

In the basic model, weights of different edits are equal or based on past observation. It is noticeable that Church and Gale[4] provided a more sophisticated model. They attached a probability on each unique edit. Firstly, they assumed all edits are equiprobable. Then, they iteratively run the spell checker over the training corpus to find corrections, and corrections were used to update the edit probabilities. Here, having the aid of their idea, I try to iteratively run this correction system over the target corpus and update the edit probabilities to achieve the goal. A global dictionary with keys like “e|a” was constructed to record the frequency of certain edits and was used to update the transition probability. Yet this method also has disadvantages. If the early handled words are not corrected accurately, the later words may face a poor correction system.

3 Experiments

In the experiment, a text file consisting of 1,000 sentences extracted from news articles needs to be corrected.*vocab.txt* works as the dictionary for non-word detection. The thorn lies in the calculation of the transition probability *p(x|w)*. Here, 4 methods have been used, with the differences focus on this probability in the codes. The results of different methods are below:

Table 1. Results of spell correction

Method of calculating *p(x|w)*	accuracy	Time consumption
list of counts of single-edit errors	75.4%	13539s
Peter Norvig’s shortcut	72.8%	233s
Equally weighing probability	66.9%	12794s
Iteratively updated probabilities	73.3%	14043s

Due to the calculation of all words with single edit distance for each word in the sentences, most methods are terribly time consuming. Results above show that methods of list of counts of single-edit errors perform best, which may be result of its dependence on the prior frequency of edit error

When it comes to accuracy, I only take words of edit distance 1 into consideration, in other words, some edit changes have not been included. That’s why accuracy is relatively low. Many wrong words like “aapJn”, “erpoxt” have not been detected due to the neglect of words of edit distance 2. Calculation of these candidates will consume much more time and the system becomes meaningless. Therefore, other external knowledge needs to be added to choose the candidates better and faster.

4 Conclusion

In this report, a simple spell correction system was implemented. The traditional noisy channel model was constructed. And the key points of this model are the probabilities *p(w) and p(x|w)*. For *p(w)*, a language model under bigram condition was constructed with add-one smoothing. Yet the calculation of *p(x|w)* remains a knotty problem. All the methods are time consuming if we want to take words of edit distance 2 into consideration to improve the performance.

In the future, more work can be done to improve the performance of this spell correction system. On the one hand, more edits can be taken into consideration, for example, the transition from "ent" to "ant", as Brill etal[5]mentioned in their paper. On the other hand, the insertion and deletion probability can be estimated from massive corpus, so that calculation of words of edit distance 2 can be greatly reduced.

Some ideas

After constructing this correction system, I wonder how to allow more edits possible in our system. Now, the major calculation lies in the candidate generation of words and the calculation of their transition probability. How can we take words of more edits distance into consideration and recommend smaller amount of but more accurate words to be the candidates? Maybe we can vectorize each word (into 26 dimensions, with each dimension representing the amount of that character in the word), and calculate cosine similarity between two words. In that case, word like "Taiwan" can be recommended as the candidate of "Taawin", although the edit distance is 2 or even more. Yet the idea remains to be tested.

In addition, setup for probability of no error is worth discussing. If probability of no error is relatively high, the system tends to not change the original word in the real-word correction part. On the contrary, if the probability of no error is not that high, the system may select a word or token of high frequency to replace the original word which may be originally true.

References

[1] http://norvig.com/ngrams/count_1edit.txt

[2] http://norvig.com/spell-correct.html

[3] Mays, E. , Damerau, F. J. , & Mercer, R. L. . (1990). Context based spelling correction. Information Processing & Management, 27(5), 517-522.

[4] Church, K. W. , & Gale, W. A. . (1991). Probability scoring for spelling correction. Statistics and Computing, 1(2), 93-103.

[5] Brill, E. . (2000). An Improved Error Model for Noisy Channel Spelling Correction. Proc. 38th Annual Meeting of the Association for Computarional Linguistics (ACL 2000), Tokyo, Japan.