Autocorrect with NLP

CB SUMANTH – I am certified Data Science and Analyst from Cloudy ML. Here Is my small Deep learning and Machine Learning project understand the text classifiers.

Objective:

- Autocorrect is purely based on Natural Language Processing (NLP). As the name suggests that it is programmed in order to correct spellings and errors while typing text.
- Let's assume that you have typed a word on your keyboard but if that word exists in the vocabulary of our smartphone then it will assume that you have written the right word.
- If the word exists in the history of the smartphone, it will generalize or create the word as a correct word to choose. But What if the word doesn't exist? Okay, If the word that you have typed is a nonexisting word in the history of smartphones then the autocorrect is specially programmed to find the most similar words in the history of our smartphone as it suggests.

Building an Autocorrect Feature:

Procedure to build an Autocorrect feature:

Step1: Import the all necessary libraries like pandas , numpy and re for data manipulation and pattern matching.

Step2: Import the necessary TextDistance library

To run this task, we are required some libraries. You need to install one library known as "text distance", which can be easily installed by using the pip command.

TextDistance : python library for comparing distance between two or more sequences by many algorithms.

Step 3: Upload and Process Text Data

1. Upload the Text Data:
    - If you have the text file stored locally, you'll need to upload it to your working environment.
2. Read the File:
    - Use a function to read the contents of the file.
3. Convert Each Word to Lowercase:
    - After reading the file, ensure that all words are converted to lowercase to standardize the text.

Step 4: Count Word Frequency and Create Probability Distribution

1. **Count Word Frequency**:
    - After processing the text, count the frequency of each word.
2. **Calculate Word Probabilities**:
    - Compute the probability of each word by dividing its frequency by the total number of words in the text.

Step 5: **Calculate and create the auto correct or similarity check function.**

After calculating the values, now we will define a function that returns the probability of words in a list and then defines the word as lower case and then checks if it is present in unique_words, which is an array of all the words that are not already defined.

- If it is found, then this will return 'Your word seems to be correct'.
- Otherwise, similar values are calculated by finding how many times each word appears in frequency_words and subtracting 1 from its distance from other words.
- This value is stored into similarity column and sorted descending order with head() returning only one row containing probabilities for all the words.

The code is a function that will return the probability of words in the dictionary.

We will sort the similar words based on Jaccard Distance by computing the 2 Q grams of the words. We will return the 5 most similar words order by Similarity and Probability.

Step 6: Conclusion and Result

->After all doing all the processing of similarity and probabilites Now we test the our function by giving inputs.

->As we have taken words from a book. In the same way, there are some words that 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 feature to implement in real-time.