MINI PROJECT 1

IDS 566: Advance Text Analytics
Prof. Moontae Lee
College of Business Administration, the University of Illinois at Chicago

by Team 7

Anvesh Rao Nadipelli

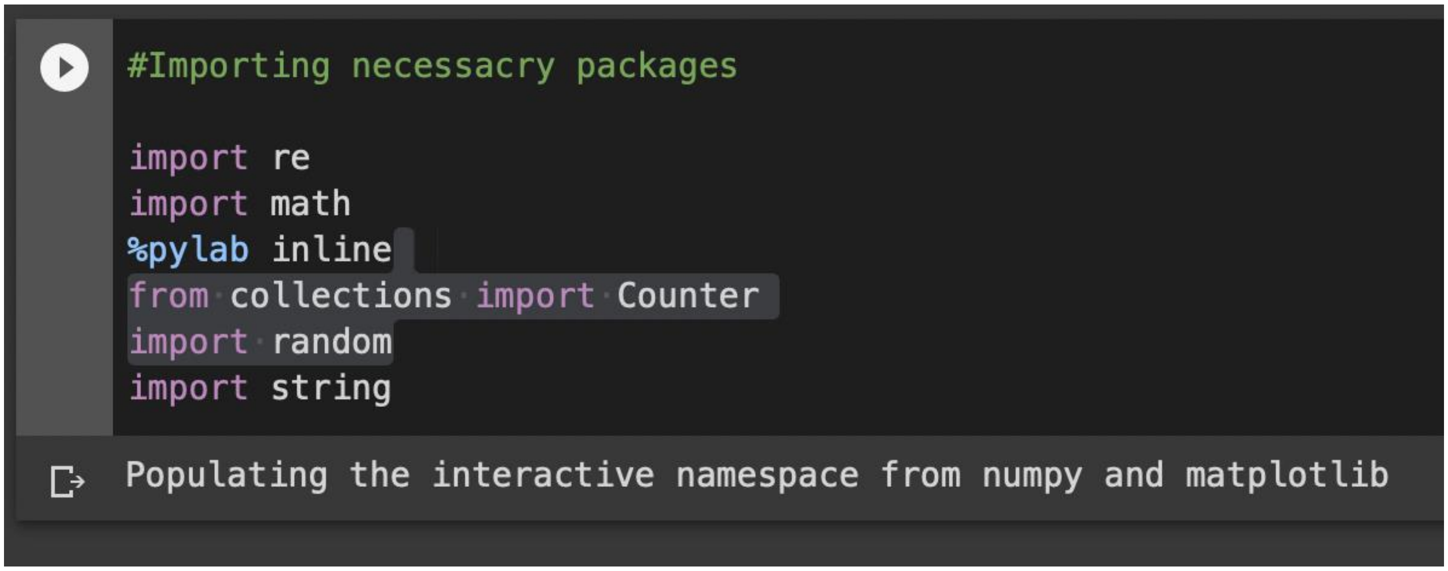
Shalaka Thackare

Nick

Yash

Spellchecker Algorithm:

The necessary libraries we had imported for the job:



```
#Importing necessacry packages

import re
import math
%pylab inline
from collections import Counter
import random
import string
```

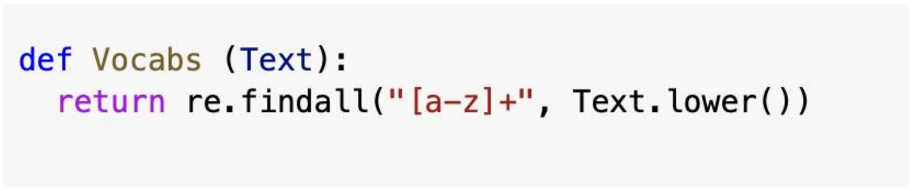
```
Populating the interactive namespace from numpy and matplotlib
```

- Re-package to consider the following to perform [[a-z] will match any lowercase ASCII letter].
- Collections package to use the counter to tokenize the words.
- String to do string operations

Data and Cleaning:

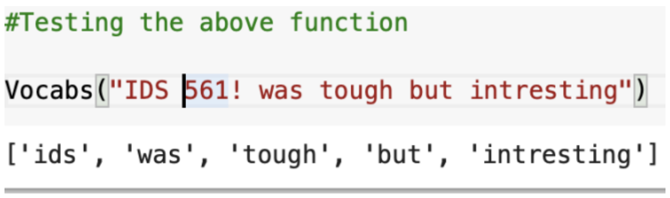
We used the “Data.text” file to load our data.
It includes 6488666 characters and 1105285 words.

Normalized all the capitals to small letter words, ignored the numbers, symbols, and punctuations present in the Data, and created all the set strings containing alphabets using the following function.



```
def Vocabs (Text):
  return re.findall("[a-z]+", Text.lower())
```

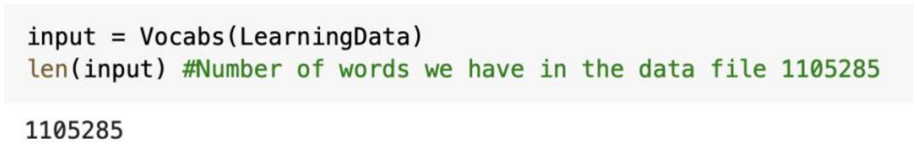
Output for Vocab Example



```
#Testing the above function

Vocabs("IDS 561! was tough but intresting")

['ids', 'was', 'tough', 'but', 'intresting']
```

Tokenization of the strings:

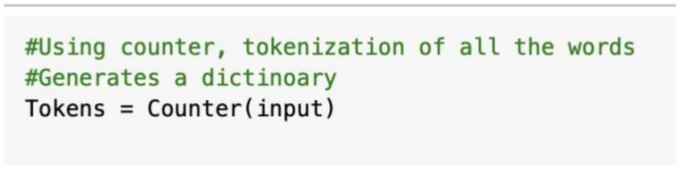
Loaded our data set into input variable using vocab function.
The following code can obtain the number of words in the file:



```
input = Vocabs(LearningData)
len(input) #Number of words we have in the data file 1105285
```

```
1105285
```

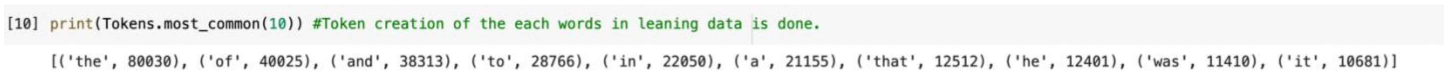
Now from NLP, tokenization of all the words present in the vocab done by the following function:



```
#Using counter, tokenization of all the words
#Generates a dictinoary
Tokens = Counter(input)
```

Resulting in Dictionary with words as keys and tokens (number of times a word appeared in the learning data) as items.

The most common words in the data:



```
[10] print(Tokens.most_common(10)) #Token creation of the each words in leaning data is done.
     [('the', 80030), ('of', 40025), ('and', 38313), ('to', 28766), ('in', 22050), ('a', 21155), ('that', 12512), ('he', 12401), ('was', 11410), ('it', 10681)]
```

Autocorrect and Spell Correction Model:

Our Autocorrect model takes the text data file to construct a dictionary of words, assuming those words exist in English language.

It also determines the likelihood of the word occurrence in English text.

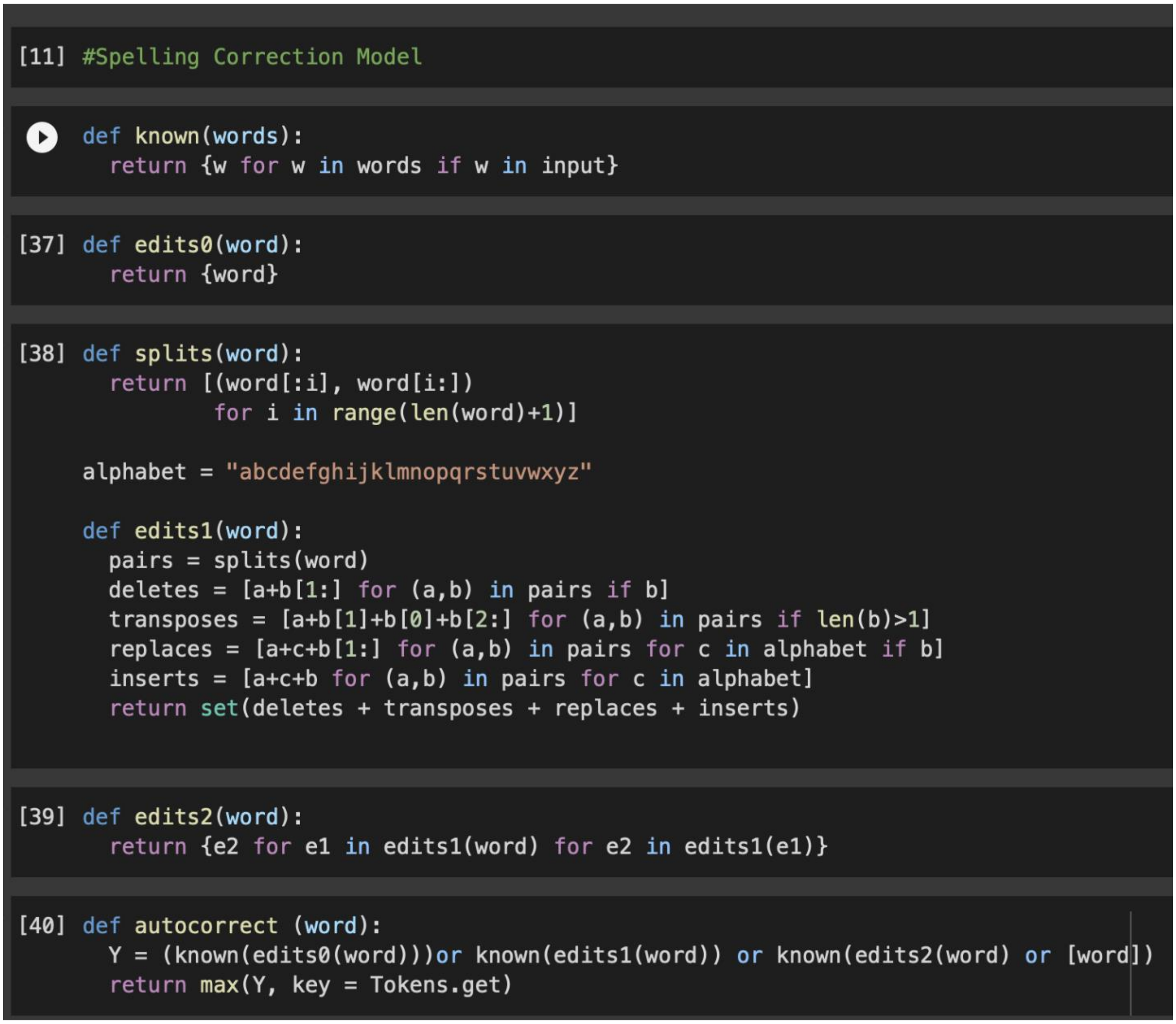
When an inaccurately spelled word is passed to the autocorrection function, the algorithm generates an edit distance 1 version of it by doing one of the following operations:

Delete, transpose, replace, or insert.

Similarly, it just uses the function on the created words again to get a list of words with an edit distance of 2.

The next step is to select the most appropriate term from the created list of corrected words; this is done by using the probability of each word occurring from the previously saved probabilities.

For example, if it had to choose between 'the' and 'then,' it would prefer 'the' because it occurs more frequently in text.



```
[11] #Spelling Correction Model
```

```
def known(words):
  return {w for w in words if w in input}
```

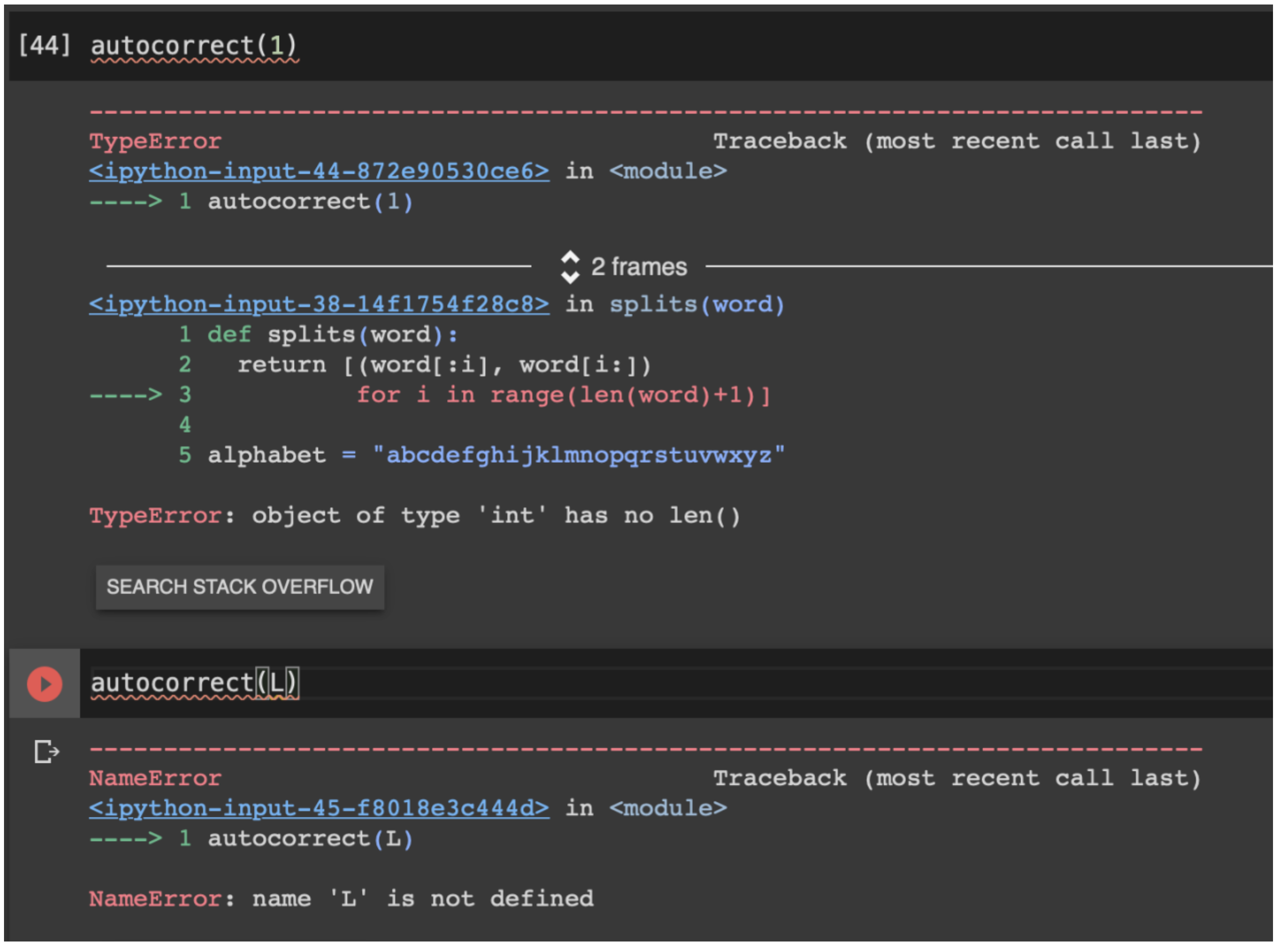
```
[37] def edits0(word):
       return {word}
```

```
[38] def splits(word):
       return [(word[:i], word[i:])
               for i in range(len(word)+1)]

     alphabet = "abcdefghijklmnopqrstuvwxyz"

     def edits1(word):
       pairs = splits(word)
       deletes = [a+b[1:] for (a,b) in pairs if b]
       transposes = [a+b[1]+b[0]+b[2:] for (a,b) in pairs if len(b)>1]
       replaces = [a+c+b[1:] for (a,b) in pairs for c in alphabet if b]
       inserts = [a+c+b for (a,b) in pairs for c in alphabet]
       return set(deletes + transposes + replaces + inserts)
```

```
[39] def edits2(word):
       return {e2 for e1 in edits1(word) for e2 in edits1(e1)}
```

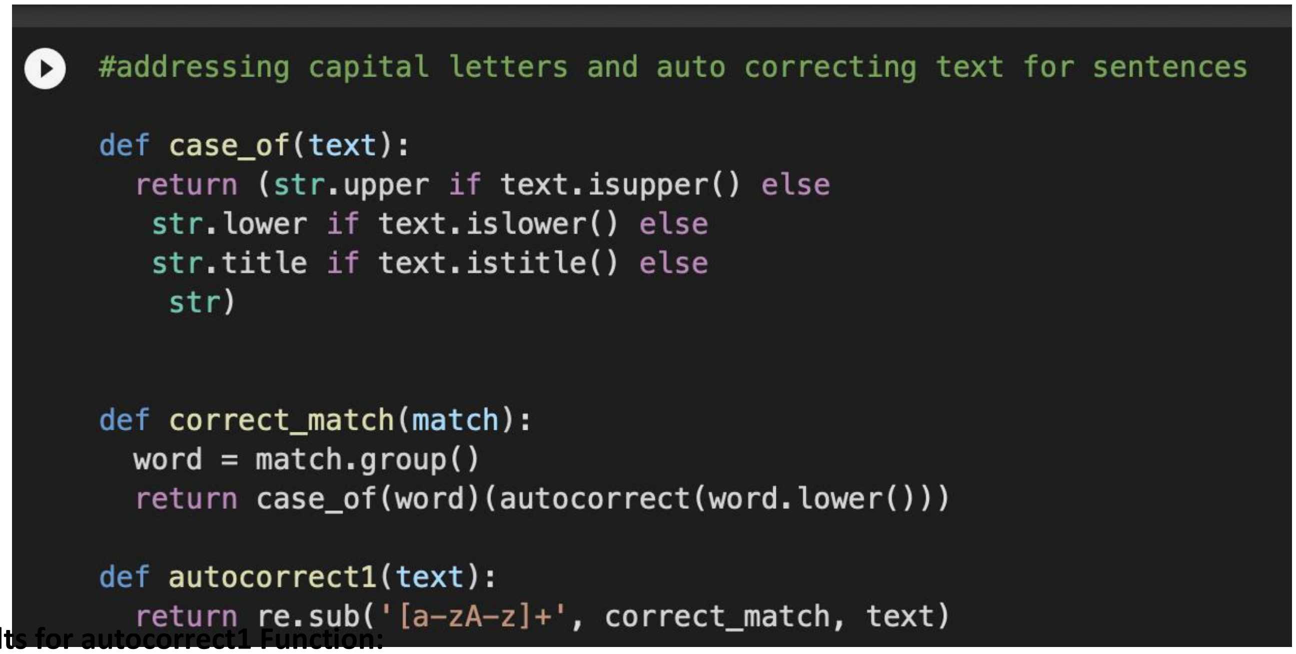
```
[40] def autocorrect (word):
       Y = (known(edits0(word)))or known(edits1(word)) or known(edits2(word) or [word])
       return max(Y, key = Tokens.get)
```

The problem with the above autocorrect function is that it will show an error whenever we use capital letter words, special characters, and numbers.



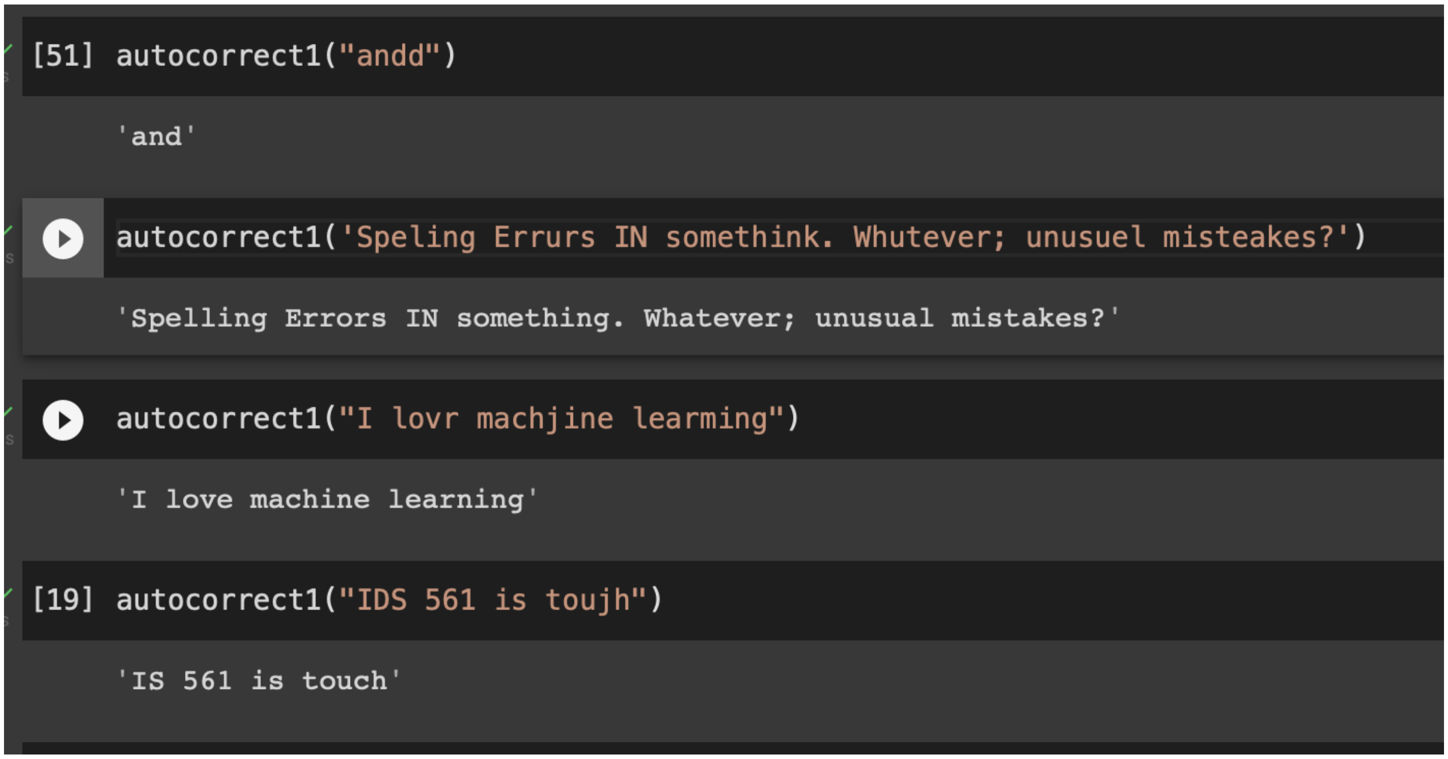
```
[44] autocorrect(1)

     ---------------------------------------------------------------------------
     TypeError                                 Traceback (most recent call last)
     <ipython-input-44-872e90530ce6> in <module>
     ----> 1 autocorrect(1)

                                    2 frames
     <ipython-input-38-14f1754f28c8> in splits(word)
           1 def splits(word):
           2   return [(word[:i], word[i:])
     ----> 3           for i in range(len(word)+1)]
           4 
           5 alphabet = "abcdefghijklmnopqrstuvwxyz"

     TypeError: object of type 'int' has no len()

      SEARCH STACK OVERFLOW

     autocorrect(L)

     ---------------------------------------------------------------------------
     NameError                                 Traceback (most recent call last)
     <ipython-input-45-f8018e3c444d> in <module>
     ----> 1 autocorrect(L)

     NameError: name 'L' is not defined
```

To handle this problem, we defined the following functions, which handled special characters, all types of alphabets (both capital and small), and numbers.



```
#addressing capital letters and auto correcting text for sentences

def case_of(text):
  return (str.upper if text.isupper() else
   str.lower if text.islower() else
   str.title if text.istitle() else
    str)


def correct_match(match):
  word = match.group()
  return case_of(word)(autocorrect(word.lower()))

def autocorrect1(text):
  return re.sub('[a-zA-z]+', correct_match, text)
```

Results for autocorrect1 Function:



```
[51] autocorrect1("andd")

     'and'

     autocorrect1('Speling Errurs IN somethink. Whutever; unusuel misteakes?')

     'Spelling Errors IN something. Whatever; unusual mistakes?'

     autocorrect1("I lovr machjine learming")

     'I love machine learning'

[19] autocorrect1("IDS 561 is toujh")

     'IS 561 is touch'
```

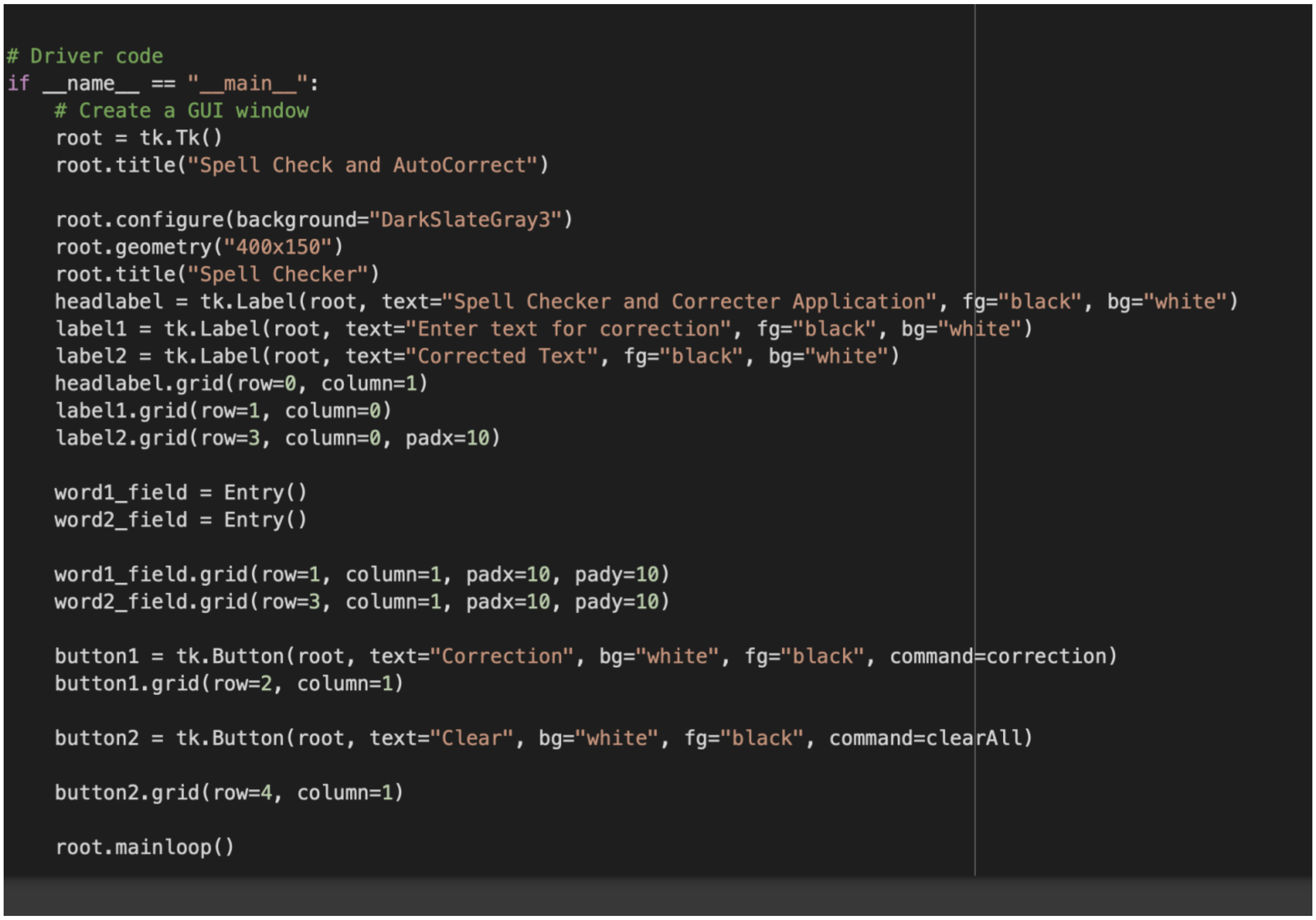
The one problem with this model will be, it's output will always be from the learnt data. Words outside of the learnt data, will be corrected to words present in the learnt data.
Eg [19] IDS is converted to IS. Tough is converted to touch.

<u>User interface</u>

We had used below python packages to generate the GUI Interface from python.

"The below code needs to be runned on PyCharm to a local machine to develop an GUI".

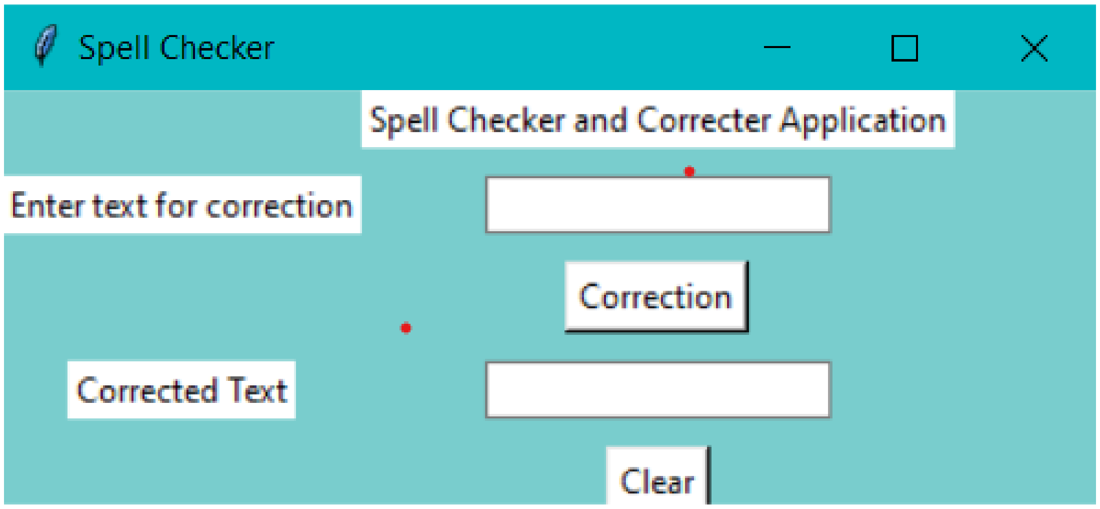
We had used tinkter packages to develop the GUI. Main driver code for the GUI Window:



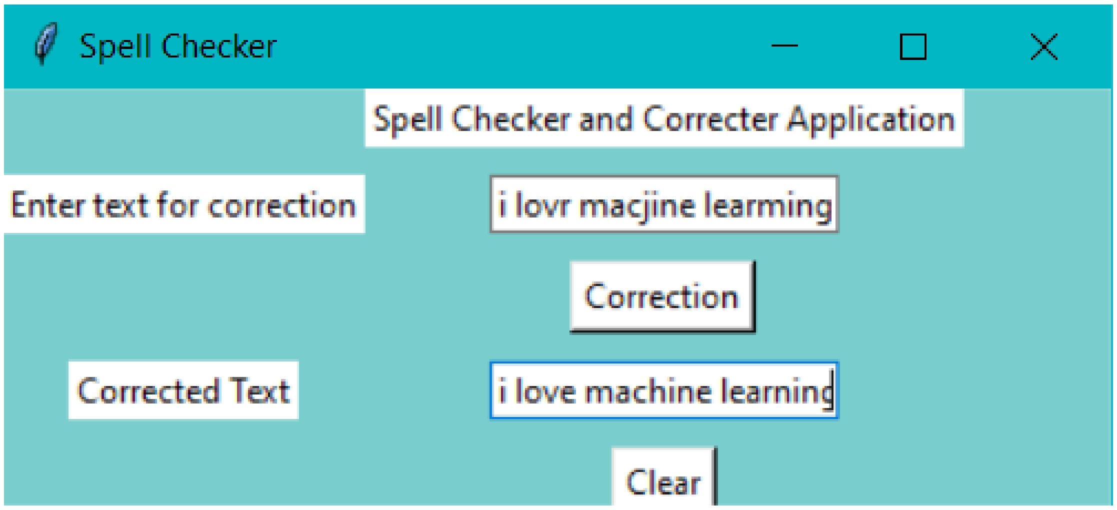
```
# Driver code
if __name__ == "__main__":
    # Create a GUI window
    root = tk.Tk()
    root.title("Spell Check and AutoCorrect")

    root.configure(background="DarkSlateGray3")
    root.geometry("400x150")
    root.title("Spell Checker")
    headlabel = tk.Label(root, text="Spell Checker and Correcter Application", fg="black", bg="white")
    label1 = tk.Label(root, text="Enter text for correction", fg="black", bg="white")
    label2 = tk.Label(root, text="Corrected Text", fg="black", bg="white")
    headlabel.grid(row=0, column=1)
    label1.grid(row=1, column=0)
    label2.grid(row=3, column=0, padx=10)

    word1_field = Entry()
    word2_field = Entry()

    word1_field.grid(row=1, column=1, padx=10, pady=10)
    word2_field.grid(row=3, column=1, padx=10, pady=10)

    button1 = tk.Button(root, text="Correction", bg="white", fg="black", command=correction)
    button1.grid(row=2, column=1)

    button2 = tk.Button(root, text="Clear", bg="white", fg="black", command=clearAll)

    button2.grid(row=4, column=1)

    root.mainloop()
```

W

Output:



Spell Checker
Spell Checker and Correcter Application
Enter text for correction
Correction
Corrected Text
Clear



Spell Checker
Spell Checker and Correcter Application
Enter text for correction
Correction
Corrected Text
Clear

i lovr macjine learming

i love machine learning