SPELL CHECKER

Vibhakti V. Bhaire, Ashiki A. Jadhav, Pradnya A. Pashte, Mr. Magdum P.G

ComputerEngineering, Rajendra Mane College of Engineering and Technology

Abstract- Spell Checker project adds spell checking and correction functionality to the windows based application by using autosuggestion technique. It helps the user to reduce typing work, by identifying any spelling errors and making it easier to repeat searches .The main goal of the spell checker is to provide unified treatment of various spell correction. Firstly the spell checking and correcting problem will be formally describe in order to provide a better understanding of these task. Spell checker and corrector is either stand-alone application capable of processing a string of words or a text or as an embedded tool which is part of a larger application such as a word processor. Various search and replace algorithms are adopted to fit into the domain of spell checker. Spell checking identifies the words that are valid in the language as well as misspelled words in the language. Spell checking suggests one or more alternative words as the correct spelling when a misspelled word is identifies.

Keyword*-Net Bean, Edit Distance.*

I. INTRODUCTION

Many search engines, web browsers and web applications provide Spell Checking as an useful feature. It is usually provided as a dropdown menu of choices below the textbox in which the user is typing. The goal is to anticipate the text the user intends to type so that instead of typing in the rest of the word, one of the alternatives can be selected. Thus we develop such application by using data structure like tries for storing dictionary and edit distance algorithm for spelling correction which provides advantage of autosuggest, so that queries need not be typed in their entirety, queries need not be remembered verbatim and wrongly typed queries can be overridden without recomposing them.

This project aims to develop this autosuggest feature for windows based application. For example, most popular search engines like Google which run on machine clusters and use lists of popular queries from their logs to provide related suggestions to users. This project aims to implement these intensive functionalities in the windows based application

II. BASIC PROCESS OF SPELL CHECKER

Spell checker scans the text and selects the words contained in it after that compares each word with a known list of correctly spelled words (i.e. a dictionary). This might contain a list of words, or it might also contain an additional information, such as word division points or lexical and grammatical attributes.

For handling morphology language-dependent algorithm is an additional step. The spell-checker will need to consider different forms of the same word, such as verbal forms, contractions, possessives, and plurals, even for a lightly inflected language like English. For many other languages, such as those featuring agglutination and more complex declension and conjugation, this part of the process is more complicated.

A spell checker carries out the following processes:

- It scans the text and selects the words contained in it.
- It then compares each word with a known list of correctly spelled words (i.e. a dictionary). This spell checker might contain just a list of words, or it might contain additional information, such as word division points or lexical and grammatical attributes.
- A language-dependent algorithm is an additional step for handling morphology. Even for a English language, the spell-checker will need to consider different forms of the same word, such as plurals, contractions, verbal forms, and apprehensive. For many other languages, such as those featuring conjugation and more complex declension and integration, this part of the process is more complicated.

Allowing for many different forms of a word depending on its grammatical role— it is unclear whether morphological analysis—provides a significant benefit for English, though its benefits for highly synthetic languages such as German, Hungarian or Turkish are clear. The program's user interface will allow users to approve or reject replacements and modify the program's operation,as an adjunct to these components.An alternative type of spell checker uses solely statistical information, such as n-grams. This approach usually information may require a lot more runtime storageandrequires a lot of effort to obtain sufficient statistical. This method is not currently in general use.In some cases spell checkers use a fixed list of misspellings and suggestions for those misspellings; this less flexible approach is often used in paper-based correction methods.

III. FLOWCHART FOR GIVEN PROCESS

Now it is the time to coherent the research work with ideas gathered in above steps by adopting any of below suitable approaches:

A. flowchart

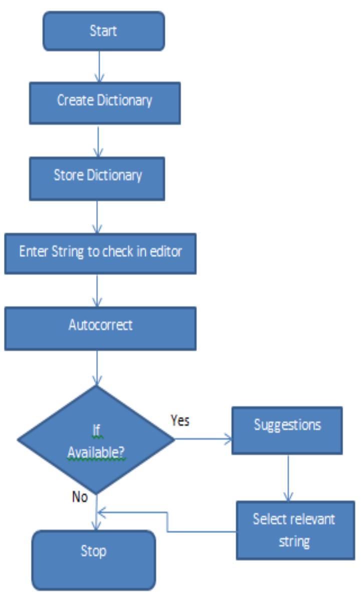
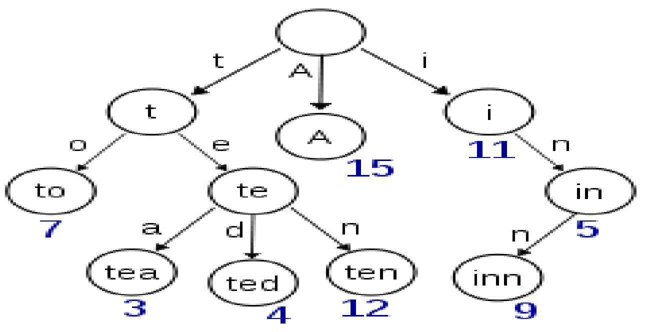


Fig 2.Flowchart for derived method

B. Data Structure (TRIE)



A trie for keys "A", "to", "tea", "ted", "ten", "i", "in", and "inn".
Fig.3 Trie

In computer science a trie, also called digital tree and sometimes radix tree or prefix tree (as they can be searched by prefixes).Trie is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings. There is no node in the tree stores the key associated with that node; instead of its position in the tree defines the key with which it is associated, unlike a binary search tree. The root is associated with the empty stringand all the descendants of a node have a common prefix of the string associated with that node. Only with leaves and some inner nodes that correspond to keys of interest, values are normally not associated with every node.

In the example shown, keys are listed in the nodes and regarding values below them. Each complete English word has an optional integer value related with it. A trie can be seen as a fatalistic fixed automation without iterations. Each finite language is generated by a trie automation, and each trie can be abstracted into a DAFSA.It is not necessary for keys to be explicitly stored in nodes.Though tries are most commonly keyed by character strings.

IV. EDIT DISTANCES ALGORITHM

The algorithm is based on dynamic programming. It allows insertions, deletions and replacements.

1. Insertion: ∂ (€;p), i.e. inserting the letter p.
2. Deletion: ∂ (p; €), i.e. deleting the letter p.
3. Replacement or Substitution: ∂ (p;r) for p ≠r, i.e. replacing p by r.
4. Transposition: ∂ (pr;rp) for p≠r, i.e. swap the adjacent letters p and r.

Definition

Mathematically, the Levenshtein distance between two strings p,ris given by $\mathrm{lev}_{p,r}(|p|,|r|)$where

$$\mathrm{lev}_{p,r}(i,j) = \begin{cases} \max(i,j) & \text{if minimum} \\ \min \begin{cases} \mathrm{levp}_{,r}(i-1,j)+1 \\ \mathrm{lev}_{p,r}(i,j-1)+1 \\ \mathrm{lev}_{p,r}(i-1,j-1)+1_{(pi!=rj)} \end{cases} & \text{otherwise} \end{cases}$$

Where $^{1}(p_{i!=}r_{j})$is the indicator function equal to 0 when $p_{i=}r_{j}$and equal to 1 otherwise.

Example- The Levenshtein distance between the words "kitten" and "sitting" is 3. The minimum edit script that transforms the former into the latter is:

1. **k**itten → **s**itten (substitution of "s" for "k")
2. sitt**e**n → sitt**i**n (substitution of "i" for "e")
3. sittin → sittin**g** (insertion of "g" at the end).

LCS distance (insertions and deletions only) gives a different distance and minimal edit script:

1. delete **k** at 0
2. insert **s** at 0
3. delete **e** at 4
4. insert **i** at 4
5. insert **g** at 6

For a total cost/distance of5 operations.

```
EDITDISTANCE(s1,s2)
1 int m [i,j]=0
2 for i=1to [s1]
3 do m [i,0]=i
4 for j=1 to [s2]
5 do m[0,j]=j
6 for i=1 to [s1]
7 do for j=1 to [s2]
8           do m[i,j]=min{m[i-1,j-1]+if(s1[i]=s2[j])then 0 else 1
9                         m[i-1,j]+1,
10                        m[i,j-1]+1}
11 return m[[s1],[s2]]
```

Figure 1.Dynamic programming algorithm for computing the edit distance between strings s1 and s2

IV.Advantages and disadvantages

Advantages

- Easy Identification.
- Identifies most of the spelling and typing errors.
- Identifies where a space between two consecutive words may have been omitted during typing by suggesting that the word is a spelling mistake.
- Offers a list of spellings as suggestions, one of which may be the correct one.
- Correct Accuracy.
- Save more time.

Disadvantages

- Does not differentiate homonyms (e.g., by – buy, their – there – there, too – to – two)so does not identify incorrect spelling of homonyms.
- May identify proper nouns as spelling mistakes (proper names of persons or places are notion the Spell-Checker dictionary, as they are not usually found in a traditional Dictionary). In such an instance, students should select "Skip" or "Ignore" in the dialog box.
- Identifies words as mistakes if they are spelled using a language form different than the program default—for example, Canadian spelling (e.g. colour) in an American default (e.g., color). In this case, inquire whether a Canadian dictionary can be installed as the default dictionary on the word processor application.
- If the user-written word contains too many errors then it may not offer any alternative spelling.
- If the error is at the starting of the word then it may not offer any alternative spelling

V. Conclusion

In this project we have to implement the features like autosuggest and spell correction for windows based application. We have to develop this application by using Java technology. Spell checker includes feature like multi-word suggestion to enhance the performance. Spell checker provide facility to reduce the typing work and avoid spelling errors as while entering the words it provides related matching suggestions to the user since saves efforts of the user to type complete words as one can select a word from the list.

Acknowledgment

It is an opportunity of immense pleasure for us to present the seminar **"**Spell Checker**"** expressing our heart left gratitude to all those who have generously offered their valuable suggestion towards the completion of the project. It's rightly said that we are built on the shoulders of others for all our achievements. Thecredit goes to our seminar guide and H.O.D.Computer Prof. L. S. Naikwhose positive attitude, moral support and encouragement lead to the success of the project. We are also graceful to our project guide Prof. P. G. Magdumfor his guidance and valuable suggestion, important to us from time to time. We are also thankful to our principal Dr. G. V. Mulgund for being very generous with his advice and encouragement. We also thank our LAB inchaerge for giving all kinds of help to us.

References

[1] IEEE Paper-SSCS: A Smart Spell Checker System Implementation Using Adaptive Software ArchitectureJournal references.

[2] Review On Error Detection and Error Correction Techniques in NLP

[3] Trie: http://en.wikipedia.org/wiki/Trie Retrieved May 15, 2012

[4] Spell correction: http://norvig.com/spell- correct.html Retrived Nov 30,2012

[5] Sandhya Vissapragada," YIOOP! Introducing AutosuggestAnd spell Check " Approved For The Department Of Computer Seience,2012.

[6] Eedit distance: http://en.wikipedia.org/wiki/ Levenshtein_distance Retrieved No 30, 2012

[7] Autosuggest http://en.wikipedia.org/wiki/ Autocomplete,Retrieved Nov 30, 2012.

[8] IEEE Paper-A Logical Framework For The Correction Of Spelling Errors In Electronicing Documents.

Authors

Vibhakti Bhaire, Computer Engineering, Mumbai University and vibhaktib15@gmail.com.
Ashiki Jadhav, Computer Engineering, Mumbai University and ashijadhav55@gmail.com.
Pradnya Pashte, Computer Engineering, Mumbai University and pradnyapashte13@gmail.com.
Mr**.** Magdum P.G**.**, pandumagdum@gmail.com,9420907385.