CS 154 Project

CorRacketify

Efficient Spelling Corrector in Racket

Team Details

1. Aman Kansal (170050027)
2. Ansh Khurana (170050035)
3. Saksham Goel (170050045)

Introduction

The aim of the project was to build an application completely in Racket which could have some practical usage. Then we started with the one which is very well developed by the various technology giants like Google and Microsoft - a Simple but not so Simple Spell Checker and wanted to see how far we can go with this in Racket ...

Problem description

Given a string made of sentences, do the following:-

1. Find all the incorrectly spelled words.
2. Suggest suitable correctly spelled words.
3. Provide an option to add to dictionary.
4. Replace the incorrect words with the correct words in the string.
5. Provide an option to search meaning of any suggested word from the internet.

Program Design

I. Spelling Check

- Use of Bloom Filter Data Structure -> bit-vector

- Each word in dictionary is pushed through 13 Hash functions to provide 13 values at which the bit-vector is made #t.
- Each given word is also pushed through the Hash functions and the respective boolean values of bit-vector is cross-checked.

II. Correct Spelling Suggestions
- Use of BK-tree -> (struct bknode (pdis val lst));
- Each element in the list is also a seperate tree;
- Each element in sub-tree has a constant DL distance with the root word;
- A given word's DL distance with root is found, say X;
- Then all trees with DL distance (X +- Tolerance) from the root are recursively; traversed.
- Works because of triangle inequality.

III. GUI
- Using racket's GUI package
- Input (of uncorrected text) and Output (Of corrected/marked text) through textboxes.
- Functions attached to buttons
- Suggestion and incorrect words in combo fields (drop-down lists)
- Button taking you to web-browser for any word meaning

Sample Input Output

1. **Input** -> "Applf is delicious" **Output** -> "{Applf} is delicious"
   After Correction -> "Apple is delicious"
2. **Input** -> "Let us play outsde today" **Output** -> "Let us play {outsde} today"
   After Correction -> "Let us play outside today"
3. **Input** -> "AnApple a day" **Output** -> {AnApple} a day"
   After Correction -> "An Apple a day"

Salient features

1. Incredibly fast spell-check- Constant time operations required vs (log N) of Binary Search.
2. Implementation of two independent Data Structures -

    a. Bloom Filter
    b. BK-Tree
3. Implemented an extremely fast non-cryptographic Hash Function in Racket.
4. Levenshtein Distance Algorithm with Damerau optimization
5. Use of Racket's
    a. net/URL package
    b. GUI package
    c. Performance hint package for inlining functions
6. Extensive string parsing

Additional Points Of Interest:

1. The spell checker can be modelled to check spellings in different languages as per the dictionary and character set supplied.
2. Add to Dictionary, and Find & Replace options.
3. Web search to find the meaning of the suggested words
4. Missed Space check

Limitations and bugs:

1. There's a tradeoff between the speed and accuracy of the spell checker.
2. The correctness of a spelling is limited to the dictionary provided -for eg: not all verb forms maybe present
3. Not all special symbols are supported.
4. Ordering of suggestions depends on the root word taken.

References:

1. GNU/Aspell Dictionary:- http://app.aspell.net/create
2. MurmurHash3:- https://github.com/aappleby/smhasher/wiki/MurmurHash3
3. Racket Documentation:- https://docs.racket-lang.org