deft Documentation

Release 0.11.2

Albert Steppi

Nov 05, 2022

Contents

Adeft (Acromine based Disambiguation of Entities from Text) is a utility for building models to disambiguate acronyms and other abbreviations of biological terms in the scientific literature. It makes use of an implementation of the Acromine algorithm developed by the NaCTeM at the University of Manchester to identify possible longform expansions for shortforms based on their text context. It allows users to build disambiguation models to disambiguate shortforms based on their text context. A growing number of pretrained disambiguation models are publically available to download through Adeft.

CHAPTER 1

Adeft modules reference

1.1 Download Trained Models

Allows models to be downloaded from the command line with

```
python -m adeft.download
```

Use

```
python -m adeft.download --update
```

to update existing models if models have changed on S3

`adeft.download.download.`**`download_models`**`(`*models=None*`)`

Download models from S3

Models are downloaded and placed into a models directory in the users home directory. Each model contains a serialized AdeftClassifier, a dictionary mapping shortforms to dictionaries mapping longform texts to groundings, and a list of canonical names for each grounding. Within the models directory, models are stored in subdirectories named after the shortform they disambiguate with escape characters used to handle characters that cannot be used in filenames and to distinguish upper and lower case for compatibility with case insensitive file systems.

Parameters **`models`** (*`Optional[iterable of str]`*) – List of models to be downloaded. Allows user to select specific models to download. If this option is set, update will be treated as True regardless of how it was set. These should be considered as mutually exclusive parameters.

`adeft.download.download.`**`download_test_resources`**`()`

Download files necessary to run tests

Downloads a test disambiguator and a set of example training data and places them in the test_resources folder of the .adeft directory. This function will error if the necessary directories do not exist. If they do not already exist they will be created when running python -m adeft.download

`adeft.download.download.`**`get_available_models`**`(`*path='/home/docs/.local/share/adeft/0.11.2/models'*`)`

Returns set of all models currently in models folder adeft.download.download.**get_s3_models**()

Returns set of all models currently available on s3

adeft.download.download.**setup_models_folder**()

Create models folder if it does not exist and download models

adeft.download.download.**setup_resources_folder**()

Make resources folder and download resources

Replaces content in existing resources folder if it already exists

adeft.download.download.**setup_test_resource_folder**()

Make test resource folders and download content

Replaces content in existing test_resource_folders if they already exist.

1.2 Disambiguate Shortforms

Implements classes to disambiguate shortforms given text context.

class adeft.disambiguate.**AdeftDisambiguator**(*classifier*, *grounding_dict*, *names*)

Bases: object

Disambiguates a particular shortform in a list of texts

Parameters

- **classifier** (py:class:*adeft.modeling.classify.AdeftClassifier*) – Machine learning model for disambiguating shortforms based upon context
- **grounding_dict** (*dict*) – Dictionary mapping shortforms to grounding_map dictionaries mapping longforms to groundings
- **names** (*dict*) – Dictionary mapping groundings to canonical names

shortforms

Shortforms to disambiguate

Type list of str

recognizers

A list of recognizers, one for each shortform, to disambiguate by searching for a defining pattern.

Type list of py:class:*adeft.recognize.AdeftRecognizer*

labels

Set of labels that the classifier is able to predict.

Type set

pos_labels

List of labels of interest. Only these are considered when calculating the micro averaged f1 score for a classifier.

Type list of str

disambiguate(*texts*)

Return disambiguations for a list of texts

First checks for defining patterns (DP) within a text. If there is an unambiguous match to a longform with a defining pattern, considers this to be the correct disambiguation with confidence 1.0. If no defining pattern is found, uses a logistic regression model to predict the correct disambiguation. If there were multiple longforms with different groundings found with a defining pattern, disambiguates to the grounding among these with highest predicted probability. If no defining pattern was found, disambiguates to the grounding with highest predicted probability.

Parameters **`texts`** (*`str or list of str`*) – fulltext or list of fulltexts in which to disambiguate shortform

Returns **result** – Disambiguations for text. For each text the corresponding disambiguation is a tuple of three elements. A grounding, a canonical name associated with the grounding, and a dictionary containing predicted probabilities for each possible grounding

Return type tuple or list of tuple

`dump`(*model_name*, *path=None*)

Save disambiguator to disk

Parameters

- **`model_name`** (*`str`*) – Model files will be saved in directory with this name.
- **`path`** (*`Optional[str]`*) – Path where model is to be stored. Defaults to current directory. Default: None

`info`()

Get information about disambiguator and its performance.

Displays disambiguations model is able to produce. Shows class balance of disambiguation labels in the models training data and crossvalidated F1 score, precision, and recall on training data. Classification metrics for multi-label data are calculated by taking the micro-average over the positive labels. This means the metrics are calculated globally by counting the total true positives, false negatives, and false positives. Positive labels are starred in in the displayed output. F1, Precision, and Recall are also shown for for each label separately. Classification metrics may not be available depending upon how the model was trained.

Returns A string representing the information about the disambigutor.

Return type str

`modify_groundings`(*new_groundings=None*, *new_names=None*)

Update groundings and standardized names

Modify groundings and standard names for the disambiguator without retraining. Cannot map two existing groundings to a single new grounding, as this leads to a nontrivial change in the model rather than just a relabeling.

Parameters

- **`new_groundings`** (*`Optional[dict]`*) – Dictionary mapping a subset of previous groundings to updated groundings. If None, no groundings are modified. Default: None
- **`new_names`** (*`Optional[dict]`*) – Dictionary mapping a subset of previous groundings to updated names. If None, no names are modified. Default: None

`update_pos_labels`(*pos_labels*)

Update which labels are considered pos_labels

Micro-averaged precision, recall, and f1 scores are also updated.

Warning: If this method is called on a disambiguator trained with a a version prior to 0.10.0, global precision, recall, and f1 will be set to NaN. Older disambiguators must be retrained to update positive labels and recompute model statistics.

Parameters **`pos_labels`** (*`list`*) – list of strs. Should be a subset of the labels produced by the underlying classifier. Check the labels attribute of the AdeftDisambiguator to see which labels are produced.

version()

Returns version string for disambiguator

Returns String of the form <adeft_version>::<timestamp>::<hash> where <hash> is the md5 hash of the grounding_dict jsonified with sorted keys.

Return type str

adeft.disambiguate.**load_disambiguator**(*shortform*, *path='/home/docs/.local/share/adeft/0.11.2/models'*)

Returns adeft disambiguator loaded from models directory

Searches folder specified by path for a disambiguation model that can disambiguate the given shortform and returns this model

Parameters

- **shortform** (*str*) – Shortform to disambiguate.
- **path** (*Optional[str]*) – Path to models directory. Defaults to adeft's pretrained models. Users have the option to specify a path to another directory to use custom models.

Returns A disambiguator that was loaded from a file. Returns None if there are no disambiguation models in the supplied folder that can disambiguate the given shortform

Return type py:class:*adeft.disambiguate.AdeftDisambiguator*

adeft.disambiguate.**load_disambiguator_directly**(*path*)

Returns disambiguator located at path

Parameters path (*str*) – Path to a disambiguation model. Must be a path to a directory <model_name> containing the files <model_name>_model.gz, <model_name>_grounding_dict.json, <model_name>_names.json

Returns A disambiguation model loaded from folder specified by path

Return type py:class:*adeft.disambiguate.AdeftDisambiguator*

1.3 Discover Longforms

1.4 Recognize Defining Patterns

Implements the disambiguation of shortforms based on recognizing an explicit defining pattern in text.

class adeft.recognize.**AdeftRecognizer**(*shortform*, *grounding_map*, *window=100*)

Bases: *adeft.recognize.BaseRecognizer*

Class for recognizing longforms by searching for defining patterns (DP)

Searches text for the pattern "<longform> (<shortform>)" for a collection of grounded longforms supplied by the user.

Parameters

- **shortform** (*str*) – shortform to be recognized
- **grounding_map** (*dict[str, str]*) – Dictionary mapping longform texts to their groundings
- **window** (*Optional[int]*) – Specifies range of characters before a defining pattern (DP) to consider when finding longforms. Should be set to the same value that was used in the AdeftMiner that was used to find longforms. Default: 100

`_trie`
Trie used to search for longforms. Edges correspond to stemmed tokens from longforms. They appear in reverse order to the bottom of the trie with terminal nodes containing the associated longform in their data.

Type `adeft.recognize._TrieNode`

class `adeft.recognize.`**`BaseRecognizer`**(*shortform*, *window=100*)
Bases: `object`

Base class for recognizers

Recognizers are built to identify longform expansions for a shortform by searching for defining patterns (DPs).

Parameters

- **`shortform`** (*`str`*) – shortform to be recognized
- **`window`** (*`Optional[int]`*) – Specifies range of characters before a defining pattern (DP) to consider when finding longforms. Should be set to the same value that was used in the AdeftMiner that was used to find longforms. Default: 100

`recognize`(*text*)
Find longforms in text by searching for defining patterns (DPs)

Parameters **`text`** (*`str`*) – Sentence where we seek to disambiguate shortform

Returns **expansions** – Set of longforms corresponding to shortform in sentence if a defining pattern is matched. Returns None if no defining patterns are found

Return type set of str

`strip_defining_patterns`(*text*)
Return text with defining patterns stripped

This is useful for training machine learning models where training labels are generated by finding defining patterns (DP)s. Models must be trained to disambiguate texts that do not contain a defining pattern.

The output on the first sentence of the previous paragraph is "This is useful for training machine learning models where training labels are generated by finding DPs."

Parameters **`text`** (*`str`*) – Text to remove defining patterns from

Returns **stripped_text** – Text with defining patterns replaced with shortform

Return type str

class `adeft.recognize.`**`OneShotRecognizer`**(*shortform*, *window=100*, ***params*)
Bases: *`adeft.recognize.BaseRecognizer`*

Identify longform expansions using subsequence matching

Uses a string matching algorithm to determine longform boundaries for a defining pattern for only a single text.

`shortform`
shortform to be recognized

Type str

`window`
Specifies range of characters before a defining pattern (DP) to consider when finding longforms. Should be set to the same value that was used in the AdeftMiner that was used to find longforms. Default: 100

Type Optional[int]

****params**
Parameters for :py:class‘adeft.score.AdeftLongformScorer‘

1.5 Learn Models

Implements classes needed to label text corpora, and learn a logistic regression model to disambiguate across a set of longforms for a given shortform.

1.5.1 Label Corpora

class `adeft.modeling.label.`**`AdeftLabeler`**(*grounding_dict*)
Bases: `object`

Class for labeling corpora

Parameters **`grounding_dict`** (*`dict of dict of str`*) – Dictionary mapping shortforms to grounding_map dictionaries mapping longforms to groundings

`recognizers`
List of recognizers for each shortform to be considered. Each recognizer identifies longforms for a shortform by finding defining matches to a defining pattern (DP)

Type list of py:class‘adeft.recognize.AdeftRecognizer‘

`build_from_texts`(*text_tuples*)
Build labeled corpus from a list of texts

Labels texts based on defining patterns (DPs)

Parameters **`text_tuples`** (*`list of tuple`*) – List of two element tuples whose first elements are texts from which we seek to build a corpus and whose second elements are identifiers associated with the texts. Each text should have a unique identifier associated to it.

Returns **corpus** – Contains a tuple for each text in the input list which contains a defining pattern. Multiple tuples correspond to texts with multiple defining patterns for longforms with different groundings. The first element of each tuple contains a training text with all defining patterns replaced with only the shortform. The second element contains a grounding label for the desired shortform within the training text that was identified through a defining pattern. The third element contains the identifier for the given training text.

Return type list

1.5.2 Classify

class `adeft.modeling.classify.`**`AdeftClassifier`**(*shortforms*, *pos_labels*, *random_state=None*)
Bases: `object`

Trains classifiers to disambiguate shortforms based on context

Fits logistic regression models with tfidf vectorized ngram features. Uses sklearns LogisticRegression and TfidfVectorizer classes. Models can be serialized and loaded for later use.

Parameters

- **`shortforms`** (*`str or list of str`*) – Shortform to disambiguate or list of shortforms to build models for multiple synomous shortforms.

- **pos_labels** (`list of str`) – Labels for positive classes. These correspond to the longforms of interest in an application. For adeft pretrained models these are typically genes and other relevant biological terms.
- **random_state** (`Optional[int]`) – Optional specification of seed used when calculating crossvalidation folds and fitting the logistic regression model. Default: None

estimator
An sklearn pipeline that transforms text data with a TfidfVectorizer and fits a logistic regression.

Type py:class:*sklearn.pipeline.Pipeline*

stats
Statistics describing model performance. Only available after model is fit with crossvalidation

Type dict

stop
List of stopwords to exclude when performing tfidf vectorization. These consist of the set of stopwords in adeft.nlp.english_stopwords along with the shortform(s) for which the model is being built

Type list of str

params
Dictionary mapping parameters to their values. If fit with cv, this contains the parameters with best micro averaged f1 score over crossvalidation runs.

Type dict

best_score
Best micro averaged f1 score for positive labels over crossvalidation runs. This information can also be found in the stats dict and is not included when models are serialized. Only available if model is fit with the cv method.

Type float

grid_search
sklearn gridsearch object if model was fit with cv. This is not included when model is serialized.

Type py:class:*sklearn.model_selection.GridSearchCV*

confusion_info
Contains the confusion matrix for each pair of labels per crossvalidation split. Only available if the model has been fit with crossvalidation. Nested dictionary, *confusion_info[label1][label2][i]* gives the number of test examples where the true label is label1 and the classifier has made prediction label2 in split i.

Type dict

other_metadata
Data set here by the user will be included when the model is serialized and remain available when the classifier is loaded again.

Type dict

version
Adeft version used when model was fit

Type str

timestamp
Human readable timestamp for when model was fit

Type str

training_set_digest
Digest of training set calculated using md5 hash. Can be used at a glance to determine if two models used the same training set.

Type str

_std
Array of standard deviations of feature values over training set. This is used to calculate feature importance

Type py:class:*numpy.ndarray*

cv (*texts*, *y*, *param_grid*, *n_jobs=1*, *cv=5*)
Performs grid search to select and fit a disambiguation model

Parameters

- **texts** (*iterable of str*) – Training texts
- **y** (*iterable of str*) – True labels for the training texts
- **param_grid** (*Optional[dict]*) – Grid search parameters. Can contain all parameters from the train method.
- **n_jobs** (*Optional[int]*) – Number of jobs to use when performing grid_search Default: 1
- **cv** (*Optional[int]*) – Number of folds to use in crossvalidation. Default: 5

Example

```
>>> params = {'C': [1.0, 10.0, 100.0],
...    'max_features': [3000, 6000, 9000],
...    'ngram_range': [(1, 1), (1, 2), (1, 3)]}
>>> classifier = LongformClassifier('IR', ['insulin receptor'])
>>> classifier.train(texts, labels, param_grid=params, n_jobs=4)
```

dump_model (*filepath*)
Serialize model to gzipped json

Parameters **filepath** (*str*) – Path to output file

feature_importances ()
Return feature importance scores for each label

The feature importance scores are given by multiplying the coefficients of the logistic regression model by the standard deviations of the tf-idf scores for the associated features over all texts. Note that there is a coefficient associated to each label feature pair.

One can interpret the feature importance score as the change in the linear predictor for a given label associated to a one standard deviation change in a feature's value. The predicted probability being given by the composition of the logit link function and the linear predictor.

Returns Dictionary with class labels as keys. The associated values are lists of two element tuples each with first element an ngram feature and second element a feature importance score

Return type dict

get_model_info ()
Return a JSON object representing a model for portability.

Returns A JSON object representing the attributes of the classifier needed to make it portable/serializable and enabling its reload.

Return type dict

predict (*texts*)
Predict class labels for a list-like of texts

predict_proba (*texts*)
Predict class probabilities for a list-like of texts

train (*texts, y, C=1.0, ngram_range=(1, 2), max_features=1000, class_weight=None*)
Fits a disambiguation model

Parameters

- **texts** (*iterable of str*) – Training texts
- **y** (*iterable of str*) – True labels for training texts
- **C** (*Optional[float]*) – L1 regularization parameter logistic regression model. Follows convention of support vector machines with smaller values corresponding to stronger regularization. Default: 1.0
- **ngram_range** (*Optional[tuple of int]*) – Range of ngram features to use. Must be a tuple of ints of the form (a, b) with a <= b. When ngram_range is (1, 2), unigrams and bigrams will be used as features. Default: (1, 2)
- **max_features** (*int*) – Maximum number of tfidf-vectorized ngrams to use as features in model. Selects top_features by term frequency Default: 1000
- **class_weight** (*Optional[dict or 'balanced']*) – Weights associated with classes in the form {class_label: weight}. If not given, all classes are supposed to have weight one.

  The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)).

  Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

adeft.modeling.classify.**load_model** (*filepath*)
Load previously serialized model

Parameters **filepath** (*str*) – path to model file

Returns **longform_model** – The classifier that was loaded from the given path.

Return type py:class:*adeft.classify.AdeftClassifier*

adeft.modeling.classify.**load_model_info** (*model_info*)
Return a longform model from a model info JSON object.

Parameters **model_info** (*dict*) – The JSON object containing the attributes of a model.

Returns **longform_model** – The classifier that was loaded from the given JSON object.

Return type py:class:*adeft.classify.AdeftClassifier*

1.6 NLP

Implements a set of natural language processing tools used to pre-process text used for finding candidate longforms, recognizing defining patterns, and learning classification models.

stopwords_min contains a small collection of stopwords for use in the alignment based `AdeftLongformScorer`. english_stopwords contains a larger collection of stopwords for use in classification and anomaly detection models.

1.7 Util

Utility functions used by Adeft internally.

`adeft.util.`**`get_candidate`**(*fragment*)

Return tokens in candidate fragment up until last excluded word

Parameters

- **`fragment`** (*`str`*) – The fragment to return tokens from.
- **`use_stemming`** (*`Optional[bool]`*) – If True, stem apply stemming to tokens. Default: True

`adeft.util.`**`get_candidate_fragments`**(*text*, *shortform*, *window=100*)

Return candidate longform fragments from text

Gets fragments of text preceding defining patterns (DPs) to search for candidate longforms. Each fragment contains either a specified range of characters before a DP, or characters up until either the start of the sentence or the end of a previous DP.

Parameters

- **`text`** (*`str`*) – Text to search for defining patterns (DP)
- **`shortform`** (*`str`*) – Shortform to disambiguate
- **`window`** (*`Optional[int]`*) – Specifies range of characters before a defining pattern (DP) to consider when finding longforms. If set to 30, candidate longforms would be taken from the string "ters before a defining pattern". Default: 100

CHAPTER 2

Tutorials

The following Jupyter notebooks serve as tutorials to use Adeft models, and train new models, respectively.

2.1 Introduction to Adeft

This notebook introduces how to use pre-trained Adeft models to disambiguate shortforms in the context of surrounding text.

Availability: https://github.com/indralab/adeft/blob/master/notebooks/introduction.ipynb

2.2 Training Adeft models

This notebook walks through an example of training a new Adeft model for a shortform of interest.

Availability: https://github.com/indralab/adeft/blob/master/notebooks/model_building.ipynb

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index