CORPUSCRIPT: AN AUTOMATED TEXT-CLEANING TOOL FOR CORPUS LINGUISTICS

Jhonatan Henrique LOPES Alves[122]
Ana Eliza Pereira BOCORNY[123]
Deise Prina DUTRA[124]
Carolina Godoi de Faria MARQUES [125]
Gustavo Leal TEIXEIRA[126]
Danilo Duarte COSTA[127]

INTRODUCTION

The process of corpus compilation remains a significant challenge in the field of corpus linguistics. This paper introduces CorpuScript, an innovative text-cleaning software aimed at aiding researchers in the process of corpus preparation. By combining software engineering with corpus linguistics methods, this tool can significantly improve the workflow for corpora compilation, specifically in the task of corpus cleaning.

The necessity for CorpuScript emerged from recurring challenges experienced by our research team, particularly during our current corpus research project, in which a considerable large number of texts needed to be cleaned before being used for data analysis.

Considering the pressing need for an automated solution that could improve the text-cleaning process in our research project, CorpuScript was carefully developed to help us accelerate the corpus compilation, while meeting the requirements outlined in our corpus design.

THEORETICAL FRAMEWORK

The importance of clean, well-prepared corpora in linguistic research is wellestablished in the literature. Biber, Conrad, and Reppen (1998) emphasize that corpusbased investigations rely on empirical analysis of large, principled collections of natural texts.

Notably, a standard procedure in corpus building is the conversion of the selected texts into plain text (ASCII or UTF-8), since this type of file format can be run in most corpus analysis tools, as mentioned in Ädel (2020) and Reppen (2022). However, Gries and Newman (2013, p. 263), point out that files will still “almost invariably require some editing for them to be used most effectively”. To ensure they can be read by computer software that are used for corpus analysis, Coxhead (2020, p. 470) stresses that text files “should be as clean as possible”.

In this context, corpus cleaning refers to the task of removing extraneous material from text data, such as headers, footers, special characters, line breaks,

---

[122] Undergraduate Student, Universidade Federal de Minas Gerais, Belo Horizonte, MG. Scholarship: FAPEMIG (APQ-01173-22)

[123] Professor, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

[124] Professor, Universidade Federal de Minas Gerais, Belo Horizonte, MG.

[125] Doctorate Student, Universidade Federal de Minas Gerais, Belo Horizonte, MG. Scholarhip: CAPES (n. 88887.939578/2024-00)

[126] Professor, Universidade Federal de Minas Gerais, Montes Claros, MG

[127] Professor, Universidade Federal do Vale do Jequitinhonha e Mucuri, Diamantina, MG

and other non-linguistic elements that do not contribute to the actual linguistic content (Weisser, 2016). Cleaning a corpus is a fundamental step in the corpus compilation process, since those unwanted items “may adversely affect the accuracy of the analytical procedures we intend to carry out, as well as impinging on the corpus’s representativeness” (McEnery and Brooks, 2022, p. 43).

Although cleaning many texts manually is rather time-consuming, it remains a frequently used method. For instance, when instructing English language students to use corpora for improving their learning, Poole (2018) suggests that they clean their texts by using the ‘find and replace tool’ in a text processor. Similarly, a non-automatic text cleaning approach was adopted in a study by Charles (2015) with students of English for Academic Purposes (EAP).

Automating the text cleaning process is certainly a highly welcomed advancement. To this end, according to Anthony (2020), high-level, functional programming languages are well-suited for the creation of brief, straightforward programs aimed at expediting the cleaning and processing of corpora. Languages such as Pearl, Python, and R have been extensively employed in corpus linguistics applications, encompassing tasks involved in corpus cleaning.

METHODS

CorpuScript was developed using Python, incorporating a suite of robust libraries to manage various aspects of text processing. The primary libraries utilized include:

1) Regular Expressions (“re” module): Fundamental for executing complex pattern matching and substitution operations essential for text cleaning.
2) SpaCy: A comprehensive natural language processing library employed for tasks such as tokenization, lemmatization, part-of-speech tagging, and stop word removal, thereby enhancing the linguistic accuracy of the text processing pipeline.
3) BeautifulSoup (bs4): Utilized for parsing and stripping HTML content from textual data, ensuring that only plain text is processed.
4) PySide6: Leveraged to develop the graphical user interface (GUI), facilitating user interaction and accessibility for researchers without programming expertise.

Additional Python Standard Libraries: Modules such as os, sys, unicodedata, logging, json, urllib.request, time, random, multiprocessing, and threading were integrated to handle file operations, system interactions, logging mechanisms, JSON data processing, network requests, time management, concurrency, and synchronization.

The core text cleaning functionality of CorpuScript is structured through a modular preprocessing pipeline, comprising the following key steps to standardize and prepare textual data for analysis:

HTML Stripping: Implemented via BeautifulSoup, this step removes any embedded HTML tags within the text, ensuring that only unformatted text is retained for subsequent processing.

Character Filtering: This module removes specified characters or sequences from the text based on user-defined parameters, allowing for the exclusion of unwanted symbols or tokens that may interfere with text analysis.

Diacritic Removal: Utilizing the unicodedata module, this process eliminates diacritical marks from characters, normalizing the text to its basic alphabetic form and enhancing consistency across different text inputs.

Script Filtering: Specific modules are employed to remove characters from nonLatin scripts, such as Greek and Cyrillic, maintaining a uniform character set within the corpus and eliminating potential noise from multilingual data.

Unicode Normalization: Applied using unicodedata.normalize with the NFKC (Normalization Form KC) standard, this step ensures that characters are represented in a consistent and compatible form, reducing discrepancies caused by varied Unicode encodings.

Whitespace Normalization: This process involves adjusting whitespace by removing unnecessary spaces preceding punctuation marks, standardizing spacing around punctuation, brackets, and braces, and collapsing multiple consecutive whitespace characters into a single space, thereby enhancing the readability and uniformity of the text.

Line Break Removal: By replacing newline characters with spaces, this module transforms multiline text into a continuous flow, which is beneficial for certain types of text analysis.

Bibliographical Reference Removal: Through the use of regular expressions, this step detects and removes bibliographical references embedded within the text, such as in-text citations, to focus the analysis on the main content.

Lowercasing: Converting all text to lowercase ensures uniformity, facilitating case-insensitive processing and comparison in subsequent analysis stages.

Lemmatization: Utilizing SpaCy's lemmatization capabilities, this module reduces words to their base or dictionary forms, which aids in consolidating different morphological variants of a word, thus improving the semantic consistency of the corpus.

Tokenization: This process involves splitting the text into sentences or words using SpaCy's tokenization tools, enabling more granular analysis and manipulation of the textual data.

Stop Word Removal: SpaCy's predefined stop word list is employed to filter out common, non-informative words, thereby focusing the analysis on more meaningful and content-rich terms.

Unicode Category Filtering: This module removes characters belonging to specific Unicode categories, such as superscript and subscript characters, further refining the text and eliminating potential formatting artifacts.

Regular Expression Substitutions: Advanced pattern matching and replacements are conducted using user-defined regular expressions, allowing for customizable and flexible text cleaning operations tailored to specific dataset requirements.

The preprocessing pipeline is designed to be highly modular and configurable, enabling users to selectively apply cleaning steps based on their specific research needs. Each preprocessing module is implemented as a distinct component, facilitating ease of maintenance, scalability, and the ability to extend or modify the pipeline as needed. This modular architecture ensures that CorpuScript can accommodate a wide range of text processing tasks, from simple cleaning

operations to more complex linguistic transformations, thereby supporting comprehensive corpus preparation for subsequent linguistic analysis.

Furthermore, CorpuScript's GUI, developed with PySide6, provides an intuitive interface for configuring processing parameters, selecting files or directories for processing, and monitoring progress through real-time feedback mechanisms. Concurrent processing capabilities, managed via Python's multiprocessing and threading modules, enable efficient handling of large datasets by leveraging multiple CPU cores. Logging functionalities ensure that all processing activities are meticulously recorded, facilitating debugging and audit trails.

RESULTS AND DISCUSSION

The implementation of CorpuScript has the potential to profoundly impact corpus compilation and research. The most striking advantage is the considerable reduction in time spent on pre-processing time.

A prime example of this efficiency gain was observed in our large-scale corpus research project. Initially, we estimated a six-month period for corpus preparation alone. However, with the introduction of CorpuScript midway through the project, we were able to complete the preparation phase in a matter of days, demonstrating the software's significant impact on research productivity.

The software's ability to maintain consistency across large volumes of text has also improved the quality of prepared corpora. By minimizing human error and ensuring uniform application of cleaning rules, CorpuScript can contribute to the reliability and validity of corpus-based studies.

CONCLUSION

CorpuScript represents a significant advancement in the field of corpus linguistics. By automating and streamlining the text cleaning process, it addresses long-standing challenges in corpus preparation, considerably reducing processing time, while minimizing human error and ensuring consistency across large corpora.

The software's impact extends beyond time-saving, enabling researchers to work with larger corpora and conduct more comprehensive analyses, thereby contributing to the advancement of corpus linguistics research.

As we continue to refine and expand the capabilities of CorpuScript, we invite collaboration and feedback from both the linguistic and software engineering communities. The goal is to further enhance its functionality and broaden its applicability to other scientific domains, ultimately contributing to more efficient, accurate, and comprehensive linguistic research. While the current version of CorpuScript has already demonstrated significant value, several points for future enhancement have been identified.

These future developments aim to further enhance the software's functionality, adaptability, and integration with existing research workflows, solidifying its role as an essential tool in corpus linguistics research.

AKNOWLEDGEMENTS: The authors wish to thank the following organizations for supporting and funding the research reported here: Federal University of Minas Gerais and Grant #APQ01173-22, Minas Gerais, Research Foundation (FAPEMIG).

REFERENCES

ÄDEL, Annelie. Corpus compilation. In: PAQUOT, Magali; GRIES, Stefan Th (Eds.). **A practical handbook of corpus linguistics**. Springer Nature, 2020.

CHARLES, Maggie. Same task, different corpus. In: BOULTON, Alex; LEŃKOSZYMAŃSKA, Agnieszka. **Multiple affordances of language corpora for datadriven learning**. John Benjamins 2015, p. 131-154, 2015.

GRIES, Stefan; NEWMAN, John. Creating and using corpora. In: PODESVA, Robert J.; SHARMA, Devyani (Ed.). **Research methods in linguistics**. Cambridge University Press, 2014.

MCENERY, Tony; BROOKES, Gavin. Building a written corpus: what are the basics?. In: O'KEEFFE, Anne; MCCARTHY, Michael (Ed.).**The Routledge handbook of corpus linguistics**. 2nd Edition. Routledge, 2022. p. 35-47.

POOLE, Robert. **A guide to using corpora for English language learners**. Edinburgh University Press, 2018.

REPPEN, Randi. Building a corpus: what are key considerations?. In: O'KEEFFE, Anne; MCCARTHY, Michael (Ed.).**The Routledge handbook of corpus linguistics**. 2nd Edition. Routledge, 2022. p. 13-20.