Parse post historic

December 12, 2022

1 Post historic

1.0.1 Description of the code:

This code parses the post-historic of all posts by the users extracted from the file. From: https://ia800107.us.archive.org/27/items/stackexchange/readme.txt

posthistory.xml

```
- Id
- PostHistoryTypeId
    - 1: Initial Title - The first title a question is asked with.
    - 2: Initial Body - The first raw body text a post is submitted with.
    - 3: Initial Tags - The first tags a question is asked with.
    - 4: Edit Title - A question's title has been changed.
    - 5: Edit Body - A post's body has been changed, the raw text is
         stored here as markdown.
    - 6: Edit Tags - A question's tags have been changed.
    - 7: Rollback Title - A question's title has reverted to a previous version.
    - 8: Rollback Body - A post's body has reverted to a previous version
         the raw text is stored here.
    - 9: Rollback Tags - A question's tags have reverted to a previous version.
    - 10: Post Closed - A post was voted to be closed.
    - 11: Post Reopened - A post was voted to be reopened.
    - 12: Post Deleted - A post was voted to be removed.
    - 13: Post Undeleted - A post was voted to be restored.
    - 14: Post Locked - A post was locked by a moderator.
    - 15: Post Unlocked - A post was unlocked by a moderator.
    - 16: Community Owned - A post has become community owned.
    - 17: Post Migrated - A post was migrated.
    - 18: Question Merged - A question has had another,
          deleted question merged into itself.
    - 19: Question Protected - A question was protected by a moderator
    - 20: Question Unprotected - A question was unprotected by a moderator
    - 21: Post Disassociated - An admin removes the OwnerUserId from a post.
    - 22: Question Unmerged - A previously merged question has had its answers
          and votes restored.
- PostId
- RevisionGUID: At times more than one type of history record can be
```

```
   recorded by a single action.  All of these will be grouped using the same RevisionGUID
- CreationDate: "2009-03-05T22:28:34.823"
- UserId
- UserDisplayName: populated if a user has been removed and no longer
   referenced by user Id
- Comment: This field will contain the comment made by the user who edited a post
- Text: A raw version of the new value for a given revision
    - If PostHistoryTypeId = 10, 11, 12, 13, 14, or 15  this column
       will contain a JSON encoded string with all users who have
       voted for the PostHistoryTypeId
    - If PostHistoryTypeId = 17 this column will contain migration details
       of either "from <url>" or "to <url>"
- CloseReasonId
    - 1: Exact Duplicate - This question covers exactly the same
        ground as earlier questions on this topic; its answers may
        be merged with another identical question.
    - 2: off-topic
    - 3: subjective
    - 4: not a real question
    - 7: too localized
```

[1]:
```
import random
import pandas as pd
import matplotlib.pyplot as plt
import math
import time # to see how long it takes to run different parts of code
import linecache
import re

post_hist_file = 'D:/Data/stackoverflow.com-PostHistory/PostHistory.xml'
```

[4]:
```
start = time.time()

#Only necessary to do once:
with open(post_hist_file, 'r', encoding='UTF-8') as f:
    num_lines = sum(1 for line in f)
    print('Total lines:', num_lines)


end = time.time()
print((end - start)/60)
```

```
Total lines: 147880458
24.585229253768922
```

[2]:
```
num_lines = 147880458
```

[3]:
```
# Make percentages, to see how fast it goes

promille = num_lines/1000
list_of_numbers=[]

for i in range(1,1001):

    list_of_numbers.append(round(i*promille))
```

[8]:
```
# Read in both file with account id and with post id

post_id = pd.read_csv("D:/Data/posts_binary_num.csv")


pid = list(post_id["x"])
```

[9]:
```
pidstr = []
# Make into string instead of list!
for i in range(len(pid)):
    pidstr.append(str(pid[i]))
```

[10]:
```
# Make a list with all usernames in, in the same format as in the big line


# Make the same for postid
list_of_search_strings_pid = []

for i in range(len(pid)):
        list_of_search_strings_pid.append(str('PostId="' + pidstr[i] + '"'))
```

[17]:
```
start = time.time()

print(start)
print("old was 2619min")

# Make a regex-string, which contains all of the user-numbers randomly drawn
temp_pid = '(?:% s)' % '|'.join(list_of_search_strings_pid)

# Make a list which can contain all of the posts
list_of_posts_pid = []

# Make i and b == 0, which will be used to count during the loop
i = 0
b = 0

# Open the file and read each line. Due to prior loss of data, the file is saved␣
 ↪at every quintile
```

```
with open(post_hist_file, 'r', encoding = 'UTF-8') as f:
    for line in range(num_lines):
        i = i+1
        z = f.readline()
        if  re.search(temp_pid, z):
                list_of_posts_pid.append(z)
        if i == round(promille*10):
            pd.DataFrame(list_of_posts_pid).to_csv("D:/Data/pid_1.csv")
        if i == round(promille*200):
            pd.DataFrame(list_of_posts_pid).to_csv("D:/Data/pid_2.csv")
        if i == round(promille*400):
            pd.DataFrame(list_of_posts_pid).to_csv("D:/Data/pid_3.csv")
        if i == round(promille*600):
            pd.DataFrame(list_of_posts_pid).to_csv("D:/Data/pid_4.csv")
        if i == round(promille*800):
            pd.DataFrame(list_of_posts_pid).to_csv("D:/Data/pid_5.csv")
        if i in list_of_numbers:
            b = b+1
            print(b)


pd.DataFrame(list_of_posts_pid).to_csv("D:/Data/pid_new.csv")

end = time.time()
print((end - start)/60)
```

```
1668065736.0676079
old was 2619min
1 - 1000
8727.911344512304 min (total)
```

[18]:
```
# Read in the saved data
pid = pd.read_csv("D:/Data/pid_new.csv")
```

[19]:
```
# Make the posts into a list

pid = pid['0'].tolist()
```

[20]:
```
# Parse PID

# A list for every variable in the file
Id  = []
PostHistoryTypeId  = []
PostId  = []
RevisionGUID  = []
CreationDate  = []
UserId  = []
UserDisplayName  = []
```

```
Comment  = []
Text = []
CloseReasonId = []

# Same as the list - but just as column-names
columnnames = ["Id", "PostHistoryTypeId", "PostId", "RevisionGUID",␣
 ↪"CreationDate", "UserId" , "UserDisplayName", 'Comment', "Text",␣
 ↪"CloseReasonId"]
# And a list of the lists (of columns)
listoflistcolumns = [Id, PostHistoryTypeId, PostId, RevisionGUID, CreationDate,␣
 ↪UserId , UserDisplayName, Comment, Text, CloseReasonId]


# Then a for-loop, to make the parsing a bit smaller (space-wise)
for i in range(len(pid)):
    for listname in range(len(listoflistcolumns)):
        if " "+columnnames[listname]+'="'in pid[i]:
            z = pid[i].split(" "+columnnames[listname]+'="')[1].split('"')[0]
            listoflistcolumns[listname].append(z)
        else:
            listoflistcolumns[listname].append("missing")


# Make a dataframe out of the list of lists
df_pid = pd.DataFrame(Id)

# Name the columns
for listname in range(len(listoflistcolumns[1:])):
    df_pid[columnnames[listname+1]] = listoflistcolumns[listname+1]
```

[21]:
```
# Save dataframes:
df_pid.to_csv("D:/Data/df_pid.csv")
```