DeComposition Corpus – Documentation

In this documentation, we briefly describe the DeComposition Corpus (DCC), as well as how to align the DCC with users' own instances of the Penn Parsed Corpora of Historical English (PPCHE) and the York-Toronto-Helsinki Corpus of Old English Prose (YCOE).

We start out with a brief description of the DCC and the information contained within it. We continue with a breakdown of the included materials which is followed by a how-to for using the included DCC Alignment Tool (DAT).

1. Introduction to the DCC

The DCC includes four separate decompositional items: *again*, *eft* (OE *again*), *re*-verbs, and almost. For each decompositional item, there is one TSV file. Every TSV file has the (column-)headers 'corpusID', 'targetID', 'l_object', position, 'token', and 'reading'. Every single row of the TSV file contains the annotational information for one annotation unit. For instance, one occurrence of *again* is one annotational unit. Moreover, the predicate *again* modifies constitutes an annotation unit (e.g. *tried* in (1)). Antecedents also constitute annotation units. In (1) this is *pushed*—the main verb of the predicate permitting the inference that the presupposition that 'Alice tried before' is satisfied in the context:

(1) Alice pushed against the door but it wouldn't budge.
Alice tried again.

- **corpusID** — corresponds to the corpus IDs as provided by the underlying corpora (YCOE, PPCME2, PPCEME, PPCMBE) and indicates the location of any annotation unit.

- **targetID** — is a number assigned to each decompositional target item (e.g. one use of *again*). It will be identical across all annotation units associated with the current target item. For instance, if the use of *again* in (1) is the 42nd instance of *again* in the dataset, all three annotation units in (1) share the targetID '42'.

- **l_object** — contains the object language annotation unit; targets are capitalized (with respect to (1), "AGAIN"), target predicates are enclosed in asterisks ("*tried*"), antecedents are enclosed in underscores ("_pushed_").

- **position** — indicates the zero-based position of the annotation unit within the corpus token; for (1), that is 1 for *pushed*, 2 for *again*, and 1 for *tried*.

- **token** — contains the corpus token——(i) object language items only, (ii) without the syntactic parse, (ii) in their surface order.

- **reading** — the classification of the target in terms of its reading; for instance, “rep” (for ‘repetitive’) with respect to (1).

The annotations for every annotation unit will be packed into a DC tag (‘decomposition tag’). For instance, derived from the toy example in (1), the relevant tags would come out as in (2)–(4).

During the alignment process, these will be appended to the POS tag in the syntactic parse as shown in (5)–(7).

Multiple DC tags can be appended to a POS tag:

(2) @DC42:rep:ANT:1
(3) @DC42:rep:PRD:1
(4) @DC42:rep:TRG:2

(5) (VBD@DC42:rep:ANT:1 pushed)
(6) (VBD@DC42:rep:PRD:1 tried)
(7) (ADV@DC42:rep:TRG:2 again)

Based on our testing, these DC tags merge relatively seamlessly with the base versions of the corpora. The character “@” is unused in both the YCOE and PPCHE data. Once merged, they can be queried with the default software CorpusSearch—if the absence of the usual POS-tag boundary is kept in mind.

2. What is included:

The DCC-package in its current version consists of the following major components:

1. – DCC (V1.0): The TSV files containing the DCC annotations. (Due to restrictions with our preliminary hosting partner, the files might end on the suffix “.csv”.)

2. – DCC Alignment Tool (‘DAT’, V0.1, ‘2511_align_DCC_w_psd.py’): A Python script to align the DCC with users’ own instances of the PPCHE and the YCOE.

3. How to use DAT (DCC Alignment Tool):

Recommended setup:

The recommendation is to create a folder, a working directory (WD), containing at least:

1. – a subfolder named ’01_tsv’ containing the TSV-files,

2. – the Python file containing the DAT,

3. – folders names ‘psd_YCOE’, ‘psd_PPCME’, ‘psd_PPCEME, and ‘psd_PPCMBE’, containing the relevant set of .PSD files,

and optionally

4. – empty output folders named ‘psd_DC_YCOE’, ‘psd_DC _PPCME’, ‘psd_DC _PPCEME, and ‘psd_DC _PPCMBE’.

The alignment tool will create the output folders should they not exist in the WD.

The user can change the alignment script as they see fit (adapt paths, change the file endings, etc).

How to run DAT:

DAT was written and tested with Python 3.9. It relies only on three standard modules: os, re (version 2.2.1) and pandas (version 2.2.3).

Upon starting it (with an IDE or from shell), the user is prompted to specify the desired DCC item and the desired (sub-)corpus for the alignment.

During alignment, the script outputs information on major operational steps as progress reporting during the alignment process.