

Validation Workflow – *ParlLawSpeech*

Overview

This file documents the validation workflow applied to the ParlLawSpeech dataset. The purpose of the validation process is to ensure the internal consistency, completeness, and structural integrity of the collected data.

More specifically, the validation procedures are designed to ensure that:

- Observations are unique and identifiers are consistent.
- Missing data and empty text fields are systematically identified.
- Scraped documents correspond to official legislative records.
- Links between speeches, bills, and laws are correctly captured.
- Metadata fields follow consistent formatting rules.
- Text content is clean and accurately reflects the original documents.
- Potential outliers or anomalies in legislative text length are identified and inspected.

The workflow consists of six main stages:

1. General completeness and duplicate checks

- 1.1. Speeches
- 1.2. Bills
- 1.3. Laws

2. Validation of scraped data coverage

- 2.1. Speeches
- 2.2. Bills
- 2.3. Laws

3. Links between the corpora

- 3.1. Links between bills and laws
- 3.2. Links between bills, laws, and speeches

4. Validation of metadata variables

- 4.1. Speeches: party, agenda, date, speaker, chair, and period variables
- 4.2. Bills: initiator, committee, status, and period variables
- 4.3. Laws: period and adoption date variables

5. Manual inspection of text content

- 5.1. Speeches
- 5.2. Bills
- 5.3. Laws

6. (Exploratory) Outlier detection

While the steps described below reflect the general validation workflow applied across the dataset, not all procedures were required for every country. Some checks were only relevant in specific cases depending on the structure and availability of source data, and additional validation steps were performed where necessary. The workflow presented here therefore summarizes the standard validation procedures implemented across the project.

The validation was implemented using R. Examples of code snippets are provided in this document.

0 Setup and Data Loading

The validation scripts rely primarily on the **tidyverse** ecosystem and several supporting packages for text processing and visualization. A fixed random seed is defined to ensure reproducibility of all random sampling procedures used throughout the validation workflow.

```
1 # Clear memory
2 rm(list = ls())
3 options(scipen = 999)
4
5 # Load packages
6 library(tidyverse)
7 library(lubridate)
8 library(quanteda)
9 library(ggplot2)
10 library(writexl)
11 library(stringr)
12 library(ggpubr)
```

```

13
14 # Set directory paths
15 data_path <- "PATH_TO_DATA/"
16 val_path <- "PATH_TO_VALIDATION_OUTPUT/"
17
18 # Load datasets
19 speeches <- readRDS(paste0(data_path, "Corpus_speeches.RDS"))
20 laws <- readRDS(paste0(data_path, "Corpus_laws.RDS"))
21 bills <- readRDS(paste0(data_path, "Corpus_bills.RDS"))
22
23 # Set random seed for reproducible validation samples
24 set.seed(01011999)

```

1 General Completeness and Duplicate Checks

The first stage examines whether observations are unique and whether key variables contain missing values.

1.1 Speeches

Uniqueness checks are performed for the entire dataset as well as for key identifier variables.

```

1 # Check uniqueness of rows
2 nrow(speeches) == nrow(unique(speeches))
3
4 # Check duplicates in key variables
5 length(speeches$speech_ID) -
6   length(unique(speeches$speech_ID))
7
8 length(speeches$procedure_ID) -
9   length(unique(speeches$procedure_ID))
10
11 length(speeches$text) -
12   length(unique(speeches$text))

```

Missing text values and empty strings are also examined.

```

1 unique(is.na(speeches$text))
2 which(speeches$text == "")
3 min(nchar(speeches$text))

```

1.2 Bills

Similar checks are conducted for the legislative proposals dataset.

```

1 # Check uniqueness
2 nrow(bills) == nrow(unique(bills))
3
4 # Check duplicates
5 length(bills$bill_text) -
6   length(unique(bills$bill_text))
7
8 # Missing text
9 length(which(is.na(bills$bill_text)))
10 length(which(bills$bill_text == ""))

```

1.3 Laws

For adopted legislation, additional checks ensure that identifiers and texts are unique.

```

1 nrow(laws) == nrow(unique(laws))
2
3 length(laws$procedure_ID) ==
4   length(unique(laws$procedure_ID))
5
6 length(laws$law_ID) ==
7   length(unique(laws$law_ID))

```

2 Validation of Scraped Data Coverage

To ensure that all relevant legislative documents have been collected, counts of speeches, bills, and laws in the dataset are compared with official sources as well as alternative sources, where available.

Listed below are possible variations of filtering mechanisms applied to verify observation counts across different groupings.

2.1 Speeches

```

1 speeches_party_period <- speeches %>%
2   group_by(party, period) %>%
3   summarize(number_of_speeches_data = n()) %>%
4   ungroup()

```

The resulting counts are manually compared to the number of speeches that can be identified through alternative sources such as party websites or filtering options on official parliamentary databases.

2.2 Bills

```
1 bills_periods <- bills %>%
2   group_by(period) %>%
3   summarize(number_of_bills_data = n()) %>%
4   ungroup()
```

The resulting counts are manually compared to the number of bills listed on the official parliamentary website or alternative sources for each legislative period.

2.3 Laws

```
1 laws_year <- laws %>%
2   mutate(year = year(adoption_date)) %>%
3   filter(year >= START_YEAR & year <= END_YEAR) %>%
4   group_by(year) %>%
5   summarize(number_of_laws_data = n()) %>%
6   ungroup()
```

The resulting counts are compared with the number of laws listed in official legislative databases after filtering the database for the same year range.

3 Links Between the Corpora

Next, links between speeches, bills, and laws are verified.

3.1 Links Between Bills and Laws

Bills and laws are linked through shared legislative procedure identifiers. The relationship is not necessarily one-to-one, as a single bill may result in multiple legal acts and vice versa. Country-specific exceptions and differences in the temporal coverage of the corpora are also taken into account when evaluating unmatched observations.

To validate the linkage, the bill and law corpora are joined using the common procedure identifier. Where multiple procedure identifiers are stored in one observation, the identifiers are first separated into individual rows and whitespace is removed. A full join retains all observations from both corpora and therefore allows bills without an adopted law and laws without a corresponding bill record to be identified.

```
1 bill_law_links <- bills %>%
2   select(procedure_ID,
3         title_bill,
4         status,
5         initiation_date,
6         bill_text) %>%
```

```

7 separate_rows(procedure_ID, sep = ",") %>%
8 mutate(procedure_ID = str_trim(procedure_ID)) %>%
9 full_join(
10   laws %>%
11     select(procedure_ID,
12            title_law,
13            adoption_date,
14            law_text) %>%
15     separate_rows(procedure_ID, sep = ",") %>%
16     mutate(procedure_ID = str_trim(procedure_ID)),
17   by = "procedure_ID"
18 )

```

Laws without a corresponding bill record are identified through missing bill metadata in the joined dataset. These observations are manually inspected to determine whether they reflect linkage errors, country-specific exceptions, or differences in the temporal coverage of the corpora.

```

1 unmatched_laws <- bill_law_links %>%
2   filter(is.na(title_bill),
3          !is.na(title_law))

```

Bills without corresponding law records are not generally treated as linkage errors, since legislative proposals may be rejected, withdrawn, or remain unfinished at the end of the observation period. However, bills with a legislative status indicating adoption are manually inspected to verify that a corresponding law has been correctly linked. As status labels differ across parliaments, the relevant values are defined separately for each corpus.

```

1 unmatched_bills <- bill_law_links %>%
2   filter(!is.na(title_bill),
3          is.na(title_law))
4
5 adopted_without_law <- unmatched_bills %>%
6   filter(status %in% adopted_status)

```

3.2 Links Between Bills, Laws and Speeches

The linkage between speeches and legislative documents is validated by checking whether procedure identifiers referenced in speeches occur in the bill and law datasets.

As with bills and laws, these relationships are not necessarily one-to-one. A single speech may refer to multiple legislative procedures, while multiple speeches usually refer to the same legislative procedure. The specific structure of these relationships depends on the institutional context and parliamentary practices in each country.

Where multiple procedure identifiers are stored in one speech observation, the identifiers are first separated into individual rows and whitespace is removed. The resulting speech links are then

aggregated by legislative procedure. This retains both the number and dates of the parliamentary debates associated with each procedure.

```
1 speech_link_summary <- speeches %>%
2   filter(!is.na(procedure_ID),
3         procedure_ID != "") %>%
4   select(procedure_ID, date) %>%
5   separate_rows(procedure_ID, sep = ",") %>%
6   mutate(procedure_ID = str_trim(procedure_ID)) %>%
7   distinct(procedure_ID, date) %>%
8   group_by(procedure_ID) %>%
9   summarize(
10     number_of_debate_dates = n_distinct(date),
11     debate_dates = paste(
12       sort(unique(date)),
13       collapse = "; "
14     ),
15     .groups = "drop"
16   )
17
18 bill_speech_links <- bills %>%
19   select(procedure_ID,
20         title_bill,
21         initiation_date,
22         status) %>%
23   separate_rows(procedure_ID, sep = ",") %>%
24   mutate(procedure_ID = str_trim(procedure_ID)) %>%
25   full_join(
26     speech_link_summary,
27     by = "procedure_ID"
28   )
```

Speech identifiers without a corresponding entry in the bill corpus are identified through missing bill metadata in the joined data. These observations are manually inspected to determine whether they reflect incorrectly extracted identifiers, formatting inconsistencies, or country-specific exceptions.

```
1 unmatched_speech_ids <- bill_speech_links %>%
2   filter(is.na(title_bill)) %>%
3   select(procedure_ID,
4         number_of_debate_dates,
5         debate_dates)
```

Any unmatched identifiers are manually investigated with reference to the original parliamentary sources. Institutional exceptions and differences in the temporal coverage of the corpora are taken into account when evaluating these cases.

3.3 Human Validation of Linkage Quality

To complement the structural linkage checks described above, an independent manual validation of the linkage quality is performed.

For each parliament, a simple random sample of 50 legislative procedures is drawn from the bills corpus, irrespective of whether the procedures are linked to speeches or laws in the final dataset.

```
1 link_validation_sample <- bills %>%
2   select(procedure_ID,
3         title_bill,
4         initiation_date,
5         bill_link) %>%
6   slice_sample(n = 50) %>%
7   mutate(
8     status_manual = NA_character_,
9     number_of_debate_dates_manual = NA_integer_,
10    debate_dates_manual = NA_character_,
11    adoption_date_manual = as.Date(NA),
12    law_title_manual = NA_character_
13  )
14
15 write_xlsx(
16   link_validation_sample,
17   paste0(val_path,
18         "manual_link_validation_sample.xlsx")
19 )
```

The sampled procedures are provided to independent coders who have not been involved in the development of the extraction and linkage pipeline. Only the information required to locate the procedure in the official parliamentary sources is provided. The speech and law links contained in PLS are not provided to the coders.

Using the official parliamentary sources, the coders independently record the proposal status, whether and when the proposal was debated in parliament, and, where applicable, the adoption date and title of the corresponding law.

After coding, the annotated table is joined to the linkage datasets created above. The bill–speech linkage supplies the automatically derived debate dates and their number, while the bill–law linkage supplies the status and metadata of the corresponding law.

```
1 manual_validation <- read_xlsx(
2   paste0(val_path,
3         "manual_link_validation_completed.xlsx")
4 )
5
6 manual_link_comparison <- manual_validation %>%
7   left_join(
```

```

8   bill_speech_links %>%
9     select(procedure_ID,
10           status,
11           date),
12   by = "procedure_ID"
13 ) %>%
14 left_join(
15   bill_law_links %>%
16     select(procedure_ID,
17           adoption_date,
18           title_law) %>%
19     distinct(),
20   by = "procedure_ID"
21 )

```

Potential discrepancies between the independently collected information and the PLS metadata are then identified manually. All potential discrepancies are manually inspected against the original parliamentary sources. If systematic errors are identified, the corresponding extraction or linkage scripts are revised and the validation process is repeated.

3.4 Human Validation of Linkage Quality

To complement the structural linkage checks described above, an independent manual validation of the linkage quality is performed.

For each parliament, a simple random sample of 50 legislative procedures is drawn from the bills corpus, irrespective of whether the procedures are linked to speeches or laws in the final dataset. The variables included in the sample differ slightly across parliaments depending on the information available from the respective official sources.

```

1 link_validation_sample <- bills %>%
2   select(title_bill,
3         initiation_date,
4         bill_link,
5         procedure_ID) %>%
6   slice_sample(n = 50) %>%
7   mutate(
8     status_manual = NA_character_,
9     debate_date_manual = NA_character_,
10    adoption_date_manual = NA_character_,
11    law_title_manual = NA_character_)
12
13 write_xlsx(
14   link_validation_sample,
15   paste0(val_path, "manual_link_validation_sample.xlsx")
16 )

```

The sampled procedures are provided to independent coders who have not been involved in the development of the extraction and linkage pipeline. Only the information required to locate the procedure in the official parliamentary sources is provided. The automatically generated speech and law links contained in PLS are not provided to the coders.

Using the official parliamentary sources, the coders independently reconstruct each legislative procedure and record the proposal status, whether and when the proposal was debated in parliament, and, where applicable, the adoption date and title of the corresponding law.

After coding, the manually collected information is compared with the corresponding bill, speech, and law records in PLS. Bill-speech and bill-law links are assessed separately, following the linkage structure described in Sections 3.1 and 3.2. Any discrepancies are manually inspected to determine whether they reflect errors in the extraction or linkage procedures. If systematic errors are identified, the corresponding extraction scripts are revised and the validation process is repeated.

4 Manual Inspection of Text Content

To identify potential scraping errors or truncated texts, a stratified sample of documents is drawn from each parliamentary period. Extreme cases based on text length are also included in the sample.

```
1 speeches_stratified <- speeches %>%  
2   group_by(period) %>%  
3   sample_n(10) %>%  
4   mutate(word_count = str_count(text, " "))
```

The sampled texts are exported for manual inspection.

```
1 write_xlsx(speeches_stratified,  
2           paste0(val_path, "speeches_stratified.xlsx"))
```

Equivalent procedures are applied to the bill and law corpora.

This validation step is conducted as an iterative process. Random text samples are repeatedly extracted from the corpora and manually inspected to identify recurring error patterns, such as incorrectly encoded characters, remaining headers or footers, truncated texts, or cases where multiple target texts were incorrectly merged into a single observation. Identified error patterns are subsequently addressed through programmatic adjustments to the text-cleaning procedures and the process is repeated until no further systematic errors are observed in random samples.

As part of this step, sentence segmentation is also tested to ensure that the cleaned texts can be reliably tokenized for downstream text analysis. This involves comparing sentence counts obtained through regular expression-based heuristics with those produced by sentence tokenization tools

(e.g., `quanteda`). Large discrepancies are manually inspected to detect remaining formatting or cleaning issues.

5 (Exploratory) Outlier Detection

Finally, potential anomalies in the textual data are examined using simple descriptive diagnostics. These diagnostics help identify cases where the scraped or processed texts may contain remaining errors, such as truncated texts, duplicated content, or incomplete cleaning.

First, differences between bill texts and the corresponding adopted laws are examined. Large discrepancies in text length may indicate issues such as incomplete scraping or incorrect document matching.

```
1 billXlaw <- select(bills, procedure_ID, bill_text) %>%
2   right_join(select(laws,
3     procedure_ID,
4     law_text),
5   by = "procedure_ID")
6
7 billXlaw$char_law <- nchar(billXlaw$law_text)
8 billXlaw$char_bill <- nchar(billXlaw$bill_text)
9
10 billXlaw$char_diff <- sqrt(
11   (billXlaw$char_law - billXlaw$char_bill)^2
12 )
```

Large discrepancies between text lengths are flagged and inspected manually.

Additional descriptive diagnostics are used to detect unusual text patterns. For example, text length is compared with the number of punctuation characters. As punctuation generally scales with text length, strong deviations from this relationship may indicate remaining formatting errors or truncated texts.

```
1 bills$num_words <- str_count(bills$bill_text, "\\S+")
2 bills$num_punctuation <- str_count(bills$bill_text, "[[:punct:]]")
3
4 ggplot(bills, aes(x = num_punctuation, y = num_words)) +
5   geom_point() +
6   geom_smooth(method = "lm", se = FALSE) +
7   labs(
8     x = "Number of Punctuation Characters",
9     y = "Number of Words",
10    title = "Text Length vs Punctuation"
11  ) +
12  theme_minimal()
```

Another diagnostic compares the length of cleaned texts with the corresponding raw texts. Large deviations between raw and cleaned text length may indicate that important text sections were removed during the cleaning process.

```
1 num_raw_words <- sapply(strsplit(bills$text, "\\s+"), length)
2 num_cleaned_words <- sapply(strsplit(bills$bill_text, "\\s+"), length)
3
4 word_count_data <- data.frame(
5   num_raw_words = num_raw_words,
6   num_cleaned_words = num_cleaned_words
7 )
8
9 ggplot(word_count_data, aes(x = num_raw_words, y = num_cleaned_words)) +
10   geom_point() +
11   geom_smooth(method = "lm", se = FALSE) +
12   labs(
13     x = "Number of Words (Raw Text)",
14     y = "Number of Words (Cleaned Text)",
15     title = "Cleaned vs Raw Text Length"
16   ) +
17   theme_minimal()
```

An additional diagnostic compares text length with the number of sentences to verify that sentence boundaries are detected correctly.

```
1 bills$num_sentences <- str_count(bills$bill_text, "[.!?]")
2
3 ggplot(bills, aes(x = num_sentences, y = num_words)) +
4   geom_point() +
5   geom_smooth(method = "lm", se = FALSE) +
6   labs(
7     x = "Number of Sentences",
8     y = "Number of Words",
9     title = "Text Length vs Sentence Count"
10  ) +
11  theme_minimal()
```

Observations that strongly deviate from these expected relationships are manually inspected to determine whether they reflect genuine document characteristics or remaining data-processing errors.