BNB INTERSHIP

AUTOCORRECTION

Presentation by Arham Ahmed, David Abraham, and Heet Jani

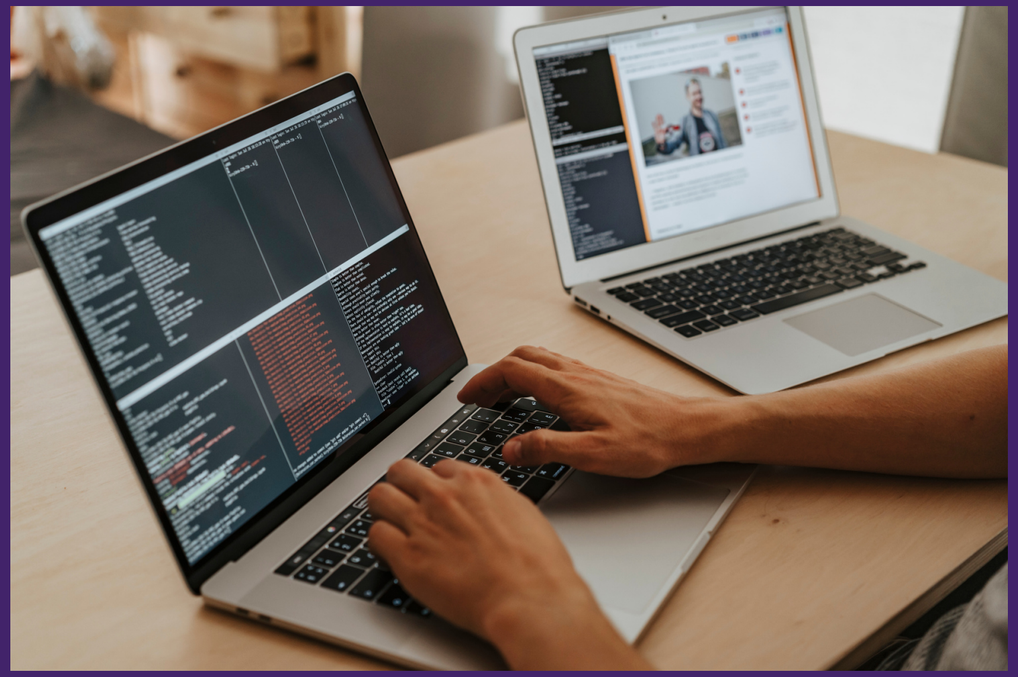
DESCRIPTION

What is **Autocorrect?**

Autocorrect is a type of software program that identifies miss-spelled words, uses algorithims to identify the words most likely to have been intended, and edits the text
for you.

KEEP BROWSING TO LEARN MORE

What Do You Know about The Advantages of Autocorrect?



2022 PRESENTATION

STUDIO SHODWE

Autocorrect has Countless Advantages

- It increases productivity
- Discreet and easy to use
- Improves time management
- Intelligent Corrections
- Gives one the write word and spells it correctly

Technical

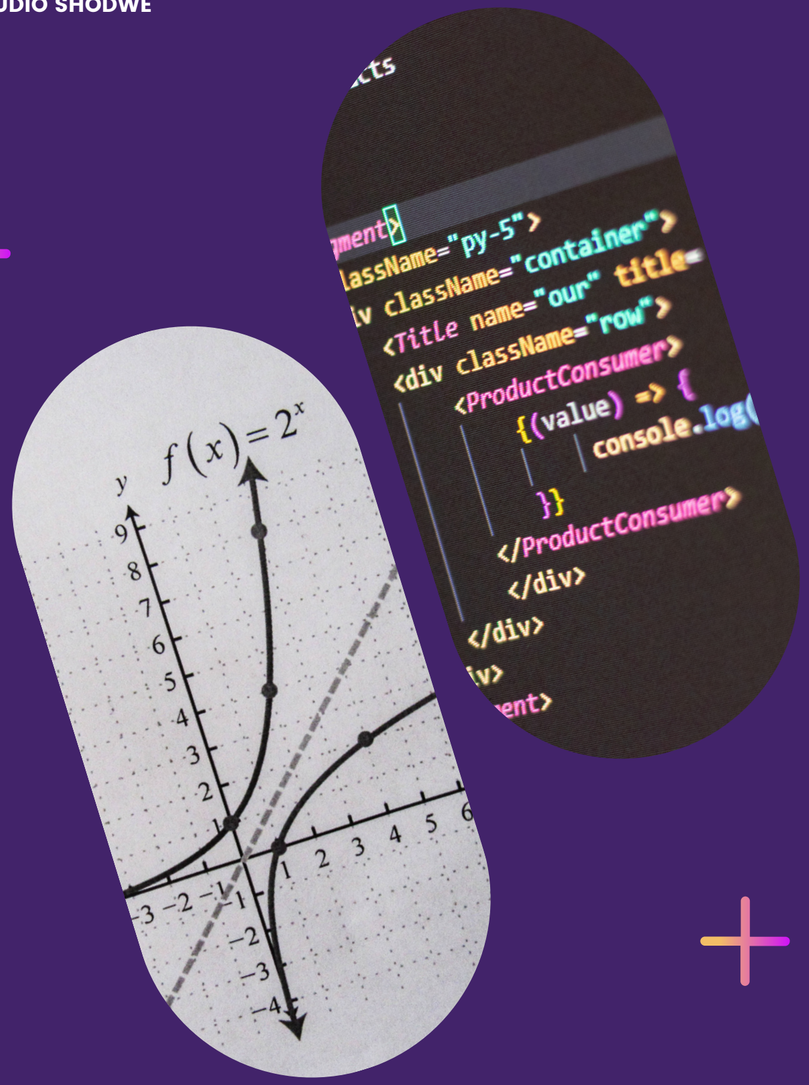
Details

Why use Artificial Intelligence?

How AI is used in Autocorrect

Computers have used machine learning algorithms in order to predict the word being typed based on the relevant context, and the data that the algorithm has collected is processed locally on your device.

STUDIO SHODWE



How does it Actually Work?

01

When an incorrect word is given to the model it matches the percentages given by the formula underneath and the probabilities the model thinks is the correct word and then giving a list or single of word(s) of the words with the highest probable outcome.

P(w) = C(w)/V

P(w)- the probability of a word w.

C(w) - number of times (frequency) word appears in the vocabulary dictionary.

V - the total sum of words in the dictionary.

Data Collection & EDA

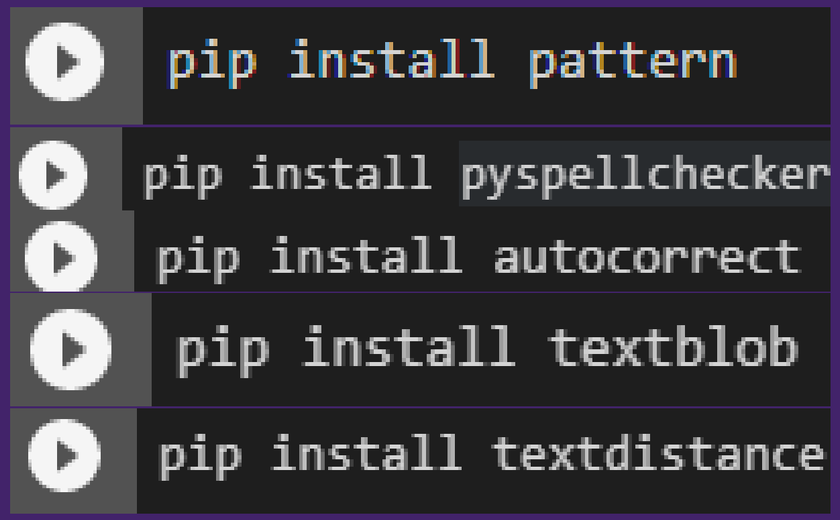
In order to take our model further in the functioning process, we needed to gather useful, appropriate data . After extensive research, we were able to come up with a file containing 1001 words of the most popular vocab used by individuals today.

Details of our Code

Overview

We will be explaining the code in chunks at a time to give a as detailed overview as possible

CONTINUING...



First of all...

We need to download the necessary libraries to assist the model

LEARN MORE

Starting Off...

```
# Step 1: Data Preprocessing
import re  # regular expression
from collections import Counter
import numpy as np
import pandas as pd

# Implement the function process_data which
# 1) Reads in a corpus
# 2) Changes everything to lowercase
# 3) Returns a list of words.

w = [] #words
with open('sample.txt','r',encoding="utf8") as f:
    file_name_data = f.read()
    file_name_data = file_name_data.lower()
    w = re.findall('\w+', file_name_data)

v = set(w) #vocabulary
print(f"The first 10 words in our dictionary are: \n{w[0:10]}")
print(f"The dictionary has {len(v)} words ")
```

To begin,

We import all the necessary libraries then assign certain variables to reading and accessing the data file we have linked. To check, we make the model say the first 10 words in the document and the number of words in the dictionary

Output:

This piece of code will produce an outcome saying,

"The first 10 words in our dictionary are:

['a', 'ability', 'able', 'about', 'above', 'accept', 'according', 'account', 'across', 'act']

The dictionary has 1001 words "

II

Now we implement the function that allows certain details to be stored in variables. Such as the 'get_count' function that returns a dictionary of word versus frequency'.

```
 #Implement the function process_data
def get_count(words):
    word_count_dict = {}
    for word in words:
        if word in word_count_dict:
            word_count_dict[word] += 1
        else:
            word_count_dict[word] = 1
    return word_count_dict
word_count_dict = get_count(w)
print(f"There are {len(word_count_dict)} key values pairs")
```

```
There are 1001 key values pairs
```

```
[12] word_count = get_count(w)
     print(f"The dictionary has  {len(word_count)} key values pairs")
```

```
The dictionary has  1001 key values pairs
```

This piece of code helps us to evaluate the probability of any word to appear from the data file. This function will come very-handy later on in the code.

```
def get_probs(word_count_dict):
    probs = {}
    m = sum(word_count_dict.values())
    for key in word_count_dict.keys():
        probs[key] = word_count_dict[key] / m
    return probs
```

These four edit functions play a pivotal role

In our code

```python
def replace_letter(word):
    split_l = []
    replace_list = []
    for i in range(len(word)):
        split_l.append((word[0:i], word[i:]))
    alphabets = 'abcdefghijklmnopqrstuvwxyz'
    replace_list = [a + l + (b[1:] if len(b) > 1 else '') for a, b in split_l if b for l in alphabets]
    return replace_list

replace_l = replace_letter(word='can')
```

```python
def insert_letter(word):
    split_l = []
    insert_list = []
    for i in range(len(word) + 1):
        split_l.append((word[0:i], word[i:]))
    letters = 'abcdefghijklmnopqrstuvwxyz'
    insert_list = [a + l + b for a, b in split_l for l in letters]
    # print(split_l)
    return insert_list
```

```python
def SwitchLetter(word):
    split_l = []
    switch_l = []
    for i in range(len(word)):
        split_l.append((word[0:i], word[i:]))
    switch_l = [a + b[1] + b[0] + b[2:] for a, b in split_l if len(b) >= 2]
    return switch_l

switch_word_l = SwitchLetter(word="eta")
```

```python
def DeleteLetter(word):
    delete_list = []
    split_list = []
    for i in range(len(word)):
        split_list.append((word[0:i], word[i:]))
    for a, b in split_list:
        delete_list.append(a + b[1:])
    return delete_list

delete_word_l = DeleteLetter(word="cans")
```

- Delete Letter
- Replace Letter
- Switch Letter
- Insert Letter

STUDIO SHODWE

Last but not Least

This last piece of code integrates all the four operations(Delete, Insert, Replace, Switch) and allows the model to interchange and use the different functions when needed. The next chunk of code controls a parameter, which allows the number of edits applied to a string. Finally, matching the probabilities against a hypothesized word and returns that outcome.

```
# combining the edits
# switch operation optional
def edit_one_letter(word, allow_switches=True):
    edit_set1 = set()
    edit_set1.update(DeleteLetter(word))
    if allow_switches:
        edit_set1.update(SwitchLetter(word))
    edit_set1.update(replace_letter(word))
    edit_set1.update(insert_letter(word))
    return edit_set1

# edit two letters
def edit_two_letters(word, allow_switches=True):
    edit_set2 = set()
    edit_one = edit_one_letter(word, allow_switches=allow_switches)
    for w in edit_one:
        if w:
            edit_two = edit_one_letter(w, allow_switches=allow_switches)
            edit_set2.update(edit_two)
    return edit_set2

# get corrected word
def get_corrections(word, probs, vocab, n=2):
    suggested_word = []
    best_suggestion = []
    suggested_word = list(
        (word in vocab and word) or edit_one_letter(word).intersection(vocab) or edit_two_letters(word).intersection(
            vocab))
    best_suggestion = [[s, probs[s]] for s in list(reversed(suggested_word))]
    return best_suggestion

my_word = input("Enter any word:")
probs = get_probs(word_count)
tmp_corrections = get_corrections(my_word, probs, v, 2)
for i, word_prob in enumerate(tmp_corrections):
    print(f"word {i}: {word_prob[0]}, probability {word_prob[1]:.6f}")
```

STUDIO SHODWE

Example

In this example, the model will prompt you with a text box. If one writes any miss-spelled word, the model will show a list of options along with their probability.


```
Enter any word:Runt
word 0: put, probability 0.000999
word 1: fund, probability 0.000999
word 2: cut, probability 0.000999
word 3: want, probability 0.000999
word 4: gun, probability 0.000999
word 5: run, probability 0.000999
word 6: must, probability 0.000999
word 7: just, probability 0.000999
word 8: out, probability 0.000999
word 9: unit, probability 0.000999
word 10: but, probability 0.000999
```

What we have Accomplished

01
whole month

03
members

100%
Effort

After all this while we are proud of what of we have accomplished. From learning the fundamentals of python to programming a Autocorrection Model. We wrote a report afterwards and posted it in Github.

Finally,

To conclude, after rigorous yet informative classes, we were able to enhance and hone our coding skills to the extent of actually constructing our own working AI model. From only reading and wondering about the words "Artificial Intelligence" to programming a fully functional model we have come a long way in our journey. We would not have been able to reach this far without the guidance and insructions of our teachers. To all the teachers who guided us along the way thank you for your help.