CLIMATE CHANGE DISCOURSE IN AUSTRIAN PARLIAMENTARY DEBATES

Setting up the Database

Tools used for this step:

- XQuery for querying multiple XML files
- Db Browser SQLite for storing, counting, and visualizing the results

STEP 1: Set up a project folder

Your Folder structure should be:

project >	input >	climate_terms.xml *List of Climate Terms*
		ParlaMint-AT-listPerson.xml *List of metadata for all the politicians, included as a TEI XML file in the ParlaMint-AT corpus*
		ParlaMint-AT_1998-*.xml *All debates from the ParlaMint-AT corpus that are relevant to the query*
	queries >	*Empty Folder where the XQuery files and the SQL query files will be stored*
	output >	*Empty Folder where the SQLite database and the results of the queries will be stored*

Set up the project folder in the Oxygen TEI Editor:

- Open Oxygen XML Editor.
- Create a new project (called “Project.xpr” or else) and choose the “project” folder as project directory

STEP 2: Prepare your TEI/XML data

Because the ParlaMint corpus is already available as cleaned, well-structured TEI data, we save ourselves this step.

STEP 3: Set up XQuery

For a clean approach, two XQuery files are created:

- **CCD_setup_tables.xquery:**
  - Creates the tables
  - Inserts the previously defined climate terms in the table climate_terms

- **CCD_XML_to_SQL.xquery:**
  - Processes all XML files at once
  - Inserts debates, speakers, quotes, etc. in the tables

CCD_setup_tables.xquery (runs only once!)

- Open your project (Project.xpr) in the Oxygen XML Editor.
- Create a new XQuery-file (*File > New > XQuery*) and name it *CCD_setup_tables.xquery*. It should be located in your Project Folder.
- Copy and paste the code below into your XQuery-file:

Setup XQuery, Declare Namespace.

```
xquery version "3.1";
declare namespace tei = "http://www.tei-c.org/ns/1.0";
```

Load an external file (../input/ParlaMint-AT-listPerson.xml) into the variable $persons. The file is part of the ParlaMint-AT corpus and contains metadata about all the speakers.

```
let $persons := doc('../input/ParlaMint-AT-listPerson.xml')
```

Load an external file (../input/climate_terms.xml) into the variable $climate-terms. The file contains climate terms agreed upon by the project team including regular expressions used to narrow down the results. Additional terms may be added. The terms could also be inserted directly into the code (like this: let $climate-terms := ('Klima', 'Dekarbonisierung').

```
let $climateTermsDoc := doc('../input/climate_terms.xml')
```

Load the content within the <term> elements into the variable $terms.

```
let $terms := $climateTermsDoc//term
```

Create the tables with CREATE TABLE statements. The IF NOT EXISTS clause ensures the tables are only created if they do not already exist. The structure ensures data integrity through primary and foreign key constraints.

```
let $createTables := (
  "CREATE TABLE IF NOT EXISTS climate_terms (id INTEGER PRIMARY KEY, climate_term TEXT NOT NULL);",
  "CREATE TABLE IF NOT EXISTS debates (id INTEGER PRIMARY KEY, date TEXT NOT NULL, sitting_number TEXT, legislative_period TEXT);",
  "CREATE TABLE IF NOT EXISTS speaker (id TEXT PRIMARY KEY, first_name TEXT, last_name TEXT, party_affiliation TEXT, sex TEXT);",
  "CREATE TABLE IF NOT EXISTS quotes (id INTEGER PRIMARY KEY, quote TEXT NOT NULL, debates_id INTEGER NOT NULL, speaker_id TEXT NOT NULL, FOREIGN KEY (debates_id) REFERENCES debates(id), FOREIGN KEY (speaker_id) REFERENCES speaker(id));",
  "CREATE TABLE IF NOT EXISTS quotes_climate_terms (id INTEGER PRIMARY KEY, quotes_id INTEGER NOT NULL, climate_terms_id INTEGER NOT NULL, FOREIGN KEY (quotes_id) REFERENCES quotes(id), FOREIGN KEY (climate_terms_id) REFERENCES climate_terms(id));"
)
```

INSERT climate terms. The concat() function is used to join multiple strings into a single string. With INSERT OR IGNORE an ID will be skipped if it already exists in the table.

```
let $termInserts :=
  for $term in $terms
  return concat(
    "INSERT OR IGNORE INTO climate_terms (id, climate_term) VALUES (",
    $term/@id, ", '", normalize-space($term), "');"
  )
```

Join the two sequences into a single output. "" is an empty string, separating the table creation and insert statements. "
" represents a line break.

```
return string-join((
  $create-tables,
  "",
  $climate-term-inserts), "&amp;#10;")
```

- Once your code is finished, save your XQuery file.

- Run your XQuery over any XML file from your dataset.
  - In the project panel, right-click on any XML file you want to run your XQuery against
  - Choose *Transform > Configure Transformation Scenario…*
  - In the dialog that opens, click "New" to create a new scenario & choose "XQuery" as transformation type
  - In the scenario configuration:
    - XQUery file: Select your .xq file (your script)
    - XML Input file: Should be pre-filled with the XML file you right clicked
  - In the "Output" panel:
    - Check "Save as" and insert your output directory; the name of the output file should end with .sql (e.g. *CCD_tables.sql*).
  - Click "OK" to save the scenario

- Choose the scenario out of the list of scenarios and click on “Apply transformation” to run it

CCD_XML_to_SQL.xquery (Run over all the XML files in the ParlaMint AT corpus to be analysed)

- Open your project (Project.xpr) in the Oxygen XML Editor, if not still open
- Create a new XQuery-file (*File > New > XQuery*) and name it *CCD_XML_to_SQL.xquery*. It should be located in your Project Folder.
- Copy and paste the code below into your XQuery-file:

Setup XQuery, Declare Namespace.

```
xquery version "3.1";
declare namespace tei = "http://www.tei-c.org/ns/1.0";
```

Load external files.

```
let $persons := doc('../input/ParlaMint-AT-listPerson.xml')
let $climateTermsDoc := doc('../input/climate_terms.xml')
```

Loop over all TEI files in the input directory and extract key metadata for each parliamentary debate. For each file, the script retrieves the debate ID, date, sitting number and legislative period. These values are used to generate SQL INSERT statements for the debates table. The collection() function loads all TEI XML files matching the pattern *ParlaMint-AT_*.xml* in the specific folder. xs:integer(...) is used to ensure that the debate ID is treated as a numeric value. XPath expressions are used to find relevant elements (e.g. <meeting>) and attributes (e.g. @when). concat(...) is used to build SQL statements as strings, which will be collected and later executed or written to a file.

```
let $all-results :=
for $doc in collection('../input?select=ParlaMint-AT_*.xml;recurse=no')//tei:TEI
let $file-name := tokenize(base-uri($doc), '/')[last()]
let $matches := analyze-string($file-name,
'^ParlaMint-AT_(\d{4})-(\d{2})-(\d{2}).*-([0-9]{5})\.xml$')//fn:match
let $date := string-join(($matches/fn:group[@nr = "1"], $matches/fn:group[@nr = "2"],
$matches/fn:group[@nr = "3"]), "")
let $number := $matches/fn:group[@nr = "4"]
let $debate-id := xs:integer(concat($date, $number))
let $debate-date := $doc//tei:sourceDesc//tei:bibl//tei:date/@when/string()
let $sitzung := $doc//tei:meeting[@ana = '#parla.lower #parla.sitting'][1]/@n/string()
let $periode := $doc//tei:meeting[contains(@ana, '#parla.term') and @xml:lang = 'de'][1]/@n/string()
```

Insert debate metadata:

```
let $debate-insert := concat("INSERT OR IGNORE INTO debates (id, date, sitting_number,
legislative_period) VALUES (", $debate-id, ", '", $debate-date, "', '", $sitzung, "', '", $periode, "');")
```

Prepare quote insertion (looping over <seg> elements), extract quote text and filter by climate terms. Ignore interruptions:

```
let $quote-inserts :=
  for $seg at $i in $doc//tei:seg
  let $text := normalize-space(string-join(
    $seg//text()[
      not(ancestor::tei:vocal) and
      not(ancestor::tei:kinesic) and
      not(ancestor::tei:note)
    ], ' '))
  where some $term in $terms satisfies contains(lower-case($text), lower-case(string($term)))
```

Extract the speaker ID from the closest <u>, then find the corresponding <person> in the personList file. Extract and escape speaker attributes:

```
  let $u := $seg/ancestor::tei:u[1]
  let $speaker-id := substring-after($u/@who, '#')
  let $person := $persons//tei:person[@xml:id = $speaker-id]
  let $first := replace(string-join($person/tei:persName/tei:forename, " "), "'", "''")
  let $last := replace(string-join($person/tei:persName/tei:surname, " "), "'", "''")
  let $party := substring-after($person/tei:affiliation[contains(@ref,'#parliamentaryGroup.')][1]/@ref,
'#parliamentaryGroup.')
  let $sex := replace($person/tei:sex/@value/string(), "'", "''")
```

Prepare quote entry:

```
let $quote := replace($text, "'", "''")
  let $quote-id := $debate-id * 100000 + $i
```

Return SQL statement:

```
  return (
    concat("INSERT OR IGNORE INTO speaker (id, first_name, last_name, party_affiliation, sex) VALUES
('", $speaker-id, "', '", $first, "', '", $last, "', '", $party, "', '", $sex, "');"),
    concat("INSERT INTO quotes (id, quote, debates_id, speaker_id) VALUES (", $quote-id, ", '", $quote, "',
", $debate-id, ", '", $speaker-id, "');"),
```

Link the quote to matching climate terms:

```
  for $term at $term-id in $terms
    where contains(lower-case($text), lower-case(string($term)))
    return concat("INSERT INTO quotes_climate_terms (quotes_id, climate_terms_id) VALUES (",
$quote-id, ", ", $term-id, ");")
  )
```

Returns all SQL statements:

```
return
    (
    $debate-insert,
    distinct-values($quote-inserts)
    )
```

Combines all results into a string of SQL commands:

```
return
```

```
string-join(distinct-values($all-results), "&#10;")
```

- Once your code is finished, save your XQuery file.

- Run your XQuery
    - choose any XML file from your dataset (or any XML file at all - the file you choose will be ignored by the script anyway); with *collections(...)//tei:TEI*, all the files in the input directory that match this pattern: *ParlaMint-AT_*.xml* will be processed
    - Choose *Transform > Configure Transformation Scenario…*
    - In the dialog that opens, click “New” to create a new scenario & choose “XQuery” as transformation type
    - In the scenario configuration:
        - XQUery file: Select your script *CCD_XML_to_SQL.sql*
        - XML Input file: Insert “${currentFileURL}”
    - In the “Output” panel:
        - Check “Save as” and insert your output directory; the name of the output file should end with .sql (e.g. *CCD_XML_to_SQL.sql*).
    - Click “OK” to save the scenario
    - Choose the scenario out of the list of scenarios and click on “Apply transformation” to run it

STEP 4: Build the Database in the DB Browser for SQLite

1. Create the actual .db file & the tables

- In the DB Browser for SQLite, open a new Project & call it e.g. *CDD.db*
- To build the tables & fill the climate terms table:
  - Go to “Execute SQL”
  - Go to “Open SQL file(s)” and open *CCD_setup_tables.sql*
  - Delete this part from the code:
    <?xml version="1.0" encoding="UTF-8"?>
  - Click on “Execute all/selected SQL”
  - Close the SQL-file

2. Populate the database

- In the “Execute SQL” file, go to “Open SQL file(s)” and open *CCD_XML_to_SQL.sql*
- Delete this part from the code:
  <?xml version="1.0" encoding="UTF-8"?>
- Click on “Execute all/selected SQL”
- Close the SQL-file

3. Inspect the data visually & clean it if necessary

Because the regular expressions from the climate terms list were not fully incorporated within the XQuery implementation and several false positive entries were identified, data cleaning and refinement is needed. Additionally, some relevant terms still need to be added to the climate terms list:

```
DELETE FROM quotes_climate_terms
WHERE climate_terms_id = 17
  AND quotes_id IN (
            SELECT q.id
            FROM quotes q
            WHERE
            LOWER(q.quote) LIKE '%viktor klima%' OR
            ....
          ...
            LOWER(q.quote) LIKE '%sachliches klima%'
  );

DELETE FROM quotes
WHERE id NOT IN (
  SELECT DISTINCT quotes_id FROM quotes_climate_terms
);

INSERT INTO quotes_climate_terms (quotes_id, climate_terms_id)
SELECT q.id, 8
FROM quotes q
```

```
WHERE LOWER(q.quote) LIKE '%energieeffizient%'
  AND NOT EXISTS (
    SELECT 1
    FROM quotes_climate_terms qct
    WHERE qct.quotes_id = q.id
      AND qct.climate_terms_id = 8
  );

INSERT INTO quotes_climate_terms (quotes_id, climate_terms_id)
SELECT q.id, 18
FROM quotes q
WHERE LOWER(q.quote) LIKE '%kohlenstoffdioxid%' OR '%kohlenstoff%'
  AND NOT EXISTS (
    SELECT 1
    FROM quotes_climate_terms qct
    WHERE qct.quotes_id = q.id
      AND qct.climate_terms_id = 8
  );

INSERT INTO quotes_climate_terms (quotes_id, climate_terms_id)
SELECT q.id, 25
FROM quotes q
WHERE LOWER(q.quote) LIKE '%pariser übereinkommen%'
  AND NOT EXISTS (
    SELECT 1
    FROM quotes_climate_terms qct
    WHERE qct.quotes_id = q.id
      AND qct.climate_terms_id = 8
  );
```

It is also needed to correct speaker entries that contain redundant name variants:

```
UPDATE speaker SET first_name = 'Michael', last_name = 'Bernhard' WHERE id = 'PAD_83124';
UPDATE speaker SET first_name = 'Nikolaus', last_name = 'Scherak' WHERE id = 'PAD_83125';
UPDATE speaker SET first_name = 'Ewa', last_name = 'Ernst-Dziedzic' WHERE id = 'PAD_87146';
UPDATE speaker SET first_name = 'Petra', last_name = 'Vorderwinkler' WHERE id = 'PAD_05647';
UPDATE speaker SET first_name = 'Anna Elisabeth', last_name = 'Aumayr' WHERE id = 'PAD_00041';
UPDATE speaker SET first_name = 'Peter', last_name = 'Schieder' WHERE id = 'PAD_01613';
UPDATE speaker SET first_name = 'Daniela', last_name = 'Holzinger-Vogtenhuber' WHERE id = 'PAD_83111';
UPDATE speaker SET first_name = 'Dagmar', last_name = 'Belakowitsch-Jenewein' WHERE id = 'PAD_35468';
UPDATE speaker SET first_name = 'Pia Philippa', last_name = 'Strache' WHERE id = 'PAD_44127';
UPDATE speaker SET first_name = 'Werner', last_name = 'Amon' WHERE id = 'PAD_02819';
UPDATE speaker SET first_name = 'Martin', last_name = 'Graf' WHERE id = 'PAD_02834';
UPDATE speaker SET first_name = 'Gabriele', last_name = 'Binder-Maier' WHERE id = 'PAD_00118';
UPDATE speaker SET first_name = 'Benita-Maria', last_name = 'Ferrero-Waldner' WHERE id = 'PAD_03130';
UPDATE speaker SET first_name = 'Susanne', last_name = 'Riess-Passer' WHERE id = 'PAD_01688';
UPDATE speaker SET first_name = 'Eva', last_name = 'Glawischnig-Piesczek' WHERE id = 'PAD_08240';
UPDATE speaker SET first_name = 'Wolfgang', last_name = 'Pirklhuber' WHERE id = 'PAD_08245';
UPDATE speaker SET first_name = 'Ridi Maria', last_name = 'Steibl' WHERE id = 'PAD_02018';
```

Analysis

1. Term Frequency Analysis

Which terms dominate the climate discourse?

In Execute SQL, write & execute:

```
SELECT
   ct.climate_term,
   COUNT(qct.id) AS frequency
FROM
   quotes_climate_terms qct
JOIN
   climate_terms ct ON qct.climate_terms_id = ct.id
GROUP BY
   ct.climate_term
ORDER BY
   frequency DESC;
```

Output: Ranked list of the most frequently occurring climate terms in the debate corpus

Tipp: The Plot View (View > Plot) allows you to generate a graph (x = climate_term, y1 = frequency, y2 = year)

How did the frequency for the climate terms change?

ALL terms, by year - In Execute SQL, write & execute:

```
SELECT
   ct.climate_term,
   SUM(CASE WHEN strftime('%Y', d.date) = '1996' THEN 1 ELSE 0 END) AS freq_1996,
   SUM(CASE WHEN strftime('%Y', d.date) = '1997' THEN 1 ELSE 0 END) AS freq_1997,
   SUM(CASE WHEN strftime('%Y', d.date) = '1998' THEN 1 ELSE 0 END) AS freq_1998,
   SUM(CASE WHEN strftime('%Y', d.date) = '1999' THEN 1 ELSE 0 END) AS freq_1999,
  SUM(CASE WHEN strftime('%Y', d.date) = '2019' THEN 1 ELSE 0 END) AS freq_2019,
  SUM(CASE WHEN strftime('%Y', d.date) = '2020' THEN 1 ELSE 0 END) AS freq_2020,
   SUM(CASE WHEN strftime('%Y', d.date) = '2021' THEN 1 ELSE 0 END) AS freq_2021,
   SUM(CASE WHEN strftime('%Y', d.date) = '2022' THEN 1 ELSE 0 END) AS freq_2022
FROM
   quotes_climate_terms qct
JOIN
   climate_terms ct ON qct.climate_terms_id = ct.id
JOIN
   quotes q ON q.id = qct.quotes_id
JOIN
   debates d ON q.debates_id = d.id
GROUP BY
   ct.climate_term
ORDER BY
   freq_1996 + freq_1997 + freq_1998 + freq_1999 + freq_2019 + freq_2020 + freq_2021 + freq_2022 DESC;
```

TOP 5 TERMS with the most significant changes in frequency, comparing 1996-1999 to 2019-2022 - In Execute SQL, write & execute:

```
SELECT
   ct.climate_term,
   SUM(CASE WHEN strftime('%Y', d.date) BETWEEN '1996' AND '1999' THEN 1 ELSE 0 END) AS
freq_1996to1999,
   SUM(CASE WHEN strftime('%Y', d.date) BETWEEN '2019' AND '2022' THEN 1 ELSE 0 END) AS
freq_2019to2022,
   ABS(
      SUM(CASE WHEN strftime('%Y', d.date) BETWEEN '1996' AND '1999' THEN 1 ELSE 0 END) -
      SUM(CASE WHEN strftime('%Y', d.date) BETWEEN '2019' AND '2022' THEN 1 ELSE 0 END)
   ) AS freq_diff
FROM
   quotes_climate_terms qct
JOIN
   climate_terms ct ON qct.climate_terms_id = ct.id
JOIN
   quotes q ON q.id = qct.quotes_id
JOIN
   debates d ON q.debates_id = d.id
GROUP BY
   ct.climate_term
ORDER BY
   freq_diff DESC
LIMIT 5;
```

2. Speaker Analysis

"Who contributes most to climate discourse?" (Top 10)

In Execute SQL, write & execute:

```
SELECT
   s.first_name || ' ' || s.last_name AS speaker_name,
   s.party_affiliation,
   COUNT(DISTINCT qct.quotes_id) AS climate_quote_count
FROM
   quotes_climate_terms qct
JOIN
   quotes q ON q.id = qct.quotes_id
JOIN
   speaker s ON q.speaker_id = s.id
GROUP BY
   s.id
ORDER BY
   climate_quote_count DESC
LIMIT 20;
```

"Is there variation by party or sex?"

PARTY:

Overall - In Execute SQL, write & execute:

```
SELECT
   s.party_affiliation,
   COUNT(DISTINCT qct.quotes_id) AS climate_quote_count
FROM
   quotes_climate_terms qct
JOIN
   quotes q ON q.id = qct.quotes_id
JOIN
   speaker s ON q.speaker_id = s.id
GROUP BY
   s.party_affiliation
ORDER BY
   climate_quote_count DESC;
```

To compare entries from 1996-1999 with entries from 2019-2022, write & execute:

```
SELECT
   s.party_affiliation,
   SUM(CASE WHEN strftime('%Y', d.date) BETWEEN '1996' AND '1999' THEN 1 ELSE 0 END) AS
quotes_1996_1999,
   SUM(CASE WHEN strftime('%Y', d.date) BETWEEN '2019' AND '2022' THEN 1 ELSE 0 END) AS
quotes_2019_2022
FROM
   quotes_climate_terms qct
JOIN
   quotes q ON q.id = qct.quotes_id
JOIN
   debates d ON q.debates_id = d.id
JOIN
   speaker s ON q.speaker_id = s.id
GROUP BY
   s.party_affiliation
ORDER BY
   quotes_1996_1999 DESC;
```

SEX:

Overall - In Execute SQL, write & execute:

```
SELECT
   s.sex,
   COUNT(DISTINCT qct.quotes_id) AS climate_quote_count
FROM
   quotes_climate_terms qct
JOIN
   quotes q ON q.id = qct.quotes_id
JOIN
   speaker s ON q.speaker_id = s.id
GROUP BY
   s.sex
ORDER BY
   climate_quote_count DESC;
```

To compare entries from 1996-1999 with entries from 2019-2022, write & execute:

```
SELECT
   s.sex,
   SUM(CASE WHEN strftime('%Y', d.date) BETWEEN '1996' AND '1999' THEN 1 ELSE 0 END) AS
quotes_1996_1999,
   SUM(CASE WHEN strftime('%Y', d.date) BETWEEN '2019' AND '2022' THEN 1 ELSE 0 END) AS
quotes_2019_2022
FROM
   quotes_climate_terms qct
JOIN
   quotes q ON q.id = qct.quotes_id
JOIN
   debates d ON q.debates_id = d.id
JOIN
   speaker s ON q.speaker_id = s.id
GROUP BY
   s.sex
ORDER BY
   quotes_1996_1999 DESC;
```