Trimodal IEMOCAP Documentation

Contents

Module `trimodal-fusion-for-iemocap`

Functions

Function `Bimodal`

```
def Bimodal()
```

Trains bimodal layers using the input feature vector and creates the bim_act.pickle file. The bimodal models and fusion layers are defined herein

Returns: None

Function `Trimodal`

```
def Trimodal()
```

Trains the trimodal layers Returns: result: a 3D numpy array of predictions of model in one hot encoded form (test_size by sequence_length by number of classes) test_label: a 3D numpy array of true labels in one hot encoded form (test_size by sequence_length by number of classes) test_mask: a 2D numpy array telling which inputs to ignore while calculating accuracies

Function `createOneHot`

```
def createOneHot(
    train_label,
    test_label
)
```

this function takes a np array of labels and converts it into a one hot encoded 2D matrix for use with categorical_crossentropy loss

Args: train_label: a 1d numpy array of integers corresponding to labels test_label: a 1d numpy array of integers corresponding to labels

Returns —-= a pair of 2D numpy arrays of size (dimension 0 of corresponding input, max label value in correspodning input+1)

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 `load_bim_acts`

```
def load_bim_acts()
```

loads the pickle file saved at "./bim_act.pickle" containing a dictionary of bimodal activations

Returns: merged_train_data: 3D numpy array of input to trimodal layers merged_test_data: 3D numpy array to test trimodal layers train_label: 3D numpy array of one hot encoded labels test_label: 3D numpy array of one hot encoded labels to test trimodal layers train_mask: 2D numpy array telling utternaces to ignore in train data test_mask: 2D numpy array telling utternaces to ignore in test data max_len: maximum sequence length

Function `load_unim_acts`

```
def load_unim_acts()
```

load unimodal activations saved at "../input/multimodal-sentiment/unimodal.pickle" The pickle file must contain a dictionary of numpy arrays having the feature vectors with keys: 'audio_train', 'video_train', 'audio_test', 'text_train', 'test_mask', 'test_label', 'video_test', 'train_mask', 'text_test', 'train_label'

Returns: merged_train_data: 3D numpy array of input to trimodal layers merged_test_data: 3D numpy array to test trimodal layers train_label: 3D numpy array of one hot encoded labels test_label: 3D numpy array of one hot encoded labels to test trimodal layers train_mask: 2D numpy array telling utternaces to ignore in train data test_mask: 2D numpy array telling utternaces to ignore in test data

Function `plot_clf_report`

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

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>
)
```

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,
    normalize=True
)
```

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 `report_plot`

```
def report_plot(
    result,
    test_label,
    test_mask
)
```

Generates various classification stats in terminal

Args: result: a 3D numpy array of predictions of model in one hot encoded form (test_size by sequence_length by number of classes) test_label: a 3D numpy array of true labels in one hot encoded form (test_size by sequence_length by number of classes) test_mask: a 2D numpy array telling which inputs to ignore while calculating accuracies

Returns: None
