Proceedings of ParlaCLARIN III @ LREC2022 pages 35–38
Marseille, 20 June 2022

Error Correction Environment for the Polish Parliamentary Corpus

Maciej Ogrodniczuk, Michał Rudolf, Beata Wójtowicz, Sonia Janicka

Institute of Computer Science, Polish Academy of Sciences
Warsaw, Poland

maciej.ogrodniczuk@ipipan.waw.pl, michal@rudolf.waw.pl,
beata.wojtowicz@ipipan.waw.pl, s.janicka@student.uw.edu.pl

Abstract

The paper introduces the environment for detecting and correcting various kinds of errors in the Polish Parliamentary Corpus. After performing a language model-based error detection experiment which resulted in too many false positives, a simpler rule-based method was introduced and is currently used in the process of manual verification of corpus texts. The paper presents types of errors detected in the corpus, the workflow of the correction process and the tools newly implemented for this purpose. To facilitate comparison of a target corpus XML file with its usually graphical PDF source, a new mechanism for inserting PDF page markers into XML was developed and is used for displaying a single source page corresponding to a given place in the resulting XML directly in the error correction environment.

Keywords: parliamentary data, error correction, Polish

1. Introduction

The Polish Parliamentary Corpus[1] (Ogrodniczuk, 2018; Ogrodniczuk and Nitoń, 2020) contains proceedings of the Polish parliament from the last 100 years of its modern history, currently of over 800M tokens. The process of adding data to the XML corpus has been heterogeneous, ranging from almost-direct inclusion of newest born-digital data already available in clean formats (such as HTML) to tedious correction of automatically OCR-ed image-based PDF files containing older materials (before 1990).

Even though the latter have already been manually verified by human proof-readers at the time of their OCR, the process still resulted in many problems of various types, including structural errors (such as retained unnecessary header information) or typographical errors (e.g. corresponding to words present in dictionary but invalid in the given context). This motivated another correction round in a new environment, developed especially for detection and correction of errors in the corpus. Below we describe the process of analysing corpus texts, present the correction environment and various add-ons improving the proofreading work.

2. Error Candidate Detection

Two experiments have been carried out before the decision was made about the target method of error candidate detection in the corpus. Since precision of error detection seems to be the most important factor of such task, two models were tested. The language model-based, intended to verify how the newest transformer models for Polish can cope with a straightforward task requiring considerable precision, was compared to a simple rule-based model, long known for its precision.

[1]Pol. Korpus Dyskursu Parlamentarnego, see `clip.ipipan.waw.pl/PPC`.

2.1. Language Model-Based Error Candidate Detection

Language models have been successfully used for OCR post-correction for Polish e.g. at PolEval 2021 (Kobyliński et al., 2021)[2]. One of the submitted solutions, ranked second best (Wróbel, 2021), was tested in an experiment to find error candidates in the Polish Parliamentary Corpus. The solution was based on a sequence to sequence model using T5 architecture (Raffel et al., 2020) and a publicly available PLT5 LARGE language model for Polish[3]. Unfortunately, even though the model was successful in discovering and correcting such cases as two words glued together, missing or excessive spaces and several types of grammatical errors, the number of false positives (most likely caused by a different training domain) rendered its use impractical.

2.2. Rule-Based Error Candidate Detection

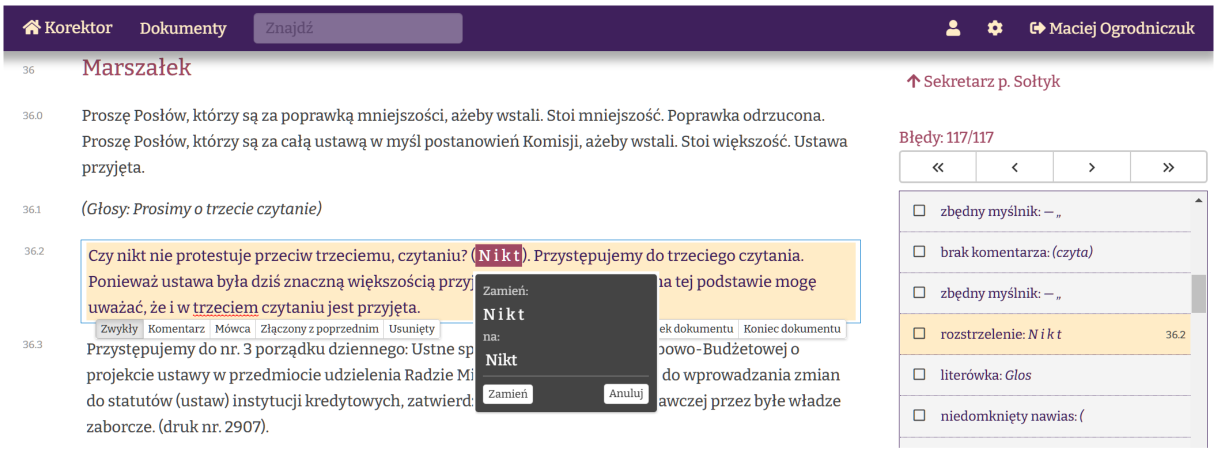
To eliminate excessive false positives, a rule-based solution was implemented. It consists of several modules intended to detect various classes of errors.

Structural errors are mostly merged enumerations or speaker names treated as normal text, leading to assigning utterances to a wrong speaker. In some cases supposed speaker labels are in fact standard text.

Comments and metadata were marked in original texts with simple brackets (e.g. *Thank you. (Applause)* which led to many conversion errors as sometimes the brackets were also used in the text, usually containing numbers or statistics, for example *(about 98%)*. This

[2]See also `http://2021.poleval.pl/tasks/task3`.

[3]`https://huggingface.co/allegro/plt5-large`



36 Marszałek

36.0 Proszę Posłów, którzy są za poprawką mniejszości, ażeby wstali. Stoi mniejszość. Poprawka odrzucona. Proszę Posłów, którzy są za całą ustawą w myśl postanowień Komisji, ażeby wstali. Stoi większość. Ustawa przyjęta.

36.1 *(Głosy: Prosimy o trzecie czytanie)*

36.2 Czy nikt nie protestuje przeciw trzeciemu, czytaniu? (N i k t). Przystępujemy do trzeciego czytania. Ponieważ ustawa była dziś znaczną większością przyj a tej podstawie mogę uważać, że i w trzeciem czytaniu jest przyjęta.

36.3 Przystępujemy do nr. 3 porządku dziennego: Ustne sp owo-Budżetowej o projekcie ustawy w przedmiocie udzielenia Radzie Mi do wprowadzania zmian do statutów (ustaw) instytucji kredytowych, zatwierd awczej przez byłe władze zaborcze. (druk nr. 2907).

Figure 1: Interface of the error correction environment: replacement suggestion for a spaced-out word

problem was solved by adding the rules concerning common comment phrases.

Punctuation errors made a separate category with many subclasses such as wrong or unmatched quotation marks or brackets, wrong types of hyphenation (minuses vs. n- or m-dashes etc.), excessively hyphenated words etc.

Broken or unfinished paragraphs mostly resulted from conversion errors but could also denote missing content.

Misspellings resulting in out-of-vocabulary strings were also quite common, particularly in older sessions which were often printed on low-quality paper. Such cases are detected using Morfeusz 2 (Kieraś and Woliński, 2017), the most efficient morphological analyser for Polish. Due to a high number of proper names, this step was limited to lowercase tokens[4].

Common OCR errors or typos corresponding to highly improbable in-dictionary words were also detected with a set of special rules. This included low-frequency words having a high-frequency orthographic neighbour, e.g. *glosowanie* (*glossing*) instead of *głosowanie* (*voting*) or rare grammatical cases, e.g. *sytemu* (*satiated* DAT) instead of *systemu* (*system* GEN) or *tyko* (*beanpole* VOC) instead of *tylko* (*only*).

Other types of errors included missing or redundant spaces, remains of non-textual elements such as tables or footnotes (which were to be removed from the proceedings), characters outside the common character set or spaced-out words.

[4]Different spelling conventions used in pre-war texts may also result in false positives (since corpus creators decided to retain the original spelling from the official parliamentary proceedings). For the correction process it means that the proofreaders have to consult the rules prevailing at the time of the sitting.

3. Error Correction Process

Annotations produced by the rule-based system were distributed to human proofreaders in a newly implemented Web-based error correction environment[5]. After logging in and selecting a text allocated for verification, they were supposed to read the text in the left/center pane (see Figure 1), consult the list of detected potential errors displayed in the right pane and correct or discard them. Apart from resolving hinted problems the proofreaders were also asked to assign speeches to speakers, distinguish the speeches from extra-textual events, mark the opening and closure of the sitting, correct annotations of misidentified comments, mark undelivered speeches etc.
The following three-step procedure was used, starting from the most to the least important issues:

1. correcting the structure of the text (distinguishing between speaker information, comments and spoken text, divided into paragraphs); errors in this layer can disrupt search results for large blocks of text so they are the most painful

2. correcting the structure of the sentence (typos, punctuation errors, hyphenation etc.): these types of errors can spoil the analysis of the sentence and cause misinterpretation of ambiguous words

3. checking the consistency of the corrections throughout the text; obviously different terms of office had different stenographers and slightly different conventions; some of them change even within a single sitting.

The interface of the error correction environment offers several functions facilitating the task such as opening the source PDF, adding general comments to the document or searching for a phrase or a certain identifier in

[5]`https://korektor.rudolf.waw.pl`, authorized access only.

the document. Sections of text selected for correction are marked in colour.

After clicking the highlighted text, a pop-up may appear with a correction suggestion (see the dark box in Figure 1). The proofreader might select to accept the proposed change or discard it. Even when there is no suggestion available, the proofreader may edit the content in place or change the structure of the text using one of the buttons (under the yellow box in Figure 1):

- Zwykły (*Plain*), i.e. the content of the speech
- Komentarz (*Comment*), used for fragments which relate to non-textual events, such as the beginning/end of the meeting, applause etc.
- Mówca (*Speaker*), marks a given passage as an identifier of the person speaking (usually their name and position)
- Złączony z poprzednim (*Merged with previous paragraph*), joins the current paragraph with the preceding one into a single, continuous text
- Usunięty (*Deleted*), deletes the paragraph.

Changes can be cancelled by clicking on the back arrow icon that appears to the right of the modified paragraphs.

The right pane provides a list of all automatically detected errors in the document. Clicking an item displays its corresponding paragraph in the center pane. Once the error has been corrected, the line is marked with a tick. Suggestions can also be ignored.

Apart from just looking at suggestions, the proofreaders were instructed to read the whole text and correct erroneous words, typos, spelling mistakes, unnecessary punctuation marks and other similar issues not detected by the rule-based system. When not certain, they were supposed to refer to the original text in the PDF file (using the integrated mechanism for locating a given text in a PDF document, see Section 5.) and keep the original spelling.

4. Inserting Page Markers into XML

In some cases the correction process requires looking into the graphical PDF source. Without knowing which page to look on, it might pose an enormous difficulty to locate the exact occurrence of the word on phrase in a multi-page, non-searchable document. This situation motivated a sub-project based on the assumption that the results of any (even considerably dirty) OCR could be, with a reasonable accuracy, compared with the clean XML text to insert page boundary markers.

In order to perform OCR, the PDF files needed to be converted into JPG format first. Then, the open source optical character recognition engine Tesseract[6] (Smith, 2007) was used to extract strings consisting of last five words of each page, which identified this page boundary. Those identifiers were subsequently stored in a list covering all the pages in a given document.

At this point, the actual procedure of inserting page boundary markers could start and each string on the list was searched for in the corresponding XML TEI file. In order to counteract possible errors resulting from imprecise text recognition, the search was fuzzy and allowed for the Levenshtein distance of six between the extracted string and the XML TEI file. When the given string was found, a page boundary marker with the page number was inserted, and the next string was searched for, starting where the previous one was found.

The XML TEI files in the Polish Parliamentary Corpus omit some parts of the original PDF files, such as tables or indices, and sometimes include blank pages. This had to be taken into account in order to maintain correct page numeration, and was successfully implemented. Moreover, pages in the PDF files frequently end with word breaks marked with a hyphen. The goal of the project was to avoid splitting words with the markers and insert them either before or after the words in question. Therefore, the need to establish whether a page ended with a whole or split word arose. In this respect, the OCR results turned out to be unreliable, as the hyphens often remained undetected in the recognized text. Therefore, the text in XML TEI files needed to be examined and the index to insert page boundary marker adjusted. As a result, the markers were effectively inserted following the word partially relegated to the next page in the PDF files.

The results of inserting page boundary markers proved satisfactory in the case of documents consisting of long chunks of text. The quality of OCR performed by Tesseract generally sufficed to detect the appropriate spot for page boundary markers. Occasional problems occurred for files of worse quality, but such an issue arising on one page did not prevent the next marker from being inserted correctly.

The following issues, however, remained unresolved:

Extremely short paragraphs In the Polish Parliamentary Corpus, the text spoken by each person is located in a different tag. Consequently, in the case of a page ending with an extremely short paragraph (e.g. a one-word statement preceded by a statement made by another person), the string identifying page boundary consists of words belonging to two different tags; it may also include the elements classified as a tag attribute in an XML TEI file rather than its content (e.g. the name of the person speaking). As the strings identifying page boundaries were searched for inside one tag at a time, in such particular cases markers with page numbers could not be inserted.

Repetitive phrases Another problem was posed by documents with numerous repetitive phrases, such as names of decrees or laws. Such files, however, were limited in length and number, and presumably do not amount to a high percentage in the whole corpus.

[6] `https://github.com/tesseract-ocr/tesseract`

5. Current Findings

The process of correcting errors in the corpus has been running for several months now so we can try to analyze its effectiveness. First of all, Table 1 presents the number of errors of various categories discovered in the subset of the corpus already assigned to proofreaders. It contains the proceedings of Sejm (lower house), with 2898 texts dated between 1919 and 2019. The vast majority of errors are related to punctuation which may result from different typing conventions (of quotation marks or brackers) but also OCR problems (hyphenation). However, the most important (from the perspective of corpus users) are structural errors, resulting in assigning utterances to wrong speakers or treating comments as spoken data.

Structural errors	71 790
unmarked speakers	32 481
enumerations	20 857
Comments and metadata	18 452
Punctuation errors	427 830
wrong quotation marks	314 772
hyphenation errors	102 103
bracket problems	6175
other punctuation problems	4780
Broken or unfinished paragraphs	121 182
Misspellings	113 170
Common OCR errors and typos	3827
Other errors	40 680
spacing problems	24 843
non-textual elements	15 720
spaced-out words	117
All errors	778 479

Table 1: Detected error counts, by class

Table 2 presents the effectiveness of rule-based error detection measured with proofreader reaction (accepted vs. ignored system suggestions). The number is a fraction of all detected errors since only approx. 30% of the assigned data is currently corrected. In our opinion the acceptance rate of errors discovered by the model seems reasonably high.

Accepted suggestions	195 416	87%
Ignored suggestions	28 486	13%
All suggestions	223 902	100%

Table 2: Error detection effectiveness

6. Looking to the Future

The environment was designed to integrate various error detection and text correction mechanisms so it inadvertently becomes the main corpus editing tool for Polish parliamentary data. One direction of its development are obviously improvements in the current error discovery, both in terms of its scope, e.g. to include detection of incomplete documents (without the formal end of the meeting) and technical capabilities, e.g. plugging in new methods of error detection capable of discovering other types of errors (e.g. syntactic errors, difficult misspellings etc.)

On the other hand, since the environment already proved to offer non-technical users the opportunity to edit corpus texts in a straightforward way, it is planned to be extended with new functions for adding longer fragments of text (confirmed to be missing) or marking up the formal structure of the meeting (agenda items).

Acknowledgements

The work reported here was financed under the 2014–2020 Smart Development Operational Programme, Priority IV: Increasing the scientific and research potential, Measure 4.2: Development of modern research infrastructure of the science sector, No. POIR.04.02.00-00C002/19, "CLARIN — Common Language Resources and Technology Infrastructure".

We would like to thank Krzysztof Wróbel for his language model-based error candidate detection experiment (see Section 2.1.).

Bibliographical References

Kieraś, W. and Woliński, M. (2017). Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego. *Język Polski*, XCVII(1):75–83.

Kobyliński, Ł., Kieraś, W., and Rynkun, S. (2021). PolEval 2021 Task 3: Post-correction of OCR Results. In (Ogrodniczuk and Kobyliński, 2021), pages 85–91.

Ogrodniczuk, M. and Kobyliński, Ł., editors. (2021). *Proceedings of the PolEval 2021 Workshop*. Institute of Computer Science, Polish Academy of Sciences.

Ogrodniczuk, M. and Nitoń, B. (2020). New Developments in the Polish Parliamentary Corpus. In Darja Fišer, et al., editors, *Proceedings of the Second ParlaCLARIN Workshop*, pages 1–4. ELRA.

Ogrodniczuk, M. (2018). Polish Parliamentary Corpus. In Darja Fišer, et al., editors, *Proceedings of the LREC 2018 Workshop* ParlaCLARIN: Creating and Using Parliamentary Corpora, pages 15–19. ELRA.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Smith, R. (2007). An Overview of the Tesseract OCR Engine. In *Proceedings of the 9th International Conference on Document Analysis and Recognition, vol. 2*, pages 629–633. IEEE Computer Society.

Wróbel, K. (2021). OCR Correction with Encoder-Decoder Transformer. In (Ogrodniczuk and Kobyliński, 2021), pages 97–102.