The IMS Open Corpus Workbench (CWB)

DH 1: Sprache und Text

Philipp Heinrich & Stephanie Evert

Lehrstuhl für Korpus- und Computerlinguistik
Friedrich-Alexander-Universität Erlangen-Nürnberg

Erlangen, den 26.07.2022

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

PHILOSOPHISCHE FAKULTÄT
UND FACHBEREICH THEOLOGIE

Korpuslinguistik und CWB

https://cwb.sourceforge.io/

- wichtigste Methoden der **Korpuslinguistik**:
  - **Konkordanz** (KWIC)
  - **Keywords**
  - **Kollokationen**
- IMS Open Corpus Workbench (**CWB**): Open-Source Tools zur Indexierung und Suche in großen linguistisch aufbereiteten Textkorpora
- Hauptkomponente: **Corpus Query Processor** (CQP), verarbeitet Anfragen in einer Corpus Query Language (CQL, alias CQP-Syntax)
- zusätzlich: Tools zur Enkodierung, Indexierung, Komprimierung, Dekodierung, Erstellung von Häufigkeitslisten und Satzalignment
- CWB ist Backend für Softwarepaketen wie **CQPweb**, **polmineR** und **cwb-ccc**
  - diese Tools implementieren z.B. Keyword- und Kollokationsanalysen

1. Einführung
   - Hintergrund
   - Installation

2. Indexierung von Korpora
   - Datenmodell
   - Kochbuch

3. Verwendungsbeispiele
   - CQP & CWB-Kommandozeilentools
   - Python: `cwb-ccc`

CQLs & CQLF

- CQP-Syntax ist *eine* mögliche Anfragesprache

  → dokumentiert im CQP Query Language Manual
- verschiedene Softwarelösungen nutzen unterschiedliche Anfragesprachen:
  - CQP (CQPweb, SketchEngine)
  - AQL (ANNIS)
  - COSMAS II QL (DeReKo)
- manche Resourcen sind in der Praxis nur über Software verfügbar, die *eine spezielle* CQL verwendet (z.B. DeReKo, DWDS-Korpora, . . .)
- Anfragesprachen sind unterschiedlich ausdrucksstark:
  - finde alle Wörter, die auf *-ität* enden
  - finde alle Sätze, die sowohl *COVID* als auch *Bayern* enthalten
  - finde alle Substantive, die durch *gut* modifiziert werden

  → Arbeit an einer Corpus Query Lingua Franca (CQLF, ISO 24623)

CWB: Geschichte

Work in Progress (alias *the Duke Nukem of corpus query tools*)

1993-2004 Entwicklung am IMS, Universität Stuttgart

2005 Code als Open-Source veröffentlicht (GNU GPL)

2001-2010 Weiterentwicklung und Refactoring zu einer stabilen Version

2010 Relase v3.0

seit 2010 parallele Arbeit an verschiedenen Features

- v3.1: Windows-Support
- v3.2: Unicode-Support (UTF-8)
- v3.4: Integration und Erweiterungen → stabile Version

ca. heute Release v3.5 LTS

- keine weiteren Features geplant → Ziggurat-Projekt

Installation: Quellen

- CWB v3.5 verfügbar über CWB Homepage

`https://cwb.sourceforge.io/`

  - Download z.Zt. noch direkt von `https://sourceforge.net/projects/cwb/files/cwb/cwb-3.5/`

- Quellcode zur lokalen Kompilation

`cwb-3.5.0-src.tar.gz`

- Binärpakete für Debian/Ubuntu, RedHat, MacOS (Intel, Arm) und Windows, z.B.

`cwb-3.5.0-macos-10.13-x86_64.tar.gz`

Installation: Linux

- Debian / Ubuntu

```
$ dpkg -i cwb_3.5.0-1_amd64.deb
$ dpkg -i cwb-dev_3.5.0-1_amd64.deb
```

- RedHat-Varianten

```
$ rpm -i cwb-3.5.0-1.fc35.x86_64.rpm
$ rpm -i cwb-devel-3.5.0-1.fc35.x86_64.rpm
```

Installation: MacOS

- Standard: Paketmanager HomeBrew (`https://brew.sh/`)

```
$ brew install cwb3
```

  - momentan noch mit `$ brew install cwb3 --HEAD`

- Alternativ: selbständige Binärpakete (keine Installation notwendig)
  - Intel 64-bit: `cwb-3.5.0-macos-10.13-x86_64.tar.gz`
  - Apple Silicon: `cwb-3.5.0-macos-11.0-arm64.tar.gz`

  - einfach irgendwo entpacken und das Unterverzeichnis `bin/` zu `$PATH` hinzufügen

Installation: Perl API

- CWB/Perl API enthält auch einige nützliche Kommandozeilen-Skripte
  - `cwb-make`, `cwb-regedit`, `cwb-convert-to-utf8`, `cwb-align-import`, ...
- Voraussetzung: `cwb-config` kann von Kommandozeile aufgerufen werden
  - Test mit `$ cwb-config -v`

```
$ cpan CWB CWB::CL
```

  - Konfiguration beim ersten Aufruf von `cpan` → einfach Defaults akzeptieren
  - speziellere Module: `$ cpan CWB::CQI CWB::Web`

Installation: Quellcode selber kompilieren

- Quellcode-Archiv von CWB-Homepage herunterladen und entpacken
- `INSTALL` gibt detaillierte Anweisungen, um Quellcode unter Unix zu kompilieren (siehe auch `INSTALL-MACOS` und `INSTALL-WIN`)
  1. einige benötigte Bibliotheken (sog. *dependencies*) müssen zuerst mit einem Paketmanager oder von Hand installiert werden
  2. `PLATFORM` und `SITE` in `config.mk` anpassen (oder `local_config.mk` anlegen)
  3. `$ make all`
  4. `$ make install`
- Installationsskript für gängige Linux-Distributionen:
  1. `$ sudo ./install-scripts/install-linux`
- Üblicher Installationspfad ist `/usr/local`
  - globale Corpus Registry dann unter `/usr/local/share/cwb/registry`

Dokumentation & Hilfe

`https://cwb.sourceforge.io/documentation.php`

- **CQP Query Language Manual**
  - vollständige Dokumentation der CQP-Syntax (→ auch für CQPweb)
  - Benutzung von CQP als interaktives Suchwerkzeug
- **Corpus Encoding Manual**
  - Enkodierung, Indexierung, ... von Korpora
  - Häufigkeitslisten, N-Gramme, Satzalignment

CWB Mailing List mit vielen netten Leuten (auch Andrew & Stephanie)

`http://devel.sslmit.unibo.it/mailman/listinfo/cwb`

1. Einführung
   - Hintergrund
   - Installation

2. Indexierung von Korpora
   - Datenmodell
   - Kochbuch

3. Verwendungsbeispiele
   - CQP & CWB-Kommandozeilentools
   - Python: `cwb-ccc`

Datenmodell

corpus position	word form	ID	part of speech	ID	lemma	ID
(0)	<text> value = “id=42 lang="English"”					
(0)	<text_id> value = “42”					
(0)	<text_lang> value = “English”					
(0)	<s>					
0	An	0	DET	0	a	0
1	easy	1	ADJ	1	easy	1
2	example	2	NN	2	example	2
3	.	3	PUN	3	.	3
(3)	</s>					
⋮	⋮					
(13)	</text_lang>					
(13)	</text_id>					
(13)	</text>					

<text id="i82764a3ab82bf0d5aa0e2c8506c22dc0" yw="2020_week21" duplicated="duplicate" retweet_status="False" counts="7538">
<tweet id="1262866103660883970" ymd="20200519" retweet_status="False" cohort_idx="i82764a3ab82bf0d5aa0e2c8506c22dc0" duplicated="duplicate" in_reply_status="False" cohort_clear="False_duplicate_2020_week21" ymd_hms="20200519_220156" screen_name="Host24Space" created_at="Tue May 19 22:01:56 +0000 2020" retweet_source="nan" year="2020" yw="2020_week21" nr_retweets="0" ym="202005" unix_time="1589925716" in_reply_source="nan" lang="de" in_reply_user="nan" user_idx="1239490346691760128">

<s>
Coronavirus NN
: $.
16.273 CARD
Fälle NN
in APPR
Österreich NE
– $(
Neues ADJA
Volksblatt NN
https://t.co/nHcWiY8Nol URL
</s>

</tweet>
<tweet id="1262866369885937664" ymd="20200519" retweet_status="False" cohort_idx="i82764a3ab82bf0d5aa0e2c8506c22dc0" duplicated="duplicate" in_reply_status="False" cohort_clear="False_duplicate_2020_week21" ymd_hms="20200519_220300" screen_name="Konvolutius" created_at="Tue May 19 22:03:00 +0000 2020" retweet_source="nan" year="2020" yw="2020_week21" nr_retweets="0" ym="202005" unix_time="1589925780" in_reply_source="nan" lang="de" in_reply_user="nan" user_idx="1136297640407248897">

<s>
Ein ART
#EU HST
#Konjunktuprogramm HST
nach APPR
Beendigung NN
der ART
#Coronakrise NN
in APPR
Forschung NN
+ $(
Bildung NN
? $.
</s>
<s>
Wo PWAV
ist VAFIN
der ART
Zusammenhang NN
? $.
</s>
:

```
<corpus>
<text id="t737732567194030080" source_day="dunknown" type="thread" resolution="unresolved">
<linear_thread>
<tweet id="t739608546355118081" timestamp="20160606_000313" screen_name="gareth91550033" thread_resolution="unresolved" in_reply_status="1" nr_duplicates="1" in_reply_source="737732567194030080" duplicate_status="0" thread_source="737732567194030080" nr_rts_db="0" duplicate_source="739608546355118081" nr_replies="1" user_idx="1133674159" ymd="20160606" language="en">
@JohnHWolfe     USR     @       @johnhwolfe     @
<intj>
well    UH      !       well    !
</intj>
<np>
you     PRP     P       you     O
</np>
<vp>
won't   MD      V       won't   V
win     VB      V       win     V
</vp>
and     CC      &amp;   and     &amp;
<np>
you     PRP     P       you     O
</np>
<advp>
no      RB      D       no      D
why     UH      R       why     R
</advp>
!!!!!   .       .       !!!!!   ,
</tweet>
<tweet id="t739750760779546624" timestamp="20160606_092819" screen_name="JohnHWolfe" thread_resolution="unresolved" in_reply_status="1" nr_duplicates="1" in_reply_source="739608546355118081" duplicate_status="0" thread_source="737732567194030080" nr_rts_db="0" duplicate_source="739750760779546624" nr_replies="0" user_idx="1729685305" ymd="20160606" language="en">
@gareth91550033 USR     @       @gareth91550033 @
<np>
#Remain PRP     V       #remain V
</np>
<vp>
Do      VBP     V       do      V
indeed  RB      R       indeed  R
know    VB      V       know    V
</vp>
<advp>
why     WRB     R       why     R
</advp>
<np>
we      PRP     P       we      O
brexit_v20190522.vrt.gz
```

Indexierung

1. Vertical Text Format in CWB-Binärformat konvertieren:

```
$ cwb-encode -d $data -R $regfile -f $infile
  -xsB -c utf8
  -P lemma ...
  -S text:0+id+...-S tweet:0+id+...-S vp ...
```

2. Indexierung & Komprimierung (Perl API)

```
$ cwb-make -M 4096 -V CORPUS
```

Es bietet sich an, für jedes Korpus ein *Indexierungsskript* zu erstellen.

- am bequemsten als Perl-Skript mit `CWB::Encoder`

Indexierung

1. Datenverzeichnis erstellen

   ```
   $ mkdir /path/to/cqp/data/CORPUS
   ```

2. Indexierungsskript aufrufen

   ```
   $ ./CORPUS-encode.sh
   ```

3. Hat's geklappt?

   ```
   $ cwb-describe-corpus -s CORPUS
   ```

Zum Ausprobieren

- `holmes.sh`
  Shell-Skript zur Indexierung eines kleinen Beispielkorpus
- `holmes.vrt`
  Eingabedatei im Vertical Text Format
- bei Aufruf von CQP, ... immer `-r $registry` angeben

- Ein größeres, bereits indexiertes Beispielkorpus für CQP-Übungen

```
https://cwb.sourceforge.io/temp/Dickens-1.0.tar.gz
```

1. Einführung
   - Hintergrund
   - Installation

2. Indexierung von Korpora
   - Datenmodell
   - Kochbuch

3. Verwendungsbeispiele
   - CQP & CWB-Kommandozeilentools
   - Python: `cwb-ccc`

Grundlagen

- Häufigkeitsverteilungen (mehr im Corpus Encoding Manual):

```
$ cwb-lexdecode -r $registry -f -s -P lemma CORPUS
  | sort -nr -k 1 | head -20
```

- CQP starten:

```
$ cqp -eC -r $registry
```

Grundlagen

- Häufigkeitsverteilungen (mehr im Corpus Encoding Manual):
  ```
  $ cwb-lexdecode -r $registry -f -s -P lemma CORPUS
    | sort -nr -k 1 | head -20
  ```
- CQP starten:
  ```
  $ cqp -eC -r $registry
  ```
- verfügbare Korpora zeigen:
  ```
  > show corpora
  ```
- Korpusinformationen anzeigen:
  ```
  > info CORPUS
  ```
- Korpus aktivieren:
  ```
  > CORPUS
  ```
- Attribute des aktivierten Korpus anzeigen:
  ```
  > show cd
  ```

Abfragen

Suchen nach p-Attributen

```
> "Corona"
> ("Covid19") | "Covid19"
> "C|co((rona)|(vid)).*(19)?"
> [word="Corona"]
> [lemma="Corona"]
> [lemma="Corona" & pos="N"]
> [lemma="Corona" & pos="N.*"]
> [lemma="corona" %cd & word!="Corona"]
```

Konkordanz

Anzeige-Optionen

```
> set Context 20
> set Context 5 words
> set Context tweet
> show +lemma -word
> show +tweet_id
> sort by word %cd
> count by lemma %cd
```

Named Query Results

```
> show named
> Covid = Last
> Bayern = [lemma="Bayern"]
> cat Last
> cat Last 5 10
> size Last
> count Last by lemma %cd
> intersection Corona Bayern
> union Corona Bayern
```

Ankerpunkte

- Zwei *Ankerpunkte* können zur Markierung wichtiger Positionen verwendet werden

  1. *Target* in Anfrage:
     ```
     > [lemma="die"] @[pos="ADJD"] [pos="NN"]
     ```
  2. *Keyword* in Ergebnissen:
     ```
     > set Last keyword matchend
     ```

- seit v3.4.16: auch *Keyword* in Anfrage (mit `@1`):
  ```
  > [lemma="die"] @[pos="ADJD"] @1[pos="NN"]
  ```

CQP Dump

- Ergebnis einer Suchanfrage ist eine Tabelle von Korpuspositionen (*cpos*)

```
> [pos="ADJD"]+ @[pos="NN"]? [pos="NN"]
> dump Last
```

Dump enthält vier Spalten

match	*matchend*	*target*	*keyword*
174460	174461	-1	-1
175685	175686	-1	-1
176715	176716	-1	-1
179244	179245	-1	-1
180121	180123	180122	-1
180983	180985	180984	-1

Wortlisten und Macros

- **Wortlisten** können Suchanfragen vereinfachen

```
> define $Wochentage= "Montag Dienstag ..."
> define $Wochentage < "Wochentage.txt"
> [lemma="letzt"] [lemma=$Wochentage]
```

- komplizierte Suchanfragen können als **Macros** für weitere Verwendung gespeichert werden

```
MACRO np(0)
  [pos = "ART"]
  ([pos = "ADV"]? [pos = "ADJD"])*
  [pos = "NN"]
;
```

Weitere Features

- Suchstrategie
  - wie viele Tokens sollen umfasst werden?

```
> set MatchingStrategy longest
> set MatchingStrategy shortest
> set MatchingStrategy standard
```

- Ankerpunkte ($\geq$ v3.4.16)
  - bis zu 10 Ankerpunkte können gesetzt werden:

```
> @0[pos="ADJD"]+ @1[pos="NN"] @2[pos="V.*"]?
```

  - nur zwei davon sind zeitgleich aktiv:

```
> set ank 0; set ant 1;
```

- Zufallsstichproben

```
> sort Last randomize
> reduce Last to 10%
```

- ...

1. Einführung
   - Hintergrund
   - Installation
2. Indexierung von Korpora
   - Datenmodell
   - Kochbuch
3. Verwendungsbeispiele
   - CQP & CWB-Kommandozeilentools
   - Python: `cwb-ccc`

cwb-ccc

https://github.com/ausgerechnet/cwb-ccc

- Python-basiertes Interface zur CWB
- Installation: `$ pip3 install cwb-ccc`
- Features: Konkordanz, Kollokationen, Keywords

Suchanfragen

```
> from ccc import Corpus
> corpus = Corpus(corpus_name="SZ_2009_14")
> angela = corpus.query(
    r'[lemma="Angela"]? [lemma="Merkel"]'
  )
```

Konkordanz

- verschiedene Formatierungen (*kwic*, *simple*, *dataframes*, ...)
- Visualisierung (mehrerer) Ankerpunkte
- Kontextbeschränkung über Fenstergröße und s-Attribute

```
> angela.breakdown()
> angela.concordance(form="kwic")
> angela.concordance(
    form="dataframes",
    p_show=["word", "lemma"],
    s_show=["text_id"]
  )
```

Kollokate

- Kollokate werden direkt aus den Ergebnissen einer Suchanfrage berechnet
- Standard-AMs implementiert (Log-Likelihood, Dice, χ^2, ...)
- Kookkurrenztabelle auch für eigene Verarbeitung verwendbar

```
> angela.collocates()
> angela.collocates(
    window=5,
    order="log_likelihood",
    cut_off=10
  )
```

Keywords

- externe Programme sind besser als die CWB dazu geeignet, Subkorpora aus Metadaten zu bilden
- entsprechende Text-IDs können direkt übergeben werden

```
> ids = set(meta.loc[
    (meta["ressort"] == "Panorama")
  ].index.values)
> subcorpus = corpus.query_s_att("text_id", values=ids)
> subcorpus.keywords()
```

```
 [lemma="viel"] [word="Dank" %cd] [pos="APPR"]
[pos="ART|PPOSAT"] [lemma="Aufmerksamkeit"] "."
```