import numpy as np

from sklearn.preprocessing import StandardScaler

import csv

import re

import os

import matchzoo as mz

print(mz.\_\_version\_\_)

task = mz.tasks.Ranking()

print(task)

import pandas as pd

bdf=pd.read\_csv('example.csv') #This will be predefined in the user input any csv

def inputs(bdf,i):

s=(bdf.iloc[i]['ttxt'])

fs= open("temp.txt","w+")

fs.write(s)

fs=open('temp.txt',"r+")

rd=fs.read()

cleaned=clean(rd)

return cleaned

#text\_file = open("sample.txt", "wt")

#n = text\_file.write(s)

def clean(btxt):

import re, string

#txt= open("uH3h7NAk.txt", "r+")

#txt= open(fname, "r+")

#txt=txt.read()#Reading

txt=btxt

print(txt)

txt=str(txt.encode(encoding = 'UTF-8',errors = 'strict')) #UTF-8 Encoding

clean = re.sub(r"[0-9,.;@#?/&%"",(),[],!&$]+\ \*", " ", txt.replace('b','')) #Removes unncessary stuff for word vectors

print(clean)

s=clean

exclude = set(string.punctuation)

table = str.maketrans("","")

regex = re.compile('[%s]' % re.escape(string.punctuation)) #Punctuations

text= regex.sub('', s)

import nltk

#nltk.download('stopwords')

from nltk.corpus import stopwords #Stop Words influencing the game

cachedStopWords = stopwords.words("english")

text = ' '.join([word for word in text.split() if word not in cachedStopWords]).lower()

return text

#Sends the dataframe here and parses for each documnet contained in iloc[0,1,2....n]

for i in range(len(bdf)):

res=inputs(bdf,i)

print(res)

train\_raw = mz.datasets.toy.load\_data(stage='train', task=task)

test\_raw = mz.datasets.toy.load\_data(stage='test', task=task)

type(train\_raw)

train\_raw.left.head()

train\_raw.right.head()

train\_raw.relation.head()

train\_raw.frame().head()

preprocessor = mz.preprocessors.BasicPreprocessor()

preprocessor.fit(train\_raw)

preprocessor.context

train\_processed = preprocessor.transform(train\_raw)

test\_processed = preprocessor.transform(test\_raw)

train\_processed.left.head()

vocab\_unit = preprocessor.context['vocab\_unit']

print('Orig Text:', train\_processed.left.loc['Q1']['text\_left'])

sequence = train\_processed.left.loc['Q1']['text\_left']

print('Transformed Indices:', sequence)

print('Transformed Indices Meaning:',

'\_'.join([vocab\_unit.state['index\_term'][i] for i in sequence]))

mz.models.list\_available()

ranking\_task = mz.tasks.Ranking(loss=mz.losses.RankCrossEntropyLoss(num\_neg=4))

ranking\_task.metrics = [

mz.metrics.NormalizedDiscountedCumulativeGain(k=3),

mz.metrics.NormalizedDiscountedCumulativeGain(k=5),

mz.metrics.MeanAveragePrecision()

]

model = mz.models.DSSM()

model.params['input\_shapes'] = preprocessor.context['input\_shapes']

model.params['task'] = ranking\_task

model.params['mlp\_num\_layers'] = 3

model.params['mlp\_num\_units'] = 300

model.params['mlp\_num\_fan\_out'] = 128

model.params['mlp\_activation\_func'] = 'relu'

model.guess\_and\_fill\_missing\_params()

model.build()

model.compile()

x, y = train\_processed.unpack()

test\_x, test\_y = test\_processed.unpack()

model.fit(x, y, batch\_size=32, epochs=800)

data\_generator = mz.DataGenerator(train\_processed, batch\_size=32)

model.fit\_generator(data\_generator, epochs=5, use\_multiprocessing=True, workers=4)

model.evaluate(test\_x, test\_y)

model.predict(test\_x)

print(np.array(test\_x))

scaler = StandardScaler()

print(scaler.fit(test\_x))

model.save('my-model')