AutoCorrect Task

Introduction:

The task is to make a pipeline to make an autocorrect for the English language.

As requested The problem is that we need to deliver a pipeline to the autocorrect system, I delivered **two approaches** to this problem one of them is using a tool called **neuspell** and the other one is by calculating the **distance between the wrong words and correct words in our vocab**.

Data:

The neuspell approach data:

* I used a kaggle dataset obtained <https://www.kaggle.com/bittlingmayer/spelling>

The data format is a correct file and a corrupt file

The data size is 36k words

The training size 28k words

The test data size 7k words

The data used in the Levenshtein distance approach

* NLTK dataset Austen-emma.txt as a random choice to generate → up to 100k

Tools:

The neuspell approach tools:

* Spacy
* Neuspell repo – > https://github.com/neuspell/neuspell
* Kaggle to download the dataset

The Levenshtein distance tools:

* Nltk

Process:

The neuspell approach:

1. First, we cloned the repo of neuspell to use it for spell correction
2. Then we tried the model to infer with its pre-trained checkpoint
3. Then i downloaded the data from Kaggle to use it in training the tool
4. I processed the data to fit the format the tool accepts which is a .txt file with clean words and another for corrupted words (misspelled)
5. The checkpoint is used to infer the test cases

The Levenshtein distance approach:

1. First I downloaded the Emma dataset to extract the vocab we will reference to
2. Then I calculated the count and the probability of each word in the text
3. Then each word in the test cases is passed throw a test, if this word is in the vocab or not.
4. If not then its a misspelled word
5. The word will calculate the Levenshtein distance with each word in the vocab and we will return the words with the closest distance to it (sorted with each word probability to appear in the text)

* This approach leverage the naive bayes algorithm.

Test Cases:

The test cases have miss spelled sentences in both approaches but the first approach attempt to correct it, the second approach suggests the best words that fit.

Challenges:

Installing the repo of neuspell and leveraging its capabilities by reading the docs

Learned lessons:

Using spacy to extract various data from the text.

References:

* <https://www.kaggle.com/bittlingmayer/spelling>
* <https://github.com/neuspell/neuspell>
* Natural language processing specialization probabilistic models :

https://github.com/amanjeetsahu/Natural-Language-Processing-Specialization/tree/master/Natural%20Language%20Processing%20with%20Probabilistic%20Models