A human-in-the-loop pipeline for cleaning the South African Multilingual Learner Corpus of Academic Texts (SAMuLCAT) with XML tagging

Bronwyn J. Bowles-King
South African Centre for Digital Language Resources (SADiLaR)
North-West University (NWU)
South Africa
bronwynbowlesking@gmail.com

Abstract

Empirical analyses of student assignments require clean, structured data for human and machine handling, but raw digital submissions can contain unforeseen character-encoding artefacts, formatting inconsistencies, and boilerplate. The paper summarises the approach and outcomes of a data-cleaning exercise for a corpus of South African student writing. Python regular expressions, rule-based heuristics, and audit logs were employed to distinguish student prose and repair character corruptions. The automated stage achieved over 80% accuracy in XML tagging, with post-editing resolving edge cases. Encoding repairs account for 1.03% of the characters in the corpus, and extraneous data removed amounts to 13% of the original text.

1 Introduction

The South African Multilingual Learner Corpus of Academic Texts (SAMuLCAT) was first introduced by Carstens and Eiselen (2019) as an evidence base for language planning and development in higher education. The corpus was designed for research into academic literacy performance among students from diverse linguistic backgrounds. SAMuLCAT is a large-scale repository of university assignments, including essays and formative writing tasks, as empirical evidence of academic writing development and error patterns. The current SAMuLCAT dataset includes 395 Afrikaans assignments (158 257 words total) and 852 English assignments (457 410 words total) written by students in 2018 and 2019.

However, raw digital student submissions require cleaning before meaningful corpus-mining can be conducted. This article briefly covers the data-cleaning approach, methods, tools and results to date in preparing the texts for advanced analysis.

2 Methods

The data-cleaning approach includes correcting character-encoding issues, reconstructing broken student records, and separating original student writing from institutional data (e.g., assignment questions and prompts, submission dates, course codes, and plagiarism declarations), tables of contents, word counts, and reference lists.

Python functions, regular expressions (regexes), and Extensible Markup Language (XML)[1] were used together to tag text chunks by paragraph or single-line text. Instead of using if-else or boolean logic, for example, a state machine controlled the identification and tagging functions, with a look-ahead guard to preview text to follow before assigning a tag (Hulden et al., 2024; Käppel et al., 2021). The process was more effective when the tags were applied automatically to batches of assignments grouped by language, structure, and topic, followed by manual correction as needed.

The general XML schema applied is outlined below, although not all tags were needed for every assignment batch:

- Preamble ( ) and postamble ( ): Generally single lines of assignment or submission metadata before or after the main text, such as dates, course codes and student names, or appendices.
- Table of contents headings (<head>) and individual entries (<item>).
- Headings (<head>): Targets the student's chosen or required headings (e.g., "Introduction", "Inleiding", "Conclusion", "Slot").

[1] See Desmet et al. (2005), Ebeling and Heuboeck (2008), and Fitch et al. (2008) on the use of XML for academic texts.

This work is licensed under CC BY SA 4.0. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/deed.en
The copyright remains with the authors.

- Paragraphs (<p>): Main assignment text.

- Word count (<word count>): Various ways students included word counts (e.g., "Words 255", "Aantal woorde: 971").

- List of references heading (<head>) and individual sources (<bibl>).

- Declaration (<p>): Standardised institutional boilerplate invariably beginning with "Declaration".

Many of these text chunks and sections could be identified based on obvious trigger words (e.g., "References", "Bronnelys", names of months indicating dates, names of courses). One- to three-word preamble lines containing no trigger words often indicated student names. For quality control, tagged text chunks in an assignment batch were consolidated into a spreadsheet. This allowed a human annotator to sort, evaluate and update the tags fairly quickly while checking for text encoding artefacts and identifying unusual cases.

Exceptions generally related to irregular formatting, such as lengthy headings with terminal punctuation, atypical date formats, or fragmentary text. Once the audit spreadsheet was corrected, the tabular data was reconstructed into XML and plain text documents for future research use. The student-written headings and paragraphs were extracted for error-tagging and further analysis.

Incorrectly encoded characters are sometimes called *mojibake*. Mojibake was found in the corpus files received for processing and eventually traced back to the original consolidated XML file containing the full corpus. It is not known how the errors arose, but they may be due to imperfect data extraction and file transfers between applications and operating systems (Linux, Windows and Mac). Opening the files received with different encoding settings did not correct the problem. Errors were most apparent for quotation marks, dashes and Afrikaans diacritics. Curly or smart quotation marks and dashes may fracture into multi-character mojibake when decoded, such as â€œ and â€ for opening and closing double quotation marks. Apostrophes appear as â€™, and en- and em-dashes are sometimes rendered as â€“ and â€”.

The Afrikaans indefinite article ('n) can be represented as a single symbol, but this regularly decodes into a composite artefact (Å‰) in Windows-1252 environments. Apple's encoding standard, MacRoman, can also produce composite artefacts (Õn) for this character and for en-dashes (Ì•n) in plain text files.

To mark pronunciation and morphology, Afrikaans relies upon the circumflex (^), as in "tersiêr", and the trema (¨), as in "strategieë". Byte corruption can convert characters such as ê and ë to Ãª and Ã¯, respectively. However, mojibake is readily normalised with regexes, a mapping dictionary for multi-character sequences, or packages such as ftfy (Speer, 2019).[2]

3 Results

The target set for automatic XML tagging of the corpus by paragraph or single-line text was at least 80% accuracy overall, with up to 20% manual correction. Accuracy here refers to the percentage of automatically applied tags that did not need to be corrected compared to those that required manual correction.

For the Afrikaans assignments, 5 950 tags were assigned to text chunks and 1 169 had to be manually corrected, giving an accuracy of 80.35%. For the English assignments, 13 748 tags were assigned and 1 706 were corrected, with an accuracy of 87.59%. For the combined corpus, the result was thus 85.4% accuracy. Accuracy was sometimes below the target for larger batches of assignments with longer, more variable content. However, accuracy was often higher for shorter assignments, smaller batches under 20 assignments, and those with more consistent formatting. Further changes to the functions to increase accuracy above this at times led to diminishing returns and eventually poorer performance (Banko and Brill, 2001; Gorman and Bedrick, 2019).

About 13% of the total word count was removed after being tagged and verified as preamble, assignment instructions, tables of contents, word counts, plagiarism declarations, bibliographies, and so forth, leaving behind the majority as student-written headings and paragraphs. Encoding issues accounted for 1.03% of the corpus by character count.

[2] See also Erickson (2021) and Wartena (2025) for more on handling encoding errors.

4 Conclusion

In summary, rule-based text-cleaning heuristics, character-encoding normalisation, tabular audit logs, and XML standards were used to prepare the SAMuLCAT data for future research as an empirical window into the development of academic literacy at South African universities. The pipeline achieved an overall automatic annotation accuracy of 85.4% for the whole corpus, meeting the 80% minimum target, although the performance for the Afrikaans assignments was slightly lower than for the English. As with other digital humanities projects dealing with real-world data, the SAMuLCAT data-cleaning exercise shows that some direct human intervention is necessary. Manual auditing is unlikely to fall away entirely because the corpus must be suitable for further human interaction (Fort, 2016; Monarch, 2021).

For increased accuracy, training a machine learning model or integrating a large language model (LLM) into the process may be effective, but this has not yet been pursued due to computational limitations. After data cleaning, the text will proceed to part-of-speech and error tagging, and various natural language processing (NLP) tools can be applied to gain insight into the corpus. Opportunities to engage with the corpus and examine the results directly will remain valuable throughout this process, along with seeking new means to clean student writing more efficiently.

References

Banko, Michele, and Eric Brill. 2001. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33.

Carstens, Adelia, and Roald Eiselen. 2019. Designing a South African Multilingual Learner Corpus of Academic Texts (SAMuLCAT). *Language Matters*, 50(1):64–83.

Desmet, Christy, Ron Balthazor, Robert Cummings, Nelson Hilton, Angela Mitchell, and Alexis Hart. 2005. <emma>: Re-forming Composition with XML. *Literary and Linguistic Computing*, 20(S):25–46.

Ebeling, Signe Oksefjell, and Alois Heuboeck. 2008. Encoding document information in a corpus of student writing: the British Academic Written English corpus. *Corpora*, 2(2):241–256.

Erickson, Seth. 2021. Plain text & character encoding: A primer for data curators. *Journal of eScience Librarianship*, 10(3):e1211.

Fitch, Dale, Tashira Gibbs, Melissa Peet, Beth Glover Reed, and Richard Tolman. 2008. XML and ePortfolios. *Journal of Educational Technology Systems*, 36(3):319–333.

Fort, Karën. 2016. *Collaborative Annotation for Reliable Natural Language Processing: Technical and Sociological Aspects*. Wiley-ISTE, London.

Gorman, Kyle, and Steven Bedrick. 2019. We Need to Talk about Standard Splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2776–2786.

Hulden, Mans, Michael Ginn, Miikka Silfverberg, and Michael Hammond. 2024. PyFoma: A Python finite-state compiler module. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 258–265.

Käppel, Martin, Lars Ackermann, Stefan Schönig, and Stefan Jablonski. 2021. Language-independent look-ahead for checking multi-perspective declarative process models. *Software and Systems Modeling*, 20(1):1379–1401.

Monarch, Robert. 2021. *Human-in-the-Loop Machine Learning: Active learning and annotation with human and machine collaboration*. Manning Publications, Shelter Island, NY.

Speer, Robyn. 2019. ftfy: fixes text for you. *Journal of Open Source Software*, 4(39):1402.

Wartena, Christian. 2025. Character encoding and its importance for text resources. In *Harmonizing Language Data*, pages 17–34. De Gruyter, Berlin.