«گزارش کار تمرین شماره ۳»

عنوان تمرین	گزارش تکلیف شماره ۳
عنوان درس	مباحث ویژه در تجارت الکترونیک (پردازش زبان طبیعی – NLP)
نام و نام خانوادگی	امیرحسین خانیکی
شماره دانشجویی	۴۰۱۳۶۱۵۰۰۶
راه ارتباطی	۰۹۱۵۵۱۷۶۹۷۶ و t.me/khaniki_ah

بخش اول: سیستم توصیه گر

در این بخش، قرار است یک سیستم توصیه گر فیلم با استفاده از مجموعه داده فیلم TMDB 5000 ایجاد کنیم. برای ساخت این توصیه گر مبتنی بر محتوا، در گام نخست عملیات جمع‌آوری و پیش‌پردازش داده‌ها انجام و بعد از آن الگوریتم شباهت و الگوریتم تولید پیشنهادها ایجاد می‌شود. برای انجام این عملیات پردازشی کتابخانه‌های مختلفی از جمله Scikit-learn مورد استفاده قرار گرفته‌اند.

همان گونه که بیان شد، قصد داریم با استفاده از ویژگی‌های یک فیلم، فیلم‌های دیگر را به کاربر توصیه کنیم؛ بنابراین کاربر عنوان فیلمی را به سیستم می‌دهد و بر اساس پردازش‌های و معیارهای شباهت، فیلم‌هایی که ژانر و کلیدواژه‌های مشابهی دارند را به او پیشنهاد می‌دهد. مجموعه داده TMDB 5000 دارای ۴۸۰۳ ردیف و ۲۰ ستون است، اما پس از پاک کردن داده‌ها و حذف مقادیر از دست رفته، ۴۷۷۵ ردیف و ۴ ستون باقی ماند. از بین ویژگی‌های موجود در این مجموعه ویژگی شناسه یکتا، عنوان، ژانر و کلیدواژه را برای ایجاد ابر کلید (meta-key) و تولید پیشنهادها نگه می‌داریم.

```
Initial Movies: (4803, 20)
      budget                                             genres  ... vote_average  vote_count
0  237000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...  ...          7.2       11800
1  300000000  [{"id": 12, "name": "Adventure"}, {"id": 14, "...  ...          6.9        4500
2  245000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...  ...          6.3        4466
3  250000000  [{"id": 28, "name": "Action"}, {"id": 80, "nam...  ...          7.6        9106
4  260000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam...  ...          6.1        2124

[5 rows x 20 columns]
       id  ...                                           overview
0   19995  ...  In the 22nd century, a paraplegic Marine is di...
1     285  ...  Captain Barbossa, long believed to be dead, ha...
2  206647  ...  A cryptic message from Bond’s past sends him o...
3   49026  ...  Following the death of District Attorney Harve...
4   49529  ...  John Carter is a war-weary, former military ca...

[5 rows x 5 columns]
```

```
# Datasets
root_path: str = ""
#os.getcwd(encoding="utf-8")
movies_path: str = os.path.join(root_path, "datasets/tmdb_5000_movies.csv")
movies_credits_path: str = os.path.join(root_path,
"datasets/tmdb_5000_credits.csv")

# Contents
#movies: str = open(movies_path, "r", encoding="utf-8",
errors="ignore").read()
initial_movies = pd.read_csv(movies_path, sep=',')
initial_movies_credits = pd.read_csv(movies_path, sep=',')

#
"""
Answers Section
"""
#

# shapes of dataset
print("Initial Movies:", initial_movies.shape)

print(initial_movies.head(5))

# trim dataset to include relevant features
movies_set = initial_movies[['id', 'original_title', 'genres', 'keywords',
'overview']]

print(movies_set.head(5))

# omitting any null values or redundancies
movies_set.isnull().sum()
movies_set.dropna(inplace = True)
print("Cleaned Movies:", movies_set.shape)
```

همچنین ویژگی‌های ژانر و کلیدواژه به شکل object جفت داده (key-value) و از نوع json ذخیره شدند که دارای کلید یا خصیصه‌های id و name هستند. برای انجام پردازش‌ها صرفا به مقادیر name نیاز است. برای این کار و با توجه به نکات خواسته شده در تمرین (حذف فضای خالی بین کلمات ترکیبی) از ساز و کار زیر استفاده شد.

```
movies_set['genres'] = movies_set['genres'].apply(lambda x: [i['name'] for i
in eval(x)])
movies_set['keywords'] = movies_set['keywords'].apply(lambda x: [i['name'] for
i in eval(x)])
```

```
features = ['keywords', 'genres']
for i in features:
    movies_set[i] = movies_set[i].apply(clean_content)

print(movies_set['genres'].head(5))
```

حال با ترکیب ویژگی‌های ژانر و کلیدواژه، یک ابر کلید می‌سازیم. سپس با استفاده از این ابر کلید و تبدیل محتوای آن به بردار و کمک گرفتن از شباهت کسینوسی، نزدیک‌ترین و مشابه‌ترین محتواها را نسبت به مقدار ورودی فهرست و مشخص خواهیم کرد.

```
movies_set['tags'] =  movies_set['keywords']  + movies_set['genres']
#movies_set['tags'] = movies_set['tags'].apply(stem)
movies_set['tags'] = movies_set['tags'].apply(lambda x: ' '.join(x))

print(movies_set['tags'].head(5))
```

به منظور تبدیل محتوای متنی به بردار از روش TF-IDF استفاده شده است که یک بار با استفاده از توابع کتابخانه‌ای و بار دیگر به صورت دستی پردازش و محاسبه شد. سپس بر روی خروجی مرحله قبل معیار شباهت کسینوسی برای دسته‌بندی شباهت‌ها و محتواهای نزدیک به یکدیگر اعمال می‌کنیم. بطور کلی نتایج دو رویکرد محاسبه TF-IDF کمی متفاوت است؛ زیرا sklearn از نسخه smoothed-idf و بهینه‌سازی‌های کوچک دیگر استفاده می‌کند.

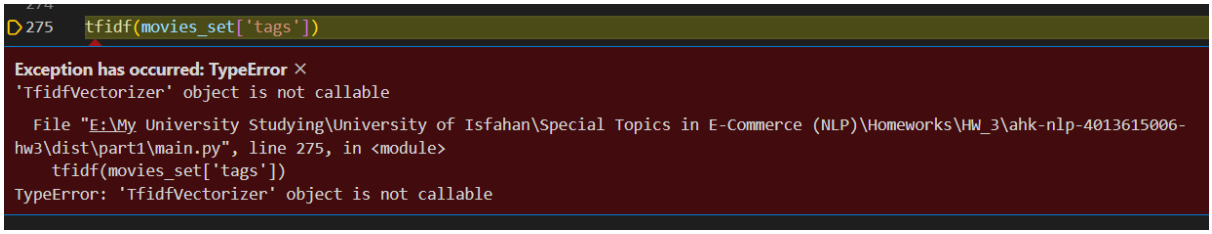
```
# computing tfidf, remove stopwords and take count of 10000 most frequent
words
tfidf = TfidfVectorizer(max_features=10000, stop_words ='english')

movies_set['original_title'] = movies_set['original_title'].fillna('')
movies_set['tags'] = movies_set['tags'].fillna('')

tfidf_title = tfidf.fit_transform(movies_set['original_title'])
tfidf_tags = tfidf.fit_transform(movies_set['tags'])

print(tfidf_title.shape)
print(tfidf_tags.shape)

# calcute the cosine similarity matrix
similarity_scores = cosine_similarity(tfidf_tags.toarray())

# create list of indices for later matching
```

```
indices = pd.Series(movies_set.index, index =
movies_set['original_title']).drop_duplicates()
```

نمونه کد تابع محاسبه دستی TF-IDF به شرح زیر است:

```
def tfidf (corpus):
    words_set = set()

    for doc in  corpus:
        words = doc.split(' ')
        words_set = words_set.union(set(words))

    n_docs = len(corpus)         #=> number of documents in the corpus
    n_words_set = len(words_set) #=> number of unique words in the

    df_tf = pd.DataFrame(np.zeros((n_docs, n_words_set)), columns=words_set)

    # compute TF
    for i in range(n_docs):
        words = corpus[i].split(' ') #=> words in the document
        for w in words:
            df_tf[w][i] = df_tf[w][i] + (1 / len(words))

    # compute IDF
    idf = {}

    for w in words_set:
        k = 0    #=> number of documents in the corpus that contain this word

        for i in range(n_docs):
            if w in corpus[i].split():
                k += 1

        idf[w] =  np.log10(n_docs / k)

    # compute TF-IDF
    df_tf_idf = df_tf.copy()

    for w in words_set:
        for i in range(n_docs):
            df_tf_idf[w][i] = df_tf[w][i] * idf[w]

    return df_tf_idf
```

تابع فوق از منظر منطقی درست است و بر روی مجموعه داده دیگری بررسی شد، اما روی این مجموعه با خطای همراه بود و خروجی مناسبی بدست نیامد.



حال بعد از انجام مراحل مقدماتی و پیش‌پردازشی همانطوری که در بالا بیان شد، الگوریتم یا تابع اصلی سیستم توصیه گر را ایجاد می‌کنیم. در این ساز و کار، با دریافت عنوان یک فیلم به عنوان محتوای ورودی مقدار آن را به بردار تبدیل کرده و سپس به فضای برداری مجموعه داده فیلم‌ها اضافه می‌شود. بعد از آن، با استفاده از تابع معیار شباهت، مجموعه داده را بر اساس نسبت شباهت به مقدار ورودی مرتب کرده و تعدادی از مشابه‌ترین فیلم‌ها (مثلا ۵ تا) را انتخاب و به عنوان پیشنهاد در خروجی معرفی می‌کنیم.

```
# generate top n recommendations list
def get_recommendations(title, n = 10, similarity_scores = similarity_scores):

    # retrieve matching movie title index
    if title not in indices.index:
        print("Movie not found.")
        return
    else:
        index = indices[title]

    # cosine similarity scores of movies in descending order
    #scores = pd.Series(cosine_sim[idx]).sort_values(ascending = False)
    similarity_scores = sorted(list(enumerate(similarity_scores[index])),
key=lambda x: x[1], reverse=True)

    # get the movie indices by the scores of the n most similar movies
    similarity_scores = similarity_scores[1:n+1]
    top_n_indices = [i[0] for i in similarity_scores]
        # top n most similar movies indexes
    # use 1:n because 0 is the same movie entered
    #top_n_idx = list(scores.iloc[1:n].index)

    # Return the top 10 most similar movies
    return movies_set['original_title'].iloc[top_n_indices]
```

```
# results
default_request = ['Mortal Kombat', "Runaway Bride", "Scream 3"]
default_top_n = 5

print ("Question Result:")
for i in default_request :
    print(f"Top {default_top_n} recommendations for ({i}) --
>\n{get_recommendations(i, default_top_n)}\n")
```

در نهایت سیستم توصیه گر تولید شده را مورد آزمایش قرار می‌دهیم. بدین منظور برای سه فیلم Scream3، Runaway Bride و Mortal Kombat پیشنهادهای ارائه شده در ادامه قابل مشاهده است.

```
Question Result:
Top 5 recommendations for (Mortal Kombat) -->
1611              Mortal Kombat: Annihilation
1670                        DOA: Dead or Alive
3856              In the Name of the King III
1001    Street Fighter: The Legend of Chun-Li
114       Harry Potter and the Goblet of Fire
Name: original_title, dtype: object

Top 5 recommendations for (Runaway Bride) -->
4115                        House of D
2325     My Big Fat Greek Wedding 2
971                 The Story of Us
1162    He's Just Not That Into You
4045          Dancer, Texas Pop. 81
Name: original_title, dtype: object

Top 5 recommendations for (Scream 3) -->
4628                            Graduation Day
4053        Friday the 13th: A New Beginning
4048                               The Calling
3902    Friday the 13th Part VI: Jason Lives
895                       Me, Myself & Irene
Name: original_title, dtype: object
```

بخش دوم: تحلیل احساسات

در این بخش از تمرین قصد داریم تحلیل احساسات را روی منبع داده Twitter Airline Sentiment انجام دهیم. منبع اصلی آن از کتابخانه Crowdflower's Data for Everyone بود. توییت‌هایی در فوریه ۲۰۱۵ از توییتر در مورد هر شرکت هواپیمایی بزرگ ایالات متحده حذف شد. گروهی از مشارکت کنندگان این محتوا را بررسی و هر توییت را در قالب مثبت، خنثی و منفی طبقه‌بندی و دلیل طبقه‌بندی منفی و همچنین امتیاز اطمینان برای برچسب اختصاص یافته را ذکر کردند. این مجموعه داده حاوی ۱۴۶۴۰ سطر و ۱۵ ویژگی اطلاعاتی (ستون) است؛ ویژگی‌های همانند شناسه توییت، احساسات، امتیاز اطمینان، دلیل منفی، نام، تعداد ری‌توییت، متن توییت، مختصات توییت، زمان توییت.

airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_gold	retweet_count	
neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	NaN	0	@Virgir @d
positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	NaN	0	@Virgir plu comm
neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	NaN	0	@Virgir I didn' Must m
negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	NaN	0	@Virgir aggr
negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	NaN	0	@Virgir and it' big ba

در بررسی ابتدایی مجموعه داده متوجه می‌شویم که تعداد توییت‌های منفی بیشتر از ترکیب مثبت و خنثی است. اینگونه می‌توان استنباط کرد که مردم بیشتر زمانی که مشکلی در پروازشان پیش می‌آید، واکنش فعال‌تری در رسانه‌های اجتماعی همانند توییتر دارند. برای انجام عملیات پردازشی به کتابخانه‌هایی همچون Numpy، Pandas، Scikit-learn و Keras استفاده شده است.

```
# Libraries
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import keras
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Dropout, Embedding, LSTM, SpatialDropout1D
from keras.callbacks import ModelCheckpoint
from keras.preprocessing.text import Tokenizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler as sc
from sklearn.metrics import roc_auc_score
```

در ادامه برای درک بهتر تفاوت احساسات برای توییت‌های مثبت، خنثی و منفی، می‌توانیم یک ابر کلمه برای هر دسته ایجاد کنیم. قبل از ایجاد این ابر کلمه، باید توییت‌ها را با حذف دسته‌های توییتر، کاراکترهای خاص، اعداد، علائم نقطه‌گذاری و هر آنچه بی تاثیر است را پاک کنیم. این به ما امکان می‌دهد کلماتی را ببینیم که نماینده هر احساس هستند در مقابل کلماتی که احتمالا در هر برچسب دیده می‌شوند.

با توجه به مسئله، فقط به متن هر متن توییت و برچسب احساسات آن نیاز است؛ بنابراین، ویژگی‌های اضافی مجموعه داده را نادیده می‌گیریم یا حذف می‌کنیم و فقط بر روی دو ستون (متن توییت و برچسب احساسات) پردازش‌ها را انجام می‌دهیم. متن هر توییت را نیز تا حد مطلوب پاکسازی می‌شود.

```
# Question 1 and 2
data = tweets[['text', 'airline_sentiment']]

default_positive_size = data[data['airline_sentiment'] == 'positive'].size
default_positive_count = 0 #data[data['airline_sentiment'] ==
'positive'].count
default_negative_size = data[data['airline_sentiment'] == 'negative'].size
default_negative_count = 0 #data[data['airline_sentiment'] ==
'negative'].count
default_neutral_size = data[data['airline_sentiment'] == 'neutral'].size
default_neutral_count = 0 #data[data['airline_sentiment'] == 'neutral'].count
# drop neutral sentiment
data = data[data.airline_sentiment != "neutral"]
data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))

# drop duplicated text
data.drop_duplicates(keep='last', inplace=True)
# drop null text
data.dropna()
tweets_list = data['text'].tolist()
cleaned_tweets_content = pd.concat([data['text'], data['airline_sentiment']],
axis = 1)

print ("Question-1 Result: ")
print (f"Positive Sentiment: Size = {default_positive_size} , Count =
{default_positive_count}")
print (f"Negative Sentiment: Size = {default_negative_size} , Count =
{default_negative_count}")
print (f"Neutral Sentiment: Size = {default_neutral_size} , Count =
{default_neutral_count}")
print (f"About Dataset:\n {data.describe()}")
print (f"Dataset Contents:\n {cleaned_tweets_content}")

word_codes, unique_words = pd.factorize(data['airline_sentiment'])
print (f"Unique Words => \n {unique_words}")
print (f"Coding => \n {word_codes}")
```

```
print(tweets_list,len(tweets_list))
data['airline_sentiment'] = pd.factorize(data['airline_sentiment'])[0]
```

```
PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE

Neutral Sentiment: Size = 6198 , Count = 0
About Dataset:
                                                    text airline_sentiment
count                                              11336             11336
unique                                             11332                 2
top     americanair continues to win ive never missed ...          negative
freq                                                   2              9086
Dataset Contents:
                                                    text airline_sentiment
1      virginamerica plus youve added commercials to ...          positive
3      virginamerica its really aggressive to blast o...          negative
4      virginamerica and its a really big bad thing a...          negative
5      virginamerica seriously would pay 30 a flight ...          negative
6      virginamerica yes nearly every time i fly vx t...          positive
...                                                  ...               ...
14633  americanair my flight was cancelled flightled ...          negative
14634           americanair right on cue with the delays          negative
14635  americanair thank you we got on a different fl...          positive
14636  americanair leaving over 20 minutes late fligh...          negative
14638  americanair you have my money you change my fl...          negative
```

به منظور تبدیل متن به بردار، برای هر کلمه منحصربه‌فرد در مجموعه داده، یک عدد را برای محاسبات برداری اختصاص داده و سپس هر توییت به عنوان یک بردار نشان داده می‌شود؛ هر عنصر حاوی یک عدد صحیح است که می‌تواند به یک کلمه خاص نگاشت شود و این کار را با استفاده از کلاس Keras Tokenizer انجام می‌گیرد که یک قطعه متن را به دنباله‌ای از اعداد صحیح تبدیل می‌کند و هر عدد صحیح شاخص یک نشانه در فرهنگ لغت است. به طور کلی، توکن سازی فرآیند تبدیل جملات به کلمات است. حجم واژگان مجموعه داده مورد بررسی ۱۵۷۶۹ کلمه است.

```
# Question 3
max_words = 10000
max_length = 200
embed_dim = 32
lstm_out = 50

tokenizer = Tokenizer(num_words = max_words, split = ' ', lower=True)
tokenizer.fit_on_texts(data['text'].values)
sequences = tokenizer.texts_to_sequences(data['text'].values)
p_sequences = pad_sequences(sequences, maxlen = max_length, padding='post')
print (f"Sequences Model => \n {sequences}")
print (f"Pad-Sequences Model => \n {p_sequences}")

x = np.array(p_sequences)
y = np.array(word_codes)
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,
random_state = 0)
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

onight waited for a delayed flight and they kept things entertaining', 'virginamerica use another browser
nse doesnt have a crossbrowser compatible website', 'virginamerica and now the flight flight booking prob
virginamerica i like the customer service but a 40 min delay just for connecting passengers seems too lon
jfk crew who moved mountains to get me home to san francisco tonight', 'virginamerica you have the absolu
y with you im delighted  thank you', 'virginamerica i need to change my flight thats scheduled in 9 hours
, 'virginamerica completely awesome experience last month boslas nonstop thanks for such an awesome fligh
 failing your customers because your check in process does not link to tsa precheck', 'virginamerica fidi
otline number and said sorry', 'virginamerica chrysichrysic your assistance yesterday when u cancelled fl
e on you', 'virginamerica another delayed flight likingyoulessandless', 'virginamerica i need to register
Sequences Model =>
 [[99, 553, 492, 1239, 2322, 1, 2, 170, 5547], [99, 64, 120, 3125, 1, 3966, 3967, 925, 15, 20, 3126, 3968
 471, 85, 21], [99, 373, 82, 270, 272, 4, 8, 6, 200, 28, 176, 25, 29, 2084, 64, 120, 2, 118, 192, 471, 85
, 3969, 5548, 199, 145, 455], [99, 190, 3, 5549, 34, 3, 50, 1369], [99, 21, 22, 319, 9, 513, 40, 76, 344,
4, 198, 148, 5551, 5552, 1369], [99, 29, 14, 534, 4, 106, 635, 221, 1537, 85, 11, 616, 206, 1, 5553, 59,
54, 51, 146, 20, 1759, 5555, 1444, 95, 114, 152, 58, 2, 1634, 455, 39, 194, 5556], [99, 5557, 959, 14, 65
 787, 107, 535, 228, 31, 106, 493, 85, 1025, 1371, 5558], [99, 3, 604, 39, 506, 1, 381, 129, 307, 9, 308,
97, 1636, 17, 19, 45], [3, 146, 99], [99, 7, 131, 66, 82, 32, 3971, 273, 5559, 92, 3, 153, 1, 112, 27, 11
1, 100, 148, 187, 1539, 67, 58, 200, 36, 351, 1, 990], [99, 3, 168, 29, 5560, 5561], [99, 3, 168, 2, 5562
8, 11, 1026, 3, 2086, 1026, 27, 11, 731, 9, 7, 104, 413, 11, 150, 455, 3, 153, 295, 1179], [99, 385, 1180
 24, 177], [99, 66, 565, 54, 396, 3128, 438, 543, 26, 361, 309, 10, 396, 401, 43, 3, 131, 3, 199, 32, 311
9, 28, 37, 63, 30, 124, 897, 277, 39, 2, 5563, 5564, 5565, 3129, 5566, 5567], [99, 698, 3, 42, 5568, 4, 4
 3, 3130, 11, 851, 456, 507, 8, 274, 204], [99, 45, 242, 1105, 3131, 10, 8, 3132, 646, 1, 330, 102, 150,
479, 91, 74, 42, 82, 1915, 1, 284, 20, 213, 3974, 460], [99, 29, 14, 106, 1138, 1371, 183, 530, 72, 1, 23
 2, 118, 138, 1, 112, 202, 170, 147, 484, 9, 5571, 5572], [99, 5573, 296, 9, 296, 202, 93, 498, 1372, 286
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

5, 102, 59, 6, 595, 19, 30, 58, 11, 194, 1927, 56], [99, 38, 104, 1302, 1029, 94, 2, 5
 1, 4009, 118, 79, 3, 183, 30, 40, 5690, 1185], [99, 2, 619, 15, 4010, 258, 5691, 28,
1384, 2098, 10, 234, 903, 990, 388, 629, 67, 5693, 515, 629, 5694], [99, 137, 22, 2664
4, 45], [99, 70, 7, 6, 2, 701, 1074, 4011, 6, 5697, 133, 593, 3168, 59, 295], [99, 38,
0, 466, 5700, 28, 1304, 2, 277, 194, 170, 70, 7, 260, 21, 48], [99, 1250, 905, 150, 9,
 [99, 620, 3, 339, 6, 449, 8, 10, 2099, 23, 1, 25, 1, 386, 6, 1445, 2666, 1075, 2667,
38, 6, 73, 627, 11, 165, 10, 20, 375, 8, 5701, 4012, 9, 772, 19, 344, 222, 160, 1302],
Pad-Sequences Model =>
 [[ 99 553 492 ...   0   0   0]
 [ 99  64 120 ...   0   0   0]
 [ 99   9  64 ...   0   0   0]
 ...
 [ 13  70   7 ...   0   0   0]
 [ 13 476  94 ...   0   0   0]
 [ 13   7  25 ...   0   0   0]]
(11336, 200) (11336,)
[[  16 1858  126 ...    0    0    0]
 [  18  392  112 ...    0    0    0]
 [  13   34  215 ...    0    0    0]
 ...
 [  13   61   36 ...    0    0    0]
 [  13    3   42 ...    0    0    0]
 [   5  120  630 ...    0    0    0]]
[[  99  109    1 ...    0    0    0]
```

در انتها و بر اساس جدول اطلاعاتی معرفی شده، مدل پنج لایه مدنظر ایجاد می‌شود و روند بهبود آموزش آن با استفاده از accuracy مورد بررسی قرار می‌گیرد.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 32)           423488

 spatial_dropout1d (SpatialD  (None, 200, 32)          0
 ropout1D)

 lstm (LSTM)                 (None, 50)                16600

 dropout (Dropout)           (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 440,139
Trainable params: 440,139
Non-trainable params: 0
_________________________________________________________________
1/1 [==============================] - 1s 639ms/step
(32, 1)
Epoch 1/5
14/14 [==============================] - 4s 82ms/step - loss: 0.2225
Epoch 2/5
14/14 [==============================] - 1s 81ms/step - loss: 0.1982
Epoch 3/5
14/14 [==============================] - 1s 77ms/step - loss: 0.1966
Epoch 4/5
14/14 [==============================] - 1s 73ms/step - loss: 0.1975
Epoch 5/5
14/14 [==============================] - 1s 73ms/step - loss: 0.1954
```