HATE-SPEECH DOCUMENTATION

Step -1 . Pulling data from AWS

- Go to (./download – script.sh) file location in terminal
- Hate_Speech → hsle folder → . /download-script.sh
- Run ./download-script.sh file in terminal

Step-2. Running .py files in hsle folder (Hatespeech → hsle)

- **py_stage1_posts.py**
  After pulling data , we have a post file which is located in (**Hatespeech → hsle → data → crowtangles-pages) and update some columns which are not used in further process by running this script.**
  Likewise, group data also perfoms the same but file location is different .**(Hatespeech → hsle → data →crowtangles- group)**

- **py_stage2_comments.py**
  All comment files are located in (Hate-speech → hsle → data → exportcomments-outputs) and stored as a xlsx file extension. Comment script file is used for updating dashboard data so it will be stored as a csv format in (Hate-speech → hsle → data → folder-name → processed) and used for later processes.

- **py_stage5_merge_and_annotate.py**
  Merge file is used for updating hate-speech dashboard, when we run Stage0_hsdata.py, then merge.csv file will be used for hate-speech data.

Step -3. Updating data in dashboard folder (Hatespeech → dashboard)

- After downloading data , go to a _current_ data _.json file which is located in (Hatespeech → hsle → src → _current _data_ .json) .This json file is as follow :
  Tips :

```
{
        "pages_groups":"groups",
        "daterange":"20201001_20201005",
        "desired_number_of_comments": 500, "sep": "~"
}
```

Generally, we have two steps before updating hate-speech dashboard.
First step : We checks our data which is either group or page , if our data is page , then we type “pages_groups”: “pages”. Otherwise , we type “page_groups”: “groups”,

Second step : We check date-range value and type its correspondence date like this
“ daterange” : "20201001_20201005" .

- Go to this file path (Hatespeech → dashboard → src) and Run (STAGE0_hsdata.py , STAGE1_hsaggregate.py, cleaned_aggregated_file.py and chi_square.py) files.

STAGE0_hsdata.py

By using merge.csv file , we are going to check post ,(page or group) files and take the relevant columns to use for hate-speech dashboard data.

STAGE1_hsaggregate.py

After running its script, we will get 5 csv files which is used for hate-speech dashboard.

cleaned_aggregated_file.py

Sometimes, we might see some irrelevant values in 5 csv files , so we do need to run its script file to avoid this issue.

chi_square.py

we apply chi_square equation to get its value.

Step -4. Applying data into big -query database

- Open the terminal and go to a corresponding path (hate-speech → dashboard → clean-data → aggregated),usually takes the last folder. If it is not sure, you can check a csv file inside the folder.
- Hate speech dataset name in big query database→(hatespeech-dashboard → HS_DATASET)

- Here is , terminal command Syntax

  1) big query authenication login

  - gcloud auth login

  2) Initialize big query database if it is the first time for you, otherwise no need to write it, again.
  - gcloud init

3) Generally, we have six csv files in every folder. All the files run the same syntax ,except chi_square.csv file in terminal. Chi square has only one row and replace its row in a big query database.

hsfirst-comment-effect.csv

bq load --max_bad_records 1 --skip_leading_rows=1 --noreplace – source_format=CSV -field_delimiter=$(printf '~') HS_DATASET.hsfirst-comment-effect.csv hsfirst-comment-effect.csv

hspost-effect.csv

bq load --max_bad_records 1 --skip_leading_rows=1 --noreplace – source_format=CSV -field_delimiter=$(printf '~') HS_DATASET.hspost-effect.csv hspost-effect.csv

lex-topic-page-time.csv

bq load --max_bad_records 1 --skip_leading_rows=1 --noreplace – source_format=CSV -field_delimiter=$(printf '~') HS_DATASET.lex-topic-page-time lex-topic-page-time.csv

page-hs-ratio-stage.csv

bq load --max_bad_records 1 --skip_leading_rows=1 --noreplace – source_format=CSV -field_delimiter=$(printf '~') HS_DATASET.page-hs-ratio-stage page-hs-ratio-stage.csv

page-reach.csv

bq load --max_bad_records 1 --skip_leading_rows=1 --noreplace – source_format=CSV -field_delimiter=$(printf '~') HS_DATASET.page-reach page-reach.csv

chi_square.csv (an overwritten file)

bq load --max_bad_records 1 --skip_leading_rows=1 --replace --source_format=CSV hatespeech-dashboard:HS_DATASET.CHI2 chi_square.csv

Step-5. Refreshing hate-speech dashboard

- Go to hate-speech dashboard and click a refresh button.

FINISH

Calculate accuracy for hate speech comment

File Path : Hate-speech → annotate → hs_accuracy.ipynb

- Get data from HSLE-Data Sheet

- used **three csv files** (annotate_here, annotate_2, annotate_3) from HSLE Data Sheet.

- Applied data from Column-B (LexFound) , Column-G (NewInSentences) and Column-I (ISHSL)

- Filtered by Column-I (ISHSL) whether HS Comment or not.

-Calculated a HS Comment's Accuracy by using this formula :

Accuracy =(HS Comment's Count)/(HS Comment's Count+ No HS Comment's Count)

Calculate for hate-speech lexicon accuracy

File Path : Hate-speech →annotate → lexicon_accuracy.py

step 1 : count the number of occurrence in each lexicon word

step 2 : store sum of the number of occurrence in each lexicon word as a list

step 3 : lexicon's accuracy formula :

lexicon_word = (count the number of HS comment in each lexicon)/(count the number of HS comment in each lexicon+count the number of No HS comment in each lexicon)

step 4 : save as hatespeech_accuracy.csv file in (Hate-speech → annotate → accuracy-folder)

Calculate weight_value for hate-speech lexicon

File Path : Hate-speech → annotate → (calculated_weight_func.py , calculation_weight.py)

- Get data from HSLE-Data Sheet

- used **three csv files** (annotate_here, annotate_2, annotate_3) from HSLE Data Sheet.

- Applied data from Column-B (LexFound) ,Column-F (MsgUniSeg), Column-G (NewInSentences)

- performed weight_calculation function in calculated_weight_func.py :

step 1 : check LexFound and NewInSentences lexicon

step 2 : take those lexicon's accuracy value

step 3 : Sum the occurrence of lexicon in these two columns

- applied by calculation_weight.py to call weight_calculation function from calculated_weight_func.py

-save a file as weight.csv file in (Hate-speech → annotate → accuracy-folder)

Calculate weight value for hate-speech sentence

File Path : Hate-speech → annotate → (sentence_weight_func.py,sentence_weight.py)

- Get data from HSLE-Data Sheet

- used **three csv files** (annotate_here, annotate_2, annotate_3) from HSLE Data Sheet.

- Applied data from Column-B (LexFound) ,Column-F (MsgUniSeg), Column-G (NewInSentences)

- performed weight_calculation_function in sentence_weight_func.py :

step 1 : check its MsgUniSeg (message)
step 2 : take those lexicon's accuracy value
step 3 : count the number of lexicon in each MsgUniSeg , then multiply with its lexicon count and accuracy in MsgUniSeg.

- applied by sentence_weight.py to call weight_calculation_function from sentence_weight_func.py

-save a file as sentence-weight.csv file in (Hate-speech → annotate → accuracy-folder)

Hate-speech Comment Prediction by using a random forest classifier

Steps :

- **Collected from weight.csv file and applied two columns(weighted_value, IsHS) only. Performed data preprocessing step**
- **Manipulated four models(Random-Forest, LogisticRegression,AdaBoostClassifier and KNeighborsClassifier)**
- **Selected the best model to predict a label value whether a HS Comment or No HS Comment**
- **Calculated a confusion matrix (Accuracy, Precision,Recall, F1-Score)**
- **Predict a model by using example.csv file which will predict its comments whether hs comment or not.**

Lexicon test data prediction file → lexfound_test_data_model.ipynb
Lexicon train data prediction file → lexfound_train_data_model.ipynb
sentence test data prediction file → sentence_test_data_model.ipynb
sentence test data prediction file → sentence_test_data_model.ipynb

Filter by weight.csv and sentence-weight.csv files in excel before starting a data cleaning step :

Tips :

- Applying by filter function to perform ‘IsHS’, when its column value is 2 or empty, dropped its row.
- Applying by filter function to check ‘Weighted_value’ column, if its value is a string value , then replace 0.0 .