Documentation for the nlpred Library

Model.py

- Black Box Method **getModels**(trueSource, falseSource, isRedditData: bool = True, max_enet_iterations: int = 4900, range_iterations: int = 5, focus_iterations: int = 10, start: int = 10, stop: int = 50, step: int = 5, focus: int = 10, appendFullModelList: bool = False):

  Prepares textual input for model creation, then creates Model objects for various topic quantities.
  The generated models are evaluated and five possible optimal model candidates are returned in a Dictionary object.

  The "best" model depends on the original corpus and purpose of the model, so it is left to the user to determine which of the five models returned by this method best suits their purposes.

  If the user is unsure of which model to use, it would behoove them to simply choose the model with the highest AUC, stored as "AUC" in the return Dictionary.

  Note that this is the main method of the nlpred package; it can be used as a black box for model creation by simply passing the arguments "trueSource", "falseSource", and "isRedditData" while leaving the rest of the arguments as their defaults.

  @arg **trueSource** Either:
  1) if isRedditData = true, the filename of the JSON file containing "true" Strings, including the .json extension
  2) if isRedditData = false, an array of Strings which are known to be associated with a "true"/"1" prediction

  @arg **falseSource** Either:
  1) if isRedditData = true, the filename of the JSON file containing "false" Strings, including the .json extension
  2) if isRedditData = false, an array of Strings which are known to be associated with a "false"/"0" prediction

  @arg **isRedditData** Boolean:
  1) True indicates the data is given as reddit JSON objects, and _getCorpusFromReddit will be called.
  2) False indicates the data is given as vectors of Strings, and _getCorpusFromVectors will be called.
  Note that redditData is TRUE by default.

@arg **max_enet_iterations** the maximum number of iterations for the Stochastic Average Gradient Descent ('saga') solver to go through for the elastic net CV. Default is 4900. Lower numbers drastically decrease runtime for large datasets, but may create inferior models.

@arg **range_iterations** the number of models to generate for each topic quantity in range(start, stop+1, step = step).
Setting this to 0 will disable the model generation dictated by start, stop, and focus. i.e., models will be only generated in the Focus range.
Must be a non-negative integer. Default is 5.

@arg **focus_iterations** the number of models to generate for each topic quanity in the interval [focus-5, focus+5].
Setting this to 0 will disable extra model generation in the focus range.
Must be a non-negative integer. Default is 20.

@arg **start** the smallest quantity of topics to generate a model with. Must be an integer. Default is 10.

@arg **stop** the largest quantity of topics to generate a model with. Must be an integer. Default is 50.

@arg **step** the integer value to count by for the topic quantities to generate models with. Must be an integer. Default is 5.
e.g., if step = 5, start = 10, and stop = 50, then models will be generated for each of the following topic quantities: [10,15,20,25,30,35,40,45,50].

@arg **focus** the user's best guess for the optimal number of topics. Must be an integer. Default is 10.
Additional models will be generated for each topic quantity in the interval [focus-4, focus+4].

@arg **appendFullModelList** boolean; if True, the full list of models generated sorted by AUC in descending order will be appended to the return Dictionary.
Default is False.

@return A Dictionary containing five candidates for an optimal model, stored as Model objects:

1) "AUC", the model with the highest AUC
2) "acc", the model with the highest accuracy
3) "rec", the model with the highest recall
4) "acc_1se", the model with the highest accuracy among models whose AUC is within 1 standard error of the maximum
5) "rec_1se", the model with the highest recall among models whose AUC is within 1 standard error of the maximum

If optional argument appendFullModelList = True (default is False):

6) "modelList", a list containing Model objects for all models generated, sorted by AUC in descending order.

- Class Model
  - Instance Variables:
    - "topicQuantity", the number of topics used in .this model
    - "fit", the fitted model itself
    - "topicLDAModel", the LDA object for the fitted topic model
    - "confusion_matrix", the confusion matrix of the fitted model
    - "accuracy", the classification rate, i.e. the percent of correct predictions; or, the number of correct predictions made divided by the total number of predictions made.
    - "recall", a.k.a sensitivity; P(Y=1 | X=1), proportion of true samples correctly identified as true. i.e., the number of true samples correctly identified divided by the total number of true samples. The recall is intuitively the ability of the classifier to find all the positive samples.
    - "AUC", a general measure of fit for the model as per the chosen predictive threshold for "true" vs "false" from logreg predictive output.
    - "id", a unique identifier for the model. Note that this is a UUID object. The actual id may be found in “id.hex”. The UUID wrapper object is kept for aesthetic reasons with regard to printing, etc.
  - Instance Methods:
    - **predict**(self, listOfDocStrings: List[str]):

      Predicts True/False (1/0) for each document (String) in the List listOfDocStrings argument using the model stored in this Model object.

      @arg **listOfDocStrings** a List of Strings wherein each String contains the text of a document to predict True/False from

      @return a DataFrame containing three columns:

      1) 'paper_text', the original input textual data

      2) 'paper_text_processed', the processed (tokenized with punctuation and capitalization removed) textual data

      3) 'value', the 1/0 (True/False) prediction for the 'paper_text' entry in the same row

    - **save**(self, fileName: str = "my_model.pkl"):

      Saves the model as a .pkl file.

      @arg **fileName** the desired filename, including the .pkl extension. e.g., "my_model.pkl"

@postState the model object will be saved as fileName in the current working directory.

- **print**(self):

  Prints to stdout a summary of the model.

- Static Methods
  - **loadModel**(fileName: str) -> Model:

    Loads a model saved as a .pkl file

    @arg **fileName** the file name of the saved model, including the .pkl extension. e.g., "my_model.pkl"

    @return a Model object of the model saved in the specified .pkl file.

  - **predictFromFile**(fileName: str, listOfDocStrings: List[str]):

    Predicts True/False (1/0) for each document (String) in the List listOfDocStrings argument using the model stored in the indicated file location.

    @arg **fileName** the file name of the saved model, including the .pkl extension. e.g., "my_model.pkl"

    @arg **listOfDocStrings** a List of Strings wherein each String contains the text of a document to predict True/False from

    @return a DataFrame containing three columns:

    1) 'paper_text', the original input textual data

    2) 'paper_text_processed', the processed (tokenized with punctuation and capitalization removed) textual data

    3) 'value', the 1/0 (True/False) prediction for the 'paper_text' entry in the same row

  - **getCompDF**(modelList: List[Model]):

    Takes a list of Model objects and places them in a pandas.DataFrame, sorted by AUC, for easy comparison.

    @arg **modelList** a list of Models to create the DataFrame from

    @return a pandas.DataFrame object containing pertinent statistics for model comparison, sorted by AUC of the models in descending order.

Each row of the DataFrame represents a Model from the argument modelList.
Note that the index of the data frame rows correspond to the index of the model in the argument modelList.

TextPrep.py

- **preparePredData**(docsToPredict: List[str]):

Prepares a list of Strings for prediction using a previously created model.

@arg **docsToPredict** a list of Strings containing the documents to have predictions made from

@return A Dictionary with two elements:

1) "processedCorpusDataFrame", a data frame containing two columns:
    I) 'paper_text', the original input textual data
    II) 'paper_text_processed', the processed (tokenized with punctuation and capitalization removed) textual data
2) "count_data", the token counts from the processed textual data to be used for prediction

- **prepareTestTrainData**(trueSource, falseSource, isRedditData: bool = True):

Prepares textual input for model creation. Supports both reddit JSON input and vectors of Strings.

@arg **trueSource** Either:

1) if isRedditData = true, the filename of the JSON file containing "true" Strings, including the .json extension
2) if isRedditData = false, an array of Strings which are known to be associated with a "true"/"1" prediction

@arg **falseSource** Either:

1) if isRedditData = true, the filename of the JSON file containing "false" Strings, including the .json extension
2) if isRedditData = false, an array of Strings which are known to be associated with a "false"/"0" prediction

@arg **isRedditData** Boolean:

1) True indicates the data is given as JSON objects, and _getCorpusFromReddit will be called.
2) False indicates the data is given as vectors of Strings, and _getCorpusFromVectors will be called.

Note that redditData is TRUE by default.

@return A Dictionary with two elements:

1) "processedCorpusDataFrame", a data frame containing three columns:
    I) 'paper_text', the original input textual data
    II) 'value', the corresponding 1/0 (i.e., true/false) value associated with the 'paper_text' entry in the same row
    III) 'paper_text_processed', the processed (tokenized with punctuation and capitalization removed) textual data
2) "count_data", the token counts from the processed textual data to be used in model creation