Conc Library Command Reference
DIGI405 Corpus Analysis Labs

Reference Guide

February 25, 2026

Contents

1 Introduction

This document provides a comprehensive reference for commands used in the DIGI405 Corpus Analysis Labs (2.1–2.3). The primary tool is the `Conc` library, which provides methods for concordancing, collocation analysis, n-gram analysis, and keyword analysis.

2 Corpus Management Commands

2.1 Corpus Class

2.1.1 `Corpus().load(path)`

Purpose: Load a previously saved corpus from disk.
Arguments:

- `path` (str): Absolute or relative path to the saved `.corpus` file

Returns: Corpus object
Example:

```
corpus = Corpus().load('/srv/corpora/quake-stories-v2.corpus')
```

2.1.2 `Corpus().build_from_files(source, save_path, **kwargs)`

Purpose: Build a new corpus from source text files.
Arguments:

- `source` (str): Path to source files or zip file
- `save_path` (str): Directory where corpus will be saved
- `name` (str, optional): Name for the corpus
- `description` (str, optional): Description of the corpus
- `standardize_word_token_punctuation_characters` (bool, optional): Normalize punctuation

Returns: Corpus object
Example:

```
corpus = Corpus(name='My␣Corpus',
    description='Sample␣corpus').build_from_files(
    'data/texts.zip', '/srv/corpora/')
```

2.1.3 `corpus.summary()`

Purpose: Display statistics about the corpus (number of texts, tokens, types, etc.).
Arguments: None
Returns: Summary display
Example:

```
corpus.summary()
```

2.2 ListCorpus Class

2.2.1 `ListCorpus().load(path)`

Purpose: Load a lightweight frequency list representation of a corpus (used for reference corpora in keyword analysis).
Arguments:

- `path` (str): Path to the `.listcorpus` file

Returns: ListCorpus object
Example:

```
reference = ListCorpus().load('/srv/corpora/bnc.listcorpus')
```

3 Conc Object Initialization

3.1 `Conc(corpus)`

Purpose: Initialize a concordance analysis object with a target corpus.
Arguments:

- `corpus` (Corpus): The corpus to analyze

Returns: Conc object
Example:

```
conc = Conc(corpus)
```

3.2 `conc.set_reference_corpus(reference_corpus)`

Purpose: Set a reference corpus for keyword analysis.
Arguments:

- `reference_corpus` (Corpus or ListCorpus): Reference corpus for comparison

Returns: None
Example:

```
conc.set_reference_corpus(reference_corpus)
```

4 Concordance Commands

4.1 `conc.concordance(query, **kwargs)`

Purpose: Generate concordance lines showing the query word/phrase in context.
Arguments:

- `query` (str): Word or phrase to search for
- `context_length` (int, optional): Number of words of context on each side (default: 5)
- `order` (str, optional): Sort order for concordance lines

  - `'1R2R3R'`: Sort by 1st, 2nd, 3rd word to the right
  - `'1L2L3L'`: Sort by 1st, 2nd, 3rd word to the left
  - `'1L1R'`: Sort by 1st left, then 1st right
  - `'node'`: Sort by the search term itself

- `page_current` (int, optional): Current page number (default: 1)
- `page_size` (int, optional): Number of lines per page (default: 20)
- `filter_context_str` (str, optional): Filter to show only lines containing this string
- `filter_context_length` (int, optional): Window size for filter (default: 5)

Returns: Concordance table object
Examples:

```
# Basic concordance
conc.concordance('home', context_length=8).display()

# Sorted by words to the right
conc.concordance('home', order='1R2R3R', page_size=50).display()

# Filtered concordance
conc.concordance('home', filter_context_str='journey',
    filter_context_length=5).display()
```

5 Collocation Analysis Commands

5.1 `conc.collocates(query, **kwargs)`

Purpose: Generate statistical collocation analysis for a query word.
Arguments:

- `query` (str): Target word for collocation analysis
- `effect_size_measure` (str, optional): Statistical measure to rank collocates
  - `'mutual_information'`: MI score (privileges exclusive associations)
  - `'logdice'`: logDice coefficient
- `context_length` (int or tuple, optional): Window size for collocates
  - Single int: symmetric window (e.g., 5 = 5L and 5R)
  - Tuple: asymmetric window (e.g., (5, 0) = 5L only, (0, 1) = 1R only)
- `min_collocate_frequency` (int, optional): Minimum co-occurrence frequency (default: 5)
- `statistical_significance_cut` (float, optional): p-value threshold (e.g., 0.05, 0.01, 0.001)
- `order` (str, optional): Sort order
  - `None`: Sort by effect size measure (default)

  - `'collocate_frequency'`: Sort by co-occurrence frequency
  - `'frequency'`: Sort by overall frequency
  - `'log_likelihood'`: Sort by statistical significance

- `page_current` (int, optional): Current page number (default: 1)
- `page_size` (int, optional): Number of rows per page (default: 20)

Returns: Collocation table object
Examples:

```
# Basic collocation with MI
conc.collocates('home', effect_size_measure='mutual_information',
    context_length=5, min_collocate_frequency=5).display()

# Using logDice
conc.collocates('home', effect_size_measure='logdice',
    context_length=5).display()

# Asymmetric window (only left context)
conc.collocates('home', context_length=(5, 0)).display()

# With statistical significance filter
conc.collocates('time', statistical_significance_cut=0.0001).display()

# Sorted by frequency
conc.collocates('home', order='collocate_frequency').display()
```

6 N-gram Analysis Commands

6.1 `conc.ngrams(query, **kwargs)`

Purpose: Generate frequency table of n-gram clusters containing a specific query word.
Arguments:

- `query` (str): Word to find in n-grams
- `ngram_length` (int, optional): Length of n-grams (default: 3)
  - 2 = bigrams
  - 3 = trigrams
  - 4 = quadgrams, etc.
- `ngram_token_position` (str, optional): Position of query word in n-gram
  - `'LEFT'`: Query appears at the start of n-gram
  - `'RIGHT'`: Query appears at the end of n-gram
  - `'MIDDLE'`: Query appears in middle positions
- `page_current` (int, optional): Current page number (default: 1)
- `page_size` (int, optional): Number of rows per page (default: 20)

Returns: N-gram table object
Examples:

```
# Trigrams starting with "I"
conc.ngrams('i', ngram_length=3,
    ngram_token_position='LEFT').display()

# Trigrams ending with "home"
conc.ngrams('home', ngram_length=3,
    ngram_token_position='RIGHT').display()

# 4-grams containing "time"
conc.ngrams('time', ngram_length=4).display()
```

6.2 `conc.ngram_frequencies(**kwargs)`

Purpose: Generate frequency table of most common n-grams in the entire corpus (not limited to a specific word).

Arguments:

- `ngram_length` (int, optional): Length of n-grams (default: 3)
- `page_current` (int, optional): Current page number (default: 1)
- `page_size` (int, optional): Number of rows per page (default: 20)

Returns: N-gram frequency table object

Examples:

```
# Most common 4-grams
conc.ngram_frequencies(ngram_length=4).display()

# View page 20 of trigrams
conc.ngram_frequencies(ngram_length=3,
    page_current=20).display()
```

7 Keywords Analysis Commands

7.1 `conc.keywords(**kwargs)`

Purpose: Generate keyword analysis comparing target corpus to reference corpus.

Note: Requires `conc.set_reference_corpus()` to be called first.

Arguments:

- `page_current` (int, optional): Current page number (default: 1)
- `page_size` (int, optional): Number of rows per page (default: 20)

Returns: Keywords table object containing:

- **Frequency**: Raw frequency in target corpus
- **Frequency Reference**: Raw frequency in reference corpus
- **Normalized Frequency**: Frequency per million words in target
- **Normalized Frequency Reference**: Frequency per million words in reference
- **Relative Risk**: Ratio of normalized frequencies (>1 = overuse, <1 = underuse)

- **Log Ratio**: Intuitive effect size measure
- **Log Likelihood**: Statistical significance measure

Examples:

```
# Set reference corpus
conc.set_reference_corpus(bnc_corpus)

# Generate keywords
conc.keywords(page_size=50).display()

# Compare two sub-corpora
conc_labour = Conc(labour_corpus)
conc_labour.set_reference_corpus(national_corpus)
conc_labour.keywords().display()
```

8 Display Methods

All table-generating methods return objects that support the `.display()` method.

8.1 `.display()`

Purpose: Render the results table in the notebook output.
Arguments: None
Returns: None (displays output)
Example:

```
conc.concordance('home').display()
```

9 Common Parameter Patterns

9.1 Pagination

Most commands support pagination for large result sets:

- `page_current`: Which page to display (1-indexed)
- `page_size`: How many results per page

9.2 Context Windows

Context can be specified as:

- **Integer**: Symmetric window (e.g., 5 = 5 words left and right)
- **Tuple**: Asymmetric window (e.g., (3, 5) = 3 left, 5 right)

9.3 Sorting

Many tables can be sorted by different columns using the `order` parameter. Valid values vary by command but commonly include field names or special codes.

10 Workflow Examples

10.1 Complete Collocation Workflow

```
# 1. Load corpus
corpus = Corpus().load('path/to/corpus.corpus')

# 2. Initialize Conc
conc = Conc(corpus)

# 3. Generate collocation table
conc.collocates('home',
    effect_size_measure='mutual_information',
    context_length=5,
    min_collocate_frequency=5).display()

# 4. Inspect a specific collocate in concordance
conc.concordance('home',
    filter_context_str='journey',
    filter_context_length=5,
    order='1R2R3R').display()
```

10.2 Complete N-gram Workflow

```
# 1. Overwhelmed by concordance lines
conc.concordance('i', context_length=8).display()
# Output: 12,000+ lines!

# 2. Use n-grams to identify most common patterns
conc.ngrams('i', ngram_length=3,
    ngram_token_position='LEFT').display()

# 3. Inspect interesting n-gram in concordance
conc.concordance('i␣had␣to', context_length=8).display()

# 4. View overall n-gram frequencies
conc.ngram_frequencies(ngram_length=4,
    page_current=1).display()
```

10.3 Complete Keywords Workflow

```
# 1. Load target corpus
target = Corpus().load('quake-stories.corpus')

# 2. Load reference corpus
reference = ListCorpus().load('bnc.listcorpus')

# 3. Initialize and set reference
conc = Conc(target)
conc.set_reference_corpus(reference)

# 4. Generate keywords
conc.keywords(page_size=50).display()
```

```
# 5. Analyze a specific keyword
conc.collocates('earthquake',
    effect_size_measure='mutual_information').display()

conc.concordance('earthquake',
    order='1L2L3L').display()

conc.ngrams('earthquake', ngram_length=3).display()
```

11 File Operations

11.1 Python File Operations Used

11.1.1 `shutil.copy(source, destination)`

Purpose: Copy a file from source to destination.
Example:

```
import shutil
import os

source_file = '/srv/source-data/corpus.zip'
destination = os.path.join(os.getcwd(),
    os.path.basename(source_file))
shutil.copy(source_file, destination)
```

11.1.2 `os.path.join(path1, path2, ...)`

Purpose: Join path components intelligently (handles OS differences).

11.1.3 `os.path.basename(path)`

Purpose: Extract filename from a full path.

11.1.4 `os.getcwd()`

Purpose: Get current working directory.