Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 10 (17): 1963-1967, 2010
ISSN 1812-5654
© 2010 Asian Network for Scientific Information

Using Modified Fixed Point Numbers Representations in Spelling Checker

[1]R.F. Alwan and [2]O. Al-Hadithi
[1]Faculty of Information Technology, Philadelphia University,
P.O. Box 1, Amman-19392, Jordan
[2]Delmond College, Bahrain

Abstract: This study is concerned with the process of designing and implementing an English text spelling checker which can detect and correct the misspelled words. The spelling checker is built using a modified version of Fixed Point Numbers Representations Technique (FPNRT), which is a compression method that is suitable to this kind of applications. The modified FPNRT will transform the dictionary words into numeric values and then stores them in a data base file of that form, which is considered to be a new way for storing a dictionary. This method gives comparatively good compression ratio and very quick in compression/decompression process. Therefore, this method is good for files which are frequently read and updated. This approach will minimize the memory space required to store the dictionary by achieving a compression ratio of 37% on average and also enhance the performance of the spelling checker by maximizing its speed which takes O(lg n/5). This approach is also provide a new way for building a list of candidate words that may be used to correct the misspelled words.

Key words: Spelling checker, spelling corrector, data compression, spelling errors, dictionary based, statistical-analysis methods

INTRODUCTION

Several software systems (such as: Professional Writer, Windows, Word Perfect, etc.) have programs that can handle an English text for potential spelling errors and can often point out a probable correct spelling. There are two types of spelling programs: spelling checkers and spelling correctors. The function of a spelling checker is to identify those words which are incorrectly spelled. While a spelling corrector can do both, detect the misspelled words and try to determine the most likely correctly spelled word which was meant (Angell *et al*., 1983; James and Daniel, 1990; Ringlstetter *et al*., 2007). In practice, many methods are used to detect spelling errors. Some of these methods depend on a statistical analysis of word occurrences to reject word forms that are statistically improbable, while others are depended on pre-constructed dictionaries which are known as Dictionary-Based Spelling Checker (DBSC) methods that containing either full words or word fragments to verify the spelling.

Most spelling checkers methods are used dictionaries rather than statistical methods (Mihov and Schulz, 2004; Gollapudi and Panigrahy, 2006; Salomon, 2007; Reynaert, 2008; Cole and Lewenstein, 2004). These Spelling Checkers methods are as follows:

- Statistical-Analysis method
- Simple method of DBSC
- Words frequency method
- DBSC method using tree technique
- DBSC method using hashing table technique

Procedures that have been developed so far will both detect and correct misspelled text words. All these procedures consist of using dictionaries of pre-stored word forms. The correction methods are as follows (Sitbon *et al*., 2007, 2008a, b; Reffle *et al*., 2008; Wilcox-O'Hearn *et al*., 2008):

- Dictionary common misspelling method
- Simple correction method (word variants method)
- Phonetic (soundex) method
- N-gram encoding method
- Damerau-Levenshtein (DL) metric method
- Fourth-Generation Language (4 GL) method

SPELLING CHECKER SYSTEM

The spelling checker system has three components:

- Data base (dictionary)
- Spelling checker routine
- Spelling corrector routine

Corresponding Author: Dr. Raad F. Alwan, Faculty of Information Technology, Philadelphia University, P.O. Box 1, Amman-19392, Jordan Tel: +962777426528

To illustrate the design and the implementation of the spelling checker program, we have to describe each one of the above components.

Data base (dictionary): The dictionary or the Data Base (DB) is the most important component of the spelling checker program. This is due to the structure of the dictionary which determines the type of the look-up strategy (search method) that is used in the spelling checker routine.

In the dictionary implementation, we transform each word into a numeric value before storing it in the dictionary. The transformation of the system dictionary words is done by using the modified FPNRT. Hence, the numeric value of the word (f) will be calculated using the following formula:

$$f = \sum_{i=0}^{n-1} (ASC - 65) * 26^i \qquad (1)$$

where, n represents a word length, ASC is the ASCII code of any letter, i is the letter's position in the word.

This transformation is considered as an encoding operation. The algorithm for encoding the dictionary is shown in Algorithm 1.

The encoding formula shows that each word has a unique value (f) in the stored dictionary. By subtracting the constant number (65) from the ASCII value of each letter, we can reduce the value of (f) in order to minimize the memory space needed to store the value (f).

```
Algorithm encodingDi ctionary (dictionary )
   input :dictionay represents the word to be encoded
   output :convert and distribute words in subfiles
   whil e(not eof(dictio nary))
   {  $f$ = convert(wo rd);
      if ($f \ge 0$ and $f \le 255$) then
           put f in DB1 in ascending order
      else if ($f \ge 256$ and $f \le 65515$) then
           put f in DB2 in ascending order
      else if ($f \ge 65516$ and $f \le 2147483647$ )then
           put f in DB3 inascending order
      elseif ($f \ge 2147483649$ and $f \le 9223372036\ 854775800$) then
           put f in DB4 in ascending order
      else
           put f in DB5 in ascending order
    }
end.
Algorithm Convert(s)
   input :s string represents the word to be encoded
   output : f numeric value of the word using FPNRT
   $f = 0$;
   for i = 0 tolength(s);
        $f = f + \left( ASC(s[i] - 65) * 26^i \right)$;
  return f;
end.
```

Algorithm 1: Encoding dictionary algorithm

The following example illustrates the transformation process of a word COMPUTER into a numeric value using the modified FPNRT algorithm shown in Eq. 1:

n = 8, ASCII of C = 67, 0 = 79, M = 77, P = 80, U = 85, T = 84, E = 69, R = 82

$$f = \begin{bmatrix} (67-65)*26^0 + (79-65)*26^1 + (77-65)*26^2 + \\ (80-65)*26^3 + (85-65)*26^4 + (84-65)*26^5 + \\ (69-65)*26^6 + (82-65)*26^7 \end{bmatrix}$$
$$= 138{,}011{,}593{,}878$$

The advantages of using this technique (which stores the words in DB as a numeric value) are as follows: Reducing the memory space that is needed to store the dictionary. Suppose we want to store the word COMPUTER in the memory in the normal form. The word needs 8 bytes of memory space to store it, while using the modified FPNRT will minimize the number of bytes needed to store the word and therefore the memory space required would be less. The space needed can be calculated as follows:

We calculate the value f of the word using Eq. 1:

f(COMPUTER) = 138, 011, 593, 878.

To calculate the number of bits (n) required to store f, we use the following equation:

Since: $f = 2^n$
Therefore:

$$n = \frac{\lg(f)}{\lg(2)} = 3.32 * \lg(f)$$
$$= 3.32 * \lg(138{,}011{,}593{,}878)$$
$$\cong 37 \text{ bits}$$

So, the number of bytes (N) required to store the word COMPUTER using the modified FPNRT is:

$$N = \frac{n}{8} = \frac{37}{8} \cong 5 \text{bytes}$$

We can gain a high speed in text spelling checking, due to the use of the binary search method as dictionary look-up strategy.

We can gain a high speed in text spelling correcting thorough minimize the construction time to build the list of words that are close in spelling to the misspelled word.

In order to build the dictionary for the system, we transform the words of the dictionary to its equivalent numeric values using the modified FPNRT, then we divide it into five parts as follows:

DB1: This file stores words whose numeric values are: $0 \le f \le 255$, such as: f(BE) = 105

DB2: This file stores words whose numeric values are: $256 \le f \le 65{,}515$ such as: f(ARE) = 3,146

DB3: This file stores words whose numeric values are: $65{,}515 \le f \le 2{,}147{,}483{,}647$ such as: f(SPELL) = 5,223,184

DB4: This file stores words whose numeric values are: $2{,}147{,}483{,}647 \le f \le 9{,}223{,}372{,}036{,}854{,}775{,}800$ suchas: f(COMPUTER) = 138,011,593,878.

DB5: This file stores the words whose numeric values are: $f > 9{,}223{,}372{,}036{,}854{,}7775{,}800$

Because there is no data type that can store a numeric value of a word whose length is more than 13 letters, we divide the word into two parts. The first part is the length of 13 letters and the second part is the length of the rest letters of the words. So, the first field is used to store the numeric value of the first part of the word and the second field is used to store the numeric value of the second part of the words such as the word CINEMATOGRAPHICAL:

F(CINEMATOGRAPH) = 723,151,536,781,827,142;
F(ICAL) = 193396

The reason for dividing the dictionary in five files is to ease the search during the spell checking process.

Spelling checker routine: This is the second component of our system which detects the misspelled words in a text file.

The algorithm of this component is shown in Algorithm 2, where a decoded text file is read word by word. Then each word (numeric value) of this text file will be looked-up in right part of the dictionary. If the word is not in that part, then the spelling checker routine will be called.

The process of detecting misspelled words is done in a short time using the binary search technique. Therefore, the time complexity of the algorithm of the look-up strategy is O(lg n/5).

Spelling corrector routine: In spelling corrector routine, we need to transform back a word(s) from its numeric form (as it is stored in the dictionary) into its normal form (letters form). The algorithm of Algorithm 3 shows the process of decoding the word(s), where the function CHR() transforms the numeric value into its corresponding character form.

```
AlgorithmSpellingChecker(textfilename)
    Input: text filename
    Output: either correctentryor listof candidates
    whilenoteof(textflename)
    {
        Boolean p;
        Strings;
        readaword fromtext fileputins;
        f =convert(s);
        if (f ≥ 0  and  f ≤ 255)the  {
            D = 1
            p = BinarySearch(1,length(DB1), f)
        }
        elseif(f ≥ 256 and  f ≤ 65515) then {
            D = 2
            p = BinarySearch(1,length(DB2),f)
        }
        elseif(f ≥ 65516and f ≤ 2147483647)then {
            D = 3
            p = BinarySearch(1,length(DB3),f)
        }
        else
            if (f ≥ 2147483648and f ≤ 9223372036854775800)then {
                D = 4
                p = BinarySearch(1,length(DB4),f)
            }
            else {
                D = 5
                p = BinarySearch(1,length(DB5),f)}
                if(!= p)then
                    callspellingCorrector(f,D);
            }
    }
end.
```

Algorithm 2: Spelling checker algorithm

```
Algorithm decoding (n)
  Input: c: array represents candidate list
         N: array length
  Output: print the list of candidate words
  String x = ;
  for (i = 1 to n)
      while (c [I]! = 0) do  {
          $s = \left[\frac{c\,[i]}{26}\right]$
          w = (c[i]-s*26)
          x = x + CHR (w + 65)
          c [I] = s
      }
      print x
end.
```

Algorithm 3: Decoding algorithm

```
Algorithm SpellingCorrector(f,d)
        Input: f : value of the misspelled word
                 d : database subfile
        Output: c : array containslist of candidate words
        i= 1;
        while(i ≤ length(d))do  {
            $x = |f - d[i]|$;
            if (x > 26)then  {
                if (xis integer  and  (x mod 2) = 0)  then
                        $x = x/26$;
            }
            else
                if (x is integer) then  {
                    c[i] = f;
                    size = size + 1;
                }
        }
        call decoding(c,size);
end.
```

Algorithm 4: Spelling corrector algorithm

The spelling corrector is a subprogram that corrects the misspelled words in a text file. The algorithm of this subprogram is shown in Algorithm 4. When the spelling checker routine detects a misspelled word, the spelling checker program will call the spelling corrector routine. The spelling corrector routine, then will look-up in the dictionary to construct a list of the nearest neighbor(s) of the incorrectly spelled word, as candidates for the correct spelling words.

As a first step in the operation of selecting a word to be included in the candidate list to correct the misspelled word, we subtract the correct word from the misspelled word and then take the absolute value of the subtraction result. Then we start dividing the result by the constant (26) until the result becomes less than 26.

If at any stage of our division the result became an odd number, then we have to stop the process and we reject this word. Now, if the result is an integer number less than 26 then the word will be selected to be a candidate in the list of the words to correct the misspelled word. On the other hand, if the result is less than 26 and it is a real number, then the word will be rejected.

The following example illustrates how the words are selected to construct a list of candidates for the misspelled word. We will take the misspelled word SPULL and the correct words are SPILL and SPILE.

|f(SPULL)-f(SPILL)| = |5234000- 5225888| = 8112

Since, 8112>26 and 8112 is integer and even number, Therefore we divide it by 26:

$$\left(\frac{8112}{26}\right) = 312$$

Since, 312>26 and 312 is integer and even number, Therefor:

$$\left(\frac{312}{26}\right) = 12$$

Since: 12<26 and 12 it is an integer number,

Therefore, SPILL is selected to be included in the list of candidate words to correct the misspelled word SPULL.

|f(SPULL)-f(SPILE)|=|5234000-2027056| = 3206944

Since, 3206944>26 and 3206944 is an integer and even number, therefor:

$$\left(\frac{3206944}{26}\right) = 123344$$

Since, 123344> 26 and 123344 is an integer and even number, therefore:

$$\left(\frac{123344}{26}\right) = 4744$$

Since, 4744>26 and 4744 is an integer and even number, Therefore:

$$\left(\frac{4744}{26}\right) = 182.46$$

Since, 182.46>26, but 182.46 is not an integer number, therefore: SPILE is not selected for the list of candidate words to correct the misspelled word SPULL.

CONCLUSIONS

From this study the following conclusions are drawn:

- In spelling checker, the dictionary is considered as the most important part of the system. The structure and the size of the dictionary will determine the performance of the spelling checker
- We found that the modified FPNRT can enhance the system and makes it ready for use with other languages after doing simple modifications on the formula of the above method

- Using the modified FPNRT to store the dictionary words in numeric form has made the system more efficient than the others because it reduces the space needed for storing the dictionary and maximizes the accessing speed

REFERENCES

Angell, R.C., G.E. Freund and P. Willett, 1983. Automatic spelling correction using a trigram similarity measure. Inform. Process. Manage., 19: 255-261.

Cole, R.L. and G.M. Lewenstein, 2004. Dictionary matching and indexing with errors and dont cares. Proceedings of the 36th Annual ACM Symposium on Theory of Computing, (AASTC'04), Chicago, IL, USA., pp: 91-100.

Gollapudi, S. and R. Panigrahy, 2006. A Dictionary for Approximate String Search and Longest Prefix Search. CIKM, Arlington, Virginia, USA., pp: 5-11.

James, K.M. and J.M. Daniel, 1990. A tale of three spelling checkers. Software-Practice Experience, 20: 625-630.

Mihov, S. and U. Schulz, 2004. Fast approximate search in large dictionaries. Comput. Linguis, 30: 451-477.

Reffle, U., A. Gotscharek and C. Ringlstetter, 2008. Successfully detecting and correcting false friends using channel profiles. AND, 303: 17-33.

Reynaert, M., 2008. All and only, the errors: More complete and consistent spelling and ocr-error correction evaluation. Proceedings of the International Conference on Language Resources and Evaluation, (CILREC'08), Tilburg University, pp: 1867-1872.

Ringlstetter, C., K. Schulz and Y. Mihov, 2007. Adaptive text correction with web-crawled domain-dependent dictionaries. ACM Trans. Speech Language Process., 4: 17-17.

Salomon, D., 2007. Data Compression the Complete Reference. 4th Edn., Springer-Verlag, Berlin.

Sitbon, L., P. Bellot and P. Blache, 2007. Phonetic based sentence level rewriting of questions typed by dyslexic spellers in an information retrieval context. Proceedings of the Interspeech, Antwerp, (IA'07), Belgium, pp:1-4.

Sitbon, L., P. Bellot and P. Blache, 2008a. A corpus of real-life questions for evaluating robustness of qa systems. Proceedings of the 6th Edition of the Language Resources and Evaluation Conference, (LREC'08), Marrakech, Morocco, pp: 1-4.

Sitbon, L., C. Meinajaries and P. Bellot, 2008b. How to cope with questions typed by dyslexic users. ACM, Singapore, 303: 1-8.

Wilcox-O'Hearn, L., G. Hirst and A. Budanitsky, 2008. Real-word spelling correction with trigrams: A reconsideration of the mays, damerau and mercer model. Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics, Feb. 17-23, Haifa, Israel, pp: 605-616.