Tribhuwan University

Institute of Science and Technology

(Software Engineering)

A Project On

Auto-correction using NLP in Python

Submitted By

Nirajan Thakuri

Roll no. 6

6th semester

Submitted To

Santosh Rijal

SUPERVISOR'S RECOMMENDATION

The project work report entitled "Auto-correction using NLP in Python" submitted by Nirajan Thakuri, Aakash Bhusal and Sunil Tamang of CSIT 6th semester, is prepared under my supervision as per the procedure and format requirements laid by Faculty of CSIT, Kathmandu College of Technology. I, therefore, recommend the project work report for evaluation.

Santosh Rijal

Date: 2079/04/10

ENDORSEMENT

I hereby endorse the project work report entitled "Auto-correction using NLP in Python" submitted by Nirajan Thakuri, Aakash Bhusal, Sunil Tamang of CSIT 6th semester, of Kathmandu College of Technology, on partial fulfillment of the requirement for the project of Software Engineering.

Santosh Rijal

Supervisor

Date: 2079/04/25

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our supervisor Mr. Santosh Rijal, who gave us golden opportunity to do this wonderful project on the topic “Auto-correction using NLP in Python”, which also helped us in doing a lot of research and we came to know about so many new things we are really thankful to them. Similarly, we would also like to thank our parents and friends who helped us a lot in finalizing this report within the limited time frame.

Nirajan Thakuri
Aakash Bhusal
Sunil Tamang
CSIT 6th Semester

ABSTRACT

This project dealt with the Auto-correction of words using Natural Language Processing (NLP) in Python. This project provides the fundamentals for how the Auto-correction feature works and how we get recommended similar words when we spell the word incorrectly. Auto-correction is the saving grace for most people. Auto-correction can help us to improve our message. This project will teach you how the Auto-correction feature we are using nowadays works. For example, “How are yoi” contain the misspelled word “yoi” which autocorrect will identify and change to “you”. On the other hand “How are me” would not be considered error an as there is no misspelled word. In this project, we extract the python libraries like pandas, and NumPy and first calculate the frequency of the word from the dataset we used. Then we calculate the probability (relative frequency) of each word and we find Q gram word using Jaccard distance then list out the similar word to our input word. We test whether our word is correct or not by giving correct and incorrect spelling. If the spelling is correct it says correct else it displays similar words.

Keyword: #Python #NLP #Autocorrection

TABLE OF CONTENT:

1. INTRODUCTION

Natural language processing (NLP) is a subject of computer science—specifically, a branch of artificial intelligence (AI)—concerning the capacity of computers to interpret text and spoken words in the same manner that humans can [1]. NLP combines statistical, machine learning, and deep learning models with computational linguistics—rule-based modeling of human language. With the use of these technologies, computers are now able to process human language in the form of text or audio data and fully "understand" what is being said or written, including the speaker's or writer's intentions and sentiment.The following are some examples of NLP uses: Sentiment Analysis, Fake News Detection, Machine Translation, Question and Answering (Q&A), Chatbot, and many other features are available [2].

Have you ever wondered how the Autocorrect features on a Smartphone keyboardwork? Almost every smartphone maker now includes an autocorrect feature in theirkeyboards, regardless of price. The main purpose of this project is the autocorrect feature. It’s similar but not the exact copy, to the smartphones that we are using nowadays. This would be a Natural Language Processing implementation on a smaller dataset, such as a book [2].

2. PROBLEM STATEMENT

Let’s say you wrote a word on your keyboard; if the word appears in our smartphone's lexicon, it will be assumed that you typed the correct term. It makes no difference if you type a name, a noun, or any other word you desire. What if, though, the word doesn't exist? If you write a term that does not exist in the historyof smartphones, the autocorrect is carefully engineered to discover the most comparable words in our smartphone's history, as it advises [2].

3. OBJECTIVE

- Identify Misspelled Word
- Find ‘n’ Strings Edit distance away
- Filtering of Candidates
- Calculate Probabilities of Words

4. LITERATURE REVIEW

Autocorrect is a technological breakthrough that practically everyone uses regularly, and often without realizing it. I'd guess that on a typical workday, I makeapproximately 50 adjustments to emails and documents, whether through spell- checking, automatic correction, or grammar manipulation. Regardless of how muchwe despise the process on occasion, it is a tremendous source of assistance for us all. But where did it come from? Who created this technology, it's such an impressive concept? [3]

Dean Hachamovitch, the originator, was a gentleman who was assigned to the Word department when he first started working for Microsoft. Dean Hachamovitchbased his concept on the lexicon, which was already present in Word. The dictionary, Hachamovitch noticed, might be used to rectify typing errors. By pressing the left arrow and F3, he was able to change the word "the" to "the." He later utilized the spacebar after discovering that the spacebar is a key that can be used to divide words and compel adjustments. [3]

Hachamovitch discovered a few typical mistakes, such as separate vs. separate. This was the birth of autocorrect, and given Microsoft's (MS) supremacy as a firm, it's hardly surprising it was such a success. He also attempted to alleviate the miseryof the inadvertent caps lock by guaranteeing that it would automatically adjust afterpressing the spacebar, transforming Dan Lewis into Dan Lewis. One day, Hachamovitch modified his manager's autocorrect lexicon so that when "Dean" was typed, it was replaced with "Mike," a coworker of Dean's, and vice versa. This jokemarked the beginning of the amusing side of autocorrect. [3]

The caps lock correction tool had a major flaw: how would it handle exceptions such as CDs and other things that needed to be capitalized? An intern, ChristopherThorpe, was in charge of compiling a master list of these specific words. Thorpe created a script that included all of Microsoft's dictionary manual entries, which hesubsequently incorporated into a corpus after some editing. The list began with words like "abuzz" and "acidhead." Word became much more efficient in later versions, transforming problematic homophone words like "there was" to "there was." [3]

For obvious reasons, Word sought to incorporate obscenity into its innovative function without actually delivering a correction. The workaround was to expand the list of "words that should not be flagged or suggested." In turn, if you enter a supposedly nasty term incorrectly, the word is highlighted in red and marked as incorrect, but no alternatives are offered. Many people, particularly politicians, began to notice the Cupertino effect about the year 1997. This is where Cupertino would mark "cooperation" as wrong and issue a correction. [3]

The Cupertino effect, which refers to an inaccurate suggestion or replacement by a spellchecker, autocorrect, or predictive text tool, is now a real thing, and it alwaysamazes me how these names come around. The pleasure of compiling these lists, whether it's the "Words that should neither be flagged nor suggested" or the modernterms has exceeded Mr. Thorpe's, with petabytes (2^{50} bytes) of public words beingstatistically assessed to determine when one is popular enough to become a replacement. When selecting whether a word can be provided as an alternative or autocorrected, many factors are taken into account, including keyboard placement,phonetic similarity, and simply the popularity of the word. Some possibilities mayeventually be abandoned in favor of others. [3]

Apple now employs a "contextual" autocorrect system that recognizes the languageyou use with certain people and not with others, such as friends vs coworkers. Theimportance of autocorrect is now widely recognized. For example, a Language Logblog commenter described hearing a full Asian-based vernacular where the local youth exploited the first autocomplete option to construct a kind of secret slang. [3]

Following in the footsteps of Word's autocorrect and spell-checker, a slew of different businesses and concepts have adapted its success or overall functionality for their purposes. For example, Google Docs has a function that allows users to program abbreviations, which are then automatically typed out in full when they are used, so WLAC would become "we love autocorrect." Mobile phones have been

using the technology alongside predictive text since the beginning of the decade,with Apple even having patents for how their devices process text entry. [3]

5. METHODOLOGY

5.1.Dataset

We will need a dictionary to create an autocorrect system that compares entered words to past use to determine whether they are correct or not. [4]

We'll use a **sample.txt** [5] file from the project folder that contains the top 1000 words for this project.

5.2. Algorithm

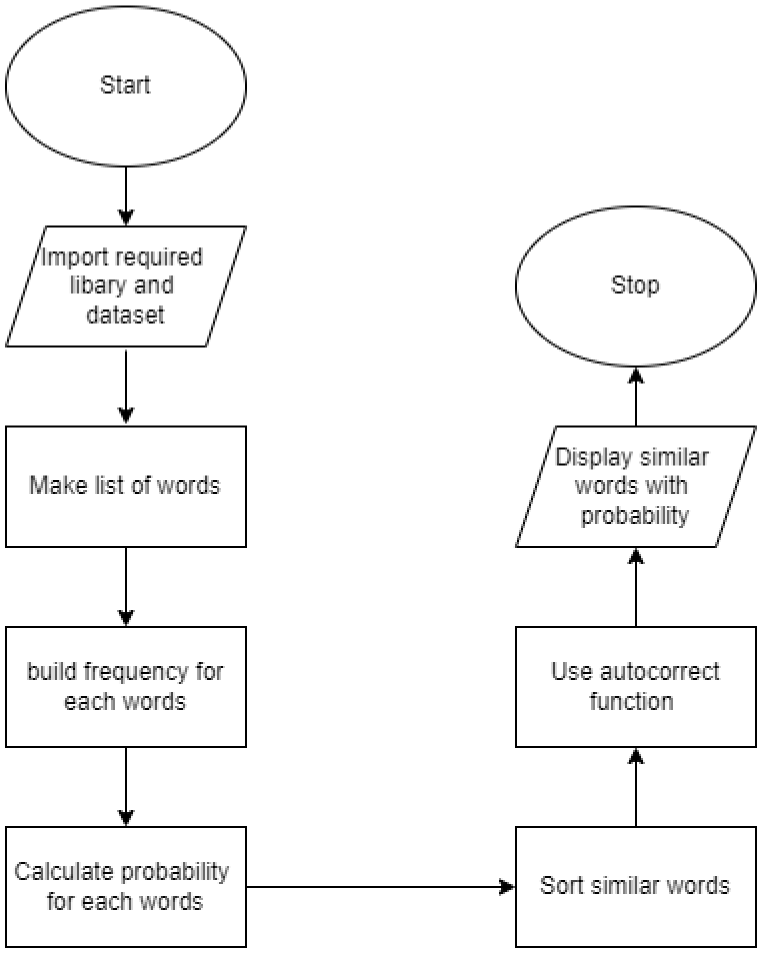

Start
Import required libary and dataset
Make list of words
build frequency for each words
Calculate probability for each words
Sort similar words
Use autocorrect function
Display similar words with probability
Stop

Fig: Flowchart for Auto-correction

Steps:

- Install the library textdistance, which can be installed using the pipcommand.
- Import the necessary packages and dataset of a book.
- Make a list of words and build frequency for each word using a counter function.

- Probability of occurrence for each word is equal to its relative frequency.

  $P(w) = C(w)/V$

  Where,

  P(w) = ProbabilityC(w = frequency
  V = total word in dictionary

- Sort similar words using Jaccard distance and return 5 most similar words.
- Find a similar word using the Auto-correction function which displays the probability of occurrence and similarity of the word.

5.3.Natural Language Processing

Natural Language Processing (NLP) is a branch of Artificial Intelligence that allows computers to understand and process natural human language. Large amounts of natural language data may be evaluated and interpreted by computers thanks to the usage of a programming language in NLP. [4]

It opens the door for increased productivity and engagement in a number of sectors, including:

- Search autocorrect and autocomplete.
- Language translation and grammar checkers.
- Chatbots and social media monitoring.
- Email filtering and voice assistants.

5.4.Schedule

Activity work	1	2	3	4	5	6	7	8	9	10	11	12
Study and Analysis	4 weeks	🟦	🟦	🟦								
Data Collection				2 weeks	🟦							
Implementation					3 weeks	🟦	🟦					
Testing							2 weeks	🟦				
Documentation								3 weeks	🟦	🟦		
Review										2 weeks	🟦	
Presentation												1 weeks

6. TOOLS USED

6.1.Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structures, dynamic typing, and dynamic binding. Additionally, Python's support of modules and packages facilitates modular programs and reuse of code [6].

6.2.Jupyter Notebook

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality [7].

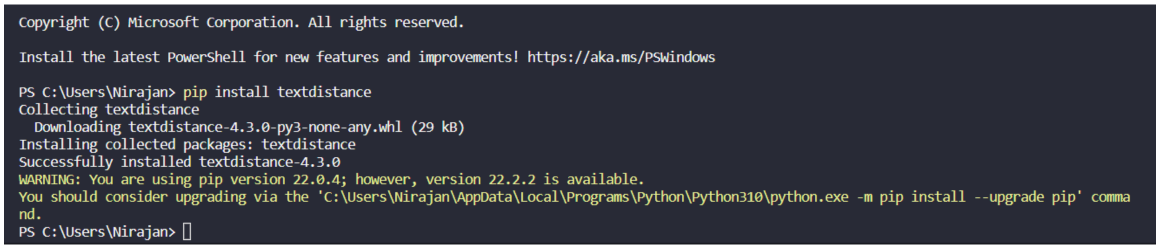
7. IMPLEMENTATION

Our smartphone analyzes historical data to determine if the words we have entered are correct or not. Therefore, in order to activate the Autocorrect capability, we must need a few words.

To explain it realistically, I will use a piece from a book. Let's begin the task of creating a Python autocorrect model.

7.1.Installing Libraries

Some libraries are necessary for us to complete this assignment. I'm planning to utilize extremely broad machine learning libraries. Therefore, all of these libraries—aside from one—should be present and installed on your machine. You must install the "text distance" library, which is simple to install by using the pip command.



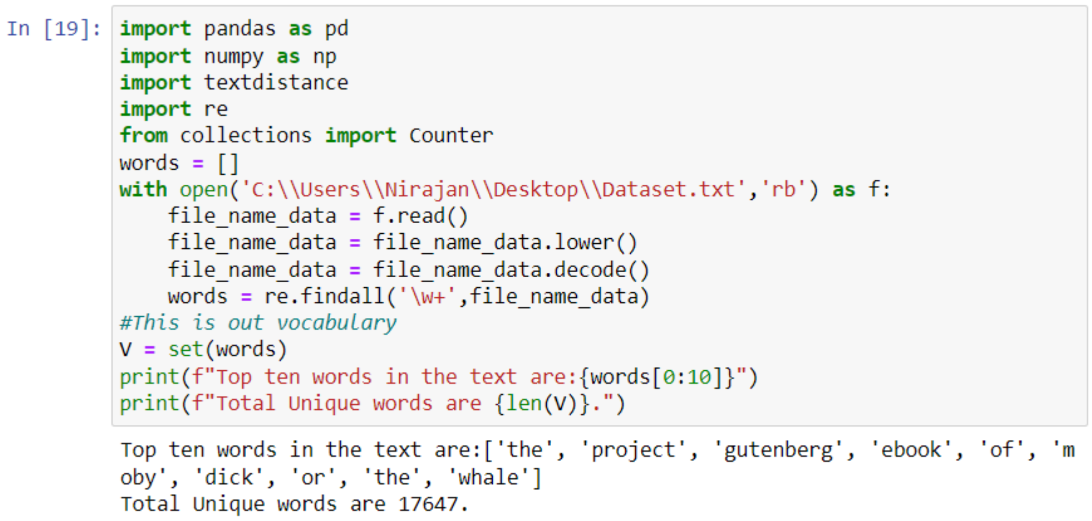
```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Nirajan> pip install textdistance
Collecting textdistance
  Downloading textdistance-4.3.0-py3-none-any.whl (29 kB)
Installing collected packages: textdistance
Successfully installed textdistance-4.3.0
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is available.
You should consider upgrading via the 'C:\Users\Nirajan\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' comma
nd.
PS C:\Users\Nirajan> 
```

7.2.Importing libraries and coding

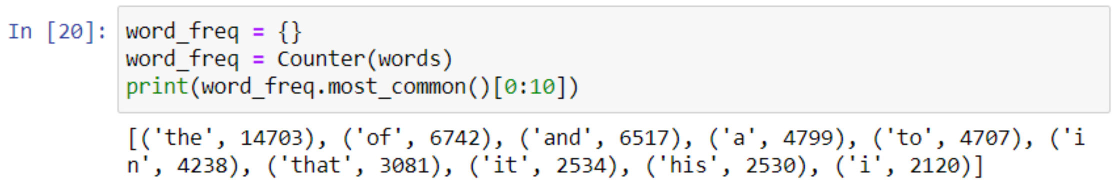
Let's begin by reading our text file and importing all the required packages and libraries.



```
In [19]: import pandas as pd
         import numpy as np
         import textdistance
         import re
         from collections import Counter
         words = []
         with open('C:\\Users\\Nirajan\\Desktop\\Dataset.txt','rb') as f:
             file_name_data = f.read()
             file_name_data = file_name_data.lower()
             file_name_data = file_name_data.decode()
             words = re.findall('\w+',file_name_data)
         #This is out vocabulary
         V = set(words)
         print(f"Top ten words in the text are:{words[0:10]}")
         print(f"Total Unique words are {len(V)}.")

         Top ten words in the text are:['the', 'project', 'gutenberg', 'ebook', 'of', 'm
         oby', 'dick', 'or', 'the', 'whale']
         Total Unique words are 17647.
```

As you can see from the code above, we have created a list of terms. Next, we will calculate their frequency, which is made simple by utilizing Python's "counter function":

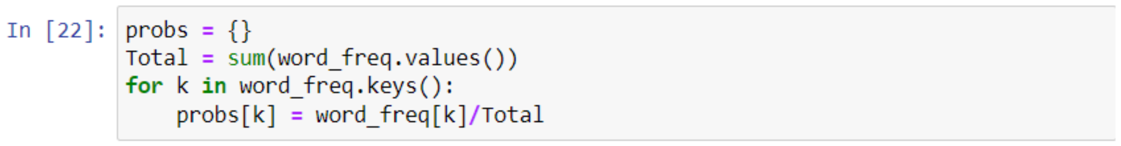


```
In [20]: word_freq = {}
         word_freq = Counter(words)
         print(word_freq.most_common()[0:10])

         [('the', 14703), ('of', 6742), ('and', 6517), ('a', 4799), ('to', 4707), ('i
         n', 4238), ('that', 3081), ('it', 2534), ('his', 2530), ('i', 2120)]
```

Relative frequency of Words

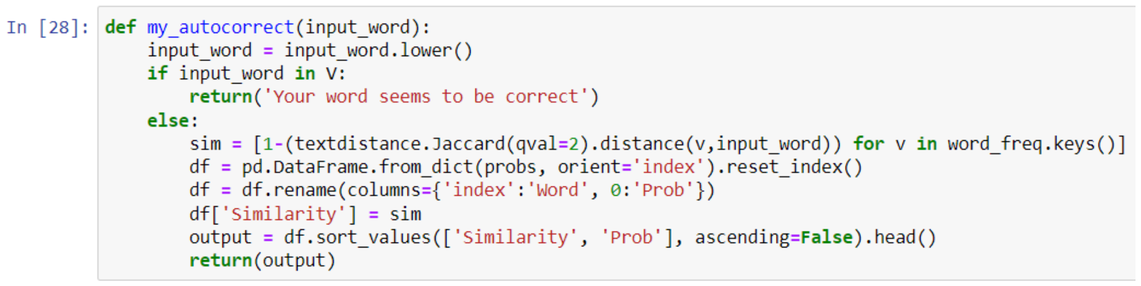
Now that we have found the probabilities, which are equal to the relative frequencies of the words, we want to discover the occurrence of each word:



```
In [22]: probs = {}
         Total = sum(word_freq.values())
         for k in word_freq.keys():
             probs[k] = word_freq[k]/Total
```

Finding Similar Words

We will thus calculate the two grams Q of the words and then arrange comparable words according to the "Jaccard Distance." Then, in order by similarity and probability, we will return the following five words: -

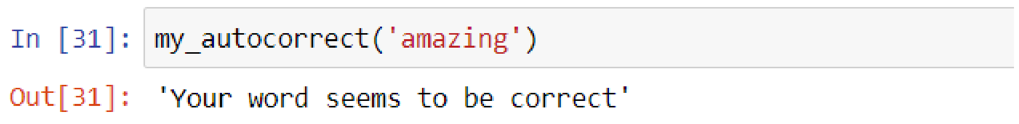


```
In [28]: def my_autocorrect(input_word):
             input_word = input_word.lower()
             if input_word in V:
                 return('Your word seems to be correct')
             else:
                 sim = [1-(textdistance.Jaccard(qval=2).distance(v,input_word)) for v in word_freq.keys()]
                 df = pd.DataFrame.from_dict(probs, orient='index').reset_index()
                 df = df.rename(columns={'index':'Word', 0:'Prob'})
                 df['Similarity'] = sim
                 output = df.sort_values(['Similarity', 'Prob'], ascending=False).head()
                 return(output)
```

7.3.Testing

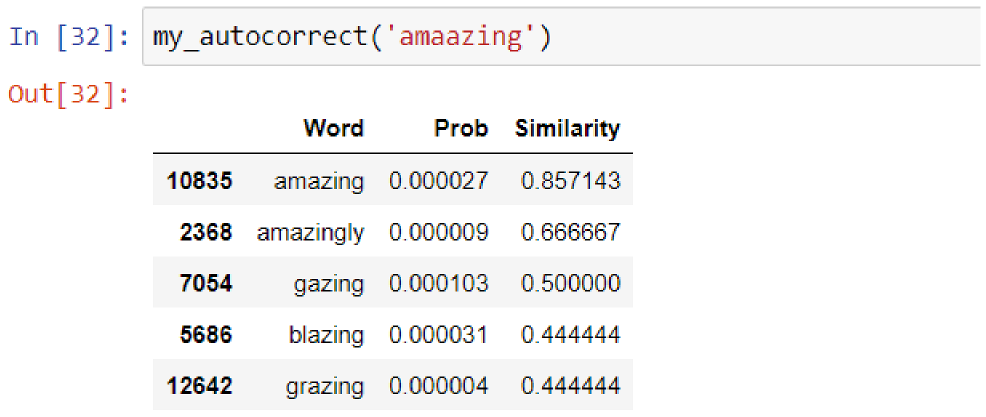
Let us find some similar words by using our autocorrect function:

Checking correct spelling:



```
In [31]: my_autocorrect('amazing')
Out[31]: 'Your word seems to be correct'
```

Checking incorrect spelling:



```
In [32]: my_autocorrect('amaazing')
Out[32]:
```

	Word	Prob	Similarity
10835	amazing	0.000027	0.857143
2368	amazingly	0.000009	0.666667
7054	gazing	0.000103	0.500000
5686	blazing	0.000031	0.444444
12642	grazing	0.000004	0.444444

This is how autocorrect feature works.

8. CONCLUSION

As we have taken words from a book, some words are already present in the vocabulary of the smartphone and then some words it records while the user starts typing using the keyboard.

You can use this autocorrect feature to implement in real-time.

9. REFERENCE

[1]A. Kadlaskar, "Autocorrect features using NLP," Analytics Vidhya, 08 November 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/11/autocorrect-feature-using-nlp-in- python/..

[2]"What is Natural Language Processing?," IBM Cloud Education, 02 July 2020. [Online]. Available: https://www.ibm.com/cloud/learn/natural-language-processing.

[3]D. Lewis, "The History and Origin of Autocorrect," Factsite, 17 March 2021. [Online]. Available: https://www.thefactsite.com/history-of-autocorrect/.

[4]A. Lia, "Using an Autocorrect feature using NLP with Python," 4 December 2021. [Online]. Available: https://www.section.io/engineering-education/building-autocorrect-feature-using-nlp-with-python/.

[5]A. Lia, "Autocorrect_System/sample.txt," 15 November 2021. [Online]. Available: https://github.com/dentex22/Autocorrect_System/blob/main/sample.txt.

[6]"What is Python?," Teradata, [Online]. Available: https://www.teradata.com/Glossary/What-is-Python#:~:text=Python%20is%20an%20interpreted%2C%20object,British%20comedy%20group%20Monty%20Python..

[7]"Project Jupyter," Jupyter, [Online]. Available: https://jupyter.org.