Natural Language Processing

Spell Corrector Project

Marius Mafteuta
Berkay URAS
Sude YILMAZ

Abstract

This project explores the development of a spell corrector using multiple approaches, including SymSpell, SpellChecker, contextual correction with spaCy and by using a text file as a train data for the correction. The project aims to create an efficient spell correction tool and evaluate its effectiveness. A graphical user interface (GUI) is also developed to provide an interactive platform for users.

2. Introduction

2.1 Aim

The aim of this project is to develop a spell correction tool that can effectively identify and correct misspellings in English text using different methodologies. The tool should be user-friendly and accurate, providing an intuitive interface for users to input text and receive corrected versions.

2.2 Scope

The project covers the implementation and comparison of three different spell correction techniques:

- SymSpell-based correction
- SpellChecker library-based correction
- Contextual spell correction using spaCy

Additionally, the project includes the development of a GUI to facilitate easy interaction with the spell correction tool. The scope also involves loading dictionaries, handling user input, displaying corrections, and maintaining a history of corrected text.

2.3 Methodology

The methodologies used in this project include:

- Implementation of NLP techniques by using Shakespeare's Works as a train data for the corrector.

- SymSpell: A fast and efficient spell correction algorithm that uses a precomputed dictionary and offers suggestions based on edit distances.
- SpellChecker: A library that uses a simple yet effective approach to spell correction by leveraging word frequency and edit distance.
- SpaCy with contextualSpellCheck: Utilizes the spaCy NLP framework combined with contextualSpellCheck to provide context-aware spell correction.

The project was developed using Python through JupyterNotebook by using different libraries like tkinter for GUI development, symspellpy, spellchecker, and spacy libraries for spell correction algorithms.

I. Theoretical Part

3. Theory

The spell correction problem can be approached through various methodologies, each with its advantages and limitations:

NLP Techniques

Using raw NLP techniques involves creating a custom dataset from Shakespeare's works. It involves generating a vocabulary set, calculating word frequencies, and developing functions to create potential word edits. This method applies techniques like deletion, insertion, replacement, and switching of letters to suggest corrections.

SymSpell

SymSpell uses a precomputed dictionary of known words and their frequencies, employing edit distance algorithms to find the closest match to a given misspelled word. This compact dictionary allows for fast lookup, making SymSpell suitable for real-time applications.

SpellChecker

This library uses a frequency-based approach to suggest corrections, relying on the assumption that more common words are more likely to be the intended correct form. SpellChecker also uses edit distance to generate possible corrections.

Contextual Correction with SpaCy

SpaCy is a powerful NLP framework that supports various language processing tasks. By integrating contextualSpellCheck, SpaCy can correct misspellings based on the context in which they appear. This approach goes beyond simple edit distance and frequency, considering the semantic context of words.

II. Practical Part

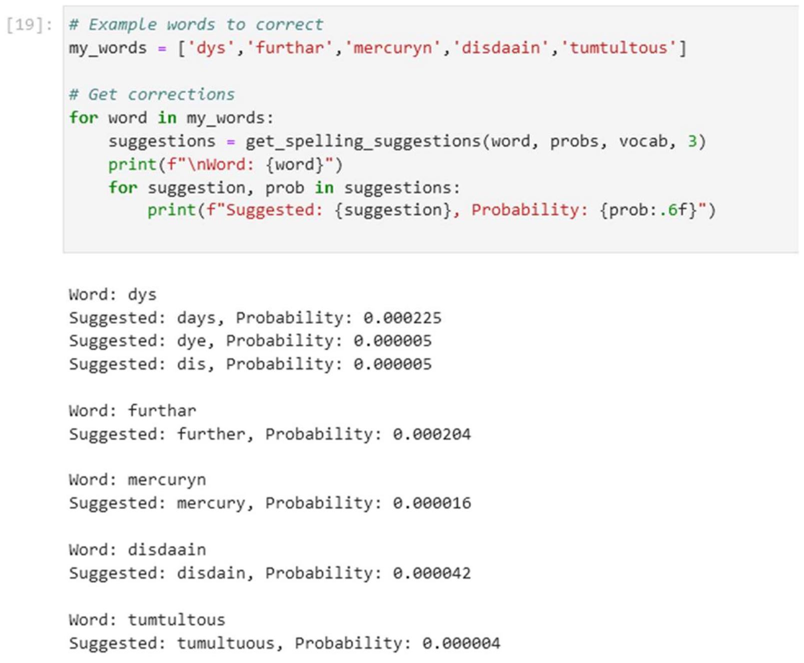
4. Implementation and Resultsù

The practical implementation of the spell correction project involved developing and integrating multiple correction methodologies and creating a user-friendly graphical user interface (GUI). This section details the specific methods used, the implementation process, and the results obtained.

NLP Techniques Using Shakespeare's Works

NLP techniques were applied to develop a spell corrector using a dataset of Shakespeare's works. The following steps outline the process:

- **Dataset Preparation**: Shakespeare's works were loaded and preprocessed to create a list of words in lowercase, forming the vocabulary.
- **Word Frequency Calculation**: A word frequency dictionary was created to calculate the probability of each word based on its frequency in the dataset.
- **Edit Distance Functions**: Functions to generate possible edits of a word (deletions, insertions, replacements, and switches) were implemented to propose potential corrections.
- **Spelling Suggestions**: The function get_spelling_suggestions was used to generate and rank possible corrections based on word probabilities and edit distances.



```
[19]: # Example words to correct
my_words = ['dys','furthar','mercuryn','disdaain','tumtultous']

# Get corrections
for word in my_words:
    suggestions = get_spelling_suggestions(word, probs, vocab, 3)
    print(f"\nWord: {word}")
    for suggestion, prob in suggestions:
        print(f"Suggested: {suggestion}, Probability: {prob:.6f}")
```

```
Word: dys
Suggested: days, Probability: 0.000225
Suggested: dye, Probability: 0.000005
Suggested: dis, Probability: 0.000005

Word: furthar
Suggested: further, Probability: 0.000204

Word: mercuryn
Suggested: mercury, Probability: 0.000016

Word: disdaain
Suggested: disdain, Probability: 0.000042

Word: tumtultous
Suggested: tumultuous, Probability: 0.000004
```

SymSpell Implementation

The correct_spellings_symspell function initializes the SymSpell object, loads the dictionary, and processes user input to provide corrections. Example usage demonstrates correcting phrases by suggesting the most likely correct forms based on the dictionary.

SymSpell is known for its fast and memory-efficient spell correction capabilities. Here's how it was implemented:

- **Initialization and Setup**: The SymSpell object is initialized with parameters like maximum edit distance and prefix length, which influence the correction process.
- **Loading the Dictionary**: A precompiled dictionary of words (frequency_dictionary_en_82_765.txt) is loaded into the SymSpell object. This dictionary contains words and their frequency counts, which help in determining the most likely corrections.
- **Correction Process**: The lookup_compound function of SymSpell is used to suggest corrections. This function considers the entire input text and provides a corrected version based on the dictionary and edit distance.

SpellChecker Implementation

The correct_spellings_spellchecker function initializes the SpellChecker object, processes user input, and outputs corrected text. It also includes functionality to handle special commands like clearing history.

- **Initialization**: The SpellChecker object is created, which includes a default dictionary of English words.
- **Correction Process**: For each word in the input text, the correction function of SpellChecker is called to find the most likely correct word based on frequency and edit distance.

Contextual Correction with SpaCy

The project incorporates a model from spaCy, enhancing it with contextualSpellCheck for context-aware corrections. Example usage highlights how the model identifies and corrects misspellings within the context of a sentence.

SpaCy, a powerful NLP framework, combined with contextualSpellCheck, provides context-aware corrections that consider the surrounding words in the text:

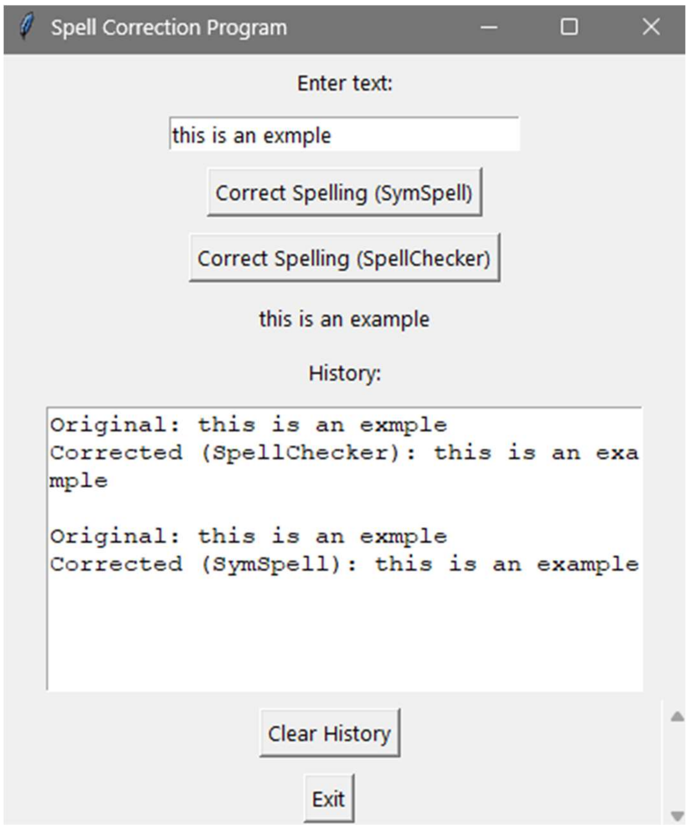
- Model Setup: The spaCy model (en_core_web_sm) is loaded, and contextualSpellCheck is added to its pipeline. This enhances the model's ability to identify and correct misspellings within the context of a sentence.

- Correction Process: Input text is processed by the spaCy model, which generates suggestions and corrections considering the context. The suggestions_spellCheck attribute provides details of the corrections made, while outcome_spellCheck gives the final corrected text.
- Example Usage: The code demonstrates the usage of spaCy for contextual corrections, highlighting how it corrects phrases and maintains the intended meaning of sentences.

```
1
{dys: 'day'}
I came home so that as I would rather participate in the function the next day.
```

GUI Development

The GUI, built with tkinter, provides an interactive platform for users to enter text and receive corrected output. It includes input fields, buttons for triggering corrections, and a display area for correction history.



```
Original: this is an exmple
Corrected (SpellChecker): this is an exa
mple

Original: this is an exmple
Corrected (SymSpell): this is an example
```

Summary

The project successfully achieved its goal of developing a versatile spell correction tool, with each implemented method demonstrating its strengths. SymSpell offers fast and efficient corrections, while SpellChecker provides a balance of simplicity and effectiveness. Additionally, SpaCy's contextual corrections enhance accuracy in complex sentences. The GUI improves the tool's usability, making it accessible and easy to use. Future improvements could involve integrating more advanced NLP techniques and expanding the dictionary to cover a wider range of words and contexts.

BIBLIOGRAPHY

- *"symspellpy 6.3.8," PyPI, https://pypi.org/project/symspellpy/6.3.8/.*

- *"pyspellchecker," PyPI, https://pypi.org/project/pyspellchecker/.*

- *"Create a Spell Check with NLP," Scaler Topics, https://www.scaler.com/topics/nlp/create-a-spell-check-with-nlp/.*