CorrectOCR Documentation

Mikkel Eide Eriksen, for Copenhagen City Archives

May 25, 2022

CONTENTS:

CHAPTER

ONE

WORKFLOW

Usage of CorrectOCR is divided into several successive tasks.

To train the software, one must first create or obtain set of matching original uncorrected files with corresponding known-correct "gold" files. Additionally, a dictionary of the target language is needed.

The pairs of (original, gold) files are then used to train a HMM model that can then be used to generate k replacement candidates for each token (word) in a new given file. A number of heuristic decisions are configured based on whether a given token is found in the dictionary, are the candidates preferable to the original, etc.

Finally, the tokens that could not be corrected based on the heuristics can be presented to annotators either via *CLI* or a HTTP `HTTP server`. The annotators' corrections are then incorporated in a corrected file.

When a corrected file is satisfactory, it can be moved or copied to the gold directory and in turn be used to tune the HMM further, thus improving the k-best candidates for subsequent files.

CHAPTER

TWO

COMMANDS

2.1 Global Arguments

If the global arguments are not provided on the command line, *CorrectOCR.ini* and environment variables are checked (see *CorrectOCR.config*).

2.1.1 Workspace

These arguments configure the *Workspace*, ie. where the documents are located.

```
usage: python -m CorrectOCR [-h] [--rootPath PATH] [--originalPath PATH]
                            [--goldPath PATH] [--trainingPath PATH]
                            [--docInfoBaseURL URL] [--nheaderlines N]
                            [--language LANGUAGE]
                            [--combine_hyphenated_images [COMBINE_HYPHENATED_IMAGES]]
```

2.1.1.1 Named Arguments

--rootPath	Path to root of workspace
--originalPath	Path to directory of original, uncorrected docs
--goldPath	Path to directory of known correct "gold" docs
--trainingPath	Path for generated training files
--docInfoBaseURL	Base URL that serves info about documents
--nheaderlines	Number of lines in corpus headers Default: 0
--language	Language of text
--combine_hyphenated_images	Generate joined images for hyphenated tokens

2.1.2 Resource

These arguments configure the *ResourceManager*, eg. dictionary, model, etc.

```
usage: python -m CorrectOCR [-h] [--resourceRootPath PATH]
                            [--hmmParamsFile FILE] [--reportFile FILE]
                            [--heuristicSettingsFile FILE]
                            [--multiCharacterErrorFile FILE]
                            [--memoizedCorrectionsFile FILE]
                            [--correctionTrackingFile FILE]
                            [--dictionaryFile FILE] [--ignoreCase]
```

2.1.2.1 Named Arguments

--resourceRootPath Path to root of resources

--hmmParamsFile Path to HMM parameters (generated from alignment docs via build_model command)

--reportFile Path to output heuristics report (TXT file)

--heuristicSettingsFile Path to heuristics settings (generated via make_settings command)

--multiCharacterErrorFile Path to output multi-character error file (JSON format)

--memoizedCorrectionsFile Path to memoizations of corrections.

--correctionTrackingFile Path to correction tracking.

--dictionaryFile Path to dictionary file

--ignoreCase Use case insensitive dictionary comparisons

Default: False

2.1.3 Storage

These arguments configure the `TokenList` backend storage.

```
usage: python -m CorrectOCR [-h] [--type {db,fs}] [--db_driver DB_DRIVER]
                            [--db_host DB_HOST] [--db_user DB_USER]
                            [--db_password DB_PASSWORD] [--db DB]
```

2.1.3.1 Named Arguments

--type Possible choices: db, fs

Storage type

--db_driver Database hostname

--db_host Database hostname

--db_user Database username

--db_password Database user password

--db Database name

2.2 Commands

Correct OCR

```
usage: python -m CorrectOCR [-h] [-k K] [--force]
                            [--loglevel {CRITICAL,FATAL,ERROR,WARNING,INFO,DEBUG}]
                            [--dehyphenate DEHYPHENATE]
                            {dictionary,align,model,add,prepare,crop,stats,correct,index,
↪cleanup,server}
                            ...
```

2.2.1 Positional Arguments

command Possible choices: dictionary, align, model, add, prepare, crop, stats, correct, index, cleanup, server

Choose command

2.2.2 Named Arguments

-k Number of k-best candidates to use for tokens (default: 4)

Default: 4

--force Force command to run

Default: False

--loglevel Possible choices: CRITICAL, FATAL, ERROR, WARNING, INFO, DEBUG

Log level

Default: “INFO”

--dehyphenate Automatically mark new tokens as hyphenated if they end with a dash

2.2.3 Sub-commands:

2.2.3.1 dictionary

Dictionary-related commands.

```
python -m CorrectOCR dictionary [-h] {build,check} ...
```

Sub-commands:

build

Build dictionary.

Input files can be either `.pdf`, `.txt`, or `.xml` (in TEI format). They may be contained in `.zip`-files.

A `corpusFile` for 1800–1948 Danish is available in the `workspace/resources/` directory.

It is strongly recommended to generate a large dictionary for best performance.

See CorrectOCR.dictionary for further details.

```
python -m CorrectOCR dictionary build [-h] [--corpusPath CORPUSPATH]
                                       [--corpusFile CORPUSFILE]
                                       [--add_annotator_gold] [--clear]
```

Named Arguments

--corpusPath Directory of files to split into words and add to dictionary

--corpusFile File containing paths and URLs to use as corpus (TXT format)

--add_annotator_gold Add gold words from annotated tokens

Default: False

--clear Clear the dictionary before adding words

Default: False

check

Undocumented

```
python -m CorrectOCR dictionary check [-h] [words [words ...]]
```

Positional Arguments

words Words to check in dictionary

2.2.3.2 align

Create alignments.

The tokens of each pair of (original, gold) files are aligned in order to determine which characters and words were misread in the original and corrected in the gold.

These alignments can be used to train the model.

See CorrectOCR.aligner for further details.

```
python -m CorrectOCR align [-h]
                           (--docid DOCID | --docids DOCIDS [DOCIDS ...] | --all)
                           [--exclude EXCLUDE]
```

Named Arguments

--docid	Input document ID (filename without path or extension)
--docids	Input multiple document IDs
--all	Align all original/gold pairs Default: False
--exclude	Doc ID to exclude (can be specified multiple times) Default: []

2.2.3.3 model

Build model. # TODO # This is done with the aligned original/gold-documents. If none exist, an attempt will be made to create them.

The result is an HMM as described in the original paper.

See CorrectOCR.model for further details.

```
python -m CorrectOCR model [-h] (--build | --get_kbest GET_KBEST)
                           [--smoothingParameter N[.N]] [--other OTHER]
```

Named Arguments

--build	Rebuild model Default: False
--get_kbest	Get k-best for word with current model
--smoothingParameter	Smoothing parameters for HMM Default: 0.0001
--other	Compare against other model

2.2.3.4 add

Add documents for processing

One may add a single document directly on the command line, or provide a text file containing a list of documents.

They will be copied or downloaded to the `workspace/original/` folder.

See CorrectOCR.workspace.Workspace for further details.

```
python -m CorrectOCR add [-h] [--documentsFile DOCUMENTSFILE]
                          [--prepare_step {tokenize,align,kbest,bin,all,server}]
                          [--max_count MAX_COUNT]
                          [document]
```

Positional Arguments

document	Single path/URL to document

Named Arguments

--documentsFile	File containing list of files/URLS to documents
--prepare_step	Possible choices: tokenize, align, kbest, bin, all, server Automatically prepare added documents
--max_count	Maximum number of new documents to add from –documentsFile.

2.2.3.5 prepare

Prepare text for correction.

See CorrectOCR.workspace.Document for further details on the possible steps.

```
python -m CorrectOCR prepare [-h]
                             (--docid DOCID | --docids DOCIDS [DOCIDS ...] | --all | --
↪skip_done)
                             [--exclude EXCLUDE]
                             [--step {tokenize,rehyphenate,align,kbest,bin,all,server}]
                             [--autocrop] [--precache_images]
```

Named Arguments

--docid	Input document ID (filename without path or extension)
--docids	Input multiple document IDs
--all	Select all documents Default: False
--skip_done	Select only unfinished documents Default: False
--exclude	Doc ID to exclude (can be specified multiple times) Default: []
--step	Possible choices: tokenize, rehyphenate, align, kbest, bin, all, server Default: “all”

--autocrop	Discard tokens near page edges
	Default: False
--precache_images	Create images for the server API
	Default: False

2.2.3.6 crop

Mark tokens near the edges of a page as disabled.

This may be desirable for scanned documents where the OCR has picked up partial words or sentences near the page edges.

The tokens are not discarded, merely marked disabled so they don't show up in the correction interface or generated gold files.

If neither –edge_left nor –edge_right are provided, an attempt will be made to calculate them automatically.

```
python -m CorrectOCR crop [-h]
                          (--docid DOCID | --docids DOCIDS [DOCIDS ...] | --all)
                          [--edge_left EDGE_LEFT] [--edge_right EDGE_RIGHT]
```

Named Arguments

--docid	Input document ID (filename without path or extension)
--docids	Input multiple document IDs
--all	Prepare all original/gold pairs
	Default: False
--edge_left	Set left cropping edge (in pixels)
--edge_right	Set right cropping edge (in pixels)

2.2.3.7 stats

Calculate stats about corrected documents.

The procedure is to first generate a report that shows how many tokens have been sorted into each bin. This report can then be annotated with the desired decision for each bin, and use this annotated report to generate settings for the heuristics.

See CorrectOCR.heuristics.Heuristics for further details.

```
python -m CorrectOCR stats [-h] (--make_report | --make_settings) [--rebin]
                           [--only_done ONLY_DONE]
```

Named Arguments

--make_report	Make heuristics statistics report from finished documents
	Default: False
--make_settings	Make heuristics settings from report
	Default: False
--rebin	Rerun kbest/bin steps to compare quality (will take longer)
	Default: False
--only_done	Whether to include all or only fully annotated documents
	Default: True

2.2.3.8 correct

Apply corrections

```
python -m CorrectOCR correct [-h] (--docid DOCID | --filePath FILEPATH)
                             (--interactive | --apply APPLY | --autocorrect | --gold_
↪ready)
                             [--highlight]
```

Named Arguments

--docid	Input document ID (filename without path or extension)
--filePath	Input file path (will be copied to originalPath directory)
--interactive	Use interactive shell to input and approve suggested corrections
	Default: False
--apply	Apply externally corrected token CSV to original document
--autocorrect	Apply automatic corrections as configured in settings
	Default: False
--gold_ready	Apply gold from ready document
	Default: False
--highlight	Create a copy with highlighted words (only available for PDFs)
	Default: False

2.2.3.9 index

Generate index data

```
python -m CorrectOCR index [-h] (--docid DOCID | --filePath FILEPATH)
                           [--exclude EXCLUDE] --termFile TERMFILES
                           [--highlight] [--autocorrect]
```

Named Arguments

--docid	Input document ID (filename without path or extension)
--filePath	Input file path (will be copied to originalPath directory)
--exclude	Doc ID to exclude (can be specified multiple times) Default: []
--termFile	File containing a string on each line, which will be matched against the tokens Default: []
--highlight	Create a copy with highlighted words (only available for PDFs) Default: False
--autocorrect	Apply automatic corrections as configured in settings Default: False

2.2.3.10 cleanup

Clean up intermediate files

```
python -m CorrectOCR cleanup [-h] [--dryrun] [--full]
```

Named Arguments

--dryrun	Dont delete files, just list them Default: False
--full	Also delete the most recent files (without .nnn. in suffix) Default: False

2.2.3.11 server

Run basic JSON-dispensing Flask server

```
python -m CorrectOCR server [-h] [--host HOST] [--debug] [--profile [PROFILE]]
                            [--dynamic_images DYNAMIC_IMAGES]
                            [--redirect_hyphenated REDIRECT_HYPHENATED]
```

Named Arguments

--host	The host address
--debug	Runs the server in debug mode (see Flask docs) Default: False
--profile	Use Werkzeug profiler middleware
--dynamic_images	Should images be generated dynamically?
--redirect_hyphenated	Redirect requests for hyphenated tokens to "head"

CHAPTER

THREE

CORRECTOCR PACKAGE

3.1 Submodules

3.1.1 CorrectOCR.aligner module

3.1.2 CorrectOCR.config module

When invoked, CorrectOCR looks for a file named `CorrectOCR.ini` in the working directory. If found, it is loaded, and any entries will be considered defaults to their corresponding option. These are the defaults:

```
[configuration]
characterSet = ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
dehyphenate = true
loglevel = INFO

[workspace]
rootPath = ./
goldPath = gold/
originalPath = original/
trainingPath = training/
nheaderlines = 0
language = Danish
docInfoBaseURL =
combine_hyphenated_images = true

[resources]
resourceRootPath = ./resources/
correctionTrackingFile = correction_tracking.json
dictionaryPath = dictionary/
hmmParamsFile = hmm_parameters.json
memoizedCorrectionsFile = memoized_corrections.json
multiCharacterErrorFile = multicharacter_errors.json
reportFile = report.txt
heuristicSettingsFile = settings.json

[storage]
type = fs
db_driver =
db_host =
db_user =
```

(continues on next page)

(continued from previous page)

```
db_pass =
db_name =

[server]
host = 127.0.0.1
profile = false
dynamic_images = true
redirect_hyphenated = true
```

By default, CorrectOCR requires 4 subdirectories in the working directory, which will be used as the current `Workspace`:

- `original/` contains the original uncorrected files. If necessary, it can be configured with the `--originalPath` argument.
- `gold/` contains the known correct "gold" files. If necessary, it can be configured with the `--goldPath` argument.
- `training/` contains the various generated files used during training. If necessary, it can be configured with the `--trainingPath` argument.

Corresponding files in *original* and *gold* are named identically, and the filename without extension is considered the *file ID*. The generated files in `training/` have suffixes according to their kind.

If generated files exist, CorrectOCR will generally avoid doing redundant calculations. The `--force` switch overrides this, forcing CorrectOCR to create new files (after moving the existing ones out of the way). Alternately, one may delete a subset of the generated files to only recreate those.

The `Workspace` also has a `ResourceManager` (accessible in code via `.resources`) that handles access to the dictionary, HMM parameter files, etc.

3.1.2.1 Environment Variables

Environment variables follow the format `CORRECTOCR_<section>_<name>` in uppercase. For example, the Workspace root path can be configured by setting `CORRECTOCR_WORKSPACE_ROOTPATH`.

class CorrectOCR.config.**EnvOverride**

Bases: configparser.BasicInterpolation

This class overrides the .ini file with environment variables if they exist.

They are checked according to this format: CORRECTOCR_<section>_<key>, all upper case.

Thus, to override the storage:db_server setting, set the CORRECTOCR_STORAGE_DB_SERVER variable.

before_get(*parser*, *section*, *option*, *value*, *defaults*)

CorrectOCR.config.**setup**(*args*, *configfiles=['CorrectOCR.ini']*)

3.1.3 CorrectOCR.correcter module

3.1.3.1 Correction Interface

The annotator will be presented with the tokens that match a heuristic bin that was marked for annotation.

They may then enter a command. The commands reflect the above settings, with an additional `defer` command to defer decision to a later time.

Prefixing the entered text with an exclamation point causes it to be considered the corrected version of the token. For example, if the token is "Wagor" and no suitable candidate is available, the annotator may enter `!Wagon` to correct the word.

Corrections are memoized, so the file need not be corrected fully in one session. To finish a session and save corrections, use the `quit` command.

A `help` command is available in the interface.

See also the `Server` for a HTTP backend.

class CorrectOCR.correcter.**CorrectionShell**(*tokens*, *dictionary*, *correctionTracking*)

Bases: cmd.Cmd

Interactive shell for making corrections to a list of tokens. Assumes that the tokens are *binned*.

Instantiate a line-oriented interpreter framework.

The optional argument 'completekey' is the readline name of a completion key; it defaults to the Tab key. If completekey is not None and the readline module is available, command completion is done automatically. The optional arguments stdin and stdout specify alternate input and output file objects; if not specified, sys.stdin and sys.stdout are used.

classmethod start(*tokens*, *dictionary*, *correctionTracking*, *intro=None*)

Parameters

- **tokens** (*TokenList*) – A list of Tokens.
- **dictionary** – A dictionary against which to check validity.
- **correctionTracking** (dict) – TODO
- **intro** (Optional[str]) – Optional introduction text.

do_original(_)

Choose original (abbreviation: o)

do_shell(*arg*)

Custom input to replace token

do_kbest(*arg*)

Choose k-best by number (abbreviation: just the number)

do_kdict(*arg*)

Choose k-best which is in dictionary

do_memoized(*arg*)

do_error(*arg*)

do_linefeed(_)

do_defer(_)

Defer heuristic for another time.

do_quit(_)

3.1.4 CorrectOCR.dictionary module

class CorrectOCR.dictionary.**Dictionary**(*path=None*, *ignoreCase=False*)

Bases: Set[str]

Set of words to use for determining correctness of *Tokens* and suggestions.

Note: A Dictionary "contains" all "words" that contain at most 1 alphabetic letters, such as 8,5 or (600) or A4 .

Parameters

- **path** (Optional[Path]) – A path for loading a previously saved dictionary.
- **ignoreCase** (bool) – Whether the dictionary is case sensitive.

has_group(*group*)

Return type bool

clear()

Remove all elements from this set.

add(*group*, *word*, *nowarn=False*, *dirty=True*)

Add a new word (sans punctuation) to the dictionary. Silently drops non-alpha strings.

Parameters

- **word** (str) – The word to add.
- **nowarn** (bool) – Don't warn about long words (>20 letters).

save_group(*group*)

save(*path=None*)

Save the dictionary.

Parameters **path** (Optional[Path]) – Optional new path to save to.

clean(*word*)

Return type str

3.1.5 CorrectOCR.fileio module

class CorrectOCR.fileio.**FileIO**

Bases: object

Various file IO helper methods.

cacheRoot = None

classmethod cachePath(*name=''*)

classmethod imageCache(*name=None*)

classmethod get_encoding(*file*)

Get encoding of a text file.

Parameters **file** (Path) – A path to a text file.

Return type str

Returns The encoding of the file, eg. 'utf-8', 'Windows-1252', etc.

classmethod ensure_new_file(*path*)

Moves a possible existing file out of the way by adding a numeric counter before the extension.

Parameters **path** (Path) – The path to check.

classmethod ensure_directories(*path*)

Ensures that the entire path exists.

Parameters **path** (Path) – The path to check.

classmethod copy(*src*, *dest*)

Copies a file.

Parameters

- **src** (Path) – Source-path.
- **dest** (Path) – Destination-path.

classmethod delete(*path*)

Deletes a file.

Parameters **path** (Path) – The path to delete.

classmethod save(*data*, *path*, *backup=True*)

Saves data into a file. The extension determines the method of saving:

- *.pickle* – uses pickle.
- *.json* – uses json.
- *.csv* – uses csv.DictWriter (assumes data is list of vars()-capable objects). The keys of the first object determines the header.

Any other extension will simply write() the data to the file.

Parameters

- **data** (Any) – The data to save.
- **path** (Path) – The path to save to.
- **backup** – Whether to move existing files out of the way via *ensure_new_file()*

classmethod load(*path*, *default=None*)

Loads data from a file. The extension determines the method of saving:

- *.pickle* – uses pickle.
- *.json* – uses json.
- *.csv* – uses csv.DictReader.

Any other extension will simply read() the data from the file.

Parameters

- **path** (Path) – The path to load from.
- **default** – If file doesn't exist, return default instead.

Returns The data from the file, or the default.

3.1.6 CorrectOCR.heuristics module

3.1.6.1 Heuristics

A given token and its *k*-best candidates are compared and checked with the dictionary. Based on this, it is matched with a *bin*.

bin	1	2	3	4	5	6	7	8	9
k = orig?	T	T	T	F	F	F	F	F	F
orig in dict?	T	F	F	F	F	F	T	T	T
top *k*-best in dict?	T	F	F	T	F	F	T	F	F
lower-ranked *k*-best in dict?	–	F	T	–	F	T	–	F	T

Each *bin* must be assigned a setting that determines what decision is made:

- `o` / *original*: select the original token as correct.
- `k` / *kbest*: select the top *k*-best candidate as correct.
- `d` / *kdict*: select the first lower-ranked candidate that is in the dictionary.
- `a` / *annotator*: defer selection to annotator.

Once the report and settings are generated, it is not strictly necessary to update them every single time the model is updated. It is however a good idea to do it regularly as the corpus grows and more tokens become available for the statistics.

class CorrectOCR.heuristics.**Heuristics**(*settings*, *dictionary*)

Bases: object

Parameters

- **settings** (Dict[int, str]) – A dictionary of `bin number => heuristic` settings.
- **dictionary** – A dictionary for determining correctness of *Tokens* and suggestions.

classmethod bin(*n*)

Return type *Bin*

bin_for_word(*original*, *kbest*)

bin_tokens(*tokens*, *force=False*)

Return type bool

add_to_report(*tokens*, *rebin=False*, *hmm=None*)

report()

Return type str

class CorrectOCR.heuristics.**Bin**(*description*, *matcher*, *heuristic='annotator'*, *number=None*, *counts=<factory>*, *example=None*)

Bases: object

Heuristics bin ...

TODO TABLE

description: str

Description of bin

matcher: Callable[[str, str, *Dictionary*, str], bool]

Function or lambda which returns *True* if a given *CorrectOCR.tokens.Token* fits into the bin, or *False* otherwise.

Parameters

- **o** – Original string
- **k** – *k*-best candidate string
- **d** – Dictionary
- **dcode** – One of 'zerokd', 'somekd', 'allkd' for whether zero, some, or all other *k*-best candidates are in dictionary

heuristic: str = 'annotator'

Which heuristic the bin is set up for, one of:

- 'annotator' = Defer to annotator.
- 'original' = Select original.
- 'kbest' = Select top *k*-best.
- 'kdict' = Select top *k*-best in dictionary.

number: int = None

The number of the bin.

counts: DefaultDict[str, int]

Statistics used for reporting.

example: original, gold, kbest = None

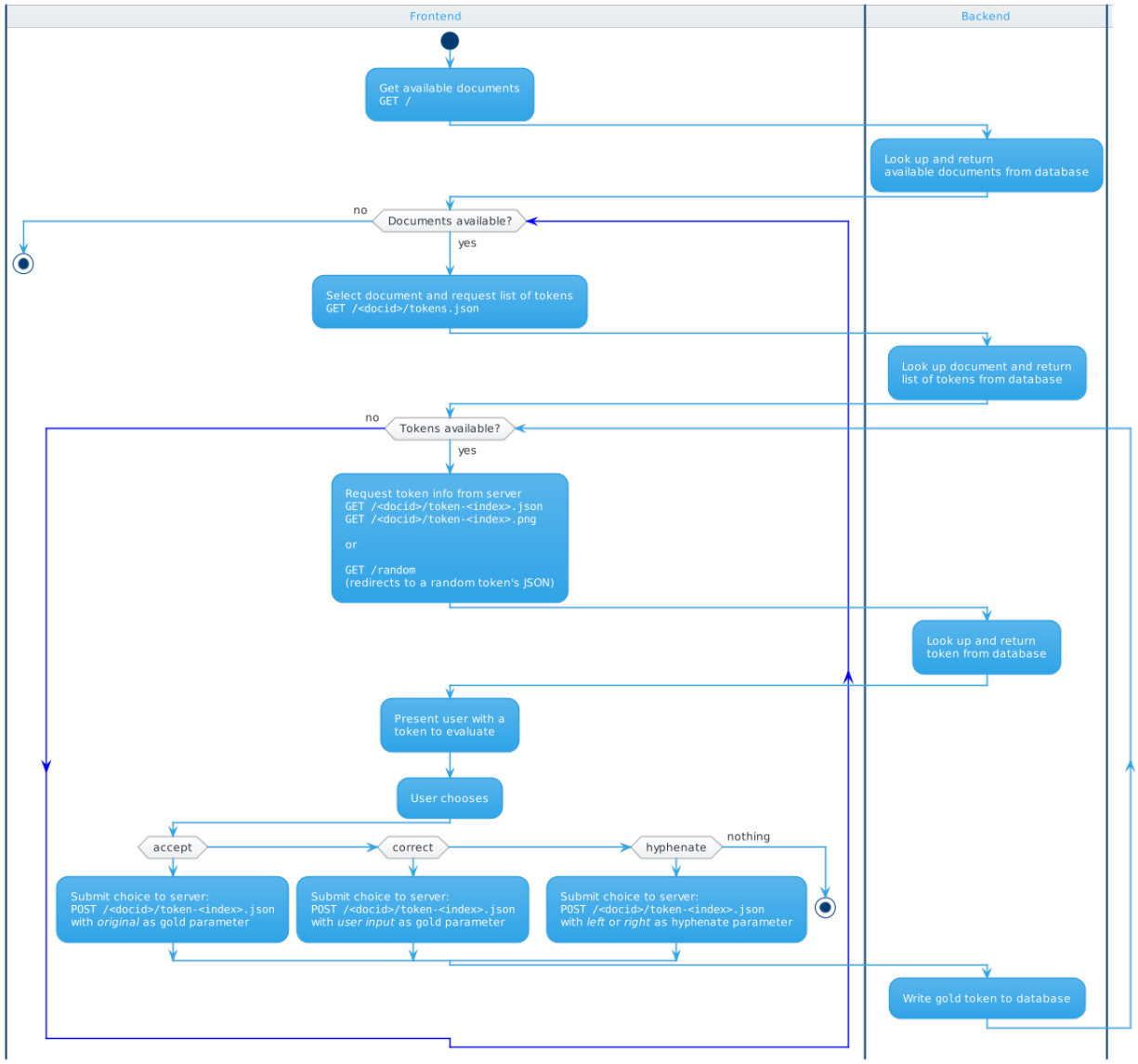
An example of a match, used for reporting.

3.1.7 CorrectOCR.model module

3.1.8 CorrectOCR.server module

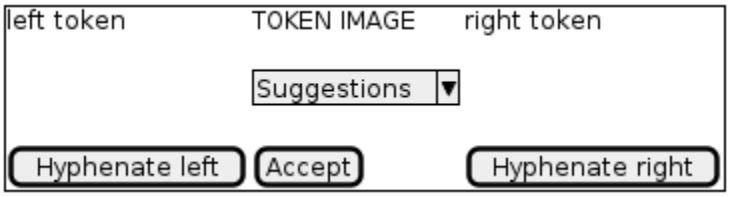
Below are some examples for a possible frontend. Naturally, they are only suggestions and any workflow and interface may be used.

3.1.8.1 Example Workflow



Open the image in a new window to view at size.

3.1.8.2 Example User Interface



The Combo box would then contain the k-best suggestions from the backend, allowing the user to accept the desired one or enter their own correction.

Showing the left and right tokens (ie. tokens with index±1) enables to user to decide if a token is part of a longer word that should be hyphenated.

3.1.8.3 Endpoint Documentation

Errors are specified according to RFC 7807 Problem Details for HTTP APIs.

Resource	Operation	Description
1 Main	*GET /*	Get list of documents
2 Documents	*GET /(string:doc_id)/tokens.json*	Get list of tokens in document
3 Tokens	*GET /random*	Get random token
	GET /(string:doc_id)/token-(int:doc_index).json	Get token
	POST /(string:doc_id)/token-(int:doc_index).json	Update token
	GET /(string:doc_id)/token-(int:doc_index).png	Get token image

GET /

Get an overview of the documents available for correction.

The list will not include documents that the backend considers 'done', but they can still be accesses via the other endpoints.

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "docid": "<docid>",
    "url": "/<docid>/tokens.json",
    "info_url": "...",
    "count": 100,
    "corrected": 87,
    "corrected_by_model": 80,
    "discarded": 10,
    "last_modified": 1605255523
  }
]
```

Response JSON Array of Objects

- **docid** (*string*) – ID for the document.
- **url** (*string*) – URL to list of Tokens in doc.
- **info_url** (*string*) – URL that provides more info about the document. See workspace.docInfoBaseURL
- **count** (*int*) – Total number of Tokens.
- **corrected** (*int*) – Number of corrected Tokens.
- **corrected_by_model** (*int*) – Number of Tokens that were automatically corrected by the model.
- **discarded** (*int*) – Number of discarded Tokens.

- **last_modified** (*int*) – The date/time of the last modified token.

GET /(string: *doc_id*)**/token-int:** *doc_index*.**json**

Get information about a specific *Token*.

Returns `404` if the document or token cannot be found, otherwise `200`.

Note: If the token is the second part of a hyphenated token, and the server is configured for it, a `302`-redirect to the previous token will be returned.

Note: The data is not escaped; care must be taken when displaying in a browser.

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "Bin": 2,
  "Heuristic": "annotator",
  "Doc ID": "7696",
  "Gold": "",
  "Hyphenated": false,
  "Discarded": false,
  "Index": 2676,
  "Original": "Jornben.",
  "Selection": [],
  "Token info": "...",
  "Token type": "PDFToken",
  "Page": 1,
  "Frame": [0, 0, 100, 100],
  "Annotation info": "...",
  "image_url": "/7696/token-2676.png"
  "k-best": {
        1: { "candidate": "Jornben", "probability": 2.96675056066388e-08 },
        2: { "candidate": "Joreben", "probability": 7.41372275428713e-10 },
        3: { "candidate": "Jornhen", "probability": 6.17986300962785e-10 },
        4: { "candidate": "Joraben", "probability": 5.52540106969346e-10 }
  },
  "Last Modified": 1605255523
}
```

Parameters

- **doc_id** (*string*) – The ID of the requested document.
- **doc_index** (*int*) – The placement of the requested Token in the document.

Return A JSON dictionary of information about the requested *Token*. Relevant keys for frontend display include *original* (uncorrected OCR result), *gold* (corrected version, if available). For further information, see the Token class.

POST /(string: *doc_id*)**/token-int:** *doc_index*.**json**

Update a given token with a *gold* transcription and/or hyphenation info.

Returns `404` if the document or token cannot be found, otherwise `200`.

If an invalid `hyphenate` value is submitted, status code `400` will be returned.

Note: If `gold` and `hyphenate` are supplied, the `gold` value will be inspected. If it contains a hyphen, the left and right parts will be set on the respective tokens. If it does not, the gold will be set on the leftmost token, and the right one discarded.

Note: If the hyphenation is set to `left`, a redirect to the new "head" token will be returned.

Parameters

- **docid** (*string*) – The ID of the requested document.
- **index** (*int*) – The placement of the requested Token in the document.

Request JSON Object

- **gold** (*string*) – Set new correction for this Token.
- **info** (*string annotation*) – Save some metadata about this correction (eg. username, date). Will only be saved if there is a gold correction.
- **hyphenate** (*string*) – Optionally hyphenate to the *left* or *right*.

Return A JSON dictionary of information about the updated *Token*. *NB*: If the hyphenation is set to `left`, a redirect to the new "head" token will be returned.

GET /(string: *doc_id*)**/token-int:** *doc_index*.**png**

Returns a snippet of the original document as an image, for comparing with the OCR result.

Returns `404` if the document or token cannot be found, otherwise `200`.

Parameters

- **docid** (*string*) – The ID of the requested document.
- **index** (*int*) – The placement of the requested Token in the document.

Query Parameters

- **leftmargin** (*int*) – Optional left margin. See `PDFToken.extract_image()` for defaults. TODO
- **rightmargin** (*int*) – Optional right margin.
- **topmargin** (*int*) – Optional top margin.
- **bottommargin** (*int*) – Optional bottom margin.

Return A PNG image of the requested *Token*.

GET /(string: *doc_id*)**/tokens.json**

Get information about the *Tokens* in a given document.

Returns `404` if the document cannot be found, otherwise `200`.

Parameters

- **docid** (*string*) – The ID of the requested document.

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "info_url": "/<docid>/token-0.json",
    "image_url": "/<docid>/token-0.png",
    "string": "Example",
    "is_corrected": true,
    "is_discarded": false,
    "requires_annotator": false,
    "last_modified": 1605255523
  },
  {
    "info_url": "/<docid>/token-1.json",
    "image_url": "/<docid>/token-1.png",
    "string": "Exanpie",
    "is_corrected": false,
    "is_discarded": false,
    "requires_annotator": true,
    "has_error": false,
    "last_modified": null
  }
]
```

Response JSON Array of Objects

- **info_url** (*string*) – URL to Token info.
- **image_url** (*string*) – URL to Token image.
- **string** (*string*) – Current Token string.
- **is_corrected** (*bool*) – Whether the Token has been corrected at the moment.
- **is_discarded** (*bool*) – Whether the Token has been discarded at the moment.
- **last_modified** (*bool*) – The date/time when the token was last modified.

GET /random

Returns a 302-redirect to a random token from a random document. TODO: filter by needing annotator

Example response:

```
HTTP/1.1 302 Found
Location: /<docid>/token-<index>.json
```

3.1.9 CorrectOCR.tokens module

class CorrectOCR.tokens.**Token**(*original*, *docid*, *index*)

Bases: abc.ABC

Abstract base class. Tokens handle single words. ...

Parameters

- **original** (str) – Original spelling of the token.
- **docid** (str) – The doc with which the Token is associated.

static register(*cls*)

Decorator which registers a *Token* subclass with the base class.

Parameters **cls** – Token subclass

docid

The doc with which the Token is associated.

index

The placement of the Token in the doc.

gold

bin: Optional[*CorrectOCR.heuristics.Bin*]

Heuristics bin.

kbest: DefaultDict[int, CorrectOCR.model.kbest.KBestItem]

Dictionary of *k*-best suggestions for the Token. They are keyed with a numerical index starting at 1, and the values are instances of KBestItem.

heuristic: Optional[str]

The heuristic that was was determined by the bin.

selection: Any

The selected automatic correction for the *heuristic*.

is_hyphenated

is_discarded

(documented in @property methods below)

annotations

A list of arbitrary key/value info about the annotations

has_error

Whether the token has an unhandled error

last_modified

When one of the gold, is_hyphenated, is_discarded, or has_error properties were last updated.

cached_image_path

Where the image file should be cached. Is not guaranteed to exist, but can be generated via extract_image()

abstract property token_info: Any

Return type Any

Returns

abstract property page: int

The page of the document on which the token is located.

May not be applicable for all token types.

Return type int

Returns The page number.

abstract property frame: int, int, int, int

The coordinates of the token's location on the page.

Takes the form [x0, y0, x1, y1] where (x0, y0) is the top-left corner, and (x1, y1) is the bottom-right corner.

May not be applicable for all token types.

Returns The frame coordinates.

property k: int

The number of *k*-best suggestions for the Token.

Return type int

is_punctuation()

Is the Token purely punctuation?

Return type bool

is_numeric()

Is the Token purely numeric?

Return type bool

classmethod from_dict(*d*)

Initialize and return a new Token with values from a dictionary.

Parameters **d** (dict) – A dictionary of properties for the Token

Return type *Token*

drop_cached_image()

extract_image(*workspace*, *highlight_word=True*, *left=300*, *right=300*, *top=15*, *bottom=15*, *force=False*)

Return type Tuple[Path, Any]

class CorrectOCR.tokens.**Tokenizer(*language*)**

Bases: abc.ABC

Abstract base class. The *Tokenizer* subclasses handle extracting *Token* instances from a document.

Parameters **language** (pycountry.Language) – The language to use for tokenization (for example, the *.txt* tokenizer internally uses nltk whose tokenizers function best with a language parameter).

static register(*extensions*)

Decorator which registers a *Tokenizer* subclass with the base class.

Parameters **extensions** (List[str]) – List of extensions that the subclass will handle

static for_extension(*ext*)

Obtain the suitable subclass for the given extension. Currently, Tokenizers are provided for the following extensions:

- `.txt` – plain old text.
- `.pdf` – assumes the PDF contains images and OCRed text.
- `.tiff` – will run OCR on the image and generate a PDF.
- `.png` – will run OCR on the image and generate a PDF.

Parameters **ext** (str) – Filename extension (including leading period).

Return type ABCMeta

Returns A Tokenizer subclass.

abstract tokenize(*file*, *storageconfig*)

Generate tokens for the given document.

Parameters

- **storageconfig** – Storage configuration (database, filesystem) for resulting Tokens
- **file** (Path) – A given document.

Return type *TokenList*

Returns

abstract static apply(*original*, *tokens*, *outfile*, *highlight=False*)

abstract static crop_tokens(*original*, *config*, *tokens*, *edge_left=None*, *edge_right=None*)

class CorrectOCR.tokens.**TokenList**(*config*, *docid=None*, *tokens=None*)

Bases: collections.abc.MutableSequence

static register(*storagetype*)

Decorator which registers a *TokenList* subclass with the base class.

Parameters **storagetype** (str) – *fs* or *db*

static new(*config*, *docid=None*, *tokens=None*)

Return type *TokenList*

static for_type(*type*)

Return type ABCMeta

insert(*key*, *value*)

S.insert(index, value) – insert value before index

abstract load()

abstract save(*token=None*)

preload()

flush()

property stats

classmethod validate_stats(*docid*, *stats*)

property consolidated: Tuple[str, str, *Token*]

A consolidated iterator of tokens, where discarded tokens are skipped, and hyphenated original/gold are included.

:returns original, gold, token

Return type Tuple[str, str, *Token*]

property server_ready

random_token_index(*has_gold=False*, *is_discarded=False*)

random_token(*has_gold=False*, *is_discarded=False*)

property overview

Generator that returns an fast overview of the TokenList.

Each item is a dictionary containing the following keys:

- `doc_id`: The document
- `doc_index`: The Token's placement in the document
- `string`: TODO
- `is_corrected`: Whether the Token has a set gold property
- `is_discarded`: Whether the Token is marked as discarded

property last_modified

dehyphenate()

CorrectOCR.tokens.**tokenize_str**(*data*, *language='english'*)

Return type List[str]

3.1.10 CorrectOCR.workspace module

class CorrectOCR.workspace.**LazyDocumentDict**(*workspace*, **args*, ***kargs*)

Bases: collections.abc.MutableMapping

class CorrectOCR.workspace.**Workspace**(*workspaceconfig*, *resourceconfig*, *storageconfig*)

Bases: object

The Workspace holds references to `Documents` and resources used by the various `commands`.

Parameters

- **workspaceconfig** – An object with the following properties:
  - **nheaderlines** (int): The number of header lines in corpus texts.
  - **language**: A language instance from *pycountry <https://pypi.org/project/pycountry/>*.
  - **originalPath** (Path): Directory containing the original docs.
  - **goldPath** (Path): Directory containing the gold (if any) docs.
  - **trainingPath** (Path): Directory for storing intermediate docs.
  - **docInfoBaseURL** (str): Base URL that when appended with a doc_id provides information about documents.

- **resourceconfig** – Passed directly to *ResourceManager*, see this for further info.
- **storageconfig** – TODO

add_doc(*doc*)

Initializes a new `Document` and adds it to the workspace.

The doc_id of the document will be determined by its filename.

If the file is not in the originalPath, it will be copied or downloaded there.

Parameters **doc** (`Any`) – A path or URL.

Return type `str`

documents(*ext=None*, *server_ready=False*, *is_done=False*)

Yields documents filtered by the given criteria.

Param ext Only include docs with this extension.

Param server_ready Only include documents that are ready (prepared).

Param is_done Only include documents that are done (all tokens have gold).

Return type `List[str]`

cleanup(*dryrun=True*, *full=False*)

Cleans out the backup files in the `trainingPath`.

Parameters

- **dryrun** – Just lists the files without actually deleting them
- **full** – Also deletes the current files (ie. those without .nnn. in their suffix).

class CorrectOCR.workspace.**CorpusFile**(*path*, *nheaderlines=0*)

Bases: `object`

Simple wrapper for text files to manage a number of lines as a separate header.

Parameters

- **path** (`Path`) – Path to text file.
- **nheaderlines** (`int`) – Number of lines from beginning to separate out as header.

save()

Concatenate header and body and save.

is_file()

Return type `bool`

Returns Does the file exist? See `pathlib.Path.is_file()`.

property id

class CorrectOCR.workspace.**JSONResource**(*path*, ***kwargs*)

Bases: `dict`

Simple wrapper for JSON files.

Parameters

- **path** – Path to load from.
- **kwargs** – TODO

save()

Save to JSON file.

class CorrectOCR.workspace.**ResourceManager**(*root*, *config*)

Bases: object

Helper for the Workspace to manage various resources.

Parameters

- **root** (Path) – Path to resources directory.
- **config** – An object with the following properties:
  - **correctionTrackingFile** (Path): Path to file containing correction tracking.
  - TODO

CHAPTER

FOUR

HISTORY

CorrectOCR is based on code created by:

- Caitlin Richter (ricca@seas.upenn.edu)
- Matthew Wickes (wickesm@seas.upenn.edu)
- Deniz Beser (dbeser@seas.upenn.edu)
- Mitchell Marcus (mitch@cis.upenn.edu)

See their article *"Low-resource Post Processing of Noisy OCR Output for Historical Corpus Digitisation"* (LREC-2018) for further details, it is available online: http://www.lrec-conf.org/proceedings/lrec2018/pdf/971.pdf

The original python 2.7 code (see `original`-tag in the repository) has been licensed under Creative Commons Attribution 4.0 (CC-BY-4.0, see also `license.txt` in the repository).

The code has subsequently been updated to Python 3 and further expanded by Mikkel Eide Eriksen (mikkel.eriksen@gmail.com) for the Copenhagen City Archives (mainly structural changes, the algorithms are generally preserved as-is). Pull requests welcome!

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

HTTP ROUTING TABLE

/

/(string:doc_id)

/random

INDEX

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

V

W