STYLOMETRIC ANALYSIS OF HETERONYMS

INTRODUCTION TO COMPUTER PROGRAMMING (PYTHON) 2023

Asil Shrara, Simone Baratella

GitHub link: https://github.com/Yolan00/ICP-Project-2023

DESCRIPTION OF THE PROJECT / **Introduction**

Our project code is designed to conduct a stylometric analysis of texts from four authors, focusing on three heteronyms of Fernando Pessoa (Alberto Caeiro, Álvaro de Campos, Ricardo Reis) and the works of Carlos Drummond for comparison. The aim is to explore Pessoa's skill in creating distinct heteronymous identities, contrasting these with the style of an independent author.

The process begins with the collection of poems from specific folders, followed by **TF-IDF vectorization** to analyse stylistic elements. The analysis is refined by excluding Portuguese stop words, allowing a focus on more substantive language elements. **Cosine similarity** measures are then applied to assess stylistic similarities and differences among the authors.

A key feature of our analysis is the use of **hierarchical clustering**, visualized through a dendrogram, which groups the poems based on stylistic similarities. This helps to differentiate the unique styles of Pessoa's heteronyms and compare them with Drummond's distinct authorial voice.

The code includes functions for counting and visualizing the most frequent words for each author, providing a visual summary of each poet's thematic focus and style.

Additionally, the analysis includes the calculation of the **stop words ratio**, **average lexical density** and three more functions for each heteronym (better discussed later on). These metrics provide insights into the complexity and richness of their vocabularies, distinguishing between the heteronyms and Carlos Drummond. Through this analysis, our project highlights the language and identity interplay in literary works, demonstrating the versatility and depth of Fernando Pessoa's heteronyms in contrast to Carlos Drummond.

DESCRIPTION OF THE PROJECT / **Background**

Our work integrates several key concepts and methodologies fundamental to stylometric analysis. The following are essential concepts that we needed to understand and apply:

- **The poets:** Fernando Pessoa was a pivotal figure in the Portuguese literary scene, contributing significantly to modernist and avant-garde movements. One of Pessoa's most distinctive literary innovations was the creation of heteronyms (distinct fictional personalities with unique voices, styles, and worldviews). These heteronyms, including Ricardo Reis, Álvaro de Campos, and Alberto Caeiro, allowed Pessoa to explore diverse perspectives within his own psyche, transforming his body of work into a rich tapestry of voices, each contributing to the broader exploration of human experience. The heteronyms, each with their own poetic identity, reveal Pessoa's intricate and multifaceted approach to artistic expression, making him an enduring and influential figure in world literature.

In expanding our literary corpus for a comprehensive analysis, we incorporated the works of Carlos Drummond de Andrade, a prominent Brazilian poet and writer. Drummond de Andrade played a pivotal role in Brazilian modernist literature and is celebrated for his profound exploration of the human condition, society, and the complexities of existence. By introducing Drummond's poems into our study alongside Fernando Pessoa's heteronyms, we aim to draw intriguing comparisons and contrasts. While Pessoa's heteronyms offer an internal kaleidoscope of perspectives within a single author, Drummond's external voice provides an external benchmark, enriching our analysis by juxtaposing Pessoa's intricate literary experimentations with the distinct style and themes of another poet who also wrote in Portuguese.

- **Stylometry:** quantitative analysis of linguistic and literary style elements to unveil patterns, authorship attribution, and other insights within written texts.

- **Text Processing in Python:** manipulating and analyzing text data using python programming techniques.

- **Natural Language Processing (NLP):** Fundamental NLP concepts, including tokenization, stop word removal, and TF-IDF vectorization.

- **Cosine Similarity and Cluster Analysis:** Techniques for measuring textual similarity and categorizing texts.

- **Statistical Measures in Textual Analysis:** Key statistical tools used in the analysis, such as stop words ratio and average lexical density.

DESCRIPTION OF THE PROJECT / **Development process**

The idea of carrying out a stylometric analysis was born mainly from curiosity and the desire to stay close to the topics covered in the master's program. We chose to focus on analysing the works of Fernando Pessoa's heteronyms due to its relevance and complexity within the realm of stylometry.

The foundation of our project was laid by Asil. Our initial task involved collecting the corpus; a dataset consisting of 20 poems from each of the four selected authors, resulting in a total of 80 texts.

We began by developing the "read_poems" function, needed for data handling and setup (with all the consequent needs related to reading and setting data). Once this groundwork was established, we started the core stylometric analysis. This phase included the creation of a list of Portuguese stop words to clean the texts and retain only the most significant words. We then applied TF-IDF vectorization, cosine similarity, and hierarchical clustering to extract meaningful insights. The construction of the dendrogram, a representation of the clustering, posed challenges, particularly in terms of visual clarity and colour labels not working. Iterations were required to optimize this aspect.

Next, we developed 5 additional functions to delve deeper into specific text analysis aspects, driven by our individual interests:

1) “count_and_save_most_frequent_words”

The **count_and_save_most_frequent_words** function analyses a collection of poems to identify and save the most 50 frequent words. It starts by merging all poems into a single string for analysis. This string is then converted to lowercase and split into individual words. If a list of stop words is provided, these are removed from the word list to focus on more meaningful words.
These top words and their frequencies are then written to a specified output file, saving this information for further use or analysis. The function returns a dictionary of these top words with their respective counts.

2) “visualize_most_frequent_words”

The **visualize_most_frequent_words** function operates by first creating a WordCloud object, customized with a specified width, height, and background colour. It then generates a word cloud using the frequencies from the **word_counts** dictionary, where each word's size in the cloud corresponds to its frequency. A matplotlib figure is set up with defined dimensions to display this word cloud. The image is rendered with bilinear interpolation for smoothness. Axes are disabled for a cleaner look, and a title is added, indicating the most frequent words for the specified author. The figure is then saved as an image file to the given path, ensuring all content is included.

3) “stop_words_ratio”

The **stop_words_ratio** function calculates the ratio of stop words in the poems. It first concatenates all the poems into a single string. This string is then converted to lowercase and split into individual words. The function counts the total number of words and separately counts the number of stop words, which are specified in the **portuguese_stop_words** list. The ratio of stop words is calculated by dividing the number of stop words by the total number of words in the text. This ratio is then returned, providing a measure of how frequently common, less informative words (stop words) are used relative to the total word count in the given text.

4) “calculate_average_lexical_density”

The **calculate_average_lexical_density** function determines the lexical density of a collection of poems by computing the ratio of unique words to the total number of words. It processes each poem in the collection, converting it to lowercase and splitting it into individual words. For each poem, it calculates the number of unique words. The function keeps a running total of both the unique words and the total word count across all poems.
After processing all poems, it calculates the average lexical density by dividing the total number of unique words by the total word count. In cases where there are no words (to avoid division by zero), it returns 0.

5) “calculate_average_word_length”

The **calculate_average_word_length** function computes the average length of words across a collection of poems. It initializes two counters: one for the total length of all words and another for the total number of words. The function then iterates through each poem in the collection. For each poem, it splits the text into individual words and calculates the length of each word. The total length of words is accumulated by summing the lengths of all words in each poem. Simultaneously, the function keeps a count of the total number of words. Once all poems have been processed, the function calculates the average word length by dividing the total length of words by the total word count. In cases where there are no words (to prevent division by zero), the function returns 0.

The final phase involved executing these functions and managing their outputs.

Although the code was now complete and functional (after having resolved all the inevitable errors that cropped up), in terms of readability and ease of use it still left a little to be desired. Simone then reworked the code by adding part 0 and creating and updating the RESULTS folder, as well as rearranging the position of the code blocks and adding more detailed comments.

Library selection and incorporation occurred incrementally, with each library chosen as needed to address specific tasks.

At the end, our code comprises four parts + 1, each serving a defined purpose:

- Part 0: Data Upload
- Part 1: Data Reading and Setup
- Part 2: Core Stylometric Analysis
- Part 3: Additional Analysis Functions
- Part 4: Additional Analysis Outputs

Each part performs a specific function necessary for the functioning of the whole but is well defined in its space to facilitate reading and possible modifications.

DESCRIPTION OF THE PROJECT / **How we divided the work**

All the code has been modified in one way or another by both of us. When one added a feature, the other could modify it to improve it or better fit the code. Despite this, some aspects of the work have been divided a little more clearly. As already mentioned, the initial idea and the structure of the core analysis are thanks to Asil. She also created 4 of the 5 auxiliary functions, of which only one was modified by Simone.

Simone put a lot of emphasis on the output/input management and optimization part, working mainly on parts 0, 1 and 4, as well as developing several iterations of the dendrogram to make it work as desired.

In general, every aspect of the work was discussed and treated by both of us thanks to the use of Google Colab which allowed us to work on the code at the same time.

The PDF file was basically written by Simone and the background was modified by Asil.

The only work carried out solely by one of the two was the creation of the README file, and the management of delivery on GitHub, but at the explicit request of Simone and in any case with the supervision of both.

FINAL NOTES/ **Bibliography**

- https://www.culturagenial.com/poemas-de-carlos-drummond-de-andrade/
- http://arquivopessoa.net/
- ICP 2023 course files
- https://stackoverflow.com/ (Every time we lost our sanity against random errors)