Submitted By:-

* Vedant Kohli [72454]

**Cluster Innovation Centre**

**University Of Delhi**

****

**Spell Corrector**

Project report

**Computational Linguistics**

**(B.Tech/B.S. in Innovation with Mathematics and IT)**

**1st semester**

**22/11/2013**

**Acknowledgement**

It gives me immense pleasure to express my gratitude towards our Computational Linguistics teacher **Mrs. Shabnam Agarwal** for her guidance, support and encouragement throughout the duration of the project.

Without her motivation and help, the successful completion of this project would not have been possible.

**Index**

1. [**Objective**](#Objective)**……………………………………………………….. 1**
2. [**Method**](#Method)**…………………………………………………………. 1**
   1. [**Platform**](#Platform)**………………………………………………… 1**
   2. [**Dependencies Required**](#Dependencies_Required)**……………………………….. 1**
   3. [**Algorithm**](#Method)**……………………………………………… 2**
      1. [**Stage 1**: Input](#Stage1)…**……………………………….. 3**
      2. [**Stage 2: Calculation**](#Stage2)**……………………………. 3**
3. [**Stage 3: Suggestion** List](#Stage3)…...**………………….... 4**
4. [**Stage 4: Output** Display](#Stage5)…….**………………..... 4**
5. [**Output**](#Output)**………………………………………………………….. 5**
6. [**Future Aspects And Applications**](#Future_Aspects_and_Applications)**……………………………. 6**

**Objective**

To build a program that would accept a word from user, and return the correct spelling of the word along with some suggestions via the Damerau Levenshtein Minimum Edit Distance Algorithm[1].

**Platform:**

Python (Ver. 2.7.5)

**Dependencies Required:**

Dictionary (COCA)[2]

**Method**

**Stage 1:**

Input a word from the user via a graphical display. Store this word as word\_user in the program. Now access each word of a pre-defined dictionary and compare the two words.

**Stage 2:**

Find the minimum edit distance between the two words through the Damerau Levenshtein Algorithm[1] as mentioned:

Initialization**:**

Make a 2D array, D

* For i=0,1……N:

D[i,0]=i

* For j=0,1……M:

D[0,j]=j

Recurring Relation:

ins=D[i-1,j] + 1

del=D[i,j-1] + 1

sub=D[i-1,j-1] + var

where var:

{ if word\_user[i-1]==word\_dict[j-1], var=0}

{else if word\_user[i-1]!=word\_dict[j-1], var=1}

if { i>1 and j>1}

and { word\_user[i-2]==word\_dict[j-1]}

and { word\_user[i-1]==word\_dict[j-2]}

trans=D[i-1,j-2] + 1

D[i,j]=min{ins, del, sub, trans}

**Stage 3:**

Filter out only those words from the dictionary which have Minimum Edit Distance less than or equal to 1. Store them in a list named Suggestions.

**Stage 4:**

Display the Suggestions in a graphical format to the user, giving him option to select one of the words in the list.

**Output**

The Program was tested for testdata, and could successfully print the list of all possible word corrections with minimum edit distance less than or equal to 1, as per the Damerau Levenshtein Minimum Edit Distance Algorithm.

**Future Aspects and Applications:**

* The program could include N-Gram Models for better spell checking.
* The program could include Parts-Of-Speech Tagging to include syntactical and semantic error rectification as well.
* The dictionary or the test set could be enhanced even more.

**References**

[1] D. Jurafsky, J. H. Martin, A. Kehler, K. Vander Linden, and N. Ward, Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, vol. 2. MIT Press, 2000.

[2] M. Davies, “The corpus of contemporary American English (COCA): 400+ million words, 1990-present,” Available Online T Httpwww Am. Org, 2008.