Mercari-Sub-I

May 27, 2020

[0]:
```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import os
import time
from py7zr import unpack_7zarchive
# from pyunpack import Archive
import shutil
import datetime
import math
from contextlib import contextmanager
import scipy
from scipy.sparse import hstack
from sklearn.preprocessing import StandardScaler
from nltk.corpus import stopwords
from tqdm import tqdm
import re
import gc
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import KFold

import tensorflow as tf
from tensorflow.keras.layers import Dense,Input
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
```

[0]:
```
os.chdir("/content/drive/My Drive/Kaggle Case Study I")
```

[0]:
```
def extract_zip(input_path,output_path):
    if not os.path.exists(output_path):
        os.makedirs(output_path)
```

```
    try:
        shutil.register_unpack_format('7zip', ['.7z'], unpack_7zarchive)
        shutil.unpack_archive(input_path, output_path)
    except Exception as e:
        shutil.unpack_archive(input_path, output_path)
```

[0]:
```
def load_data():
    train_data = pd.read_csv('train/train.tsv',sep = '\t')
    test_data = pd.read_csv('test_stg2/test_stg2.tsv',sep = '\t')
    return train_data,test_data
```

[0]:
```
def preprocess(df):
    df['name'] = df['name'].fillna('') + ' ' + df['brand_name'].fillna('')
    df['text'] = (df['item_description'].fillna('') + ' ' + df['name'] + ' ' +␣
↪df['category_name'].fillna(''))
    return df[['name', 'text', 'shipping', 'item_condition_id']]
```

[0]:
```
def main():

    start = time.time()

    # Reading the Input Data
    # extract_zip("train.tsv.7z","train/")
    # extract_zip("test.tsv.7z","test/")

    train_data,test_data = load_data()

    train_data = train_data[(train_data.price >= 3) & (train_data.price <=␣
↪2000)].reset_index(drop=True)
    print("(1) done")
  ␣
↪#############################################################################################

    # Vectorizing the Data
    scaler = StandardScaler()
    y_train = scaler.fit_transform(np.log1p(train_data['price'].values.
↪reshape(-1, 1)))

    train_data = preprocess(train_data)
    test_data = preprocess(test_data)

    Vectorizer = TfidfVectorizer(max_features=100000,token_pattern='\w+',␣
↪dtype=np.float32)
    Vectorizer.fit(train_data['name'].values)

    X_train_name = Vectorizer.transform(train_data['name'].values)
```

```
    X_test_name = Vectorizer.transform(test_data['name'].values)

    Vectorizer = TfidfVectorizer(max_features=100000,ngram_range =␣
↪(1,2),token_pattern='\w+', dtype=np.float32)
    Vectorizer.fit(train_data['text'].values)

    X_train_text = Vectorizer.transform(train_data['text'].values)
    X_test_text = Vectorizer.transform(test_data['text'].values)

    Vectorizer = OneHotEncoder(dtype=np.float32)
    X_train_ship = Vectorizer.fit_transform(train_data['shipping'].values.
↪reshape(-1,1))
    X_test_ship = Vectorizer.transform(test_data['shipping'].values.
↪reshape(-1,1))

    Vectorizer = OneHotEncoder(dtype=np.float32)
    X_train_item = Vectorizer.fit_transform(train_data['item_condition_id'].
↪values.reshape(-1,1))
    X_test_item = Vectorizer.transform(test_data['item_condition_id'].values.
↪reshape(-1,1))

    X_train_tfidf =␣
↪hstack((X_train_name,X_train_text,X_train_ship,X_train_item)).tocsr()
    X_test_tfidf = hstack((X_test_name,X_test_text,X_test_ship,X_test_item)).
↪tocsr()

    del␣
↪X_train_name,X_test_name,X_train_text,X_test_text,X_train_ship,X_test_ship,X_train_item,X_t
    del train_data
    gc.collect()

    X_train_binary, X_test_binary = [x.astype(np.bool).astype(np.float32) for x␣
↪in [X_train_tfidf, X_test_tfidf]]

    print("X_train TFIDF Shape : ",X_train_tfidf.shape)
    print("X_train Binarized Shape : ",X_train_binary.shape)
    print("X_test TFIDF Shape : ",X_test_tfidf.shape)
    print("X_test Binarized Shape : ",X_test_binary.shape)

    print("(2) done")
  ␣
↪#############################################################################################

    # Saving the Pre-processed file into a pickle file along with the test id's

    file = open("X_train_tfidf","wb")
```

```
    pickle.dump(X_train_tfidf,file)
    file.close()

    file = open("X_test_tfidf","wb")
    pickle.dump(X_test_tfidf,file)
    file.close()

    file = open("X_train_binary","wb")
    pickle.dump(X_train_binary,file)
    file.close()

    file = open("X_test_binary","wb")
    pickle.dump(X_test_binary,file)
    file.close()


␣
↪#############################################################################################
    end = time.time()
    print("Time Taken in Seconds : ", end - start)
```

[9]:
```
if __name__ == "__main__":
    main()
```

```
(1) done
X_train TFIDF Shape :  (1481658, 200007)
X_train Binarized Shape :  (1481658, 200007)
X_test TFIDF Shape :  (3460725, 200007)
X_test Binarized Shape :  (3460725, 200007)
(2) done
Time Taken in Seconds :  576.8102276325226
```

[0]:
```
train_data,test_data = load_data()
test_data.test_id.to_csv("test_id.csv",index=False)
```