Re-Energize DR3:

Code Documentation and Description:

Quality Assessment of Social Media Data

for Crisis Management

UCL Energy Institute

Islands Laboratory

15th February, 2022

Code Dependencies

This code uses Python 3.7.9 running after Miniconda3 environments. The code has been tested using a GPU NVIDIA Geforce RTX 2060. The list of dependencies are described in the file *requirements.yaml*. To install them using Anaconda/Miniconda, simply create and activate an environment:

```
conda env create --name <cool_environment_name> --file=requirements.yaml
conda activate <cool_environment_name>
```

Summary

This document describes the codes to reproduce a natural language processing framework to analyse social media data related to disaster and crisis events. The main or core part, refers to train, load, and use several machine learning models to classify messages related to certain target categories as a **filtering approach**.

These parts of the framework are briefly represented in the diagram below:

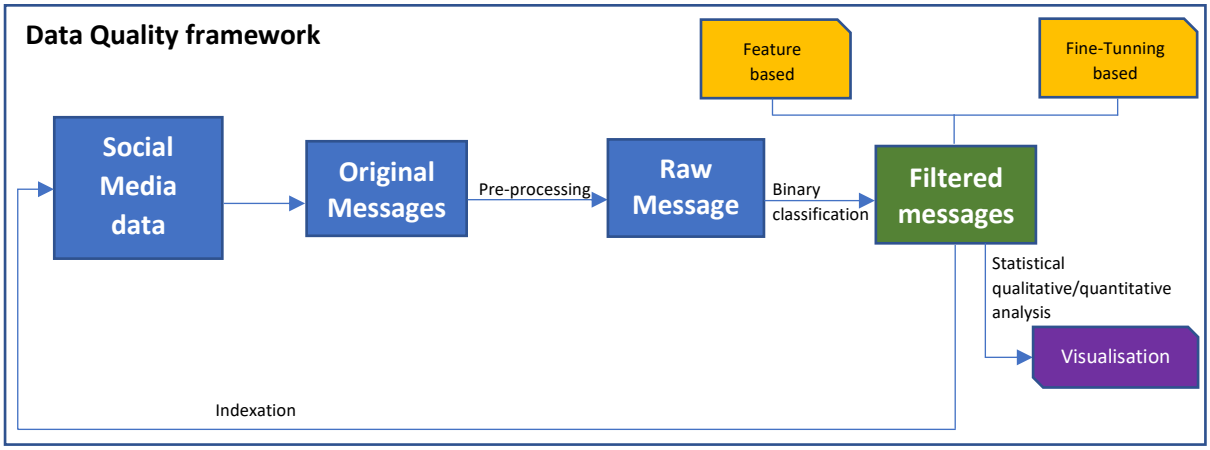

Figure 1: Overview diagram of the data quality framework.

The code is remains in a private repository in GitHub* that will be released and made it publicly available along with its corresponding citation at the Islands Laboratory Website.

* https://github.com/islandslab/NLP-Disaster

Filtering via machine learning classification of target categories

Once all dependences are installed, you aim to run the main file `disaster_response.py` shown in the image below. Before that, it is crucial to ensure parameters and paths are correct to import the data, and to select the method and target categories.

```
@author: Victor Ponce-Lopez @ UCL Energy Institute

================================================================
Filtering, Text and Sentiment Analysis, and Topic Classification
================================================================
 """
if __name__ == '__main__':

    from utils import prepareData, showWords, saveResults
    from binary import classifCat, classifySentiment
    from multiclass import LearnInferTopics
    #%matplotlib inline

    # Set parameters and import data
    path = 'data/'
    datasets = ['socialmedia_disaster', 'multilingual_disaster', 'heatwaveUK',
                'floodUK', 'stormUK', 'snowIceUK', 'hurricaneFlor', 'hurricaneNC',
                'floodAccra','mauritiusFb','UnifiedMETCEHFloodsUK']
    dataset = datasets[2]
    bMethods = ['bow', 'tfidf', 'distBert', 'all']; bMethod = bMethods[2]
    mMethods = ['lda-bow', 'lda-tfidf', 'all']; mMethod = mMethods[2]
    targets = ['severe']  #['aid_related','medical_products','other_aid','hospitals','aid_centers']  #['water','food','sh
    save, show, retrain, pos_ratio = False, True, False, 0  #(default for floods: 0.45)

    trainData, valData, testData, data, pos_ratio = prepareData(path, dataset, pos_ratio, targets)
```

The list of main parameters are described below:

- `path` : points at the `data` directory where the data files are located in JSON or CSV format.

- `datasets` : contains a list of available tested datasets. The available models specified in the parameter `bMethod` can be trained and evaluated on the annotated datasets `'socialmedia_disaster'`, `'multilingual_disaster'`, and `'UnifiedMETCEHFloodsUK'`.

    1. To **evaluate an existing model** on a dataset, first <u>your model must be stored in the correct paths to be loaded properly</u>, as explained by the end of this section 1. Then, simply set the index of the target dataset –e.g. `dataset = datasets[2]` to select the HeatwaveUK2020 data collection from the list of available `datasets`.

    2. To **train or fine-tune a new model** on the annotated datasets you must select one of the annotated datasets – e.g. `dataset = datasets[1]` will use the MDRM dataset 'Multilingual Disaster Response Messages'. Note that training a model requires high computational demand may take many hours or days - it is recommended to do this on a computing cluster. If you have a model trained already, please follow step 1. If your model already exists in the path, then it will be evaluated only (step 1) - unless the parameter `retrain` is set to `True` (see more information about this parameter below).

- `bMethod` : it specifies the method for loading, training or fine-tunning the model to use for binary classification on the dataset specified previously. You can use the methods `'bow'` or `'tfidf'` for traditional hand-crafted features, the deep learning method `'distBert'`, or all of them –e.g. `bMethod = bMethods[2]` selects the deep learning method `'distBert'`.

Please refer to the document description of data and methods for more information about feature-based and fine-tunning methods.

- The Boolean parameters `save, show` and `retrain`, indicates respectively whether to store the results of the classification approaches, to visualise their results, and to continue fine-tunning the distilBERT model with `bMethod = bMethods[2]` only and only if `'distBert'` is the selected method.

- `targets :` it specifies the target category to classify. For annotated datasets, it will specify the messages related to this category -or group of categories- as the label; these labelled messages will be assigned after running the function `'prepareData(…)'`, which prepares the data splits for training and/or testing. The values of target categories can be set to:

    1. `targets = ['disaster'] :` messages labelled as **disaster**-related information.
    2. `targets = ['floods'] :` messages labelled as **flood**-related information.
        - Note: to make it consistent, the selection of this category requires to replace the value of `renamedColumns` by `renamedColumns = ['floods']` and to replace the element `'floods'` by `'related'` from the `headerDropper` in the function `getDataInfo(dataset)` of the script `utils.py`.
    3. `targets = ['aid_related','medical_products','other_aid', 'hospitals', 'aid_centers'] :` messages labelled as related to multiple categories grouped as **medical information**.
    4. `targets = ['water','food','shelter','medical_help'] :` messages labelled as related to multiple categories grouped as **humanitarian standards**' information.
    5. `targets = ['severe'] :` messages labelled as information related to severe events.

    Note that, by default, the models to use for the target categories 'disaster', 'flood', 'medical information', and 'humanitarian standards', are those trained on the MDRM dataset `'multilingual_disaster'`. And the model to use by default for the target category 'severe' is trained on the `'UnifiedMETCEHFloodsUK'` dataset 'Unified MET and CEH records of reports from floods in the UK'. The reasons are because either 1) these are the datasets found to learn better models for these target categories or 2) the categories are not present in the other annotated datasets to learn a model from them. For example, you can still use the dataset `'socialmedia_disaster'` to train a model for the target category `'disaster'` because this dataset contains this category as annotated labels – however, you cannot use this dataset to train a model for the remainder categories because it does not contain annotated labels for any other category. Having said that, you can use an existing model already trained for a target category to classify messages of that category on any other non-annotated dataset.

    Please refer to the document description of data and methods for more information about the annotated datasets.

    The function `getDataInfo(dataset)` in the file `utils.py` contains a list of data files for each dataset. Be sure to load the proper file(s) and keep the target label in `renamedColumns` for consistency for the supervised classification approaches.

    The selection of `targets` depends on the dataset to be used and must match `modeldata`.

- `modeldata :` it refers to the directory path where the models for each target category are saved. This should match with the parameter `targets` for consistency.

* https://github.com/islandslab/NLP-Disaster

- `pos_ratio` : indicates the positivity ratio of messages belonging to the selected target category. This value is empirically set to balance the ratio of positive and negative samples for each class and hence learn more generic models. By default is set to 0 (keep all the original samples).

```
54      ##########################################################################
55      # Classify Target Category and get indexes
56      idx_bow, idx_tfidf, idx_distBert = classifCat(trainData, valData, testData,
57                                                    retrain, dataset, modeldata,
58                                                    save, pos_ratio, bMethod, show)
59      ##########################################################################
```

After running the `classifCat(…)` function, the models trained with the **BoW** and **TF-IDF** methods are saved in the root directory as simple lightweight Pickle files of the form:

model*_<method>_<modeldata>_<pos_ratio>.*pkl

– e.g. a BoW model learned for the target category 1 (`'disaster'`) on the MDRM dataset with a default value of `pos_ratio` will be saved as files named *`model_BoW_multidisaster_0.pkl`* or *`model_BoW_multidisaster_full_0.pkl`*. The suffix '_full' is for models learned also with validation data.

On the other hand, the fine-tuned models trained with the **distilBert** method are saved as checkpoints within directory folders named

distilBert_<modeldata>

– e.g. a fine-tuned **distilBERT model** learned for the target category 5(`'severe'`) on the dataset `'UnifiedMETCEHFloodsUK'` (Unified MET and CEH records of reports from floods in the UK) with a default value of will be saved within a folder named *`distilBert_Unifisevere`*.

Then, these models can be used to other non-annotated data collections to infer the messages related to the target category/event and the other messages. This inference is done via classification of all test messages into the target category. The results of a classification are saved within the folder *results* via the function `saveResults(…)`, which creates a lightweight Pickle file for each dataset. This file contains, for each method, the indexes of messages classified as relevant to the target category.

Indexation is then used as the **filtering approach** to retrieve the original messages from the data collection related to the target category.

* https://github.com/islandslab/NLP-Disaster