**Spelling Correction Tool**

**Description:** Implement a spelling correction tool using techniques like Levenshtein distance or language models.

**Introduction**:

Spell check and correction is a crucial aspect of natural language processing (NLP) that aims to automatically identify and rectify spelling errors in textual data. This project develops a spelling correction tool using unigram and bigram frequencies to suggest and select the most probable corrections for misspelled words. By analyzing a dataset of correctly spelled sentences, the tool learns word sequence likelihoods to improve text accuracy.

In Python, various techniques and libraries, such as Levenshtein distance and language models, are used to implement robust spell-checking systems. These systems enhance text accuracy and readability, playing vital roles in applications from word processors to search engines. Leveraging NLP techniques, spell check and correction tools significantly improve communication and user experience in digital environments.

**Types of errors:**

1. ****Cognitive Errors****: In this type of error the words like *piece-peace*knight-night, steal-steel are homophones (sound the same). So you are not sure which one is which.
2. ****Real Word Errors****: Sometimes instead of creating a non-word, you end up creating a real word, but one you didn’t intend. E.g, typing *buckled* when you meant *bucked*. Or if you type in *three* when you meant *there*.
3. ****Non-word Errors****: This is the most common type of error like if we type *langage* when you meant *language*; or *hurryu* when you meant *hurry.*
4. ****Short forms/Slang****: In this case may be u r just being kewl. Or you are trying hard to fit in everything within a text message or a tweet and must commit a spelling sin. We mention them here for the sake of completeness.

**Different Spell Correction Libraries**

1. ****Jamspell**** It is a modern spellchecking library. It is light-weight, fast and accurate. It consider word surroundings to make better corrections. It has following features:  
   It considers words surroundings (context) for better correction  
   Nearly 5K words per second  
   Multi-language →it’s written in C++ and available for many languages with swig bindings
2. ****Symspellpy**** The Symmetric Delete spelling correction algorithm reduces the complexity of edit candidate generation and dictionary lookup for a given Damerau-Levenshtein distance. It is six orders of magnitude faster Damerau-Levenshtein distance.) and language independent. An average 5 letter word has about 3 million possible spelling errors within a maximum edit distance of 3, but SymSpell needs to generate only 25 deletes to cover them all, both at pre-calculation and at lookup time.
3. ****Textblob****The textblob’s spelling correction is based on Peter Norvig’s “Damerau-Levenshtein distance. as implemented in the pattern library.

**Building a Spelling Correction Tool with NLP**

1. **Libraries and Imports**

The project utilizes several essential libraries:

- `nltk`: Provides robust tools for text processing, tokenization, and frequency distribution analysis.

- `json`: Facilitates loading and handling data in JSON format.

- `tqdm`: Displays progress bars during processing, enhancing the user experience.

- `time`: Measures execution time, aiding in performance evaluation.

**2. Data Preprocessing**

The data used for training and testing the model is loaded from a JSON file, which contains pairs of original (potentially misspelled) and corrected sentences. The dataset is split into training and testing sets, with the first 100 lines reserved for testing purposes. The data preprocessing steps include:

- Tokenizing the sentences to remove special characters.

- Separating the training and testing sets.

- Converting the sentences into lower case to ensure consistency.

**3.Model Used**

The spelling correction model is based on the probabilities of unigrams and bigrams:

- Unigrams: These are individual words. The unigram probabilities are calculated based on their frequency of occurrence in the training data.

- Bigrams: These are pairs of consecutive words. The bigram probabilities are determined by the frequency of word pairs in the training data.

* **Unigram and Bigram Probabilities**

The model calculates the frequency distribution of unigrams and bigrams from the training data. This statistical information forms the basis for determining the most probable corrections for misspelled words.

* **Correction Mechanism**

When a misspelled word is encountered, the model generates candidate corrections using the edit distance metric, which measures the number of changes required to transform one word into another. The best correction is selected based on the bigram probabilities involving the candidate word and its context within the sentence. This approach ensures that the corrected word fits naturally into the sentence, enhancing the overall coherence and accuracy.

* **Evaluation**

The model's performance is evaluated by comparing the corrected sentences with the actual corrected sentences in the test set. Accuracy is measured by the proportion of correctly predicted words. The evaluation process includes:

- Comparing each predicted word with the actual word.

- Calculating the average accuracy across all test sentences.

- Measuring the time taken for the evaluation process.

**4.Conclusion and Results**

The spelling correction tool was successfully developed using unigram and bigram probabilities derived from a training dataset. The tool was evaluated on a test set of 100 sentences, achieving an average accuracy of 84.19%. The model correctly predicted 13 out of 100 sentences without any errors, demonstrating its effectiveness in improving text quality. The evaluation process took approximately 63.92 seconds.

The results indicate that the tool is a promising solution for spelling correction, though there is potential for further enhancement. Future improvements could involve:

- Integrating more advanced models such as neural networks or deep learning techniques.

- Using larger and more diverse datasets for training.

- Refining the candidate selection process to consider more context-aware corrections.

By addressing these areas, the tool's accuracy and performance can be significantly enhanced, making it a valuable asset for various NLP applications.