Text Technology Project

Project Id: TWEET-SQLITE-XML

Winter 2020-2021

IMS, Universität Stuttgart

Data

Tweets with hashtag #Cyberpunk2077 collected using the twitter API and `tweepy`

Retrieved features: - created_at - text - user location - user description - geo - coordinates - place - retweet count - favorite count - lang

Retrieval count by date:

Date	Retrieved Tweet Count
Jan 4th, 2021	39490
Jan 24th, 2021	8177

XML Schemas (XSD)

We have 5 different xml schemas that can be found in the /schema folder. They are:

1. TweetDataSchema.xsd, a schema for defining the Tweets.
2. HashTagSchema.xsd, a schema only for hashtags.
3. HashTagSchemaWithTweet.xsd, a schema for hashtags with the Tweets under the hasgtag.
4. LocationSchema.xsd, a schema only for user locations.
5. LocationSchemaWithTweet.xsd, a schema for user locations with the Tweets under the user location.

XQuery

The folder /schema/XqueryResults contains 10 different Xquery methods that we have developed in order to query the xml file "all_tweets_merge.xml" which can be found in the /generated folder. The xml file was generated from the "TweetDataSchema.xsd" in the folder /schema. The /schema/XqueryResults folder contains both the xquery (xq) and the respective xml result. We used an application called BaseX to run the xquery methods.

Running Locally

Directory Structure

- `/data` : contains the csv file
- `/db` : contains the sqlite database
- `/generated` : contains the generated xml
- `/view` : scripts for generating xml views
- `/schema` : contains the schemas to validate generated xml

Setting up python env

Create a virtual env using the requirements.py file. Or if you are using Anaconda, create a conda env and then install from this file.

```
# venv
python3 -m venv ./venv
# or python depending on your os
source venv/bin/activate
# check the appropriate command for windows
pip install -r requirements.txt

# for anaconda
conda create -n ttw python=3 -y
conda activate ttw
pip install -r requirements.txt
```

Running the code

The entrypoint here is `app.py`. It'll connect to the database and then you can call `DataFactory` methods to fetch data. Use the methods for generating `XML` and validate them. Check the `generated` directory for some generated and validated xml files.

If you want to use the `DataFactory` class, which connects to the database and provides helper methods for running queries, there are commented examples in `app.py`.

In case you want to connect to the Twitter API and fetch new data you can do so by running `data/tweetdata.py`. Make sure that you've your API keys and credentials ready to run the script.

```
# credentials
consumer_key = ''
consumer_secret = ''
access_token = ''
access_token_secret = ''
```

Please make sure to use the generated directory for storing generated xml files.