Shankar Puramsetti

Affinity Answers - Assignment

.

Question 1: Imagine there is a file full of Twitter tweets by various users and you are provided a set of words that indicates racial slurs. Write a program that can indicate the degree of profanity for each sentence in the file. Write in any programming language (preferably in Python) - make any assumptions, but remember to state them.

Solution

In the context of this program, the profanity score means the number of racial slurs found in a given tweet. The program checks each word in the tweet against a set of known racial slurs and increments the profanity score for each occurrence of a racial slur. The final profanity score for a tweet is the total number of racial slurs found in the tweet.

For example, if a tweet contains two racial slurs, the profanity score for that tweet would be 2. The higher the profanity score, the more likely the tweet contains offensive language and may be inappropriate or offensive to certain audiences.

Here's an example Python program that reads in a file of Twitter tweets, checks each tweet for the presence of racial slurs, and outputs a profanity score for each tweet based on the number of racial slurs found.

Assumptions:

```
The file of Twitter tweets is in a plain text format.
Each tweet is on a separate line in the file.
The set of racial slurs is provided as a Python set of lowercase strings.
The profanity score for each tweet is calculated as the number of racial slurs found in the tweet.
```

Here's the Python code:

In [ ]:
```python
# Open the file of Twitter tweets
with open('tweets.txt', 'r') as f:
    # Read in each tweet as a separate line
    tweets = f.readlines()

# Define the set of racial slurs
racial_slurs = {'slur1', 'slur2', 'slur3', ...}

# Loop through each tweet and check for racial slurs
for tweet in tweets:
    # Split the tweet into a list of words
    words = tweet.split()

    # Initialize the profanity score to 0
    profanity_score = 0

    # Loop through each word in the tweet and check for racial slurs
    for word in words:
        if word.lower() in racial_slurs:
            # Increment the profanity score if a racial slur is found
            profanity_score += 1

    # Output the profanity score for the tweet
    print('Tweet: {}\nProfanity score: {}\n'.format(tweet.strip(), profanity_score))
```

This program reads in the file 'tweets.txt', splits each tweet into a list of words, and checks each word for the presence of racial slurs. The profanity score for each tweet is calculated as the number of racial slurs found in the tweet. Finally, the program outputs the profanity score for each tweet.

Question 2: Which is an interesting data set you discovered recently? Why is it your favorite? No datasets on Kaggle, please.

Answer:

I recently came across an intrestind dataset, The Image Spam Hunter dataset is a collection of 1,000 color images in JPEG format, with each image measuring 250 x 250 pixels. The images are a mix of spam and non-spam images, with 500 spam images and 500 non-spam images. The spam images are designed to resemble legitimate email messages, with text and images intended to trick recipients into clicking on a link or downloading a file. The non-spam images are generally benign images, such as photos of animals or landscapes. The dataset is interesting because it provides a labeled collection of images that can be used to develop and test algorithms for identifying spam images, which is a common problem in email security. The dataset can be used for a variety of applications, including developing image recognition algorithms for spam detection, analyzing the characteristics of spam images, or comparing the performance of different image classification algorithms.

Dataset: https://users.cs.northwestern.edu/~yga751/ML/ISH.htm (https://users.cs.northwestern.edu/~yga751/ML/ISH.htm)

Question 3: The following questions test your aptitude for interacting with databases. The questions are based off the following public SQL DB: https://docs.rfam.org/en/latest/database.html (https://docs.rfam.org/en/latest/database.html)

a. How many types of tigers can be found in the taxonomy table of the dataset? What is the "ncbi_id" of the Sumatran Tiger? (hint: use the biological name of the tiger)

b. Find all the columns that can be used to connect the tables in the given database

Solution:

a)

To find out how many types of tigers can be found in the taxonomy table of the dataset, we can use the following SQL query:

```
SELECT COUNT(*) FROM taxonomy WHERE species = 'Panthera tigris';
```

This query will return the total number of tiger species found in the taxonomy table of the dataset.

To find the "ncbi_id" of the Sumatran Tiger, we can modify the query as follows:

```
SELECT ncbi_id FROM taxonomy WHERE species = 'Panthera tigris' AND subtaxa = 'sumatrae';
```

This query will return the ncbi_id of the Sumatran Tiger, which is a subspecies of the tiger.

b)

To find all the columns that can be used to connect the tables in the given database, we need to look for columns that are present in multiple tables. One such column is "rfam_acc", which is present in both the "family" and "family_info" tables. This column can be used to join the two tables together.

Another column that can be used to connect the tables is "accession", which is present in both the "sequence" and "structure" tables. This column can be used to join the two tables together and link the sequence information with the corresponding structure information.

Question 4:

This question is to test your aptitude for writing small shell scripts on Unix. You are given this URL https://www.amfiindia.com/spages/NAVAll.txt (https://www.amfiindia.com/spages/NAVAll.txt). Write a shell script that extracts the Scheme Name and Asset Value fields only and saves them in a csv file.

Solution:

Here is a shell script that extracts the Scheme Name and Asset Value fields from the given URL and saves them in a csv file:

```
#!/bin/bash

# download the file and store it in a variable
NAV_FILE=$(curl -s https://www.amfiindia.com/spages/NAVAll.txt)

# extract the relevant fields and save them in a csv file
echo "$NAV_FILE" | awk -F ';' '{print $3 "," $6}' > amfi_nav.csv
```

The script first uses 'curl' to download the NAVAll.txt file from the given URL and store its contents in a variable named 'NAV_FILE'.

Then, the script uses 'awk' to extract the third and sixth fields (i.e., Scheme Name and Asset Value) from each line of 'NAV_FILE', separated by semicolons (';'), and print them in a comma-separated format.

Finally, the output is redirected to a file named 'amfi_nav.csv'.

.

In [ ]: