1.3.1_duplicate_removal

March 11, 2021

1 1.3.x Remove duplicate text chunks for each admission

2 THIS MUST BE RUN WITH PYTHON 3.7.x or the regex won't work

- custom tokenizer to split on `.` + `space` or `\n`
- saves the concatenation of unique tokens in db `transfused_notes_unique` `ctrl_notes_unique` for analysis
- last ran 8/23/2019 on AWS large instance Windows OS

[1]:
```
# -*- coding: utf-8 -*-
from datetime import datetime
import time
import re
import pandas as pd


import sys
import psycopg2
from sqlalchemy import create_engine, update, event
from tqdm import tnrange, tqdm_notebook

from time import sleep
from importlib_metadata import version

if sys.platform == "darwin":
    conn = psycopg2.connect("dbname=mimic user=xxxxxxxxxxx␣
 ↪options=--search_path=mimiciii");
    engine = create_engine('postgresql://xxxxxxxxxxx@localhost/mimic')
    path1 = './replication/'


cur = conn.cursor();
cur.execute("""SET search_path = mimiciii;""")

libraries = ['pandas','psycopg2','tqdm','scipy']
```

```
print('last ran: ',datetime.now() )
print("Python Version:", sys.version[0:7])
print( "operating system:", sys.platform)

for lib in libraries:
    print(lib + ' version: ' + version(lib))
```

```
last ran:  2019-12-25 10:37:27.773955
Python Version: 3.7.3 (
operating system: darwin
pandas version: 0.24.2
psycopg2 version: 2.7.6.1
tqdm version: 4.32.1
scipy version: 1.2.1
```

3 Transfused

- 21443

create a new table `transfused_notes_unique` in the Postgres database

[2]:
```
cur.execute("""DROP TABLE IF EXISTS mimiciii.transfused_notes_unique;

CREATE TABLE mimiciii.transfused_notes_unique
(hadm_id int,
 text varchar);""")

conn.commit();
```

get list of **hadm_id**s from the transfused admissions

[3]:
```
xf = pd.read_sql("""
SELECT hadm_id
FROM mimiciii.transfused_notes_sink """, engine)

xf_ids = xf.hadm_id.unique()
len(xf)
```

[3]:
```
21443
```

this helps us execute multiple calls to the database using pandas

[4]:
```
@event.listens_for(engine, 'before_cursor_execute')
def receive_before_cursor_execute(conn, cursor, statement, params, context,␣
 ↪executemany):
    #print("FUNC call")
    if executemany:
        cursor.fast_executemany = True
```

```
[5]: def mark_duplicates(input_tokens):
         '''
         Function uses a set() and list to generates each token with html tags added␣
      ↪to duplicate tokens.
         INPUT: input_tokens = string of tokenized text (can be sentences,␣
      ↪paragraphs, words etc)

         OUTPUT: a single token at a time (generator) until the end of the␣
      ↪input_tokens.
         '''

         # create hash of tokens
         tokens_set = set()
         tokens_set_add = tokens_set.add

         for token in input_tokens:

             #skip any empty tokens
             if token == '':
                 pass

             elif token not in tokens_set:
                 tokens_set_add(token)
                 yield token

             else:
                 pass


     def tokenize2(t):


         tok_new = []

         # find any \n followed by an uppercase letter, a number, or a dash
         sent_token =re.split(r"(?=\n\s*[A-Z1-9#-]+.*)", t)

         # replace \n with a space with a space
         sent_token = [re.sub(r"$\n+","",i) for i in sent_token] # remove from end
         sent_token = [re.sub(r"^\n", "", i) for i in sent_token] #remove from front
             # line feeds + whitespace or not
         sent_token = [re.sub(r"\s+\n\s+", " ", i) for i in sent_token]
         sent_token = [re.sub(r"\s+\n", " ", i) for i in sent_token]
         sent_token = [re.sub(r"\n\s+", " ", i) for i in sent_token]
         sent_token = [re.sub(r"\n", " ", i) for i in sent_token]

         #remove front/end whitespace
```

```
    sent_token = [i.strip(' ') for i in sent_token]


    for i in sent_token:
        if i != '':
            tok_new.append(i)

    return tok_new

def save_tokens(token):
    for enum_num, enum_token in enumerate(token):
        yield str(enum_num), enum_token

def bloatectomy(input_text):
    '''Function to tokenize,remove duplicates, and output a string.
   Tokenization is done at each period followed by a space, or a newline.

    INPUT: input_text = text to be tokenized
    OUTPUT: an string with duplicate tokens removed.
    '''
    # tokenize 1

    tok = re.split(r"(.+?\.[\s\n]+)", input_text, flags=re.DOTALL)
    # whitespace around tokens can cause a duplicate to be missed
    tok = [i.strip(' ') for i in tok]

    #tokenize 2
    new_tok = []

    for num, t in enumerate(tok):

        n_tok = tokenize2(t)

        new_tok.extend(n_tok)

    # save numbered list
    numbered_output = list(save_tokens(new_tok))

    # detect and mark/remove duplicates
    u_set = list(mark_duplicates(new_tok))

    uniq =str("\n ".join(u_set))

    return uniq
```

3.0.1 check to make sure it's working

[6]:
```
hadm_id_sample = #an hadm_id from the database
pt_all_xf = pd.read_sql("""SELECT * FROM mimiciii.transfused_notes_sink WHERE␣
 ↪hadm_id IN (hadm_id_sample);""", engine)
print(bloatectomy(pt_all_xf))
```

[6]:
```
fake_text = '''Assessment and Plan
61 yo male Hep C cirrhosis and HCC presents with probable lower GIB and
renal failure of unclear duration.
Abd pain:
-other labs: PT / PTT / INR:16.6//                         1.5, CK / CKMB /

Troponin-T:4390/40/0.21, ALT / AST:258/575, Alk Phos / T Bili:232/5.9,
ICU Care
-other labs: PT / PTT / INR:16.6//                         1.5, CK / CKMB /
   Communication:                                              Comments:
icu Care
Assessment and Plan
Chief Complaint:
61 yo male Hep C cirrhosis and HCC presents with probable lower GIB and
renal failure of unclear duration.
# Abd pain:'''
```

[7]:
```
print(bloatectomy(fake_text))
```

```
Assessment and Plan
 61 yo male Hep C cirrhosis and HCC presents with probable lower GIB and renal
failure of unclear duration.
 Abd pain:
 -other labs: PT / PTT / INR:16.6//                         1.5, CK / CKMB /
 Troponin-T:4390/40/0.21, ALT / AST:258/575, Alk Phos / T Bili:232/5.9,
 ICU Care
 Communication:                                              Comments: icu Care
 Chief Complaint:
 # Abd pain:
```

3.1 split each admission's notes into sentences, strip whitespace from edges, remove duplicates, join, store in new table

should take approx 19 minutes

[8]:
```
s = time.time()

for j in tqdm_notebook(xf_ids):

    sql = """
    SELECT hadm_id, text
    FROM mimiciii.transfused_notes_sink
        WHERE hadm_id in ({0})
    """

    # run sql query above to pull all notes for one admission (already in order␣
 ↪by date)
    sql = sql.format(j)
    xnotes = pd.read_sql(sql, engine)

    # split into tokens based on a period followed by a space or a newline \n␣
 ↪(the period and \n are included in the token)
    input_notes = xnotes.text.tolist()[0]

    unique_tokens = bloatectomy(input_notes)

    # save as a new dataframe
    xtext2 = [(j, unique_tokens)]
    xfulltext=pd.DataFrame(xtext2, columns=['hadm_id', 'text'])

    # append user and single note to the new table in database
    table_name = 'transfused_notes_unique'


    z = time.time()

    xfulltext.to_sql(table_name, con=engine, if_exists='append', chunksize=1,␣
 ↪index=False, schema='mimiciii')

    #print('per hadmid-',(time.time() - z)/60,'min')

print('total-', (time.time() - s)/60,'minutes')


conn.commit()
```

```
HBox(children=(IntProgress(value=0, max=21443), HTML(value='')))
```

```
total- 27.955767035484314 minutes
```

[9]:

```
cur.execute(""" SELECT COUNT(*), COUNT(DISTINCT hadm_id) FROM␣
 ↪transfused_notes_unique;""")

print( pd.DataFrame(cur.fetchall(), columns=[ 'total notes count', 'xf␣
 ↪admissions with notes']).to_string(index=False))
```

```
total notes count  xf admissions with notes
            21443                     21443
```

4 Control

- unique admissions = 27,888

```
[10]: cur.execute("""DROP TABLE IF EXISTS mimiciii.ctrl_notes_unique;

      CREATE TABLE mimiciii.ctrl_notes_unique
      (hadm_id int,
       text varchar);""")

      conn.commit();
```

```
[11]: xf = pd.read_sql("""
      SELECT hadm_id
      FROM mimiciii.ctrl_notes_sink """, engine)

      # get list of ids
      cxf_ids = xf.hadm_id.unique()
```

this will take about 15 minutes to run

```
[12]: s = time.time()

      for j in tqdm_notebook(cxf_ids):


          sql = """
          SELECT hadm_id, text
          FROM mimiciii.ctrl_notes_sink
              WHERE hadm_id in ({0})
          """

          # run sql query above to pull all notes for one admission (already in order␣
       ↪by date)
          sql = sql.format(j)
          cnotes = pd.read_sql(sql, engine)

           # split into tokens based on a period followed by a space or a newline \n␣
       ↪(the period and \n are included in the token)
          input_cnotes = cnotes.text.tolist()[0]

          unique_tokens = bloatectomy(input_cnotes)
```

```
    # save as a new dataframe
    xtext2 = [(j, unique_tokens)]
    xfulltext=pd.DataFrame(xtext2, columns=['hadm_id', 'text'])

    # append user and single note to the new table in database
    table_name = 'ctrl_notes_unique'


    z = time.time()

    xfulltext.to_sql(table_name, con=engine, if_exists='append', chunksize=1,␣
 ↪index=False, schema='mimiciii')

   # print('per hadmid-',time.time() - z)

print('total-', (time.time() - s)/60,'minutes')


conn.commit()
```

```
HBox(children=(IntProgress(value=0, max=27888), HTML(value='')))


total- 38.29251765012741 minutes
```

[13]:
```
cur.execute(""" SELECT COUNT(*), COUNT(DISTINCT hadm_id) FROM ctrl_notes_unique;
 ↪""")

print( pd.DataFrame(cur.fetchall(), columns=[ 'total notes count', 'ctrl␣
 ↪admissions with notes']).to_string(index=False))
```

```
 total notes count  ctrl admissions with notes
             27888                       27888
```

[14]:
```
conn.commit()
cur.close()
conn.close()
```