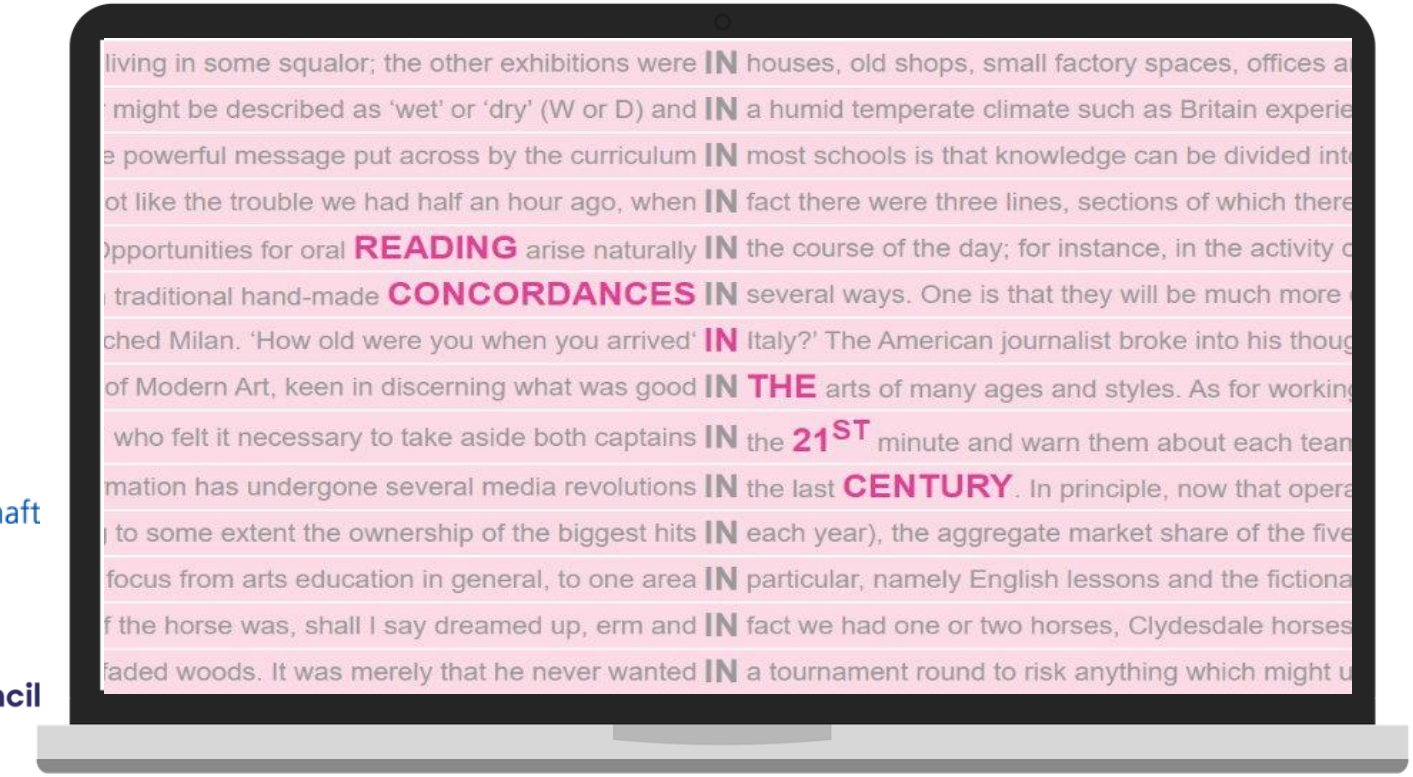
Friedrich-Alexander-Universität
Philosophische Fakultät und
Fachbereich Theologie

UNIVERSITY OF BIRMINGHAM

DFG Deutsche Forschungsgemeinschaft

UKRI Arts and Humanities Research Council



2

Reading concordances with algorithms

Konvens 2025 Tutorial | 9 Sep 2025

Reading Concordances in the 21st Century (RC21) project team

Nathan Dykes • **Stephanie Evert** • Michaela Mahlberg • **Alexander Piperski**

Overview of the tutorial

Part I

1. Concordance reading
2. Strategies for organising concordances
3. Reading concordances with algorithms
4. A mathematical framework

Part II

5. Hands-on: FlexiConc in Jupyter
6. Implementation details
7. Loading your own data

Overview of the tutorial

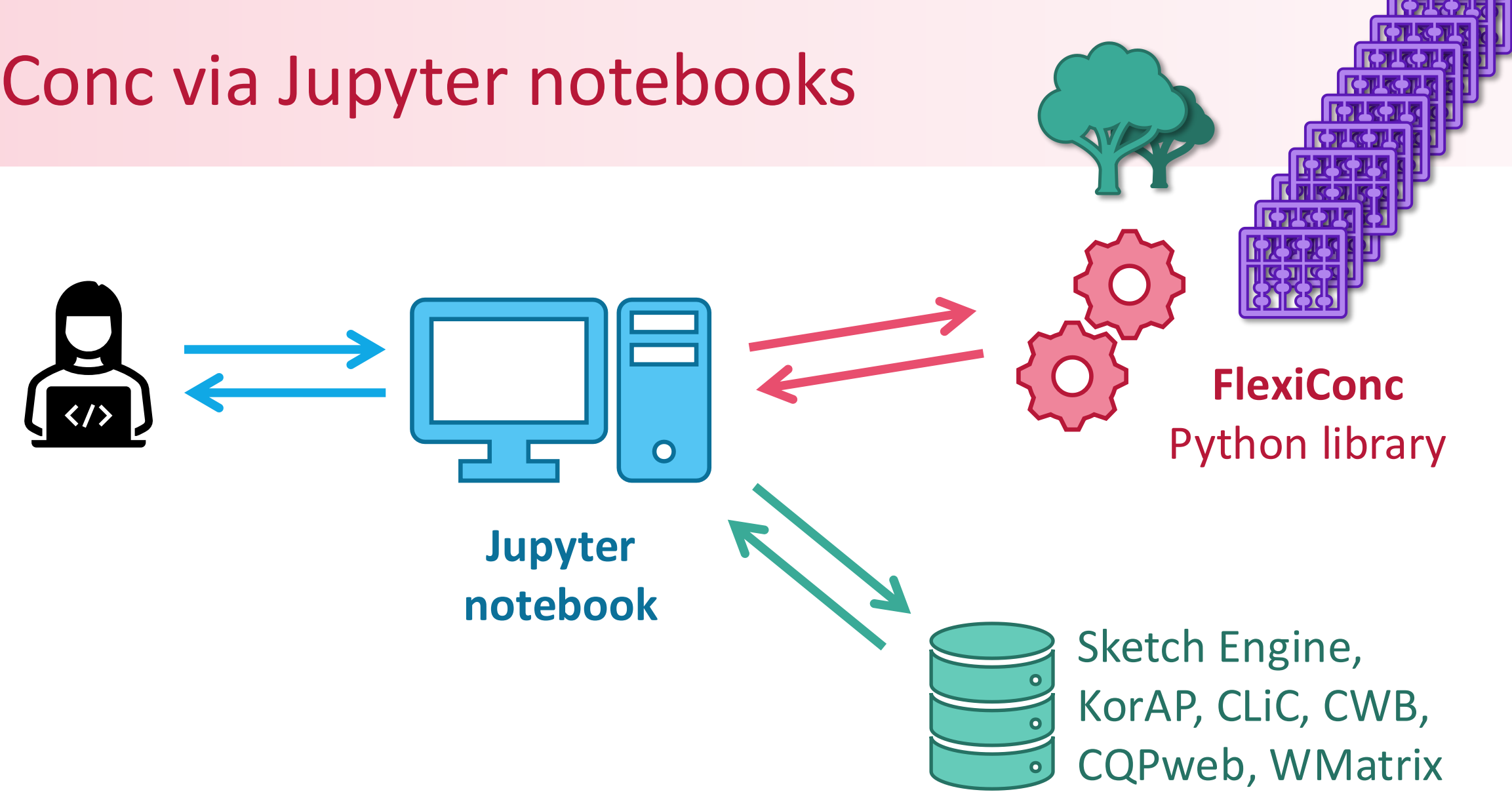
Part I

1. Concordance reading
2. Strategies for organising concordances
3. Reading concordances with algorithms
4. A mathematical framework

Part II

5. **Hands-on: FlexiConc in Jupyter**
6. Implementation details
7. Loading your own data

FlexiConc via Jupyter notebooks



https://pypi.org/project/FlexiConc/

FlexiConc via Jupyter notebooks

Why FlexiConc? ➞ benefit from algorithmic innovation!

- Use FlexiConc library from Python code in Jupyter notebook
- Combines (some) interactivity with custom programming, e.g. to carry out additional anlayses not available in concordancer
- Connect with various indexing & search backend to access your own corpora there ➞ FlexiConc has ready-made functions

- The Right Way: install Jupyter Lab + FlexiConc on your computer
- The easy way: run notebook in Google Colab … in a moment

Installing FlexiConc (Windows / MacOS)

- install Anaconda Python or miniconda (anaconda.com/download)
- create a separate environment for FlexiConc (`conda create -n FlexiConc python=3.13`)
- don't forget to activate it (`conda activate FlexiConc`)
- install PyICU from Anaconda (`conda install pyicu`)
- install JupyterLab (`conda install jupyterlab`)
- install FlexiConc and dependencies from PyPI with pip:
  ```
  pip install -U
  "flexiconc[notebooks,web,ICU,partitioning,annotation]"
  ```
- now start JupyterLab (`jupyter lab .`) … or ask us for help

Installing FlexiConc (Linux)

- create virtualenv: `python3 -mvenv --upgrade-deps venv`
- activate virtualenv: `source venv/bin/activate`
- install ICU (e.g. `sudo apt install libicu-dev` on Ubuntu)
- install JupyterLab (`pip install jupyterlab`)
- install FlexiConc and dependencies from PyPI with pip:
  ```
  pip install -U
  "flexiconc[notebooks,web,ICU,partitioning,annotation]"
  ```
- start JupyterLab in your working directory: `jupyter lab .`
- ask us for help if it doesn't work …

FlexiConc in Jupyter notebooks

- First steps

flexiconc_introduction_Konvens 2025.ipynb

Download from Google Colab or copy and use it there

https://colab.research.google.com/drive/1sAC7c0vljy6og_dSwHbvbCs3QTs4a6Lx?usp=sharing

Overview of the tutorial

Part I

1. Concordance reading
2. Strategies for organising concordances
3. Reading concordances with algorithms
4. A mathematical framework

Part II

5. Hands-on: FlexiConc in Jupyter
6. **Implementation details**
7. Loading your own data

Representation of concordances in FlexiConc

- In FlexiConc, a Concordance object must include two Pandas dataframes
  - `metadata`
    - information about concordance lines
  - `tokens`
    - information about individual tokens

Representation of concordances in FlexiConc: metadata

	line_id	doc.printed	doc.title	p.speaker
0	0	1831	Antonius und Cleopatra	ENOBARBUS
1	1	1831	Coriolanus	ERSTER BÜRGER
2	2	1832	Cymbeline	FRANZOSE
3	3	1832	Cymbeline	IMOGEN
4	4	1832	Das Wintermärchen	LEONTES
5	5	1799	Der Kaufmann von Venedig	BASSANIO
6	6	1831	Der Widerspenstigen Zähmung	SCHLAU
7	7	1890	Die beiden edlen Vettern	EMILIA
8	8	1890	Die beiden edlen Vettern	KERKERMEISTER
9	9	1832	Die lustigen Weiber von Windsor	FRAU PAGE
10	10	1832	Die lustigen Weiber von Windsor	FENTON

Representation of concordances in FlexiConc: tokens

	line_id	id_in_line	offset	word	tag	lemma	pos
125	2	10	-10	ich	PRO.Pers.Subst.1.Nom.S...	sie	p
126	2	11	-9	Euch	PRO.Pers.Subst.2.Dat.Pl.*	sie	p
127	2	12	-8	und	CONJ.Coord	und	c
128	2	13	-7	meinen	PRO.Poss.Attr.Acc.Sg.M...	meine	p
129	2	14	-6	Landsmann	N.Reg.Acc.Sg.Masc	Landsmann	n
130	2	15	-5	versöhnen	VINF.Full	versöhnen	v
131	2	16	-4	konnte	VFIN.Mod.3.Sg.Past.Ind	können	v
132	2	17	-3	;	SYM.Pun.Sent	;	x
133	2	18	-2	es	PRO.Pers.Subst.3.Nom.S...	sie	p
134	2	19	-1	wäre	VFIN.Aux.3.Sg.Past.Subj	sein	v
135	2	20	0	schade	ADJD.Pos	schade	j
136	2	21	1	gewesen	VPP.Aux.Psp	sein	v
137	2	22	2	,	SYM.Pun.Comma	,	x
138	2	23	3	wäret	VFIN.Aux.2.Pl.Past.Subj	sein	v
139	2	24	4	Ihr	PRO.Pers.Subst.2.Nom.P...	sie	p
140	2	25	5	mit	APPR	mit	i
141	2	26	6	so	ADV	so	a

Representation of algorithms in FlexiConc

- Algorithms are stored as individual .py files in the `algorithms` folder provided with the package
- Algorithms can also be loaded from user-specified folders
- Each algorithms includes an exact specification as a JSON schema, which can be updated dynamically for each node
  - name
  - description
  - algorithm type
  - arguments
  - availability of an algorithm for a node

Representation of algorithms in FlexiConc: arguments

- Schema for individual arguments
  - name
  - description
  - type
  - restrictions
    - list of possible values
    - min/max
    - …

Internal structure of an algorithm

Docstring

```
def partition_ngrams(conc, **args):
    """
    Partition concordance lines by the n-gram tuples of tokens at the specified offsets. Compare Anthony's (2018) KWIC Patterns and subsequent work.

    Parameters
    ----------
    positions : list[int]
        Offsets (relative token positions) that form the pattern.
    tokens_attribute : str, optional
        Token-attribute column used to fetch the tokens
        (default ``"word"``).
    case_sensitive : bool, optional
        Preserve original casing if *True*; otherwise tokens are converted
        to lowercase (default ``False``).

    Returns
    -------
    dict
        ``{"partitions": [ {"id": int,
                            "label": str,
                            "line_ids": list[int]}, … ]}``

    Notes
    -----
    * The ``label`` of each partition is the stringified tuple of tokens
      extracted at *positions*.
    * Partitions are ordered by descending size and, for equal sizes,
      alphabetically by their label.
    """
```

Metadata

```
32      # Metadata for the algorithm
33      partition_ngrams._algorithm_metadata = {
34          "name": "Partition by Ngrams",
35          "description": "Extracts ngram patterns from specified positions and partitions the
concordance according to their frequency in the concordance lines. Compare Anthony's (2018) KWIC
Patterns and subsequent work.",
36          "algorithm_type": "partitioning",
```

Argument schema

Dynamic list of possible values

```
37          "args_schema": {
38              "type": "object",
39              "properties": {
40                  "positions": {
41                      "type": "array",
42                      "items": {"type": "integer"},
43                      "description": "The list of positions (offsets) to extract for the ngram
    pattern."
44                  },
45                  "tokens_attribute": {
46                      "type": "string",
47                      "description": "The positional attribute to search within (e.g., 'word').",
48                      "default": "word",
49                      "x-eval": "dict(enum=list(set(conc.tokens.columns) - {'id_in_line', 'line_id',
    'offset'}))"
50                  },
51                  "case_sensitive": {
52                      "type": "boolean",
53                      "description": "If True, the search is case-sensitive.",
54                      "default": False
55                  }
56              },
57              "required": ["positions"]
58          }
59      }
```

```python
        # Step 1: Filter tokens based on the specified positions
        filtered_tokens = conc.tokens[conc.tokens["offset"].isin(positions)].copy()

        # Step 2: Apply case sensitivity
        if not case_sensitive:
            filtered_tokens[tokens_attribute] = filtered_tokens[tokens_attribute].str.lower()

        # Step 3: Aggregate ngrams by line_id as tuples of tokens at the specified positions
        ngram_dict = filtered_tokens.groupby("line_id")[tokens_attribute].apply(
            lambda x: tuple(x.tolist())
        ).to_dict()
```

```python
        # Step 4: Group line IDs by unique ngram patterns
        group_dict = {}
        for line_id, ngram in ngram_dict.items():
            group_dict.setdefault(ngram, []).append(line_id)

        # Step 5: Sort ngram patterns: first by descending size, then alphabetically by their string representation.
        sorted_group_dict = {
            str(ngram): line_ids
            for ngram, line_ids in sorted(
                group_dict.items(),
                key=lambda item: (-len(item[1]), str(item[0]))
            )
        }
```

```python
        # Step 6: Format the output as a list of dictionaries with an "id" for each group.
        result = {"partitions": [
            {
                "id": idx,
                "label": label,
                "line_ids": line_ids
            }
            for idx, (label, line_ids) in enumerate(sorted_group_dict.items())
        ]}

        return result
```

Creating custom algorithms should be doable!

Executing algorithms

- Algorithms are applied to subsets of concordance lines and produce information about:
  - selected lines;
  - ordering of lines;
  - grouping of lines;
  - token spans to be highlighted in the concordance view.
- Algorithms can only be accessed via `add_subset_node` and `add_arrangement_node`, ensuring that:
  - only permissible algorithm combinations are used;
  - any algorithm call is documented in the analysis tree;
  - the results of algorithm execution are stored correctly.

Overview of the tutorial

Part I

1. Concordance reading
2. Strategies for organising concordances
3. Reading concordances with algorithms
4. A mathematical framework

Part II

5. Hands-on: FlexiConc in Jupyter
6. Implementation details
7. **Loading your own data**

FlexiConc in Jupyter notebooks

- Loading your own data:

flexiconc_import_Konvens 2025.ipynb

Download from Google Colab or copy and use it there

https://colab.research.google.com/drive/1HuVRl748lWe65Mzl5HNge4Wwgq5xiqw2?usp=sharing

- After you've learned how to load your own data, explore a concordance of your choice. Then we'll discuss your experiences and results.

Import from CLiC

https://clic-fiction.com/

- Preparation: none

```
C = Concordance()
C.retrieve_from_clic(query=["head"], corpora="dickens")
```

Import from Sketch Engine

https://app.sketchengine.eu/

- Prerequisite: paid SkE account (free 30-day trials available)

Import from Sketch Engine

https://app.sketchengine.eu/

MANAGE CORPUS

TTA

Get more space

My account
My Sketch Engine
Settings
Local administration
Logout

CORPUS: TTA (English)

Trump Twitter Archive

Browse
View documents and folders, edit metadata

Delete
Remove corpus permanently

New corpus
Create new corpus

Stephanie Evert (ID: 80014)

SUBSCRIPTION & INVOICING | CHANGE PASSWORD

Username	SEvert
E-mail	stephanie.evert@fau.de
Account type	Multi-user account
Group account end date	April 8, 2026
Corpus storage used (words)	4,098,805 of 100,000,000 (4%)
Academic user	yes
Sketch Engine API key	78b4f7fd6c3e75e62469f535d88fcd69
	Generate new key
Account admins Admins can add or pause users and change their storage space.	Stephanie Evert

CLOSE

Import from Sketch Engine

https://app.sketchengine.eu/

https://app.sketchengine.eu/#ca?corpname=user%2FSEvert%2Ftta

TTA user/SEvert/tta created May 7, 2025 at 8:52:32 PM

Trump Twitter Archive

MANAGE CORPUS | MANAGE SUBCORPORA | COMPARE CORPORA | TEXT TYPE ANALYSIS

GENERAL INFO

Language: English

CORPUS DESCRIPTION & BIBLIOGRAPHY

TAGSET

WORD SKETCH GRAMMAR

TERM GRAMMAR

COUNTS

Tokens	1,359,510
Words	1,036,092
Sentences	377
Documents	56,570

TEXT TYPES

TEXT TYPE ANALYSIS

<doc> (2)	13
File ID , doc.id	13
File name , doc.filename	13
<text> (0)	56,570
<g> (0)	27,609
<s> (0)	377

LEXICON SIZES

word?	75,630
tag	63
lempos?	49,323
pos	9
lemma	46,084
lempos_lc	46,180
lemma_lc	42,597
lc	64,983

COMMON TAGS

adjective	J.*
adverb	RB.?
conjunction	CC
determiner	DT
noun	N.*
numeral	CD
particle	RP
preposition	IN
pronoun	PP.?
verb	V.*

All tags

Import from Sketch Engine

https://app.sketchengine.eu/

- Prerequisite: paid SkE account (free 30-day trials available)
- Generate API access token (as shown on previous slides)
- Note down full path to desired corpus (as shown on previous slides)

```
C = Concordance()
C.retrieve_from_sketchengine(
    query='[lc="fake"] [lc="news"]',
    corpus="user/SEvert/tta", # insert your path here
    api_key="[YOUR API KEY]")
```

Import from CQPweb

https://corpora.linguistik.uni-erlangen.de/cqpweb/

Your query "water and sanitation" returned 669 matches in 424 different texts (in 409,134,520 words [1,956,223 texts]; frequency: 1.64 instances per million words)

[0.107 seconds - retrieved from cache]

|< << Go!

- ✓ Choose action...
- New query
- Thin...
- Frequency breakdown
- Distribution
- Dispersion
- Sort
- Collocations...
- Download...
- Categorise...
- Save current query result...

Detailed output options

Formatting options

Choose operating system on which you will be working with the file:	UNIX (incl. Mac OS X & iOS/Android)
Print short handles or full values for text categories:	full values
Mark query results as **<<< result >>>**:	No
Size of context:	20 words each way
Download both tagged and untagged version of your results:	Yes
Write information about table columns at the beginning of file:	Yes - column headings
Format of output - KWIC or line:	KWIC
Include sub-text region boundary markers:	Yes
Include corpus positions (required for re-import)	Yes
Include URL to context display	Yes
Enter name for the downloaded file:	CQPweb_WaterSanitation_ParlUK

Please tick the text metadata categories that you want to include in your download:

Method:	Download text metadata ticked below
Select from available text metadata:	☐ Agenda ☐ Chair? ☑ Date ☐ Month/Year ☑ Party ☐ Party Facts ID ☐ Speaker ☐ Speech number ☐ Terms ☑ Year

No.	T
1	c_ 1988
2	c_7 1990
3	c_8 1990
4	c_1 1990
5	c_1 1990
6	c_1 1990
7	c_1 1990
8	c_1 1991

upporte … and some s

my righ … the trade

speciall … nes to imp

aring fo … New claus

. Ten years ago only 40 per cent . of the world popu

developing countries . That is why , in 1988 alone ,

ite around that cause . At a time when we are consi

Red Cross says that , unless Iraq immediately rece

Import from CQPweb

https://corpora.linguistik.uni-erlangen.de/cqpweb/

- Demo account credentials:
  username: **studentN**, N ∈ {1, 2, …, 15} password: **erlangen**
- Obtain desired concordance in CQPweb and save it with download action (as shown on previous slide)
- Make sure to include all relevant metadata
- Save the downloaded file in the same directory as the Jupyter notebook

```
C = Concordance()
C.load_from_cqpweb_export("CQPweb_WaterSanitation_ParlUK.txt")
```

Import from WMatrix

https://ucrel-wmatrix7.lancaster.ac.uk/

- Tag Wizard: Create your own corpus from text files in ZIP archive
- Annotated with POS, lemma, semantic concept in 9 languages
- Corpus ("folder") can be downloaded in SQLite format

Import from WMatrix

https://ucrel-wmatrix7.lancaster.ac.uk/

- Use own corpus ("folder") or copy `ESSLLI_Water_ParlUK` from library
- Main function: keyword analysis (for word, lemma, POS, concept)
- Note interesting keywords → concordance analysis in FlexiConc

Import from WMatrix

https://ucrel-wmatrix7.lancaster.ac.uk/

- Get a free WMatrix account (or use our demo account)
- Demo account credentials:
  username: **demo1@esslli.2025** password: **u73ripee4y**
- Download complete annotated WMatrix corpus
- Then use query to obtain concordance for desired keyword

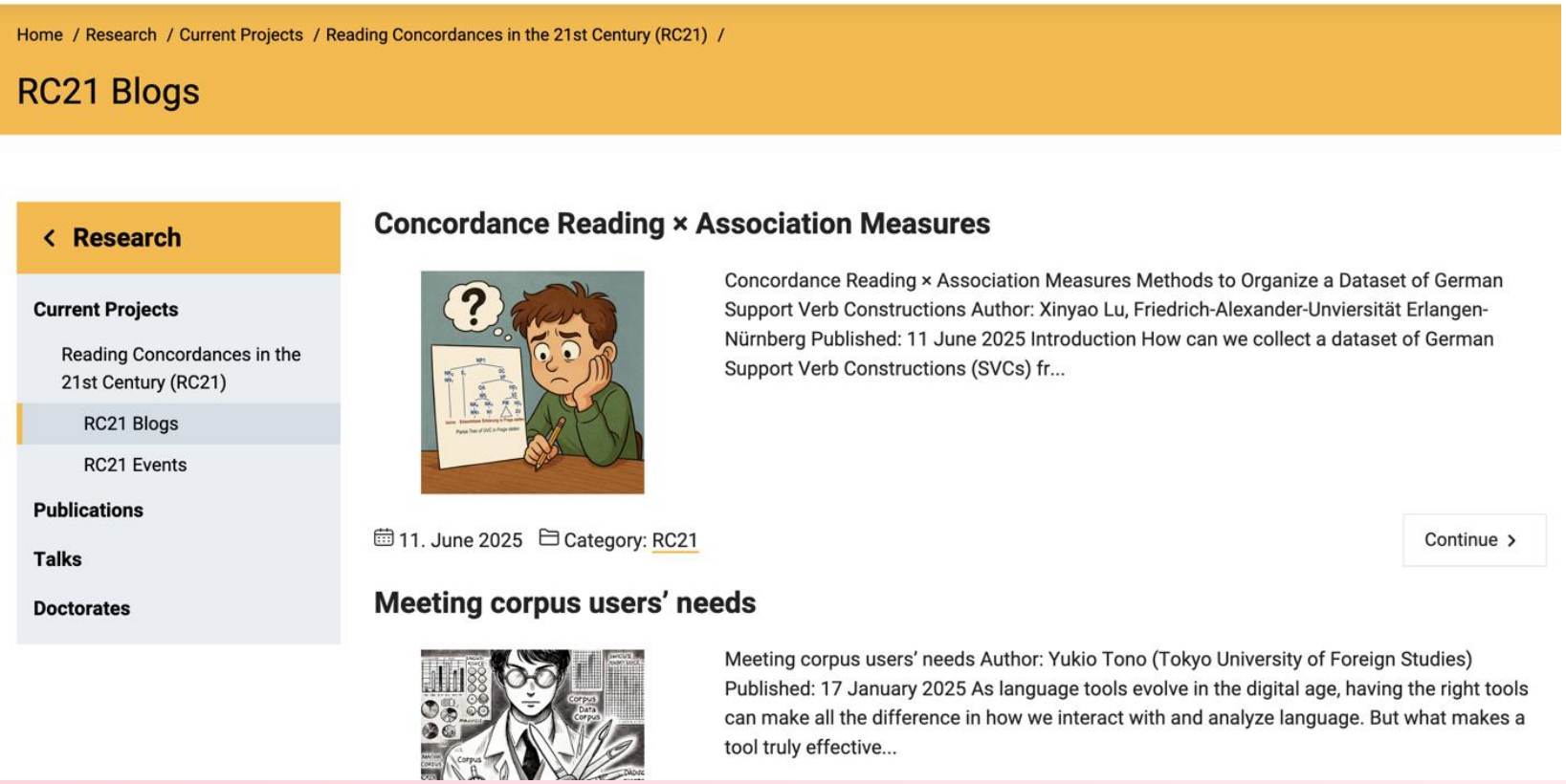
```
labour2005 = wmatrix.load(
    corpus_name="LabourManifesto2005",
    username="[USER]", password="[PASSWORD]",
    db_filename="labour2005.db")
C = labour2005.concordance_from_query("community")
```

Import your own files

```
from flexiconc import TextImport
T = TextImport()
T.load_files(
    paths=["data"],
    shorten_paths=True,
    db_name="db.sqlite",
    use_spacy=True,
    lemma=True,
    pos=True,
    tag=True
)
C = T.concordance_from_query("stomach")
```

Feedback and contributions welcome!

Please fill in our feedback form for this tutorial!
https://forms.gle/tRRVRbQHA4Qa2iFeA



Get in touch with us to contribute to our project blog!

- experience with FlexiConc
- concordance reading for your own research
- needs/ideas for algorithms