CLiC Dickens Documentation

Release 1.3

J. de Joode

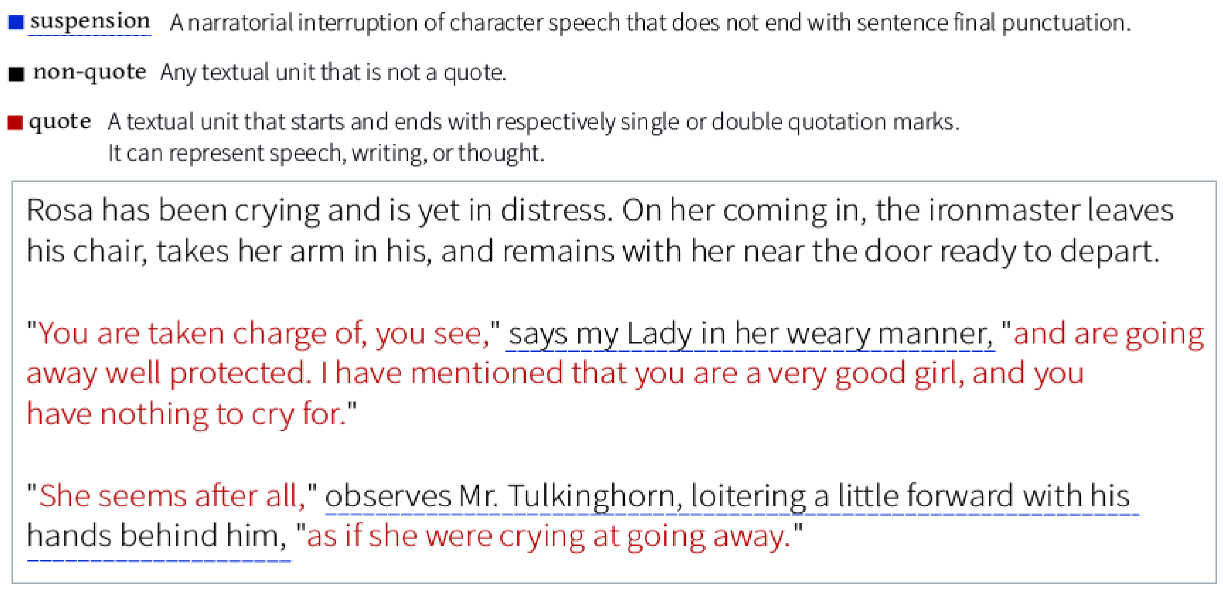
March 15, 2016

Contents:

CHAPTER ONE

CLIC FOR END-USERS

1.1 Definitions


🟦 suspension A narratorial interruption of character speech that does not end with sentence final punctuation.

⬛ non-quote Any textual unit that is not a quote.

🟥 quote A textual unit that starts and ends with respectively single or double quotation marks. It can represent speech, writing, or thought.

Rosa has been crying and is yet in distress. On her coming in, the ironmaster leaves his chair, takes her arm in his, and remains with her near the door ready to depart.

"You are taken charge of, you see," says my Lady in her weary manner, "and are going away well protected. I have mentioned that you are a very good girl, and you have nothing to cry for."

"She seems after all," observes Mr. Tulkinghorn, loitering a little forward with his hands behind him, "as if she were crying at going away."

1.2 The underlying annotation

1.3 The corpora

1.4 How to use ...

1.4.1 the concordance

1.4.2 clusters

1.4.3 keywords

1.4.4 subsets

1.4.5 patterns

1.4.6 user annotation

CHAPTER

TWO

CLIC FOR ADMINISTRATORS

2.1 Installing CLiC on your own server

For the deployment of CLiC we heavily rely on Docker. This has several benefits: it packages all the dependencies of CLiC together in a simple image and it makes a deployment much faster and possible on many different platforms.

2.1.1 Install Docker on a vanilla Ubuntu server

```
# Install Docker in a way that can easily be upgraded
sudo apt-get update

sudo apt-key adv --keyserver hkp://pgp.mit.edu:80 --recv-keys
58118e89f3a912897c070adbf76221572c52609d

# open the /etc/apt/sources.list.d/docker.list file in your favorite
editor. if the file doesn’t exist, create it. and add:

# this is not a command, but something that needs to be pasted in the
file opened

deb https://apt.dockerproject.org/repo ubuntu-trusty main

sudo apt-get update

# verify that apt is pulling from the right repository.

sudo apt-cache policy docker-engine

sudo apt-get install linux-image-generic-lts-trusty

# you need to reboot, this can cause some issues where the firewall
settings would not be saved, this was solved by loading the firewall
explicitly at the restart

sudo reboot

sudo apt-get update

sudo apt-get install docker-engine

sudo docker run hello-world
```

2.1.2 Configure Docker

```
# set UFW's forwarding policy appropriately.
# Open /etc/default/ufw file for editing.
sudo vimsu /etc/default/ufw

# Set the DEFAULT\_FORWARD\_POLICY policy to:
DEFAULT\_FORWARD\_POLICY="ACCEPT"

sudo ufw reload

#TODO check if this is persistent

# Allow incoming connections on the Docker port.
$ sudo ufw allow 2375/tcp

# Configure Docker to start on boot
# DOCS SAY $ sudo systemctl enable docker
# but I think:

sudo update-rc.d docker defaults
```

2.1.3 Quick Docker command guide

You might have to prepend `sudo` to the commands below, depending on your environment.

```
# is docker installed
docker info

# activate docker
docker-machine start default
eval "$(docker-machine env default)"

# to find localhost ping
docker-machine env default

# to build
cd ~/ImagesDocker/clic-docker/
docker build -t jdejoode/clic:v0 .

# list images
docker images

# run a docker image
docker run -d -P -i -t --name apache11 jdejoode/clic:v0 docker ps

# find info on container
docker port apache11
docker logs a5a665d32

# for live updates a la Flask
docker logs -f a6516a51sd f651

# stop all docker containers
docker stop $(docker ps -a -q)

# remove them
```

```
docker rm $(docker ps -a -\ **q**\ )

# ssh into container
docker exec -i -t fbd8112 bash

# command on actual deploy is slightly different for caching and
rounting purposes:
# -v /path/on/host:/path/in/container
docker run -p 80:8080 -v /tmp/cache:/tmp/cache ...

docker run -d -P --name clic2 -v
/bin:/clic-project/clic/dbs/dickens/indexes jdejoode/clic:v0

# to see what processes run in the container
docker top clic0

# remove untagged images
docker rmi $(docker images \| grep "^<none>" \| awk '{print $3}')

# deploy
# on macbook
# https://docs.docker.com/docker-hub/repos/
docker login
docker push jdejoode/clic:latest
```

CHAPTER

THREE

CLIC FOR DEVELOPERS

3.1 Cheshire3

3.1.1 The data-model

3.2 CLiC Concordance

CLiC Concordance based on cheshire3 indexes.

class `concordance.`**`Concordance`**

This concordance takes terms, index names, book selections, and search type as input values and returns json with the search term, ten words to the left and ten to the right, and location information.

This can be used in an ajax api.

`build_and_run_query`(*terms*, *idxName*, *Materials*, *selectWords*)

Builds a cheshire query and runs it.

Its output is a tuple of which the first element is a resultset and the second element is number of search terms in the query.

`create_concordance`(*terms*, *idxName*, *Materials*, *selectWords*)

main concordance method create a list of lists containing each three contexts left - node -right, and a list within those contexts containing each word. Add two separate lists containing metadata information: [[left context - word 1, word 2, etc.], [node - word 1, word 2, etc], [right context - word 1, etc], [chapter metadata], [book metadata]], etc.

3.3 CLiC Clusters

Tool to create wordlists based on the entries in an index.

class `clusters.`**`Clusters`**

Class that does all the heavy weighting. It makes the connection with cheshire3, uses the input parameters (indexname and subcorpus/Materials) to return a list of words and their total number of occurrences.

For instance,

```
the 98021

to 78465
```

...

or `he said 8937`

`she said 6732`

...

`list_clusters`(*idxName*, *Materials*)
Build a list of clusters and their occurrences.

Limit the list to the first 3000 entries.

3.4 CLiC Keywords

Module to compute keywords (words that are used significantly more frequently in one corpus than they are in a reference corpus).

class `keywords.`**`Keywords`**
Class to compute keywords based on an test index (the corpus of analysis), a reference index (the corpus of reference), and a P value.

`list_keywords`(*testIdxName*, *testMaterials*, *refIdxName*, *refMaterials*, *pValue*)
Return a sorted list of keywords. Limited to the first 5000 items.

3.5 CLiC Chapter Repository

Display the texts available in the cheshire3 database. Also highlight specific items that were previously retrieved with a concordance.

class `chapter_repository.`**`ChapterRepository`**
Responsible for providing access to chapter resources within Cheshire.

`get_book_title`(*book*)
Gets the title of a book from the json file booklist.json

book – string - the book id/accronym e.g. BH

`get_chapter`(*chapter_number*, *book*)
Returns transformed XML for given chapter & book

chapter_number – integer book – string - the book id/accronym e.g. BH

`get_chapter_with_highlighted_search_term`(*chapter_number*, *book*, *wid*, *search_term*)
Returns transformed XML for given chapter & book with the search highlighted.

We create the transformer directly so that we can pass extra parameters to it at runtime. In this case the search term.

chapter_number – integer book – string - the book id/accronym e.g. BH wid – integer - word index search_term – string - term to highlight

`get_raw_chapter`(*chapter_number*, *book*)
Returns raw chapter XML for given chapter & book

chapter_number – integer book – string - the book id/accronym e.g. BH

3.6 CLiC KWICgrouper

A module to look for patterns in concordances.

class `kwicgrouper.`**`Concordance`**(*term*, *text*, *word_boundaries=True*, *length=50*, *keep_punctuation=True*, *keep_line_breaks=False*)

This is a simple concordance for a text file. The input text should a string that is cleaned, for instance:

text.replace("

", " ").replace(" ", " ")

This function has two argument: the search term and the text to be searched.

The length should be an integer

classmethod **`from_multiple_line_file`**(*term*, *input_li*)

Construct a concordance that respect line breaks (rather than one that treats the text as one large string)

TODO

`list_concordance`()

List the actual concordance.

`print_concordance`()

Print the lines of a concordance. For debugging purposes.

`single_line_conc`()

Build a basic concordance based on a single string of text.

class `kwicgrouper.`**`KWICgrouper`**(*concordance*)

This starts from a concordance and transforms it into a pandas dataframe (here called textframe) that has five words to the left and right of the search term in separate columns. These columns can then be searched for and sorted.

Input: A nested list of lists looking like:

[['sessed of that very useful appendage a ', 'voice', ' for a much longer space of time than t'],

...

Each pattern needs *its own* instantiation of the KWICgrouper object because the self.textframe variable is changed in the filter method.

`args_to_dict`(*L5=None*, *L4=None*, *L3=None*, *L2=None*, *L1=None*, *R1=None*, *R2=None*, *R3=None*, *R4=None*, *R5=None*)

Helper function to use L1="a" type of functions

`conc_to_df`()

Turns a list of dictionaries with L1-R5 values into a dataframe which can be used as a kwicgrouper.

`filter_textframe`(*kwdict*)

Construct a dataframe slice and selector on the fly. This is no longer meta-programming as it does not use the eval function anymore.

This returns None if there is no textframe

`split_nodes`()

Splits the words into nodes that can be fed into a dataframe.

`kwicgrouper.`**`clean_punkt`**(*text*)

Delete punktuation from a text.

Problem: turns CAN'T into CA NT

`kwicgrouper.`**`clean_text`**(*text*)

Clean a text so that it can be used in a concordance. This includes:

- all text to lowercase
- deleting line-breaks
- tokenizing and detokenizing

`kwicgrouper.`**`concordance_for_line_by_line_file`**(*input_file*, *term*)

Takes a file that has different line breaks that cannot be ignored (for instance a file with a list of things) and makes it into a concordance

`kwicgrouper.`**`old_clean_punkt`**(*text*)

This ignores apostrophes and punctuation marks attached to the word * an alternative way would be to replace-delete the punctuation from the text

3.7 CLiC Normalizer

Defines normalizers that can be used in the cheshire3 indexing workflow.

3.8 CLiC Query Builder

Future module to handle the construction of cheshire3 CQL queries.

3.9 CLiC Web app

3.9.1 Index

This is the most important file for the web app. It contains the various routes that end users can use.

For instance

@app.route('/about/', methods=['GET']) def about():

return render_template("info/about.html")

Where /about/ is the link.

3.9.2 API

This file is an extension of index.py. It generates the raw json API that the keywords, cluster, and concordances use(d).

It needs to be refactored.

3.9.3 Models

models.py defines the SQL tables that CLiC uses. These classes provide a python interface to the SQL database so that you can write python code that automatically queries the database.

This is heavily dependent on Flask-SQLAlchmey and SQLAlchemy

CHAPTER

FOUR

REPORTING ISSUES

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX