

---

# **nlpatl Documentation**

***Release 0.0.3dev***

**Edward Ma**

**Feb 28, 2022**



## CONTENTS:

<b>1</b>	<b>Learning</b>	<b>3</b>
1.1	nlpatl.learning.mismatch_farthest_learning . . . . .	3
1.2	nlpatl.learning.semi_supervised_learning . . . . .	3
1.3	nlpatl.learning.supervised_learning . . . . .	3
1.4	nlpatl.learning.unsupervised_learning . . . . .	3
<b>2</b>	<b>Model</b>	<b>5</b>
2.1	Classification . . . . .	5
2.1.1	nlpatl.models.classification.sklearn_classification . . . . .	5
2.1.2	nlpatl.models.classification.xgboost_classification . . . . .	5
2.2	Clustering . . . . .	5
2.2.1	nlpatl.models.clustering.sklearn_clustering . . . . .	5
2.2.2	nlpatl.models.clustering.sklearn_extra_clustering . . . . .	6
2.3	Embeddings . . . . .	6
2.3.1	nlpatl.models.embeddings.sentence_transformers . . . . .	6
2.3.2	nlpatl.models.embeddings.transformers . . . . .	6
2.3.3	nlpatl.models.embeddings.torchvision . . . . .	7
2.3.4	nlpatl.models.embeddings.nemo . . . . .	7
<b>3</b>	<b>Sampling</b>	<b>9</b>
3.1	Certainty Sampling . . . . .	9
3.1.1	nlpatl.sampling.certainty.most_confidence . . . . .	9
3.2	Uncertainty Learning . . . . .	9
3.2.1	nlpatl.sampling.uncertainty.least_confidence . . . . .	9
3.2.2	nlpatl.sampling.uncertainty.entropy . . . . .	10
3.2.3	nlpatl.sampling.uncertainty.margin . . . . .	10
3.2.4	nlpatl.sampling.uncertainty.mismatch . . . . .	11
3.3	Clustering Sampling . . . . .	11
3.3.1	nlpatl.sampling.clustering.farthest . . . . .	11
<b>4</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



*nlpatl* 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.



## LEARNING

**1.1 `nlpatl.learning.mismatch_farthest_learning`**

**1.2 `nlpatl.learning.semi_supervised_learning`**

**1.3 `nlpatl.learning.supervised_learning`**

**1.4 `nlpatl.learning.unsupervised_learning`**





## 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](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\_fw=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\_fw** (*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.TorchVision()
```

**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\_tdn*. Refer to [https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/asr/speaker\\_recognition/intro.html](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*

## SAMPLING

### 3.1 Certainty 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 Uncertainty 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** probabilities 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 highest 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.





## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### n

`nlpatl.models.clustering.sklearn_clustering,`  
    5  
`nlpatl.models.embeddings.nemo,` 7  
`nlpatl.models.embeddings.sentence_transformers,`  
    6  
`nlpatl.models.embeddings.torchvision,` 7  
`nlpatl.models.embeddings.transformers,` 6  
`nlpatl.sampling.certainty.most_confidence,` 9  
`nlpatl.sampling.clustering.farthest,` 11  
`nlpatl.sampling.uncertainty.entropy,` 10  
`nlpatl.sampling.uncertainty.least_confidence,`  
    9  
`nlpatl.sampling.uncertainty.margin,` 10  
`nlpatl.sampling.uncertainty.mismatch,` 11



## INDEX

### C

`convert()` (*nlpatl.models.embeddings.nemo.Nemo* method), 8  
`convert()` (*nlpatl.models.embeddings.sentence\_transformers.SentenceTransformers* method), 6  
`convert()` (*nlpatl.models.embeddings.torchvision.TorchVision* method), 7  
`convert()` (*nlpatl.models.embeddings.transformers.Transformers* method), 7  
`nlpatl.sampling.uncertainty.margin`, 10  
`nlpatl.sampling.uncertainty.mismatch`, 11  
`MostConfidenceSampling` (class in *nlpatl.sampling.certainty*), 9

### E

`EntropySampling` (class in *nlpatl.sampling.uncertainty.entropy*), 10

### F

`FarthestSampling` (class in *nlpatl.sampling.clustering.farthest*), 11

### L

`LeastConfidenceSampling` (class in *nlpatl.sampling.uncertainty.least\_confidence*), 9

### M

`MarginSampling` (class in *nlpatl.sampling.uncertainty.margin*), 10

`MismatchSampling` (class in *nlpatl.sampling.uncertainty.mismatch*), 11

module

`nlpatl.models.clustering.sklearn_clustering`, 5

`nlpatl.models.embeddings.nemo`, 7

`nlpatl.models.embeddings.sentence_transformers`, 6

`nlpatl.models.embeddings.torchvision`, 7

`nlpatl.models.embeddings.transformers`, 6

`nlpatl.sampling.certainty.most_confidence`, 9

`nlpatl.sampling.clustering.farthest`, 11

`nlpatl.sampling.uncertainty.entropy`, 10

`nlpatl.sampling.uncertainty.least_confidence`, 9

### N

`Nemo` (class in *nlpatl.models.embeddings.nemo*), 7

`nlpatl.models.clustering.sklearn_clustering` module, 5

`nlpatl.models.embeddings.nemo` module, 7

`nlpatl.models.embeddings.sentence_transformers` module, 6

`nlpatl.models.embeddings.torchvision` module, 7

`nlpatl.models.embeddings.transformers` module, 6

`nlpatl.sampling.certainty.most_confidence` module, 9

`nlpatl.sampling.clustering.farthest` module, 11

`nlpatl.sampling.uncertainty.entropy` module, 10

`nlpatl.sampling.uncertainty.least_confidence` module, 9

`nlpatl.sampling.uncertainty.margin` module, 10

`nlpatl.sampling.uncertainty.mismatch` module, 11

### P

`predict_proba()` (*nlpatl.models.clustering.sklearn\_clustering.SkLearnClustering* method), 5

### S

`sample()` (*nlpatl.sampling.certainty.most\_confidence.MostConfidenceSampling* method), 9

`sample()` (*nlpatl.sampling.clustering.farthest.FarthestSampling* method), 11

`sample()` (*nlpatl.sampling.uncertainty.entropy.EntropySampling*  
    *method*), [10](#)

`sample()` (*nlpatl.sampling.uncertainty.least\_confidence.LeastConfidenceSampling*  
    *method*), [9](#)

`sample()` (*nlpatl.sampling.uncertainty.margin.MarginSampling*  
    *method*), [10](#)

`sample()` (*nlpatl.sampling.uncertainty.mismatch.MismatchSampling*  
    *method*), [11](#)

`SentenceTransformers` (class in *nl-*  
    *patl.models.embeddings.sentence\_transformers*),  
[6](#)

`SkLearnClustering` (class in *nl-*  
    *patl.models.clustering.sklearn\_clustering*),  
[5](#)

## T

`TorchVision` (class in *nl-*  
    *patl.models.embeddings.torchvision*), [7](#)

`train()` (*nlpatl.models.clustering.sklearn\_clustering.SkLearnClustering*  
    *method*), [5](#)

`Transformers` (class in *nl-*  
    *patl.models.embeddings.transformers*), [6](#)