Disaster Response Message Classification Pipelines

Libraries

- pandas
- numpy
- sqlalchemy
- matplotlib
- plotly
- NLTK
- NLTK [punkt, wordnet, stopwords]
- sklearn
- joblib
- flask

Project Description

- Figure Eight Data Set: Disaster Response Messages provides thousands of messages that have been sorted into 36 categories. These messages are sorted into specific categories such as Water, Hospitals, Aid-Related, that are specifically aimed at helping emergency personnel in their aid efforts.
- The main goal of this project is to build an app that can help emergency workers analyse incoming messages and sort them into specific categories to speed up aid and contribute to more efficient distribution of people and other resources.

File Description

There are three main folders:

1. data
   - disaster_categories.csv: dataset including all the categories
   - disaster_messages.csv: dataset including all the messages
   - process_data.py: ETL pipeline scripts to read, clean, and save data into a database
   - DisasterResponse.db: output of the ETL pipeline, i.e. SQLite database containing messages and categories data
2. models
   - train_classifier.py: machine learning pipeline scripts to train and export a classifier
   - model-adaboost.pkl: output of the machine learning pipeline, i.e. a trained classifier

3. app
    - run.py: Flask file to run the web application
    - templates contain html file for the web application

Analysis

Data Preparation

- Modify the Category csv; split each category into a separate column
- Merge Data from the two csv files (messages.csv & categories.csv)
- remove duplicates and any non-categorized valued
- create SQL database DisasterResponse.db for the merged data sets

Text Preprocessing

- Tokenize text
- remove special characters
- lemmatize text
- remove stop words

Build Machine Learning Pipeline

- Build Pipeline with countevectorizer and tfidtransformer
- Seal pipeline with multioutput classifier with random forest
- Train Pipeline (with Train/Test Split)
- Print classification reports and accuracy scores

Results

1. Created an ETL pipeline to read data from two csv files, clean data, and save data into a SQLite database.
2. Created a machine learning pipeline to train a multi-output classifier on the various categories in the dataset.
3. Created a Flask app to show data visualization and classify any message that users would enter on the web page.

Installation

1. Install pip

```
sudo apt-get install python-pip
```

2. Install virtualenv

```
sudo pip install virtualenv
```

3. Create a virtualenv for this project

```
virtualenv -p python3 venv
```

Activate the virtualenv and install dependencies

```
source venv/bin/activate
pip install -r requirements.txt
```

Instructions

1. Run the following commands in the project's root directory to set up your database and model.

    - To run ETL pipeline that cleans data and stores in database **python data/process_data.py data/disaster_messages.csv data/disaster_categories.csv data/DisasterResponse.db**

    - To run ML pipeline that trains classifier and saves model **python models/train_classifier.py data/DisasterResponse.db models/classifier.pkl**

    Note: Takes about 1 hour 40 minutes to finish training on a 6 core i5 8th gen CPU

2. Run the following command in the app's directory to run your web app. **python .app/run.py**

3. Go to http://0.0.0.0:3001/

4. To run data crawling and flask app **python .app/run_all_script.py**

File Structure

```
-disaster-message-classification
 - app
 | - template
 | |- master.html  # main page of web app
 | |- go.html  # classification result page of web app
 |- run.py  # Flask file that runs app
 |-run_all_scripts.py # To execute all scripts starting from data crawling

 - data
 |- disaster_categories.csv  # data to process
 |- disaster_messages.csv  # data to process
 |- process_data.py        # etl pipeline
 |- DisasterMessages.db   # database to save clean data to

 - data_crawling
 |- rss_feed.py # To collect the rssfeed
 |- google_search.py # To search in google
 |- twitter_search.py # To collect tweets
 - models
 |- train_classifier.py  # train the model
 |- model-adaboost.pkl  # saved model
```