medaCy Documentation

Release 0.6

Andriy Mulyar, Jorge Vargas, Corey Sutphin, Bobby Best, Steele F

May 19, 2020

Contents

For the latest updates, please see the project on github.

MedaCy is a medical text mining framework built over spaCy to facilitate the engineering, training and application of machine learning models for medical information extraction.

To confront the unique challenges posed by medical text medaCy provides interfaces to medical ontologies such as Metamap allowing their integration into text mining workflows. Additional help, examples and tutorials can be found in the examples section of the repository.

MedaCy does not officially support non-unix based operating systems (however we have found most functionality works on Windows).

CHAPTER 1

Trained Models

A complete listing of trained models can be found here.

CHAPTER 2

Datasets

MedaCy implements a Dataset functionality that loosely wraps a working directory to manage and version training data. See more in the examples.

CHAPTER 3

Contents

3.1 medacy package

3.1.1 medacy.__main__ module

MedaCy CLI Setup

medacy.__main__.**cross_validate**(*args*, *dataset*, *model*)
Used for running k-fold cross validations. :param args: Argparse args object. :param dataset: Dataset to use for training. :param model: Untrained model object to use.

medacy.__main__.**main**()
Main function where initial argument parsing happens.

medacy.__main__.**predict**(*args*, *dataset*, *model*)
Used for running predictions on new datasets. :param args: Argparse args object. :param dataset: Dataset to run prediction over. :param model: Trained model to use for predictions.

medacy.__main__.**setup**(*args*)
Sets up dataset and pipeline/model since it gets used by every command. :param args: Argparse args object. :return dataset, model: The dataset and model objects created.

medacy.__main__.**train**(*args*, *dataset*, *model*)
Used for training new models. :param args: Argparse args object. :param dataset: Dataset to use for training. :param model: Untrained model object to use.

3.1.2 medacy.data package

medacy.data.dataset module

A medaCy Dataset facilities the management of data for both model training and model prediction.

A Dataset object provides a wrapper for a unix file directory containing training/prediction data. If a Dataset, at training time, is fed into a pipeline requiring auxilary files (Metamap for instance) the Dataset will automatically create those files in the most efficient way possible.

Training

When a directory contains **both** raw text files alongside annotation files, an instantiated Dataset detects and facilitates access to those files.

Assuming your directory looks like this (where .ann files are in BRAT format):

```
home/medacy/data
├── file_one.ann
├── file_one.txt
├── file_two.ann
└── file_two.txt
```

A common data work flow might look as follows.

Running:

```
>>> from medacy.data import Dataset
>>> from medacy.pipeline_components.feature_overlayers.metamap.metamap import MetaMap

>>> dataset = Dataset('/home/datasets/some_dataset')
>>> for data_file in dataset:
...     (data_file.file_name, data_file.raw_path, dataset.ann_path)
(file_one, file_one.txt, file_one.ann)
(file_two, file_two.txt, file_two.ann)
>>> dataset
['file_one', 'file_two']
>>>> dataset.is_metamapped()
False
>>> metamap = MetaMap('/home/path/to/metamap/binary')
>>> with metamap:
...     metamap.metamap_dataset(dataset)
>>> dataset.is_metamapped()
True
```

MedaCy **does not** alter the data you load in any way - it only reads from it.

Prediction

When a directory contains **only** raw text files, an instantiated Dataset object interprets this as a directory of files that need to be predicted. This means that the internal Datafile that aggregates meta-data for a given prediction file does not have fields for annotation_file_path set.

When a directory contains **only** ann files, an instantiated Dataset object interprets this as a directory of files that are predictions. Useful methods for analysis include *medacy.data.dataset.Dataset.compute_confusion_matrix()*, *medacy.data.dataset.Dataset.compute_ambiguity()* and *medacy.data.dataset.Dataset.compute_counts()*.

External Datasets

In the real world, datasets (regardless of domain) are evolving entities. Hence, it is essential to version them. A medaCy compatible dataset can be created to facilitate this versioning. A medaCy compatible dataset lives a python packages that can be hooked into medaCy or used for any other purpose - it is simply a loose wrapper for this Dataset object. Instructions for creating such a dataset can be found here. wrap them.

class medacy.data.dataset.**Dataset**(*data_directory*, *data_limit=None*)
Bases: object

A facilitation class for data management.

_create_data_files()

compute_ambiguity(*dataset*)
Finds occurrences of spans from 'dataset' that intersect with a span from this annotation but do not have this spans label. label. If 'dataset' comprises a models predictions, this method provides a strong indicators of a model's in-ability to dis-ambiguate between entities. For a full analysis, compute a confusion matrix.

Parameters **dataset** – a Dataset object containing a predicted version of this dataset.

Returns a dictionary containing the ambiguity computations on each gold, predicted file pair

compute_confusion_matrix(*other*, *leniency=0*)
Generates a confusion matrix where this Dataset serves as the gold standard annotations and *dataset* serves as the predicted annotations. A typical workflow would involve creating a Dataset object with the prediction directory outputted by a model and then passing it into this method.

Parameters

- **other** – a Dataset object containing a predicted version of this dataset.
- **leniency** – a floating point value between [0,1] defining the leniency of the character spans to count as different. A value of zero considers only exact character matches while a positive value considers entities that differ by up to `ceil(leniency * len(span)/2)` on either side.

Returns two element tuple containing a label array (of entity names) and a matrix where rows are gold labels and columns are predicted labels. matrix[i][j] indicates that entities[i] in this dataset was predicted as entities[j] in 'annotation' matrix[i][j] times

compute_counts()
Computes entity counts over all documents in this dataset.

Returns a Counter of entity counts

generate_annotations()
Generates Annotation objects for all the files in this Dataset

get_labels(*as_list=False*)
Get all of the entities/labels used in the dataset. :param as_list: bool for if to return the results as a list; defaults to False :return: A set of strings. Each string is a label used.

is_metamapped()
Verifies if all fil es in the Dataset are metamapped.

Returns True if all data files are metamapped, False otherwise.

medacy.data.dataset.**main**()
CLI for retrieving dataset information

3.1.3 medacy.ner package

medacy.ner.model package

medacy.ner.model.model module

medacy.ner.model.spacy_model module

medacy.ner.model.stratified_k_fold module

medacy.ner.pipelines package

medacy.ner.pipelines.base package

medacy.ner.pipelines.base.base_pipeline module

medacy.ner.pipelines.clinical_pipeline module

medacy.ner.pipelines.drug_event_pipeline module

medacy.ner.pipelines.fda_nano_drug_label_pipeline module

medacy.ner.pipelines.systematic_review_pipeline module

medacy.ner.pipelines.testing_pipeline module

3.1.4 MedaCy NN

BiLSTM-CRF

3.1.5 medacy.relation package

3.1.6 medacy.pipeline_components package

medacy.pipeline_components.annotation package

medacy.pipeline_components.annotation.gold_annotator_component module

medacy.pipeline_components.base package

medacy.pipeline_components.base.base_component module

Pipeline Components: Learners

BiLSTM-CRF Learner

class medacy.pipeline_components.**BiLstmCrfLearner**

Bases: object

BiLSTM-CRF model class for using the network. Currently handles all vectorization as well.

Variables

- **device** – PyTorch device to use.
- **model** – Instance of BiLstmCrfNetwork to use.
- **word_embeddings_file** – File to load word embeddings from.
- **word_vectors** – Gensim word vectors object for use in configuring word embeddings.

fit(*x_data*, *y_data*)
Fully train model based on x and y data. self.model is set to trained model.

Parameters

- **x_data** – List of list of tokens.
- **y_data** – List of list of correct labels for the tokens.

load(*path*)
Load model and other required values from given path.

Parameters **path** – Path of saved model.

predict(*sequences*)
Use model to make predictions over a given dataset.

Parameters **sequences** – Sequences to predict labels for.

Returns List of list of predicted labels.

save(*path*)
Save model and other required values.

Parameters **path** – Path to save model to.

medacy.pipeline_components.lexicon package

medacy.pipeline_components.lexicon.lexicon_component module

medacy.pipeline_components.metamap package

medacy.pipeline_components.metamap.metamap module

medacy.pipeline_components.metamap.metamap_component module

medacy.pipeline_components.tokenization package

medacy.pipeline_components.tokenization.character_tokenizer module

medacy.pipeline_components.tokenization.clinical_tokenizer module

medacy.pipeline_components.tokenization.systematic_review_tokenizer module

medacy.pipeline_components.units package

medacy.pipeline_components.units.frequency_unit_component module

class medacy.pipeline_components.units.frequency_unit_component.**FrequencyUnitOverlayer**(*spacy*
Bases: medacy.pipeline_components.feature_overlayers.base.base_overlayer.BaseOverlayer

A pipeline component that tags Frequency units

_abc_impl = <_abc_data object>

dependencies = []

name = 'frequency_unit_annotator'

medacy.pipeline_components.units.mass_unit_component module

class medacy.pipeline_components.units.mass_unit_component.**MassUnitOverlayer**(*spacy_pipeline*)
Bases: medacy.pipeline_components.feature_overlayers.base.base_overlayer.BaseOverlayer

A pipeline component that tags mass units

_abc_impl = <_abc_data object>

dependencies = []

name = 'mass_unit_annotator'

medacy.pipeline_components.units.measurement_unit_component module

class medacy.pipeline_components.units.measurement_unit_component.**MeasurementUnitOverlayer**
Bases: medacy.pipeline_components.feature_overlayers.base.base_overlayer.BaseOverlayer

A pipeline component that tags Frequency units

_abc_impl = <_abc_data object>

dependencies = [<class 'medacy.pipeline_components.units.mass_unit_component.MassUnitO

name = 'measurement_unit_annotator'

medacy.pipeline_components.units.route_unit_component module

medacy.pipeline_components.units.time_unit_component module

class medacy.pipeline_components.units.time_unit_component.**TimeUnitOverlayer**(*spacy_pipeline*)
Bases: medacy.pipeline_components.feature_overlayers.base.base_overlayer.BaseOverlayer

A pipeline component that tags time units

_abc_impl = <_abc_data object>

dependencies = []

name = 'time_unit_annotator'

medacy.pipeline_components.units.unit_component module

class medacy.pipeline_components.units.unit_component.**UnitOverlayer**(*nlp*)
Bases: medacy.pipeline_components.feature_overlayers.base.base_overlayer.BaseOverlayer

A pipeline component that tags units. Begins by first tagging all mass, volume, time, and form units then aggregates as necessary.

_abc_impl = <_abc_data object>

dependencies = []

name = 'unit_annotator'

medacy.pipeline_components.units.volume_unit_component module

class medacy.pipeline_components.units.volume_unit_component.**VolumeUnitOverlayer**(*spacy_pipelin*
Bases: medacy.pipeline_components.feature_overlayers.base.base_overlayer.BaseOverlayer

A pipeline component that tags volume units

_abc_impl = <_abc_data object>

dependencies = []

name = 'volume_unit_annotator'

medacy.pipeline_components.feature_extraction package

medacy.pipeline_components.feature_extraction.feature_extractor module

3.1.7 medacy.tools package

medacy.tools.annotations module

medacy.tools.data_file module

Python Module Index

m

Index

Symbols

B

C

D

F

G

I

L

M

N

P

S

T

U

V