A SPELLING CORRECTION BASED ON PROBABILITIES

Sylvie Thouësny

Department of Computational Linguistics
Dublin City University, Dublin
sylvie.thouesny2@mail.dcu.ie

ABSTRACT

"A University of Pittsburgh study states that using the spell-check button can create problems for those who rely too heavily on the software" [1].

I strongly agree with this statement. Spelling checker should just be a flag indicating that some errors could occur at some stage, but certainly not a systematic way to correct text. Even if the purpose of Computational Linguistics is to be very close of human minds, we still have to think by ourselves.

This paper is an approach to present how probabilities can be used in order to generate suggestions for misspelt words. SpellChecker, as its name indicates, is a spell checker. It is written in Java and is easy to use. It looks at single error and generates a list of candidates for non-word errors.

Based on existing algorithms, SpellChecker detects deletion, substitution, reversal or insertion errors [2] according to the dictionary I use. Candidates are computed according to probabilities and sorted by the most likely. The user has the possibility to replace the misspelt word suggested or simply ignore it.

1. INTRODUCTION

An efficient spell checker will automatically detect non-word and real-word errors, using context to propose a word, the word the user had in mind.

In this essay I will focus on detection and correction of non-word errors and look only at words in isolation using a dictionary to help detect and correct spelling errors. Probabilities will help estimate the correct word depending on the level of noise. Correct word means the word the user had in mind but was mistyped for many reasons. There are many sources of noise, such as fat fingers that press two keys at the same time.

In this topic I will look at single error misspellings and propose a list of candidates sorted by the most likely. First of all I will present SpellChecker, the program I wrote for this project and explain briefly its main functions.

Then I will focus more precisely on the methods I used to determine and propose to the user the most likely suggestion and not only a straight list of words. Finally I will compare my program with three other spell checkers and evaluate the rate of accuracy.

2. PRESENTATION OF SPELLCHECKER

2.1. How to use SpellChecker

SpellChecker is a program that checks either a single word in the field word or a text entered in the text area. Depending on what you want to check, you click on the *"Check Word"* or *"Check Text"* button.

If a misspelling is detected in the field word and if suggestions are available, you can select the one you want in the drop down list and replace your input by the correction or you can ignore it if the word is correct but not in the dictionary.

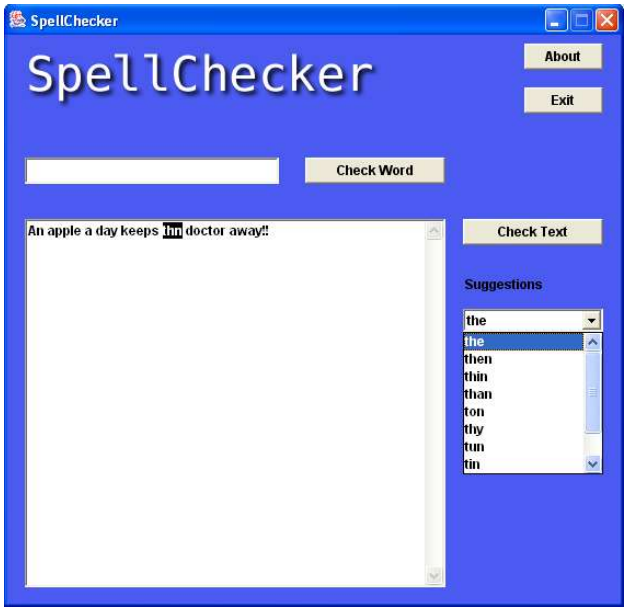

Figure 1: Main Window of SpellChecker.

If SpellChecker detects an error in the text area, the misspelt word will be highlighted. You can then also replace or ignore the suggestions proposed in the drop down list. To continue the checking, you must click on the *"Check Text"* button again until SpellChecker reaches the end of your text and displays the dialog box *"The spelling check is complete"*.

2.2. Proposing Candidates Corrections

SpellChecker takes the input string and split it into words, it ignores spaces and punctuations if the input is from the text area. Spaces and punctuations are not allowed in the field word. If any, a *"no spaces or punctuations"* comment will be displayed.

The program checks each words to see whether or not this unit is part of the dictionary. If it is, SpellChecker ignores it and goes to the next one. If it is not, SpellChecker will find a list of words that differs from the input by a single deletion, insertion, reversal or substitution error and cross-check with the dictionary.

For example, given the input *"ble"*, SpellChecker suggests a list of words as shown in Table 1:

Typo	Correction	Methods
ble	bee	substitution
ble	blue	deletion
ble	bel	reversal
ble	be	insertion

Table 1: List of suggested words

According to the dictionary, if no suggestions is found, SpellChecker returns a *"no suggestions"* comment for that entry.

The list of candidates is sorted by probability in order to give the user the most likely candidate and not only a simple list of suggestions where the first proposition could be a word rarely used.

3. DETERMINING THE MOST LIKELY

3.1. Training Corpus

To determine the most likely candidate based on probabilities, I used a training corpus. This corpus is made of extracts from three different texts. The type of texts is really important and can radically change the probability of a word. I was looking for a large range of different tokens to build the corpus.

The first one is the Seti@home Frequently Asked Questions [3], the second one is the Novel Carrie By Stephen King [4] and the last one is The Hitch Hikers Guide to Galaxy by Douglas Adams [5]. This corpus contains 4208 words.

Firstly it is used to determine how many times each word occurs in the corpus. To compute the probability of this word, I used the smoothed method in order to avoid zero probability [6]:

$$P(w)=\frac{\text{c(w)} + 0.5}{\text{N} + 0.5\text{V}}$$

Where *c* is the count of words, *w* the word, *N* the number of tokens and *V* the vocabulary's size.

It is also used to count the co-occurrences of the letters. I will need this value to calculate *P(t/w)* later on.

3.2. Matrices for Probabilities

The probabilities are calculated from four matrices [2]:

- Deletion or del[x,y], the number of times that the characters xy were entered as x.
- Insertion or ins[x,y], the number of times that x was entered as xy.
- Substitution or sub[x,y], the number of times that y was entered as x.
- Reversal or rev[x,y], the number of times that xy were entered as yx.

The training data I used to count the number of times of these methods above comes from the report of Kernighan, Church and Gale [7].

Probabilities *P(t/w)* are estimated from these 4 matrices divided by the number of times that xy appear in the training corpus. To avoid zero probability, probabilities are computed by using an add-one smoothed method:

$$P(t/w)=\frac{\text{del[wp-1, wp]+1}}{\text{c[wp-1, wp]+V}}$$

$$P(t/w)=\frac{\text{rev[wp, wp+1]+1}}{\text{c[wp, wp+1]+V}}$$

$$P(t/w)=\frac{\text{sub[tp, wp]+1}}{\text{c[wp, wp+1]+V}}$$

$$P(t/w)=\frac{\text{ins[wp-1, tp]+1}}{\text{c[wp-1, wp+1]+V}}$$

Where *c* is the count of words, *V* is the size of the alphabet, *t* a typo, *w* a word, *wp* the pth character in the word *w* and *tp* the pth character in the typo *t*.

I have slightly modified the insertion and substitution formulas. In the substitution method, I counted the number of occurences between the estimated substituted character and the next character. In the insertion method, I counted the number of occurences of letters before and after the presumed inserted character. I got better results with these modifications.

After determining all the elements above, it is easy to find the most likely corrections[6]:

$$\hat{w}=argmax\, P(t/w)\,P(w)$$

SpellChecker proposes a list with estimated words for the typo *"thn"* sorted by the most likely, see table 2 for an extract.

Corrections	Probabilities
the	0.004513058394336933
then	5.558864403414731E-5
thin	4.557095853954192E-5
than	3.6731704741374366E-5
tin	1.170823088631308E-5
ton	5.854115443156539E-6
tan	4.1416871162468025E-6

Table 2: Probabilities of candidates for *"thn"*

4. COMPARISON AND EVALUATION

4.1. Comparison

Every program that checks spelling may use different algorithms to find the most appropriate candidate. It is really interesting to compare what I have done with other programs such as Microsoft Word[8], OpenOffice[9] or SpellCheck.net [10].

SpellCheck.net inspired me for the final presentation of SpellChecker, however it seems to behave in a very curious manner. Suggestions from that program seem to be far from what I was thinking.

I do not want to achieve the same results as Microsoft Word or OpenOffice, I think their results are correct enough but could be more selective.

I compare the typo *"thn"*, see table 3, with the three of them. It is simply a way to look at what is the most likely for all of these programs.

SpellChecker	Microsoft Word	Open Office	spellcheck.net
the	then	then	than
then	than	thin	then
thin	thin	than	TN
than	ten	tn	thank

Table 3: Comparison

It is also important to say that all the probabilities depend on the choice of the training corpus and on the dictionary's quality and size.

4.2. Evaluation

I look at words in isolation, context is not taken into account yet. SpellChecker has been evaluated with 50 different tokens, see appendix 1 for the complete list. The first column contains the words I had in mind, then a different column for each of the algorithms and the position in which the correct word was in the list of candidates.

The statistics, in table 4, shows that the most accurate results are obtained with the insertion method. The average position of the correct candidates is 1.06, close to the first position in the drop down list. The variance equals 0.1 which indicates that the position is practically always the first one.

SpellChecker has a rate of accuracy of 80%, calculated from the number of first positions divided by the number of increasing the size of the dictionary and taking the context into account.

	Del.	Ins.	Sub.	Rev.
Average	1.74	1.06	1.4	1.29
Maximum	6	3	8	4
Mode	1	1	1	1
Variance	1.71	0.1	1.35	0.33

Table 4: Evaluation

Where Maximum returns the maximum value and Mode returns the most common value.

5. CONCLUSION

The SpellChecker program detects non-word errors according to the dictionary, the more entries we have in the dictionary, the better.

Say we have as entry the word *"chocolate"* to check. The spelling of this word is correct. However if this unit does not appear in the dictionary, SpellChecker will spend some time searching for suggestions according to deletion, substitution, insertion or reversal error detection. It is a waste of time and resources for correcting a non-word error, which is in fact a correct word simply not in the dictionary.

SpellChecker is right now restricted to non-word errors, it looks at single error misspellings and approaches the word as word in isolation. A way of improving this program will be to take into account context to compute probabilities. It could be the purpose of a next update.

6. REFERENCES

[1] Step Away From the Spell-Checker. (http://www.wired.com/news/print/0,1294,58058,00.html). 2003.

[2] Daniel Jurafsky, James H. Martin. *Speech and Language Processing*. International Edition. 2000.

[3] Maintainer: Mark Taylor. *Seti@home Frequently Asked Questions*. From the World Wide Web: http://www.faqs.org/faqs/seti/at-home/questions/ . 2003.

[4] Stephen King. *Carrie.* Pocket Books. 1974.

[5] Douglas Adams. *The Hitch Hikers Guide to Galaxy*. Ballantine Books. 1979.

[6] John McKenna. *Lecture's notes*. 2004.

[7] Mark D. Kernighan, Kenneth W. Church, William A. Gale. *A Spelling Correction Program Based on a Noisy Channel Model*. pp. 205-210. 1990.

[8] Microsoft. *Word*. 2002.

[9] Sun Microsystems Inc. *OpenOffice.org*. 1.1.0.

[10] Spellcheck.net (http://www.spellcheck.net)