LAB3 DOCOMENTATION

Redwan Khan and Alex Barganier

UNIVERSITY AT BUFFALO

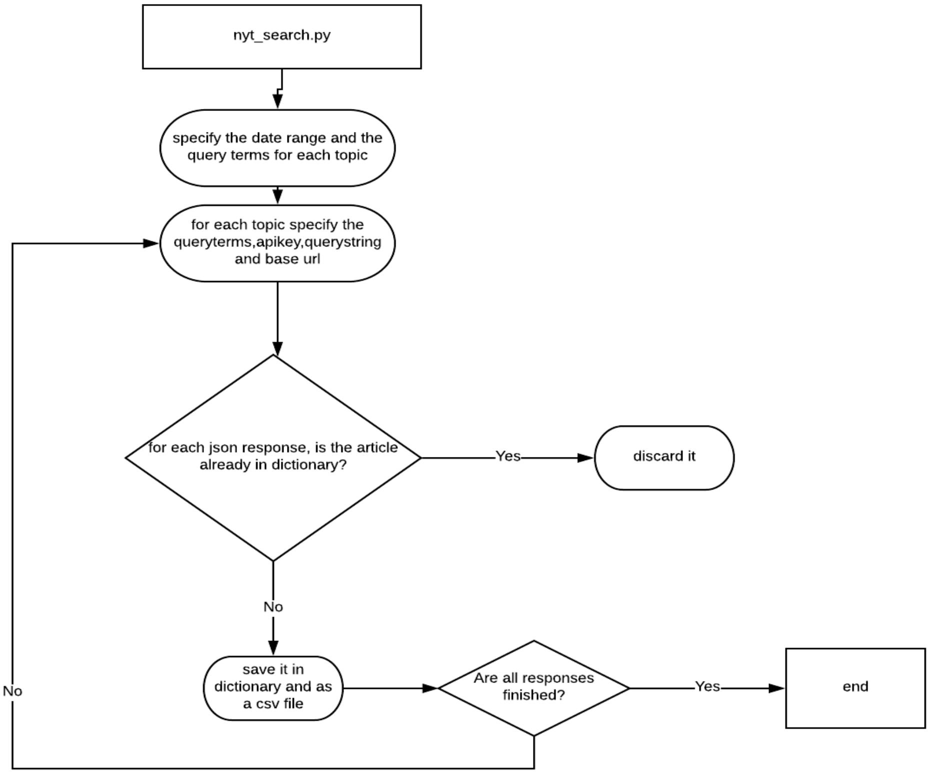
nyt_search.py

Code used for collecting news articles from NYtimes using NYtimes API

topics - structure that contains the keywords we used for generating articles related to Sports,Business, Politics and Entertainment

daterange(start_date, end_date) - specifies a range of dates between which we want to collect news articles

get_article_body(url) - collects the article text from the url specified


nyt_search.py
specify the date range and the query terms for each topic
for each topic specify the queryterms,apikey,querystring and base url
for each json response, is the article already in dictionary?
Yes
discard it
No
save it in dictionary and as a csv file
Are all responses finished?
Yes
end
No

filter_data.py

Code used to filter the collected data by removing stop words and saving each collected article in a seperate file. Even after specifying keywords some of the articles collected for politics might have actually appeared in the business section of Nytimes. So we needed to filter the articles based on their url.

filter_keywords : consists of the following fields

'**filename_prefix':** prefix of the filename,

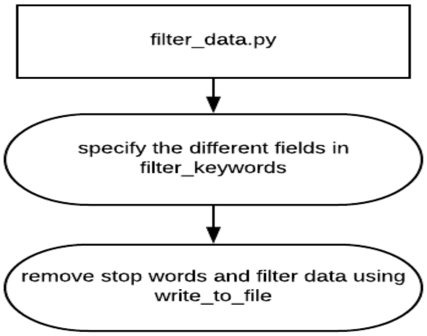
'train_dir': the place where the training data files should be saved,

'**test_dir':** the place where testing data files should be saved

'source_file': which one is the source file to look into for saving the files,

'url_keywords': what keywords to look for in an url for filtering the articles.

write_to_file(fn_prefix, dict): gets the prefix name of file and the dictionary consisting of url and content. Opens up a file with the prefix and puts the content of news articles in that file after removing stop words.


filter_data.py
specify the different fields in filter_keywords
remove stop words and filter data using write_to_file

tf_idf.py

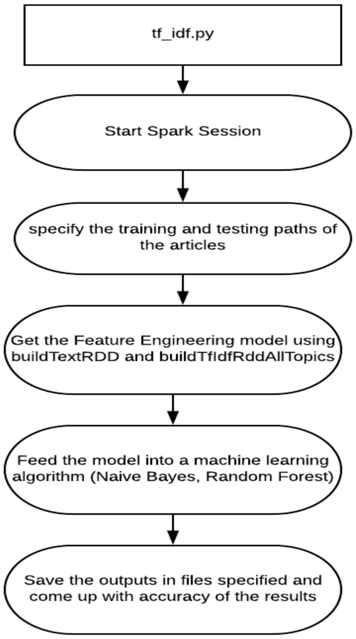
Training and Testing paths are specified for the different topics of articles. Random articles from Washington Post have also been collected and the testing paths for those articles have also been specified.The labels of the different articles have also been given.

buildTextRDD(directory, label_id): return a dataframe of label ids and articles content.

buildTfIdfRddAllTopics(business, sports, politics, entertainment): builds a dataframe for each topic and combines all of the dataframes. Returns a rescaled data of labels and features.

runTestForModel(model, test_rdd): makes a tuple of predicted label and actual label after running a machine learning model. Also gives the accuracy of the results.

Outputs saved in **EvaluationMetrics.txt, featureengineeringtestrdd.txt, featureengineeringtrainrdd.txt,PredictionNYtimes.txt, PredictionWashington.txt**


tf_idf.py
Start Spark Session
specify the training and testing paths of the articles
Get the Feature Engineering model using buildTextRDD and buildTfIdfRddAllTopics
Feed the model into a machine learning algorithm (Naive Bayes, Random Forest)
Save the outputs in files specified and come up with accuracy of the results