MyProject Documentation

Contents

Module `utils_nlp`

Functions

Function `calc_test_result`

```
def calc_test_result(
    output,
    actual
)
```

This function prints the confusion matrix (normalised i.e. rows sum to 1) and the classification report using sklearn module.

Args —-= **`output`** : numpy array representing class probabilities

`actual` actual label (may be one hot encoded or class probabilities)

Returns —-= No return value, prints confusion matrix and report to stdout

Function `get_report`

```
def get_report(
    y_true,
    y_pred,
    classes
)
```

This function parses the classification report given by sklearn to get all the row names metric values as floats and supports for each class label.

Args —-= **`y_true`** : true (numerical) labels of data

`y_pred` predicted (numerical) labels of the same data
`classes` a python list of class labels

Returns —-= class_names : a python list of class labels (here, row names from report)

plotMat numerical values (metrics) in the classification report

support the number of instances for each class_name present in report

Function `get_scores`

```
def get_scores(
    y_true,
    y_pred,
    classes
)
```

This function calculates the correct and incorrect counts for each label as a fraction to the total instances of that class.

Args —-= **`y_true`** : true (numerical) labels of data

`y_pred` predicted (numerical) labels of the same data
`classes` a python list of class labels

Returns —-= numpy array of tuple of (correct,incorrect) fractions for each class

Function `plot_clf_report`

```
def plot_clf_report(
    classes,
    plotMat,
    support,
    cmap=<matplotlib.colors.LinearSegmentedColormap object>,
    filename='classification_report'
)
```

This function plots the classification report as an image, using the parsed values from the sklearn classification report and saves the image in the current working directory.

Args —-= **`classes`** : a python list of class labels

`plotMat` numerical values (metrics) in the classification report
`support` the number of instances for each class present in report
`cmap` the color map to be used in the output image
`filename` the filename with which the plot will be saved (can be a path too)

Returns —-= No return value. Shows and saves the report image.

Function `plot_confusion_matrix`

```
def plot_confusion_matrix(
    classes,
    mat,
    normalize=True,
    cmap=<matplotlib.colors.LinearSegmentedColormap object>,
    filename='confusion_matrix'
)
```

This function plots the confusion matrix as an image, using the parsed values from the confusion matrix and saves the image in the current working directory.

Args —-= **`classes`** : a python list of class labels

`mat` numerical values (metrics) in the confusion matrix
`normalize` controls the normalization of the confusion matrix (rows sum to 1 or not)
`cmap` the color map to be used in the output image
`filename` the filename with which the plot will be saved (can be a path too)

Returns —-= No return value. Shows and saves the confusion matrix image.

Function `plot_tag_scores`

```
def plot_tag_scores(
    classes,
    scores,
    filename='tag_scores'
)
```

This function plots the histogram for tag scores and saves the image in the current working directory.

Args —-= **`classes`** : a python list of class labels

`scores` a dictionary of correct and incorrect counts for each label
`filename` the filename with which the plot will be saved (can be a path too)

Returns —-= No return value. Shows and saves the tag scores plot.

Function `re_label`

```
def re_label(
    y
)
```

re_label function takes in input a list/array of any hashable data type, for example, strings, integers etc, which represent class labels. This function replaces all the class labels with integers 0 to K-1 where K is the number of classes (distinct elements present in input). Args: y: python list/numpy array with each element a hashable data type Returns: relabeled numpy array in integer format

Function `remove_labels`

```
def remove_labels(
    X,
    y,
    labels_to_remove
)
```

remove_labels does exactly what the name suggests. This function removes instances from both X and y where that instance has a label which is provided in the argument labels_to_remove. Instances refer to rows in the data X and labels y.

Args —-= **`X`** : data having same number of instances as y

`y` numpy array of labels corresponding to the feature matrix X labels_to_remove: a list of labels which will be removed from the data as well as label list.

Returns —-= X_new : X with some removed rows

y_new y with some removed rows

Function `round_label`

```
def round_label(
    a,
    limit=1
)
```

round_label rounds off labels where the label list actually is a continuous real value depicting the intensity of the sentiment. Negative values are rounded off to nearest smaller integer Positive values are rounded off to nearest larger integer The output is constrained to be in the range [-limit,limit]

Args —-= **`a`** : real number (treated as intensity)

`limit` argument for constraining the output

Returns —-= rounded off value

Function `to_categorical`

```
def to_categorical(
    y,
    num_classes=None,
    dtype='float32'
)
```

to_categorical converts labels to their one-hot encoded representation.

Args —-= `y` : array/list of labels within range 0 to K-1 where K <= num_classes

`num_classes` the number of classes to encoded, if None, max(y)+1 is used
`dtype` dtype of the output array

Returns —-= one-hot encoded vector of shape NxK where N is number of examples and K is number of classes.

Generated by *pdoc* 0.9.2 (https://pdoc3.github.io).