1.2_concat_notes

May 26, 2021

1 1.2_concat_notes

- python 2.7.x
- from the mimic iii PostgreSQL database
- all the notes for each admission (hadm_id) get ordered by time and concatenated into one note per admission.
- create tables for transfused `transfused_notes_sink` and control `ctrl_notes_sink`

1.1 import libraries, connect to mimic database

[15]:
```
conn.commit();
cur.close();
conn.close();
```

[1]:
```
import sys

import time
from datetime import datetime
import datetime

import pandas as pd
import random

from tqdm import tnrange, tqdm_notebook
from time import sleep

from importlib_metadata import version

# things to connect to the posgres database
import psycopg2
from sqlalchemy import create_engine, update, event

POSTGRES_CONNECT = os.environ.get("POSTGRES_CONNECT")
POSTGRES_ENGINE = os.environ.get("POSTGRES_ENGINE")
conn = psycopg2.connect(POSTGRES_CONNECT)
cur = conn.cursor();
cur.execute("""SET search_path = mimiciii;""")
engine = create_engine(POSTGRES_ENGINE)
```

```
libraries = ['pandas','sqlalchemy','psycopg2','tqdm']
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:  2019-12-24 23:53:36.145380
Python Version: 3.7.3 (
operating system: darwin
pandas version: 0.24.2
sqlalchemy version: 1.3.3
psycopg2 version: 2.7.6.1
tqdm version: 4.32.1
```

1.2 1.2.1. Load Transfused Notes `xf_notes`

- Get all notes for admissions (**hadm_id**) that have been identified as transfused group `transfused_hadm_id`
- Print the total number of notes, and unique admissions **21,443**
- keep all types of timestamps **chartdate** is only a date but is present in every note

  ```
  **charttime** and **storetime** are timestamps, but are not present in every note (disch
  ```
- do not use any notes where the provider has indicated that the note is an error (**iserror**=1)
- note that there are 98 less admissions than in `transfused_hadm_id`, meaning that 98 admissions did not have any data in the `noteevents` table

[19]:
```
cur.execute("""
DROP TABLE IF EXISTS mimiciii.transfused_notes;

SELECT  B.*
INTO mimiciii.transfused_notes
    FROM mimiciii.noteevents B
    WHERE B.hadm_id IN (
            SELECT  x.hadm_id
            FROM mimiciii.transfused_hadm_id x)

    AND B.iserror IS NULL
;""")
```

Print counts for number of notes total, and number of admissions

[20]:
```
cur.execute("""SELECT COUNT(*), COUNT(DISTINCT hadm_id) FROM  mimiciii.
 ↪transfused_notes;""")
```

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

```
 total notes count  admissions
            874711       21443
```

1.3 1.2.2 One Document per Admission

For each admission, concatenate all the notes for that admission into one note (thus, each admission has one **document**). Create a table of these admission notes using the hospital admission id (hadm_id) as the identifier rather than the note id (row_id)

1.3.1 Transfused Notes by Admission `transfused_notes_sink` (without metadata, best for analysis) or or `transfused_notes_sink_metadata` (with metadata at the top of the concatenated note, best for consumption by a subject matter expert or displaying the notes)

- group by admission ID
- order by note date (**note_dt**)
- concatenate all notes for that admission ID into one string
- metadata==True: concatenate all notes and other data (date(s), provider=cgid, note, type=category,description) for that admission ID into one string
- save as transfused_notes_sink or transfused_notes_sink_metadata

1.3.2 The new table contains the following columns for each unique admission:

- hadm_id
- text (concatenate notes and/or other data)

[3]:
```
# set whether you want to include metadata at the top of each note (we don't␣
 ↪use this for the NLP, but is' useful for the viewing by SMEs)
metadata = False

if metadata==False:

    cur.execute("""DROP TABLE IF EXISTS mimiciii.transfused_notes_sink;

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

else:
    cur.execute("""DROP TABLE IF EXISTS mimiciii.transfused_notes_sink_metadata;

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

```
conn.commit();
```

1.3.3 create list of unique hadm_ids

[4]:
```
xf = pd.read_sql("""
SELECT hadm_id
FROM mimiciii.transfused_notes """, engine)

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

[4]:
```
21443
```

1.3.4 function that lets us make multiple requests to the postgres using pandas read_sql

[5]:
```
@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
```

1.3.5 function to pull notes, concatenate and save

- this will take a few hours(2.7) to run
- iterate through for each unique admission (hadm_id)
- pull all notes for an admission
- order notes by charttime , then storetime
- concatenate
- save as one big note to new table

[6]:
```
s = time.time()

for j in tqdm_notebook(xf_ids):

    if metadata == False:

        table_name = 'transfused_notes_sink'

        sql = """

        SELECT  hadm_id, chartdate, charttime, storetime, text
        FROM mimiciii.transfused_notes
```

```
            WHERE hadm_id in ({0})
        GROUP BY hadm_id, chartdate, charttime, storetime, text
        ORDER BY chartdate, charttime, storetime"""

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

        xnotes = xnotes.loc[:,'text']

    else:

        table_name = 'transfused_notes_sink_metadata'

        sql = """

        SELECT subject_id, hadm_id, chartdate, charttime, storetime, category,␣
↪cgid, description, text
        FROM mimiciii.transfused_notes
            WHERE hadm_id in ({0})
        GROUP BY subject_id, hadm_id, chartdate, charttime, storetime,␣
↪category, cgid, description, text
        ORDER BY chartdate, charttime, storetime"""

        # run sql query above to pull all notes for one admission (in order by␣
↪date)
        # concatenate notes and all other cols (metadata)
        # all the metadata gets put into one token for duplicate removal␣
↪purposes

        sql = sql.format(j)
        xnotes=pd.read_sql(sql, engine)

        xnotes.loc[:,'text2'] = xnotes.loc[:,'text']
        xnotes.iloc[:,-2] = '. '

    # put a a period + whitespace to designate the end start and end of a note ␣
↪
    xnotes['separator'] = '. '

    xtext = xnotes.to_csv(None, header=False, index=False)

    # save as a new dataframe
    xtext2 = [(j, xtext)]
    xfulltext=pd.DataFrame(xtext2, columns=['hadm_id', 'text'])
    # append hadm_id and the new single note to the new table in database
    xfulltext.to_sql(table_name, con=engine, if_exists='append', chunksize=1,␣
 ↪index=False, schema='mimiciii')

print(time.time() - s)

conn.commit()
```

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


7052.120042562485
```

Print counts for number of notes and number of admissions

```
[ ]: if metadata==False:
         cur.execute(""" SELECT COUNT(DISTINCT hadm_id) FROM transfused_notes_sink;
      ↪""")
     else:
         cur.execute(""" SELECT COUNT(DISTINCT hadm_id) FROM␣
      ↪transfused_notes_sink_metadata;""")

     print( pd.DataFrame(cur.fetchall()).to_string(index=False))
```

```
[ ]: if metadata==False:
         cur.execute(""" SELECT COUNT(*) FROM transfused_notes_sink;""")
     else:
         cur.execute(""" SELECT COUNT(*) FROM transfused_notes_sink_metadata;""")

     print( pd.DataFrame(cur.fetchall()).to_string(index=False))
```

1.4 1.2.3 Non-Transfused `ctrl_notes`

1.4.1 Get all notes for admissions (hadm_id) that have been identified as control group `ctrl_ids`

- Print the total number of notes, and unique admissions
- note that there are **27,888** admissions w/ notes (240 control group admissions did not have data in the `noteevents` table.

```
[7]: cur.execute(""" DROP TABLE IF EXISTS mimiciii.ctrl_notes;
     SELECT n.*
         INTO ctrl_notes
     FROM noteevents n
         WHERE n.hadm_ID IN (
             SELECT DISTINCT c.hadm_id
             FROM ctrl_ids c)
```

```
    AND n.iserror IS NULL

    ;""")

conn.commit()
```

Print count of total notes and admissions

```
[8]: cur.execute(""" SELECT COUNT(*), COUNT(DISTINCT hadm_id) FROM ctrl_notes;""")
print( pd.DataFrame(cur.fetchall(), columns=[ 'total notes count', 'ctrl␣
 ↪admissions with notes']).to_string(index=False))
```

```
 total notes count  ctrl admissions with notes
            535639                       27888
```

1.4.2 Control Notes by Admission `ctrl_notes_sink` or `ctrl_notes_sink_metadata`

- group by admission ID
- order by note date ('note_dt')
- concatenate all notes for that admission ID into one string
- save as ctrl_notes_sink

New table contains the following columns

- hadm_id
- text (concatenated notes + metadata (if metadata==True)

```
[9]: if metadata==False:
    cur.execute("""DROP TABLE IF EXISTS ctrl_notes_sink;

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

else:
    cur.execute("""DROP TABLE IF EXISTS ctrl_notes_sink_metadata;

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

conn.commit();
```

1.4.3 Load the unique hadm_ids (identifies each admission) and make a list

[10]:
```
ctrl_ids = pd.read_sql("""
SELECT hadm_id
FROM mimiciii.ctrl_notes""", engine)

cids= ctrl_ids.hadm_id.unique()
```

1.4.4 Function to load, concatenate and save

- this takes a few hours (2.3 hrs) to run
- iterates through each hadm_id
- pulls all notes (and other data if chosen)
- orders notes in order of charttime, then storetime
- concatenate and save in new table

[11]:
```
s = time.time()

for i in tqdm_notebook(cids):

    if metadata==False:

        table_name = 'ctrl_notes_sink'

        sql = """

        SELECT  hadm_id, chartdate, charttime, storetime,text
        FROM mimiciii.ctrl_notes
            WHERE hadm_id IN ({0})
        GROUP BY  hadm_id, chartdate, charttime, storetime, text
        ORDER BY chartdate, charttime, storetime"""

        sql = sql.format(i)
        cnotes = pd.read_sql(sql, engine)
        cnotes = cnotes.loc[:,'text']

    else:

        table_name = 'ctrl_notes_sink_metadata'

        sql = """

        SELECT subject_id, hadm_id, chartdate, charttime, storetime, category,␣
↪cgid, description, text
        FROM mimiciii.ctrl_notes
            WHERE hadm_id IN ({0})
        GROUP BY subject_id, hadm_id, chartdate, charttime, storetime,␣
↪category, cgid, description, text
```

```
        ORDER BY chartdate, charttime, storetime"""

        sql = sql.format(i)
        cnotes = pd.read_sql(sql, engine)

        cnotes.loc[:,'text2'] = cnotes.loc[:,'text']
        cnotes.iloc[:,-2] = '. '


    cnotes['separator'] = '. '

    #CONCAT NOTES
    ctext = cnotes.to_csv(None, header=False, index=False)

    #put into a data frame with hadm_id
    ctext2 = [(i, ctext)]
    cfulltext = pd.DataFrame(ctext2, columns=['hadm_id', 'text'])


    # append admission and single note to the new table in database
    cfulltext.to_sql(table_name, con=engine, if_exists='append', chunksize=1,␣
 ↪index=False, schema='mimiciii')

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

conn.commit()
```

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


total time= 87.5706007361412 min
```

Print counts of the new table

[17]:
```
if metadata==False:
    cur.execute(""" SELECT COUNT(*), COUNT(DISTINCT hadm_id) FROM␣
 ↪ctrl_notes_sink;""")
else:
    cur.execute(""" SELECT COUNT(*), COUNT(DISTINCT hadm_id) FROM␣
 ↪ctrl_notes_sink_metadata;""")

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
```

1.5 1.2.4 Clean Up, Commit, and Close

[12]:
```
conn.commit()
cur.close()
conn.close()
```