CorrectOCR Documentation

Mikkel Eide Eriksen, for Copenhagen City Archives

Sep 27, 2020

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 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

CONFIGURATION

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. For example:

```
[configuration]
characterSet = ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

[workspace]
correctedPath = corrected/
goldPath = gold/
originalPath = original/
trainingPath = training/
nheaderlines = 0

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

[storage]
type = fs
```

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.
- `corrected/` contains the corrected files generated by running the `correct` command. If necessary, it can be configured with the `--correctedPath` argument.

Corresponding files in *original*, *gold*, and *corrected* 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.

CHAPTER

THREE

COMMANDS

Commands and their arguments are called directly on the module, like so:

```
python -m CorrectOCR [command] [args...]
```

The following commands are available:

- `build_dictionary` creates a dictionary. Input files can be either `.pdf`, `.txt`, or `.xml` (in TEI format). They may be contained in `.zip`-files.
  - The `--corpusPath` option specifies a directory of files.
  - The `--corpusFile` option specifies a file containing paths and URLs. One such file for a dictionary covering 1800–1948 Danish is provided under `resources/`.
  - The `--clear` option clears the dictionary before adding words (the file is backed up first).

  It is strongly recommended to generate a large dictionary for best performance.
- `align` aligns a pair of (original, gold) files in order to determine which characters and words were misread in the original and corrected in the gold.
  - The `--fileid` option specifies a single pair of files to align.
  - The `--all` option aligns all available pairs. Can be combined with `--exclude` to skip specific files.
- `build_model` uses the alignments to create parameters for the HMM.
  - The `--smoothingParameter` option can be adjusted as needed.
- `add` copies or downloads files to the workspace. One may provide about a single file directly, or use the option to provide a list of files.
  - The `--documents` option specifies a file containing paths and URLs.
  - The `--max_count` option specifies the maximum number of files to add.
  - The `--prepare_step` option allows the automatic preparation of the files as they are added. See below.
- `prepare` tokenizes and prepare texts for corrections.
  - The `--fileid` option specifies which file to tokenize.
  - The `--all` option tokenizes all available texts. Can be combined with `--exclude` to skip specific files.
  - The `--step` option specifies how many of the processing steps to take. The default is to take all steps.
    - `tokenize` simply splits the text into tokens (words).
    - `align` aligns tokens with gold versions, if these exist.
    - `kbest` calculates *k*-best correction candidates for each token via the HMM.

    * `bin` sorts the tokens into *bins* according to the *heuristics* below.

  Each of the steps includes the previous step, and will save intermediary information about each token to CSV or a databases.

- `stats` is used to configure which decisions the program should make about each bin of tokens:
  - `--make_report` generates a statistical report on whether originals/*k*-best equal are in the dictionary, etc. This report can then be inspected and annotated with the desired decision for each *bin*.
  - `--make_settings` creates correction settings based on the annotated report.
- `correct` uses the settings to sort the tokens into bins and makes automated decisions as configured.
  - The `--fileid` option specifies which file to correct.

  There are three ways to run corrections:

  - `--interactive` runs an interactive correction CLI for the remaining undecided tokens (see *Correction Interface* below).
  - `--apply` takes a path argument to an edited token CSV file and applies the corrections therein.
  - `--autocorrect` applies available corrections as configured in correction settings (ie. any heuristic bins not marked for human annotation).
- `index` finds specified terms for use in index-generation.
  - The `--fileid` option specifies a single file for which to generate an index.
  - The `--all` option generates indices for all available files. Can be combined with `--exclude` to skip specific files.
  - The `--termFile` option specifies a text file containing a word on each line, which will be matched against the tokens. The option may be repeated, and each filename (without extension) will be used as markers for the string.
  - The `--highlight` option will create a copy of the input files with highlighted words (only available for PDFs).
  - The `--autocorrect` option applies available corrections prior to search/highlighting, as above.
- `server` starts a simple Flask backend server that provides `JSON` descriptions and `.png` images of tokens, as well as accepts `POST`-requests to update tokens with corrections.
- `cleanup` deletes the backup files in the training directory.
  - The `--dryrun` option simply lists the files without actually deleting them.
  - The `--full` option also deletes the current files (ie. those without .nnn. in their suffix).

CHAPTER

FOUR

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.

CHAPTER

FIVE

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.

CHAPTER

SIX

CORRECTOCR PACKAGE

6.1 Submodules

6.1.1 CorrectOCR.aligner module

class CorrectOCR.aligner.**Aligner**
Bases: object

alignments(*originalTokens*, *goldTokens*)
Aligns the original and gold tokens in order to discover the corrections that have been made.

Parameters

- **originalTokens** (List[str]) – List of original text strings
- **goldTokens** (List[str]) – List of gold text strings

Returns

A tuple with three elements:

- fullAlignments – A list of letter-by-letter alignments (2-element tuples)
- wordAlignments–
- readCounts – A dictionary of counts of aligned reads for each character.

6.1.2 CorrectOCR.commands module

6.1.2.1 Commands

Commands and their arguments are called directly on the module, like so:

```
python -m CorrectOCR [command] [args...]
```

The following commands are available:

- build_dictionary creates a dictionary. Input files can be either .pdf, .txt, or .xml (in TEI format). They may be contained in .zip-files.
  - The --corpusPath option specifies a directory of files.
  - The --corpusFile option specifies a file containing paths and URLs. One such file for a dictionary covering 1800–1948 Danish is provided under resources/.
  - The --clear option clears the dictionary before adding words (the file is backed up first).

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

- `align` aligns a pair of (original, gold) files in order to determine which characters and words were misread in the original and corrected in the gold.
  - The `--fileid` option specifies a single pair of files to align.
  - The `--all` option aligns all available pairs. Can be combined with `--exclude` to skip specific files.
- `build_model` uses the alignments to create parameters for the HMM.
  - The `--smoothingParameter` option can be adjusted as needed.
- `add` copies or downloads files to the workspace. One may provide about a single file directly, or use the option to provide a list of files.
  - The `--documents` option specifies a file containing paths and URLs.
  - The `--max_count` option specifies the maximum number of files to add.
  - The `--prepare_step` option allows the automatic preparation of the files as they are added. See below.
- `prepare` tokenizes and prepare texts for corrections.
  - The `--fileid` option specifies which file to tokenize.
  - The `--all` option tokenizes all available texts. Can be combined with `--exclude` to skip specific files.
  - The `--step` option specifies how many of the processing steps to take. The default is to take all steps.
    - `tokenize` simply splits the text into tokens (words).
    - `align` aligns tokens with gold versions, if these exist.
    - `kbest` calculates *k*-best correction candidates for each token via the HMM.
    - `bin` sorts the tokens into *bins* according to the *heuristics* below.

    Each of the steps includes the previous step, and will save intermediary information about each token to CSV or a databases.
- `stats` is used to configure which decisions the program should make about each bin of tokens:
  - `--make_report` generates a statistical report on whether originals/*k*-best equal are in the dictionary, etc. This report can then be inspected and annotated with the desired decision for each *bin*.
  - `--make_settings` creates correction settings based on the annotated report.
- `correct` uses the settings to sort the tokens into bins and makes automated decisions as configured.
  - The `--fileid` option specifies which file to correct.

  There are three ways to run corrections:
  - `--interactive` runs an interactive correction CLI for the remaining undecided tokens (see *Correction Interface* below).
  - `--apply` takes a path argument to an edited token CSV file and applies the corrections therein.
  - `--autocorrect` applies available corrections as configured in correction settings (ie. any heuristic bins not marked for human annotation).
- `index` finds specified terms for use in index-generation.
  - The `--fileid` option specifies a single file for which to generate an index.
  - The `--all` option generates indices for all available files. Can be combined with `--exclude` to skip specific files.

  - The `--termFile` option specifies a text file containing a word on each line, which will be matched against the tokens. The option may be repeated, and each filename (without extension) will be used as markers for the string.
  - The `--highlight` option will create a copy of the input files with highlighted words (only available for PDFs).
  - The `--autocorrect` option applies available corrections prior to search/highlighting, as above.
- `server` starts a simple Flask backend server that provides `JSON` descriptions and `.png` images of tokens, as well as accepts `POST`-requests to update tokens with corrections.
- `cleanup` deletes the backup files in the training directory.
  - The `--dryrun` option simply lists the files without actually deleting them.
  - The `--full` option also deletes the current files (ie. those without .nnn. in their suffix).

`CorrectOCR.commands.`**`build_dictionary`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_align`**(*workspace*, *config*)

`CorrectOCR.commands.`**`build_model`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_add`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_prepare`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_crop`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_stats`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_correct`**(*workspace*, *config*)

`CorrectOCR.commands.`**`make_gold`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_index`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_cleanup`**(*workspace*, *config*)

`CorrectOCR.commands.`**`do_extract`**(*workspace*, *config*)

`CorrectOCR.commands.`**`run_server`**(*workspace*, *config*)

6.1.3 CorrectOCR.correcter module

6.1.3.1 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.

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 decision for another time.

do_quit(_)

6.1.4 CorrectOCR.dictionary module

class CorrectOCR.dictionary.**Dictionary**(*path=None*, *ignoreCase=False*)
Bases: set, typing.Generic

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

Parameters

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

clear()
Remove all elements from this set.

add(*word*, *nowarn=False*)
Add a new word to the dictionary. Silently drops non-alpha strings.

Parameters

- **word** (str) – The word to add.
- **nowarn** (bool) – Don't warn about long words (>15 letters).

save(*path=None*)
Save the dictionary.

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

6.1.5 CorrectOCR.fileio module

class CorrectOCR.fileio.**FileIO**
Bases: object

Various file IO helper methods.

classmethod cachePath(*name=''*)

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.

6.1.6 CorrectOCR.heuristics module

6.1.6.1 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 => heuristic settings.
- **dictionary** – A dictionary for determining correctness of *Tokens* and suggestions.

classmethod bin(*n*)

Return type *Bin*

bin_for_token(*token*)

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

add_to_report(*token*)

report()

Return type str

class CorrectOCR.heuristics.**Bin**(*description*, *matcher*, *heuristic='a'*, *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 = 'a'
Which heuristic the bin is set up for, one of:

- 'a' = Defer to annotator.
- 'o' = Select original.
- 'k' = Select top *k*-best.
- 'd' = Select *k*-best in dictionary.

number: int = None
The number of the bin.

counts: DefaultDict[str, int]
Statistics used for reporting.

example: Token = None
An example of a matching *CorrectOCR.tokens.Token*, used for reporting.

6.1.7 CorrectOCR.model module

class CorrectOCR.model.**HMM**(*path*, *multichars=None*, *dictionary=None*)
Bases: object

Parameters

- **path** (Path) – Path for loading and saving.
- **multichars** – A dictionary of possible multicharacter substitutions (eg. 'cr': 'æ' or vice versa).
- **dictionary** (Optional[*Dictionary*]) – The dictionary against which to check validity.

property init
Initial probabilities.

Return type DefaultDict[str, float]

property tran
Transition probabilities.

Return type DefaultDict[str, DefaultDict[str, float]]

property emis
Emission probabilities.

Return type DefaultDict[str, DefaultDict[str, float]]

save(*path=None*)
Save the HMM parameters.

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

is_valid()
Verify that parameters are valid (ie. the keys in init/tran/emis match).

Return type bool

viterbi(*char_seq*)
TODO

Parameters **char_seq** (Sequence[str]) –

Return type str

Returns

kbest_for_word(*word*, *k*)
Generates *k*-best correction candidates for a single word.

Parameters

- **word** (str) – The word for which to generate candidates
- **k** (int) – How many candidates to generate.

Return type DefaultDict[int, KBestItem]

Returns A dictionary with ranked candidates keyed by 1..*k*.

generate_kbest(*tokens*, *k=4*, *force=False*)
Generates *k*-best correction candidates for a list of Tokens and adds them to each token.

Parameters

- **tokens** (TokenList) – List of tokens.
- **k** (int) – How many candidates to generate.

class CorrectOCR.model.**HMMBuilder**(*dictionary*, *smoothingParameter*, *characterSet*, *readCounts*, *remove_chars*, *gold_words*)

Bases: object

Calculates parameters for a HMM based on the input. They can be accessed via the three properties.

Parameters

- **dictionary** (*Dictionary*) – The dictionary to use for generating probabilities.
- **smoothingParameter** (float) – Lower bound for probabilities.
- **characterSet** – Set of required characters for the final HMM.
- **readCounts** – See *Aligner*.
- **remove_chars** (List[str]) – List of characters to remove from the final HMM.
- **gold_words** (List[str]) – List of known correct words.

emis: DefaultDict[str, float]
Emission probabilities.

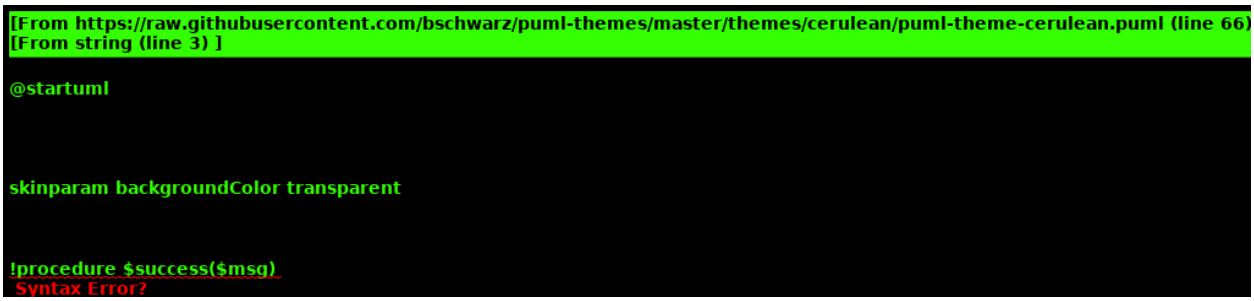
init: DefaultDict[str, float]
Initial probabilities.

tran: DefaultDict[str, DefaultDict[str, float]]
Transition probabilities.

6.1.8 CorrectOCR.server module

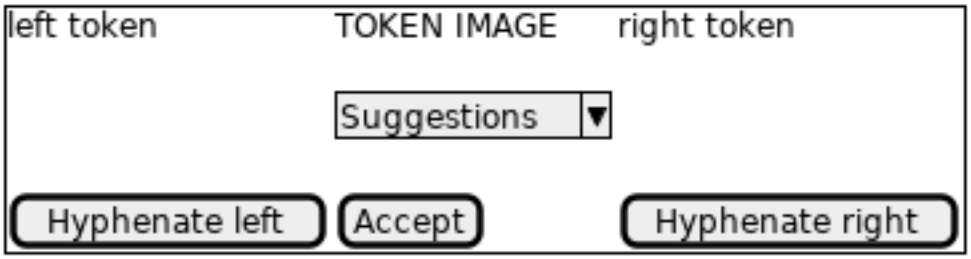
Below are some examples for a possible frontend. Naturally, they are only suggestions and any workflow and interface may be used.

6.1.8.1 Example Workflow



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

6.1.8.2 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$\pm$1) enables to user to decide if a token is part of a longer word that should be hyphenated.

6.1.8.3 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:docid)/tokens.json*	Get list of tokens in document
3 Tokens	*GET /random*	Get random token
	GET /(string:docid)/token-(int:index).json	Get token
	POST /(string:docid)/token-(int:index).json	Update token
	GET /(string:docid)/token-(int:index).png	Get token image

GET /

Get an overview of the documents available for correction.

Example response:

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

[
  {
    "docid": "<docid>",
    "url": "/<docid>/tokens.json",
    "count": 100,
    "corrected": 87
  }
]
```

Response JSON Array of Objects

- **docid** (*string*) – ID for the document.
- **url** (*string*) – URL to list of Tokens in doc.
- **count** (*int*) – Total number of Tokens.
- **corrected** (*int*) – Number of corrected Tokens.

GET /(string: *docid*)**/token-int:** *index*.**json** Get information about a specific *Token*

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

{
  "1-best": "Jornben",
  "1-best prob.": 2.96675056066388e-08,
  "2-best": "Joreben",
  "2-best prob.": 7.41372275428713e-10,
  "3-best": "Jornhen",
  "3-best prob.": 6.17986300962785e-10,
  "4-best": "Joraben",
  "4-best prob.": 5.52540106969346e-10,
  "Bin": 2,
  "Decision": "annotator",
  "Doc ID": "7696",
  "Gold": "",
  "Heuristic": "a",
      "Hyphenated": false,
      "Discarded": false,
  "Index": 2676,
  "Original": "Jornben.",
  "Selection": [],
  "Token info": "...",
  "Token type": "PDFToken",
  "image_url": "/7696/token-2676.png"
}
```

Parameters

- **docid** (*string*) – The ID of the requested document.
- **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 are *original* (uncorrected OCR result), *gold* (corrected version, if available), TODO

POST /(string: *docid*)**/token-int:** *index*.**json** Update a given token with a *gold* transcription and/or hyphenation info.

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.
- **hyphenate** (*string*) – Optionally hyphenate to the *left* or *right*.

Return A JSON dictionary of information about the updated *Token*.

GET /(string: *docid*)**/token-int:** *index*.**png** Returns a snippet of the original document as an image, for comparing with the OCR result.

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: *docid*)/tokens.json

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

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
  },
  {
    "info_url": "/<docid>/token-1.json",
    "image_url": "/<docid>/token-1.png",
    "string": "Exanpie",
    "is_corrected": false
  }
]
```

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.

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
```

6.1.9 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.

bin: Optional[Bin]
Heuristics bin.

kbest: DefaultDict[int, *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*.

decision: Optional[str]
The decision that was made when *gold* was set automatically.

selection: Any
The selected automatic correction for the *decision*.

is_hyphenated
Whether the token is hyphenated to the following token.

is_discarded
Whether the token has been discarded (marked irrelevant by code or annotator).

abstract property token_info

Return type Any

Returns

property original
The original spelling of the Token.

Return type str

property gold
The corrected spelling of the Token.

Return type str

property k
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

class CorrectOCR.tokens.**Tokenizer**(*language*, *dehyphenate*)
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*, *corrected*)

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

static exists(*config*, *docid*)

Return type bool

abstract load(*docid*)

abstract save(*token=None*)

property corrected_count

property discarded_count

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

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

class CorrectOCR.tokens.**KBestItem**(*candidate*, *probability*)
Bases: tuple

Create new instance of KBestItem(candidate, probability)

property candidate
Alias for field number 0

property probability
Alias for field number 1

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

Return type List[str]

6.1.10 CorrectOCR.workspace module

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.

- **correctedPath** (Path): Directory for saving corrected docs.

- **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

docids_for_ext(*ext*)
Returns a list of IDs for documents with the given extension.

Return type List[str]

original_tokens()
Yields an iterator of (docid, list of tokens).

Return type Iterator[Tuple[str, TokenList]]

gold_tokens()
Yields an iterator of (docid, list of gold-aligned tokens).

Return type Iterator[Tuple[str, TokenList]]

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.**Document**(*workspace*, *doc*, *original*, *gold*, *training*, *corrected*, *nheaderlines=0*)

Bases: object

Parameters

- **doc** (Path) – A path to a file.
- **original** (Path) – Directory for original uncorrected files.
- **gold** (Path) – Directory for known correct "gold" files (if any).
- **training** (Path) – Directory for storing intermediate files.
- **corrected** (Path) – Directory for saving corrected files.
- **nheaderlines** (int) – Number of lines in file header (only relevant for .txt files)

tokenFile
Path to token file (CSV format).

fullAlignmentsFile
Path to full letter-by-letter alignments (JSON format).

wordAlignmentsFile
Path to word-by-word alignments (JSON format).

readCountsFile
Path to letter read counts (JSON format).

alignments(*force=False*)
Uses the *Aligner* to generate alignments for a given original, gold pair of docs.

Caches its results in the `trainingPath`.

Parameters **force** – Back up existing alignment docs and create new ones.

Return type `Tuple`[`list`, `dict`, `list`]

prepare(*step*, *k*, *dehyphenate=False*, *force=False*)
Prepares the *Tokens* for the given doc.

Parameters

- **k** (`int`) – How many *k*-best suggestions to calculate, if necessary.
- **dehyphenate** – Whether to attempt dehyphenization of tokens.
- **force** – Back up existing tokens and create new ones.

crop_tokens(*edge_left=None*, *edge_right=None*)

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

SEVEN

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

EIGHT

REQUIREMENTS

- Python >= 3.6

For package dependencies see requirements.txt. They can be installed using `pip install -r requirements.txt`

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

HTTP ROUTING TABLE

/

/(string:docid)

/random

INDEX

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

V

W