ETUDE Engine Documentation

Paul M. Heider & the TBIC Team

Sep 08, 2021

HIgh-Level Content:

1 Documentation **1**

2 Sample Runs **3**
2.1 Basic Run 3
2.2 Specifying Annotation Configs 4
2.3 Scoring on Different Fields 5
2.4 Custom Evaluation Print-Outs 6

3 Configuring Annotation Extraction **9**

4 Dependencies **11**

5 Testing **13**

6 API Documentation **15**
6.1 args_and_configs.py Functions 15
6.2 etude.py Functions 16
6.3 scoring_metrics.py Functions 16
6.4 text_extraction.py Functions 18

7 Configuration Files **21**

8 Input Formats & Support Details **23**
8.1 Simple Plain Text 23
8.2 Structured Plain Text (e.g., csv) 23
8.3 XML Formats 24

9 Evaluating Matches **25**

10 Output Formats **27**

11 Indices and tables **29**

Python Module Index **31**

Index **33**

CHAPTER 1

Documentation

The latest documentation (compiled from the contents of the *docs* folder) can be viewed on-line: ETUDE Engine's documentation

Documentation for the ETUDE engine is managed via reStructuredText files and Sphinx. If you don't have Sphinx installed, you should check out a quick primer (First Steps with Sphinx) or install it as below:

```
## If you don't have Sphinx installed already
pip install Sphinx

## Generate a locally viewable HTML version
cd docs
make html
```

The latest version of the documentation can be generated as locally viewable HTML: file:///path/to/git/repository/docs/_build/html/index.html

CHAPTER 2

Sample Runs

2.1 Basic Run

The simplest test run requires that we specify a reference directory and a test directory. The default file matching assumes that our reference and test files match names exactly and both end in '.xml'. With just the two directory arguments, we get micro-average scores for the default metrics across the full directory.

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test
```

exact	TP	FP	TN	FN
micro-average	374.0	8.0	0.0	108.0

Note: You may get a warning if you run the previous command from a directory other than *$ETUDE_DIR*:

```
ERROR: No reference patterns extracted from config. Bailing out now.
```

This warning is because the default configuration files use relative paths. See the section below

In the next sample runs, you can see how to include a per-file score breakdown and a per-annotation-type score breakdown.

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-file
```

exact	TP	FP	TN	FN
micro-average	340.0	8.0	0.0	105.0
0005_gs.xml	31.0	0.0	0.0	0.0
0016_gs.xml	21.0	0.0	0.0	30.0
0267_gs.xml	27.0	0.0	0.0	32.0
0273_gs.xml	0.0	0.0	0.0	35.0
0389_gs.xml	26.0	8.0	0.0	8.0
0475_gs.xml	45.0	0.0	0.0	0.0
0617_gs.xml	32.0	0.0	0.0	0.0
0709_gs.xml	41.0	0.0	0.0	0.0
0982_gs.xml	95.0	0.0	0.0	0.0
0992_gs.xml	22.0	0.0	0.0	0.0
macro-average by file	340.0	8.0	0.0	105.0

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-type
```

exact	TP	FP	TN	FN
micro-average	340.0	8.0	0.0	105.0
Age	63.0	2.0	0.0	29.0
DateTime	91.0	2.0	0.0	33.0
HCUnit	61.0	4.0	0.0	15.0
OtherID	7.0	0.0	0.0	0.0
OtherLoc	1.0	0.0	0.0	4.0
OtherOrg	18.0	0.0	0.0	3.0
Patient	16.0	0.0	0.0	3.0
PhoneFax	5.0	0.0	0.0	1.0
Provider	54.0	0.0	0.0	10.0
StateCountry	14.0	0.0	0.0	7.0
StreetCity	4.0	0.0	0.0	0.0
Zip	4.0	0.0	0.0	0.0
eAddress	2.0	0.0	0.0	0.0
macro-average by type	340.0	8.0	0.0	105.0

2.2 Specifying Annotation Configs

We can use the same reference corpus to analyze annotations generated by UIMA's DateTime tutorial (see link below). A minimal run requires creating a matching dataset for the default configurations. Process the I2B2 dev set using the DateTime tutorial provided with UIMA. Then, because the output files for the I2B2 dev-annotations end in '.xml' but the UIMA tutorial files end in '.txt', you need to specify a file suffix translation rule. Also, the annotations are encoded slightly differently by the tutorial descriptor than by the I2B2 reference. As such, you will need to load a different configuration for the test directory to tell ETUDE how to find and extract the annotations. (If you run this example without the '–test-config' argument, you should see all FN matches because nothing can be extracted from the test corpus.)

Link: http://uima.apache.org/downloads/releaseDocs/2.2.2-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html#ugr.tug.aae.building_aggregates

```
export I2B2_CORPUS="/path/to/Corpora and annotations/2016 NGRID challenge (deid)/2016_
↪track_1-deidentification"

export I2B2_OUTPUT="/tmp/datetime-out"
mkdir $I2B2_OUTPUT

$UIMA_HOME/bin/runAE.sh \
  $UIMA_HOME/examples/descriptors/tutorial/ex3/TutorialDateTime.xml \
  $I2B2_CORPUS/dev-text \
  $I2B2_OUTPUT

python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $I2B2_OUTPUT \
    --by-type \
    --file-suffix ".xml" ".txt" \
    --test-config config/CAS_XMI.conf

#########   TP   FP   TN   FN
aggregate   19.0    20.0    0.0 426.0
Age 0.0 0.0 0.0 92.0
DateTime    19.0    20.0    0.0 105.0
HCUnit  0.0 0.0 0.0 76.0
OtherID 0.0 0.0 0.0 7.0
OtherLoc    0.0 0.0 0.0 5.0
OtherOrg    0.0 0.0 0.0 21.0
Patient 0.0 0.0 0.0 19.0
PhoneFax    0.0 0.0 0.0 6.0
Provider    0.0 0.0 0.0 64.0
StateCountry    0.0 0.0 0.0 21.0
StreetCity  0.0 0.0 0.0 4.0
Zip 0.0 0.0 0.0 4.0
eAddress    0.0 0.0 0.0 2.0

python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $I2B2_OUTPUT \
    --file-suffix ".xml" ".txt"

#########   TP   FP   TN   FN
aggregate   0.0 0.0 0.0 445.0
```

2.3 Scoring on Different Fields

The above examples show scoring based on the default key in the configuration file used for matching the reference to the test configuration. You may wish to group annotations on different fields, such as the parent class or long description.

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-type

python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
```

(continues on next page)

(continued from previous page)

```
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-type \
    --score-key "Parent"

python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-type \
    --score-key "Long Name"
```

exact	TP	FP	TN	FN
micro-average	341.0	7.0	0.0	104.0
Address	22.0	0.0	0.0	7.0
Contact Information	7.0	0.0	0.0	1.0
Identifiers	7.0	0.0	0.0	0.0
Locations	80.0	4.0	0.0	22.0
Names	70.0	0.0	0.0	13.0
Time	155.0	3.0	0.0	61.0
macro-average by type	341.0	7.0	0.0	104.0

exact	TP	FP	TN	FN
micro-average	340.0	8.0	0.0	105.0
Age Greater than 89	63.0	2.0	0.0	29.0
Date and Time Information	91.0	2.0	0.0	33.0
Electronic Address Information	2.0	0.0	0.0	0.0
Health Care Provider Name	54.0	0.0	0.0	10.0
Health Care Unit Name	61.0	4.0	0.0	15.0
Other ID Numbers	7.0	0.0	0.0	0.0
Other Locations	1.0	0.0	0.0	4.0
Other Organization Name	18.0	0.0	0.0	3.0
Patient Name	16.0	0.0	0.0	3.0
Phone, Fax, or Pager Number	5.0	0.0	0.0	1.0
State or Country	14.0	0.0	0.0	7.0
Street City Name	4.0	0.0	0.0	0.0
ZIP Code	4.0	0.0	0.0	0.0
macro-average by type	340.0	8.0	0.0	105.0

2.4 Custom Evaluation Print-Outs

The majority of you evaluation output customization can be handled by the above command-line arguments. However, sometimes you'll need to generate output that exactly matches some very specific formatting requirements. For these instances, ETUDE supports custom print functions. Currently, those print functions must be hard-coded into *scoring_metrics.py*. Our roadmap includes the ability to load and trigger these print functions from a standard folder to make the system much more modular. Until that point, you can see an example custom print-out that targets the 2018 n2c2 Track 1 output format. The configurations for this sample are in our sister repository: ETUDE Engine Configs for n2c2 The original evaluation script for the competition, used as a point of reference, can be found on github: Evaluation scripts for the 2018 N2C2 shared tasks on clinical NLP

```
export ETUDE_DIR=etude-engine
export ETUDE_CONFIGS_DIR=etude-engine-configs

export N2C2_DATA=/tmp/n2c2

python ${ETUDE_DIR}/etude.py \
  --reference-input ${N2C2_DATA}/train_annotations \
   --reference-config ${ETUDE_CONFIGS_DIR}/n2c2/2018_n2c2_track-1.conf \
   --test-input ${N2C2_DATA}/train_annotations \
   --test-config ${ETUDE_CONFIGS_DIR}/n2c2/2018_n2c2_track-1.conf \
   --no-metrics \
   --print-custom "2018 n2c2 track 1" \
   --fuzzy-match-flag exact \
   --file-suffix ".xml" \
   --empty-value 0.0


******************************************* TRACK 1␣
↪********************************************
                      ------------ met -------------    ------ not met -------    --␣
↪overall ---
                      Prec.   Rec.    Speci.  F(b=1)    Prec.   Rec.    F(b=1)   ␣
↪F(b=1)  AUC
           Abdominal  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
        Advanced-cad  1.0000  1.0000  0.0000  1.0000    0.0000  0.0000  0.0000    0.
↪5000  0.5000
       Alcohol-abuse  0.0000  0.0000  1.0000  0.0000    1.0000  1.0000  1.0000    0.
↪5000  0.5000
          Asp-for-mi  1.0000  1.0000  0.0000  1.0000    0.0000  0.0000  0.0000    0.
↪5000  0.5000
          Creatinine  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
       Dietsupp-2mos  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
          Drug-abuse  0.0000  0.0000  1.0000  0.0000    1.0000  1.0000  1.0000    0.
↪5000  0.5000
             English  1.0000  1.0000  0.0000  1.0000    0.0000  0.0000  0.0000    0.
↪5000  0.5000
               Hba1c  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
            Keto-1yr  0.0000  0.0000  1.0000  0.0000    1.0000  1.0000  1.0000    0.
↪5000  0.5000
      Major-diabetes  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
     Makes-decisions  1.0000  1.0000  0.0000  1.0000    0.0000  0.0000  0.0000    0.
↪5000  0.5000
             Mi-6mos  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
                      ------------------------------    ----------------------    ----
↪----------
     Overall (micro)  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
     Overall (macro)  0.7692  0.7692  0.6923  0.7692    0.6923  0.6923  0.6923    0.
↪7308  0.7308

                                                    10 files found
```

CHAPTER 3

Configuring Annotation Extraction

Several sample configurations are provided in the config/ folder. Each long name for an annotation description should be unique due to how Python's configuration parser works. XPath's should also be unique within a config file but do not programmitically need to be. The begin and end attribute are required for a pattern to be scorable.

```
[ Long Name or Description ]
Parent:           (optional; useful for merging multiple child types together for␣
↪scoring)
Short Name:  (optional; useful for displaying as column output name and merging
                       multiple XPaths into a single scoring category)
XPath:            (required; pattern used by XPath to find annotation)
Begin Attr:     (required; beginning or start offset attribute name)
End Attr:       (required; end offset attribute name)
Text Attr:      (optional; not used by anything currently)
```

Additional interesting or useful configuration files can be found in our sister repository: ETUDE Engine Configs

CHAPTER 4

Dependencies

Python module requirements for running ETUDE are included in the requirements.txt file. You should be able to install all non-default packages using pip:

```
pip install -r requirements
```

CHAPTER 5

Testing

Unit testing is done with the pytest module. Because of a bug in how tests are processed in Python 2.7, you should run pytest indirectly rather than directly:

```
python -m pytest tests/

## You can also generate a coverate report in html format
python2.7 -m pytest --cov-report html:cov_html_py2.7 --cov=./ tests/
python3.7 -m pytest --cov-report html:cov_html_py3.7 --cov=./ tests/

## The junit file is helpful for automated systems or CI pipelines
python -m pytest --junitxml=junit.xml tests
```

CHAPTER 6

API Documentation

6.1 args_and_configs.py Functions

`args_and_configs.`**`align_patterns`**(*reference_patterns*, *test_patterns*)

`args_and_configs.`**`extract_brat_patterns`**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *verbose=False*)

`args_and_configs.`**`extract_delimited_patterns`**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *verbose=False*)

`args_and_configs.`**`extract_document_data`**(*document_data*, *config*, *sect*)

`args_and_configs.`**`extract_namespaces`**(*namespaces*, *config*, *sect*)

`args_and_configs.`**`extract_patterns`**(*annotations*, *config*, *sect*, *score_key*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

`args_and_configs.`**`extract_xpath_patterns`**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

`args_and_configs.`**`extract_xpath_spanless_patterns`**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

`args_and_configs.`**`get_arguments`**(*command_line_args*)

`args_and_configs.`**`initialize_arg_parser`**()

`args_and_configs.`**`process_config`**(*config_file*, *score_key*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

`args_and_configs.`**`process_normalization_file`**(*normalization_file*)

`args_and_configs.`**`unique_attributes`**(*patterns*)

6.2 etude.py Functions

`etude.`**`align_tokens`**(*reference_folder*, *test_folder*, *args*, *file_prefix='/'*, *file_suffix='.xml'*)
Align reference and test documents by token for comparison

`etude.`**`collect_files`**(*reference_folder*, *test_folder*, *file_prefix*, *file_suffix*, *skip_missing_files_flag*)

`etude.`**`count_chars_profile`**(*reference_ns*, *reference_dd*, *reference_folder*, *test_ns*, *test_dd*, *test_folder*, *args*, *file_prefix='/'*, *file_suffix='.xml'*)

`etude.`**`count_ref_set`**(*this_ns*, *this_dd*, *this_patterns*, *this_folder*, *args*, *file_prefix='/'*, *file_suffix='.xml'*, *set_type=None*)

`etude.`**`create_output_folders`**(*reference_out*, *test_out*)

`etude.`**`generate_out_file`**(*output_dir*, *input_filename*)
Generate a well-formed full file path for writing output stats

`etude.`**`get_file_mapping`**(*reference_folder*, *test_folder*, *file_prefix*, *file_suffix*, *skip_missing_files_flag*)

`etude.`**`init_args`**()

`etude.`**`score_ref_set`**(*reference_ns*, *reference_dd*, *reference_patterns*, *reference_folder*, *test_ns*, *test_dd*, *test_patterns*, *test_folder*, *args*, *file_prefix='/'*, *file_suffix='.xml'*)

6.3 scoring_metrics.py Functions

`scoring_metrics.`**`accuracy`**(*tp*, *fp*, *tn*, *fn*)

`scoring_metrics.`**`add_missing_fields`**(*score_summary*)

`scoring_metrics.`**`document_level_annot_comparison_runner`**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *fuzzy_flag*, *scorable_attributes*)

`scoring_metrics.`**`end_comparison_runner`**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

`scoring_metrics.`**`evaluate_positions`**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_ss*, *test_ss*, *fuzzy_flag='exact'*, *use_mapped_chars=False*, *scorable_attributes=[]*, *scorable_engines=[]*, *norm_synonyms={}*)

`scoring_metrics.`**`exact_comparison_runner`**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

`scoring_metrics.`**`f_score`**(*p*, *r*, *beta=1*)

`scoring_metrics.`**`flatten_ss_dictionary`**(*ss_dictionary*, *category='(unknown)'*) `scoring_metrics.`**`fully_contained_comparison_runner`**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

`scoring_metrics.`**`get_annotation_from_base_entry`**(*annotation_entry*, *start_key*, *end_key*)

`scoring_metrics.`**`get_unique_types`**(*config*)

`scoring_metrics.`**`new_score_card`**(*fuzzy_flags=['exact'], normalization_engines=[]*)

`scoring_metrics.`**`norm_summary`**(*score_summary*, *args*)

`scoring_metrics.`**`output_metrics`**(*class_data*, *fuzzy_flag*, *metrics*, *delimiter_prefix*, *delimiter*, *stdout_flag*, *csv_out_filename*, *pretty_print_flag*)

`scoring_metrics.`**`partial_comparison_runner`**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

`scoring_metrics.`**`precision`**(*tp*, *fp*)

`scoring_metrics.`**`print_2018_n2c2_track1`**(*score_card*, *file_mapping*, *args*)

`scoring_metrics.`**`print_confusion_matrix`**(*confusion_matrix*, *file_mapping*, *reference_config*, *test_config*, *fuzzy_flag*, *args*)

`scoring_metrics.`**`print_confusion_matrix_shell`**(*confusion_matrix*, *file_mapping*, *reference_patterns*, *test_patterns*, *args*)

`scoring_metrics.`**`print_counts_summary`**(*score_card*, *file_list*, *config_patterns*, *args*, *set_type*)

`scoring_metrics.`**`print_score_summary`**(*score_card*, *file_mapping*, *reference_config*, *test_config*, *fuzzy_flag*, *args*, *norm_engine=''*)

`scoring_metrics.`**`print_score_summary_shell`**(*score_card*, *file_mapping*, *reference_config*, *test_config*, *args*)

`scoring_metrics.`**`recall`**(*tp*, *fn*)

`scoring_metrics.`**`recursive_deep_key_value_pair`**(*dictionary*, *path*, *key*, *value*)

`scoring_metrics.`**`reference_annot_comparison_runner`**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

`scoring_metrics.`**`specificity`**(*tn*, *fp*, *empty_value=None*)

`scoring_metrics.`**`update_confusion_matrix`**(*confusion_matrix*, *fuzzy_flag*, *ref_type*, *test_type*)

`scoring_metrics.`**`update_csv_output`**(*csv_out_filename*, *delimiter*, *row_content*)

`scoring_metrics.`**`update_output_dictionary`**(*out_file*, *metric_type*, *metrics_keys*, *metrics_values*) `scoring_metrics.`**`update_score_card`**(*condition*, *score_card*, *fuzzy_flag*, *filename*, *start_pos*, *end_pos*, *type*, *pivot_value=None*, *ref_annot=None*, *test_annot=None*, *scorable_attributes=None*, *scorable_engines=None*, *norm_synonyms={}*)

6.4 text_extraction.py Functions

`text_extraction.`**`align_tokens_on_whitespace`**(*dictionary*, *out_file*)

`text_extraction.`**`create_annotation_entry`**(*begin_pos=-1*, *begin_pos_mapped=None*, *end_pos=-1*, *end_pos_mapped=None*, *raw_text=None*, *pivot_attr=None*, *pivot_value=None*, *parity=None*, *tag_name=None*)

`text_extraction.`**`extract_annotations`**(*ingest_file*, *namespaces*, *document_data*, *patterns*, *skip_chars=None*, *out_file=None*)

`text_extraction.`**`extract_annotations_brat_standoff`**(*ingest_file*, *offset_mapping*, *type_prefix*, *tag_name*, *optional_attributes=[]*, *normalization_engines=[]*)

`text_extraction.`**`extract_annotations_plaintext`**(*offset_mapping*, *raw_content*, *delimiter*, *tag_name*)

`text_extraction.`**`extract_annotations_xml`**(*ingest_file*, *offset_mapping*, *annotation_path*, *tag_name*, *namespaces={}*, *begin_attribute=None*, *end_attribute=None*, *text_attribute=None*, *optional_attributes=[]*, *normalization_engines=[]*)

`text_extraction.`**`extract_annotations_xml_spanless`**(*ingest_file*, *annotation_path*, *tag_name*, *pivot_attribute*, *parity*, *namespaces={}*, *text_attribute=None*, *optional_attributes=[]*)

`text_extraction.`**`extract_brat_attribute`**(*ingest_file*, *annot_line*, *optional_attributes=[]*)

`text_extraction.`**`extract_brat_equivalence`**(*ingest_file*, *annot_line*, *optional_attributes=[]*)

`text_extraction.`**`extract_brat_event`**(*ingest_file*, *annot_line*, *tag_name*, *optional_attributes=[]*)

`text_extraction.`**`extract_brat_normalization`**(*ingest_file*, *annot_line*, *normalization_engines=[]*)

`text_extraction.`**`extract_brat_relation`**(*ingest_file*, *annot_line*, *tag_name*, *optional_attributes=[]*)

`text_extraction.`**`extract_brat_text_bound_annotation`**(*ingest_file*, *annot_line*, *offset_mapping*, *tag_name*, *optional_attributes=[]*)

`text_extraction.`**`extract_chars`**(*ingest_file*, *namespaces*, *document_data*, *skip_chars=None*)

`text_extraction.`**`extract_plaintext`**(*ingest_file*, *skip_chars*)

`text_extraction.`**`map_position`**(*offset_mapping*, *position*, *direction*)

Convert a character position to the closest non-skipped position.

Use the offset mapping dictionary to convert a position to the closest valid character position. We include a direction for the mapping because it is important to consider the closest position to the right or left of a position when mapping the start or end position, respectively.

Parameters

- **`offset_mapping`** – a dictionary mapping character positions to `None` if the character is in the skip list or to an int, otherwise
- **`position`** – current character position
- **`direction`** – 1, if moving right; -1 if moving left

Returns character position if all skipped characters were removed from the document and positions re-assigned or `None`, on KeyError

`text_extraction.`**`split_content`**(*raw_text*, *offset_mapping*, *skip_chars*)

`text_extraction.`**`write_annotations_to_disk`**(*annotations*, *out_file*)

CHAPTER 7

Configuration Files

CHAPTER 8

Input Formats & Support Details

8.1 Simple Plain Text

8.1.1 Newlines for Sentences

Local sample configuration files (under *config/*):

- *plaintext_sentences.conf*

8.2 Structured Plain Text (e.g., csv)

8.2.1 brat Annotation

The brat rapid annotation tool generates brat standoff format. Annotations are stored in a seconardy file (**.ann*) while the original text is found in a plain text file (**.txt*). This standoff format uses character offsets to locate spans: "*All offsets all [sic] indexed from 0 and include the character at the start offset but exclude the character at the end offset.*" See BioNLP Shared Task standoff format for a related format.

Limitations: The extraction engine currently only handles continous text-bound annotations for evaluation. Binary attributes can be extracted and included in the evaluation dictionary but are not scored themselves. Discontinous text-bound annotations, relations, events, multi-value attributes, normalizations, and notes are not supported.

Local sample configuration files (under *config/*):

- *brat_problems_allergies_standoff.conf*

8.3 XML Formats

8.3.1 UIMA CAS XMI

Local sample configuration files (under *config/*):

- *CAS_XMI.conf*
- *i2b2_2016_track-1.conf*
- *uima_sentences.conf*
- *webanno_phi_xmi.conf*
- *webanno_problems_allergies_xmi.conf*
- *webanno_uima_xmi.conf*

8.3.2 Other

Extra sample configuration files (via the ETUDE engine configs repository):

- *i2b2/…*
- *n2c2/n2c2_2018_track-1.conf*

CHAPTER 9

Evaluating Matches

CHAPTER 10

Output Formats

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

args_and_configs, 15

e

etude, 16

s

scoring_metrics, 16

t

text_extraction, 18

Index

A

accuracy() (*in module scoring_metrics*), 16
add_missing_fields() (*in module scoring_metrics*), 16
align_patterns() (*in module args_and_configs*), 15
align_tokens() (*in module etude*), 16
align_tokens_on_whitespace() (*in module text_extraction*), 18
args_and_configs (*module*), 15

C

collect_files() (*in module etude*), 16
count_chars_profile() (*in module etude*), 16
count_ref_set() (*in module etude*), 16
create_annotation_entry() (*in module text_extraction*), 18
create_output_folders() (*in module etude*), 16

D

document_level_annot_comparison_runner() (*in module scoring_metrics*), 16

E

end_comparison_runner() (*in module scoring_metrics*), 16
etude (*module*), 16
evaluate_positions() (*in module scoring_metrics*), 16
exact_comparison_runner() (*in module scoring_metrics*), 16
extract_annotations() (*in module text_extraction*), 18
extract_annotations_brat_standoff() (*in module text_extraction*), 18
extract_annotations_plaintext() (*in module text_extraction*), 18
extract_annotations_xml() (*in module text_extraction*), 18
extract_annotations_xml_spanless() (*in module text_extraction*), 18
extract_brat_attribute() (*in module text_extraction*), 18
extract_brat_equivalence() (*in module text_extraction*), 18
extract_brat_event() (*in module text_extraction*), 18
extract_brat_normalization() (*in module text_extraction*), 18
extract_brat_patterns() (*in module args_and_configs*), 15
extract_brat_relation() (*in module text_extraction*), 18
extract_brat_text_bound_annotation() (*in module text_extraction*), 18
extract_chars() (*in module text_extraction*), 18
extract_delimited_patterns() (*in module args_and_configs*), 15
extract_document_data() (*in module args_and_configs*), 15
extract_namespaces() (*in module args_and_configs*), 15
extract_patterns() (*in module args_and_configs*), 15
extract_plaintext() (*in module text_extraction*), 18
extract_xpath_patterns() (*in module args_and_configs*), 15
extract_xpath_spanless_patterns() (*in module args_and_configs*), 15

F

f_score() (*in module scoring_metrics*), 16
flatten_ss_dictionary() (*in module scoring_metrics*), 16
fully_contained_comparison_runner() (*in module scoring_metrics*), 16

G

generate_out_file() (*in module etude*), 16
get_annotation_from_base_entry() (*in module scoring_metrics*), 17
get_arguments() (*in module args_and_configs*), 15
get_file_mapping() (*in module etude*), 16
get_unique_types() (*in module scoring_metrics*), 17

I

init_args() (*in module etude*), 16
initialize_arg_parser() (*in module args_and_configs*), 15

M

map_position() (*in module text_extraction*), 18

N

new_score_card() (*in module scoring_metrics*), 17
norm_summary() (*in module scoring_metrics*), 17

O

output_metrics() (*in module scoring_metrics*), 17

P

partial_comparison_runner() (*in module scoring_metrics*), 17
precision() (*in module scoring_metrics*), 17
print_2018_n2c2_track1() (*in module scoring_metrics*), 17
print_confusion_matrix() (*in module scoring_metrics*), 17
print_confusion_matrix_shell() (*in module scoring_metrics*), 17
print_counts_summary() (*in module scoring_metrics*), 17
print_score_summary() (*in module scoring_metrics*), 17
print_score_summary_shell() (*in module scoring_metrics*), 17
process_config() (*in module args_and_configs*), 15
process_normalization_file() (*in module args_and_configs*), 15

R

recall() (*in module scoring_metrics*), 17
recursive_deep_key_value_pair() (*in module scoring_metrics*), 17
reference_annot_comparison_runner() (*in module scoring_metrics*), 17

S

score_ref_set() (*in module etude*), 16
scoring_metrics (*module*), 16
specificity() (*in module scoring_metrics*), 17
split_content() (*in module text_extraction*), 19

T

text_extraction (*module*), 18

U

unique_attributes() (*in module args_and_configs*), 15
update_confusion_matrix() (*in module scoring_metrics*), 17
update_csv_output() (*in module scoring_metrics*), 17
update_output_dictionary() (*in module scoring_metrics*), 17
update_score_card() (*in module scoring_metrics*), 17

W

write_annotations_to_disk() (*in module text_extraction*), 19