nlpatl Documentation

Release 0.0.3dev

Edward Ma

Feb 28, 2022

CONTENTS:

nlpgatl is a library for active learning in machine learning experiments. The goal of NLPatl is assisting user to build high quality labeled dataset. It built on top of transformers, scikit-learn and other machine learning package. It can be applied into both cold start scenario (no any labeled data) and limited labeled data scenario.

CHAPTER

ONE

LEARNING

CHAPTER

TWO

MODEL

2.1 Classification

2.1.1 nlpatl.models.classification.sklearn_classification

2.1.2 nlpatl.models.classification.xgboost_classification

2.2 Clustering

2.2.1 nlpatl.models.clustering.sklearn_clustering

class nlpatl.models.clustering.sklearn_clustering.**SkLearnClustering**(*model_name='kmeans'*, *model_config={}*, *name='sklearn_clustering'*)

Bases: nlpatl.models.clustering.clustering.Clustering

A wrapper of sci-kit learn clustering class.

Parameters

- **model_name** (*str*) – sci-kit learn clustering model name. Possible values are *kmeans*.
- **model_config** (*dict*) – Model paramateters. Refer to https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster
- **name** (*str*) – Name of this clustering

```
>>> import nlpatl.models.clustering as nmclu
>>> model = nmclu.SkLearnClustering()
```

predict_proba(*x*, *predict_config={}*)

Parameters

- **x** (*np.ndarray*) – Raw features
- **predict_config** (*dict*) – Model prediction paramateters. Refer to https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster

Returns Feature and probabilities

Return type nlptatl.dataset.Dataset

train(*x*)

Parameters **x** (*np.ndarray*) – Raw features

2.2.2 nlpatl.models.clustering.sklearn_extra_clustering

2.3 Embeddings

2.3.1 nlpatl.models.embeddings.sentence_transformers

class nlpatl.models.embeddings.sentence_transformers.**SentenceTransformers**(*model_name_or_path*, *batch_size=16*, *name='sentence_transformers'*)

Bases: nlpatl.models.embeddings.embeddings.Embeddings

A wrapper of transformers class.

Parameters

- **model_name_or_path** (*str*) – sentence transformers model name.
- **batch_size** (*int*) – Batch size of data processing. Default is 16
- **model_config** (*dict*) – Model paramateters. Refer to https://www.sbert.net/docs/pretrained_models.html
- **name** (*str*) – Name of this embeddings

```
>>> import nlpatl.models.embeddings as nme
>>> model = nme.SentenceTransformers()
```

convert(*x*)

Parameters **x** (*np.ndarray*) – Raw features

Returns Vectors of features

Return type np.ndarray

2.3.2 nlpatl.models.embeddings.transformers

class nlpatl.models.embeddings.transformers.**Transformers**(*model_name_or_path*, *batch_size=16*, *padding=False*, *truncation=False*, *nn_fwk=None*, *name='transformers'*)

Bases: nlpatl.models.embeddings.embeddings.Embeddings

A wrapper of transformers class.

Parameters

- **model_name_or_path** (*str*) – transformers model name.
- **batch_size** (*int*) – Batch size of data processing. Default is 16
- **padding** (*bool*) – Inputs may not have same size. Set True to pad it. Default is False
- **truncation** (*bool*) – Inputs may not have same size. Set True to truncate it. Default is False
- **nn_fwk** (*str*) – Neual network framework. Either pt (for PyTorch) or tf (for TensorFlow)

- **name** (*str*) – Name of this embeddings

```
>>> import nlpatl.models.embeddings as nme
>>> model = nme.Transformers()
```

convert(*x*)

Parameters **x** (*np.ndarray*) – Raw features

Returns Vectors of features

Return type np.ndarray

2.3.3 nlpatl.models.embeddings.torchvision

class nlpatl.models.embeddings.torchvision.**TorchVision**(*model_name_or_path*, *batch_size=16*, *model_config={'pretrained': True}*, *transform=None*, *name='torchvision'*)

Bases: nlpatl.models.embeddings.embeddings.Embeddings

A wrapper of torch vision class.

Parameters

- **model_name_or_path** (*str*) – torch vision model name. Possible values are *resnet18*, *alexnet* and *vgg16*.
- **batch_size** (*int*) – Batch size of data processing. Default is 16
- **model_config** (*dict*) – Model paramateters. Refer to https://pytorch.org/vision/stable/models.html
- **transform** – Preprocessing function
- **name** (*str*) – Name of this embeddings

```
>>> import nlpatl.models.embeddings as nme
>>> model = nme.TrochVision()
```

convert(*x*)

Parameters **x** (*np.ndarray*) – Raw features

Returns Vectors of features

Return type np.ndarray

2.3.4 nlpatl.models.embeddings.nemo

class nlpatl.models.embeddings.nemo.**Nemo**(*model_name_or_path='titanet_large'*, *batch_size=16*, *target_sr=16000*, *device='cuda'*, *name='nemo'*)

Bases: nlpatl.models.embeddings.embeddings.Embeddings

A wrapper of nemo class.

Parameters

- **model_name_or_path** (*str*) – nemo model name. Verifeid. *titanet_large*, *speakerverification_speakernet* and *ecapa_tdnn*. Refer to https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/asr/speaker_recognition/intro.html
- **batch_size** (*int*) – Batch size of data processing. Default is 16
- **target_sr** (*int*) – Sample rate. Audio will be resample to this value.
- **device** (*str*) – Device for processing data
- **name** (*str*) – Name of this embeddings

```
>>> import nlpatl.models.embeddings as nme
>>> model = nme.Nemo()
```

convert(*x*)

Parameters **x** (*np.ndarray*) – Raw features

Returns Vectors of features

Return type np.ndarray

CHAPTER

THREE

SAMPLING

3.1 Certainity Sampling

3.1.1 nlpatl.sampling.certainty.most_confidence

class nlpatl.sampling.certainty.most_confidence.**MostConfidenceSampling**(*threshold=0.85*, *name='most_confidence_sampling'*)

Bases: nlpatl.sampling.sampling.Sampling

Sampling data points if the confidence is higher than threshold. Refer to https://markcartwright.com/files/wang2019active.pdf

Parameters

- **threshold** (*float*) – Minimum probability of model prediction. Default value is 0.85
- **name** (*str*) – Name of this sampling

sample(*data*, *num_sample*)

Parameters

- **x** – Values of determine the sampling
- **num_sample** (*int*) – Total number of sample for labeling
- **data** (*<MagicMock id='139720627348432'>*) –

Returns Tuple of target indices and sampling values

Return type Tuple of numpy.ndarray, numpy.ndarray

3.2 Uncertainity Learning

3.2.1 nlpatl.sampling.uncertainty.least_confidence

class nlpatl.sampling.uncertainty.least_confidence.**LeastConfidenceSampling**(*name='least_confidence_sampling'*)

Bases: nlpatl.sampling.sampling.Sampling

Sampling data points according to the least confidence. Pick the lowest probabilies for the highest class. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.219.1846&rep=rep1&type=pdf

Parameters **name** (*str*) – Name of this sampling

sample(*data*, *num_sample*)

Parameters

- **x** – Values of determine the sampling
- **num_sample** (*int*) – Total number of sample for labeling
- **data** (*<MagicMock id='139720625627488'>*) –

Returns Tuple of target indices and sampling values

Return type Tuple of numpy.ndarray, numpy.ndarray

3.2.2 nlpatl.sampling.uncertainty.entropy

class nlpatl.sampling.uncertainty.entropy.**EntropySampling**(*name='entropy_sampling'*)

Bases: nlpatl.sampling.sampling.Sampling

Sampling data points according to the entropy. Pick the highest N data points

Parameters **name** (*str*) – Name of this sampling

sample(*data*, *num_sample*)

Parameters

- **x** – Values of determine the sampling
- **num_sample** (*int*) – Total number of sample for labeling
- **data** (*<MagicMock id='139720625514096'>*) –

Returns Tuple of target indices and sampling values

Return type Tuple of numpy.ndarray, numpy.ndarray

3.2.3 nlpatl.sampling.uncertainty.margin

class nlpatl.sampling.uncertainty.margin.**MarginSampling**(*name='margin_sampling'*)

Bases: nlpatl.sampling.sampling.Sampling

Sampling data points according to the margin confidence. Pick the lowest probabilies difference between the highest class and second higest class.

Parameters **name** (*str*) – Name of this sampling

sample(*data*, *num_sample*)

Parameters

- **x** – Values of determine the sampling
- **num_sample** (*int*) – Total number of sample for labeling
- **data** (*<MagicMock id='139720625888944'>*) –

Returns Tuple of target indices and sampling values

Return type Tuple of numpy.ndarray, numpy.ndarray

3.2.4 nlpatl.sampling.uncertainty.mismatch

class nlpatl.sampling.uncertainty.mismatch.**MismatchSampling**(*name='mismatch_sampling'*)
Bases: nlpatl.sampling.sampling.Sampling

Sampling data points according to the mismatch. Pick the N data points randomly.

Parameters **name** (*str*) – Name of this sampling

sample(*data1*, *data2*, *num_sample*)

Parameters

- **x** – Values of determine the sampling
- **num_sample** (*int*) – Total number of sample for labeling
- **data1** (*Union[List[str], List[int], List[float], <MagicMock id='139720625227760'>]*) –
- **data2** (*Union[List[str], List[int], List[float], <MagicMock id='139720625309488'>]*) –

Returns Tuple of target indices and sampling values

Return type Tuple of numpy.ndarray, numpy.ndarray

3.3 Clustering Sampling

3.3.1 nlpatl.sampling.clustering.farthest

class nlpatl.sampling.clustering.farthest.**FarthestSampling**(*name='farthest_sampling'*)
Bases: nlpatl.sampling.sampling.Sampling

Sampling data points according to the distances of cluster centriod. Picking n farthest data points per number of cluster. http://zhaoshuyang.com/static/documents/MAL2.pdf

Parameters **name** (*str*) – Name of this sampling

sample(*data*, *groups*, *num_sample*)

Parameters

- **x** – Values of determine the sampling
- **num_sample** (*int*) – Total number of sample for labeling
- **data** (*<MagicMock id='139720626599776'>*) –
- **groups** (*<MagicMock id='139720626615872'>*) –

Returns Tuple of target indices and sampling values

Return type Tuple of numpy.ndarray, numpy.ndarray

See modindex for API.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

INDEX

T