

# تشخیص ژانر فیلم با استفاده از Storyline فیلم

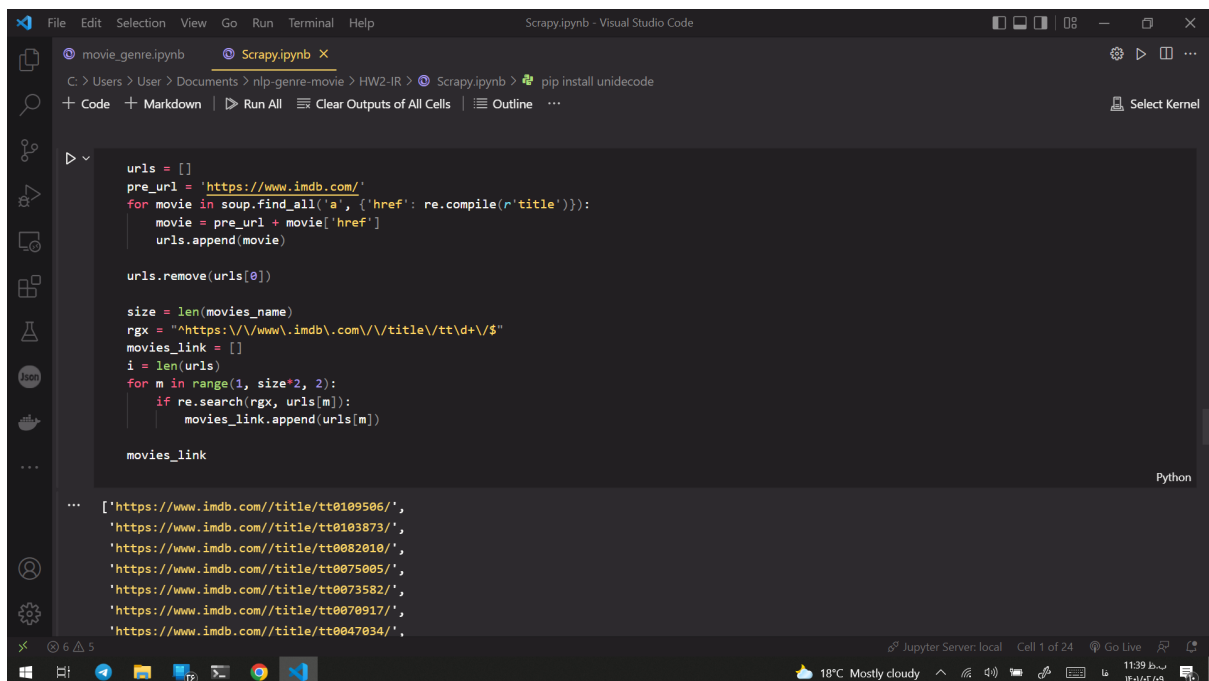
سارا زاهدی موحد

شماره دانشجویی:

98170849

در این تمرین از دیتای سایت imdb استفاده کردم تا بتوانم با گرفتن storyline، ژانر فیلم را با دقت نسبتاً خوبی تشخیص دهم.

برای این کار، اطلاعات چند فیلم (حدود 2000 فیلم) را با استفاده از کرالری که نوشتم و در فایل تمرین موجود است، جمع‌آوری کردم. اطلاعات فیلم شامل ژانر و storyline هر فیلم میشود. این اطلاعات را در یک فایل با نام dataset.json نگهداری کردم. کرال سایت و پیدا کردن لینک‌های فیلم‌های لینک (برای رجکس url):



```
urls = []
pre_url = 'https://www.imdb.com/'
for movie in soup.find_all('a', {'href': re.compile(r'title')}):
    movie = pre_url + movie['href']
    urls.append(movie)

urls.remove(urls[0])

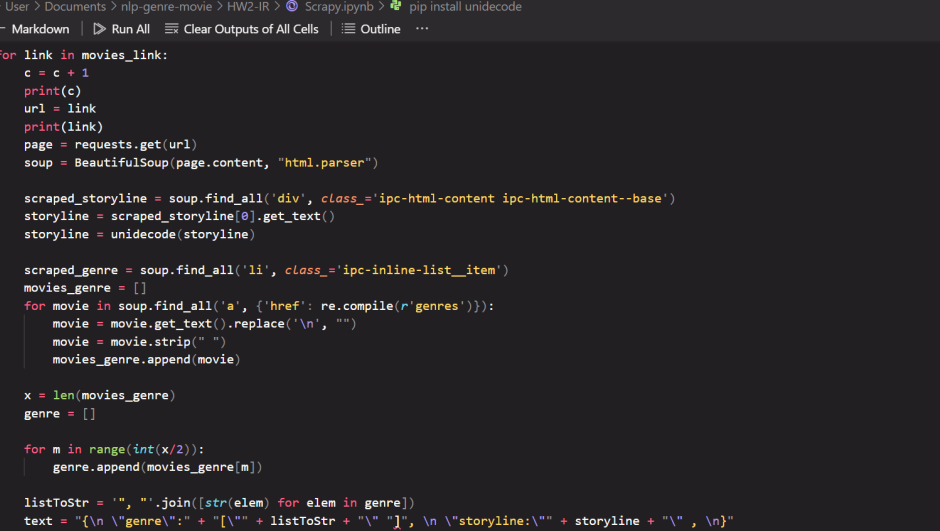
size = len(movies_name)
rgx = "^https://www.imdb.com/\\title/tt\\d+/"
movies_link = []
i = len(urls)
for m in range(1, size*2, 2):
    if re.search(rgx, urls[m]):
        movies_link.append(urls[m])

movies_link
```

Python

```
... ['https://www.imdb.com/title/tt0109506/',
'https://www.imdb.com/title/tt0103873/',
'https://www.imdb.com/title/tt0082010/',
'https://www.imdb.com/title/tt0075005/',
'https://www.imdb.com/title/tt0073582/',
'https://www.imdb.com/title/tt0070917/',
'https://www.imdb.com/title/tt0047034/']
```

پیدا کردن و ریختن استوری لاین و ژانر هر فیلم در فایل json:



```
File Edit Selection View Go Run Terminal Help
Scrapy.ipynb - Visual Studio Code

movie_genre.ipynb Scrapy.ipynb X
C:\> Users > User > Documents > nlp-genre-movie > HW2-IR > Scrapy.ipynb > pip install unicode
+ Code + Markdown | ▶ Run All | Clear Outputs of All Cells | Outline ...
Select Kernel

for link in movies_link:
    c = c + 1
    print(c)
    url = link
    print(link)
    page = requests.get(url)
    soup = BeautifulSoup(page.content, "html.parser")

    scraped_storyline = soup.find_all('div', class_='ipc-html-content ipc-html-content--base')
    storyline = scraped_storyline[0].get_text()
    storyline = unicode(storyline)

    scraped_genre = soup.find_all('li', class_='ipc-inline-list__item')
    movies_genre = []
    for movie in soup.find_all('a', {'href': re.compile(r'genres')})):
        movie = movie.get_text().replace('\n', "")
        movie = movie.strip(" ")
        movies_genre.append(movie)

    x = len(movies_genre)
    genre = []

    for m in range(int(x/2)):
        genre.append(movies_genre[m])

    listToStr = ' '.join([str(elem) for elem in genre])
    text = "{\n  \"genre\": \"[" + listToStr + "]" ,\n  \"storyline\": \"\" + storyline + \"\", \n}"
    f.write(text + '\n')
f.close()
```

```
File Edit Selection View Go Run Terminal Help movie_genre.ipynb - Visual Studio Code
C:\Users\User> Documents > nlp-genre-movie > movie_genre.ipynb M+ determine genre of the movies M+ M+ Give a storyline and get the genre > text = "John Rambo is offered the chance to ...
+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ... base (Python 3.8.8)

cleaned_tokens = [remove_stopwords(' '.join([str(elem) for elem in x['storyline']])) for x in data]

[13] ✓ 1.6s Python

tokens_len = len(cleaned_tokens)
print("number of tokens:", tokens_len)

[14] ✓ 0.5s Python

... number of tokens: 1878

+ Code + Markdown

Stemm tokens

import nltk
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()

l = len(cleaned_tokens)
for i in range(l):
    element_len = len(cleaned_tokens[i])
    for w in range(element_len):
```

تعدادی از توکن ها را چاپ کردم:

```
File Edit Selection View Go Run Terminal Help movie_genre.ipynb - Visual Studio Code
C:\Users> User > Documents > nlp-genre-movie > movie_genre.ipynb M+ determine genre of the movies M+ M+ Give a storyline and get the genre > text = "John Rambo is offered the chance to ...
+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ... base (Python 3.8.8)

tokens_len = len(cleaned_tokens)
for i in range(tokens_len):
    cleaned_tokens[i] = [item for item in cleaned_tokens[i] if len(item)>3]

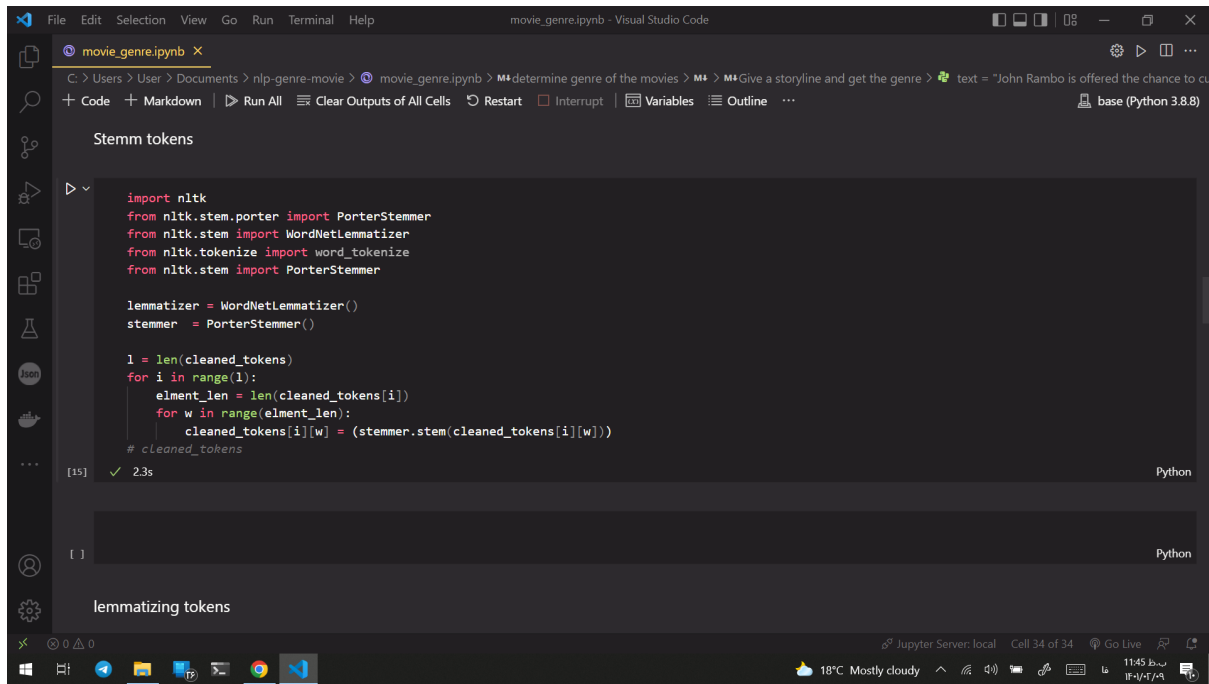
[17] ✓ 0.1s Python

print("print some tokens:\n", cleaned_tokens[0:3][0:3])

[24] ✓ 0.4s Python

... print some tokens:
[['riddler', 'sadist', 'serial', 'killer', 'begin', 'murder', 'polit', 'figur', 'gotham', 'batman', 'forc', 'investig', 'citi', 'hidden', 'corrupt',
'question', 'famili', 'involv', 'riddler', 'sadist', 'serial', 'killer', 'begin', 'murder', 'polit', 'figur', 'gotham', 'batman', 'forc', 'investig',
'citi', 'hidden', 'corrupt', 'question', 'famili', 'involv', 'riddler', 'sadist', 'serial', 'killer', 'begin', 'murder', 'polit', 'figur', 'gotham',
'batman', 'forc', 'investig', 'citi', 'hidden', 'corrupt', 'question', 'famili', 'involv', ['visionari', 'director', 'robert', 'egger', 'come',
'northman', 'action-fil', 'epic', 'follow', 'young', 'vike', 'princ', 'quest', 'aveng', 'father', 'murder', 'visionari', 'director', 'robert',
'egger', 'come', 'northman', 'action-fil', 'epic', 'follow', 'young', 'vike', 'princ', 'quest', 'aveng', 'father', 'murder', 'visionari', 'director',
'robert', 'egger', 'come', 'northman', 'action-fil', 'epic', 'follow', 'young', 'vike', 'princ', 'quest', 'aveng', 'father', 'murder'], ['seri',
'follow', 'steven', 'grant', 'mild-', 'manner', 'gift-shop', 'employe', 'becom', 'plagu', 'blackout', 'memori', 'anoth', 'life', 'steven', 'discov',
'dissoci', 'ident', 'disord', 'share', 'bodi', 'mercenari', 'marc', 'spector', 'steven/marc', 'enemi', 'converg', 'upon', 'must', 'navig', 'complex',
'ident', 'thrust', 'deadli', 'mysteri', 'among', 'power', 'egypt', 'seri', 'follow', 'steven', 'grant', 'mild-', 'manner', 'gift-shop', 'employe',
'becom', 'plagu', 'blackout', 'memori', 'anoth', 'life', 'steven', 'discov', 'dissoci', 'ident', 'disord', 'share', 'bodi', 'mercenari', 'marc',
'spector', 'steven/marc', 'enemi', 'converg', 'upon', 'must', 'navig', 'complex', 'ident', 'thrust', 'deadli', 'mysteri', 'among', 'power', 'egypt',
'seri', 'follow', 'steven', 'grant', 'mild-', 'manner', 'gift-shop', 'employe', 'becom', 'plagu', 'blackout', 'memori', 'anoth', 'life', 'steven',
'discov', 'dissoci', 'ident', 'disord', 'share', 'bodi', 'mercenari', 'marc', 'spector', 'steven/marc', 'enemi', 'converg', 'upon', 'must', 'navig',
'complex', 'ident', 'thrust', 'deadli', 'mysteri', 'among', 'power', 'egypt']]
```

در مرحله بعد، اطلاعات را نرمالایز و توکنایز کردم. سپس به هر واژه و هر ژانر، یک لیبل اختصاص دادم.



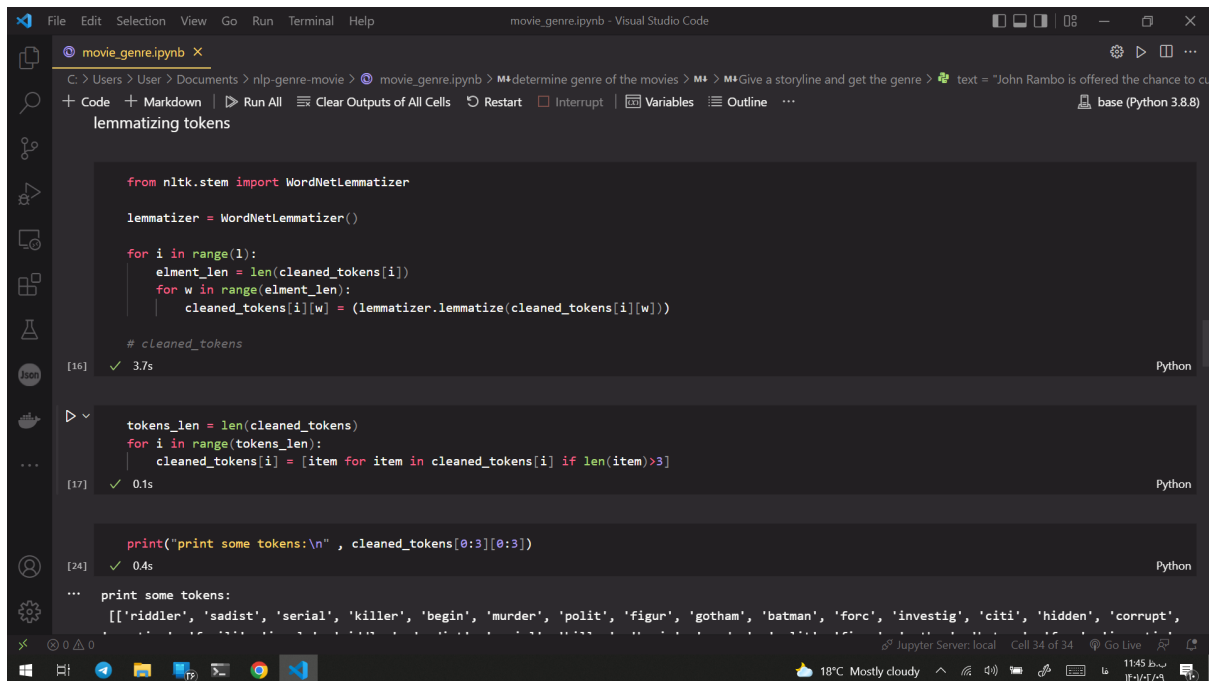
```
import nltk
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()

l = len(cleaned_tokens)
for i in range(l):
    element_len = len(cleaned_tokens[i])
    for w in range(element_len):
        cleaned_tokens[i][w] = (stemmer.stem(cleaned_tokens[i][w]))
# cleaned_tokens
```

[15] ✓ 2.3s Python

lemmatizing tokens



```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

for i in range(l):
    element_len = len(cleaned_tokens[i])
    for w in range(element_len):
        cleaned_tokens[i][w] = (lemmatizer.lemmatize(cleaned_tokens[i][w]))
# cleaned_tokens
```

[16] ✓ 3.7s Python

```
tokens_len = len(cleaned_tokens)
for i in range(tokens_len):
    cleaned_tokens[i] = [item for item in cleaned_tokens[i] if len(item)>3]
```

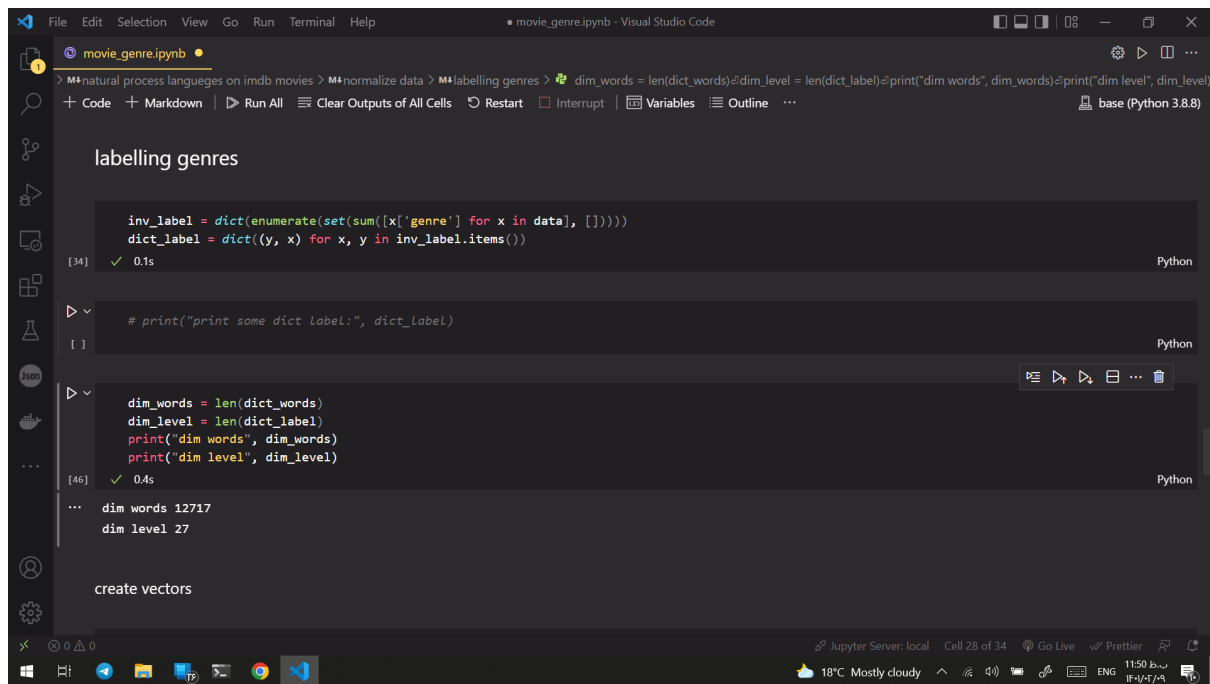
[17] ✓ 0.1s Python

```
print("print some tokens:\n", cleaned_tokens[0:3][0:3])
```

[24] ✓ 0.4s Python

```
... print some tokens:
[['riddler', 'sadist', 'serial', 'killer', 'begin', 'murder', 'polit', 'figur', 'gotham', 'batman', 'forc', 'investig', 'citi', 'hidden', 'corrupt',
```

تخصيص لیبیل:



```
File Edit Selection View Go Run Terminal Help
movie_genre.ipynb - Visual Studio Code

> M*natural process languages on imdb movies > M*naturalize data > M*labelling genres > dim_words = len(dict_words); dim_level = len(dict_label); print("dim words", dim_words); print("dim level", dim_level)

+ Code + Markdown | ▶ Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ... base (Python 3.8.8)

labelling genres

[34] ✓ 0.1s Python

# print("print some dict Label:", dict_label)

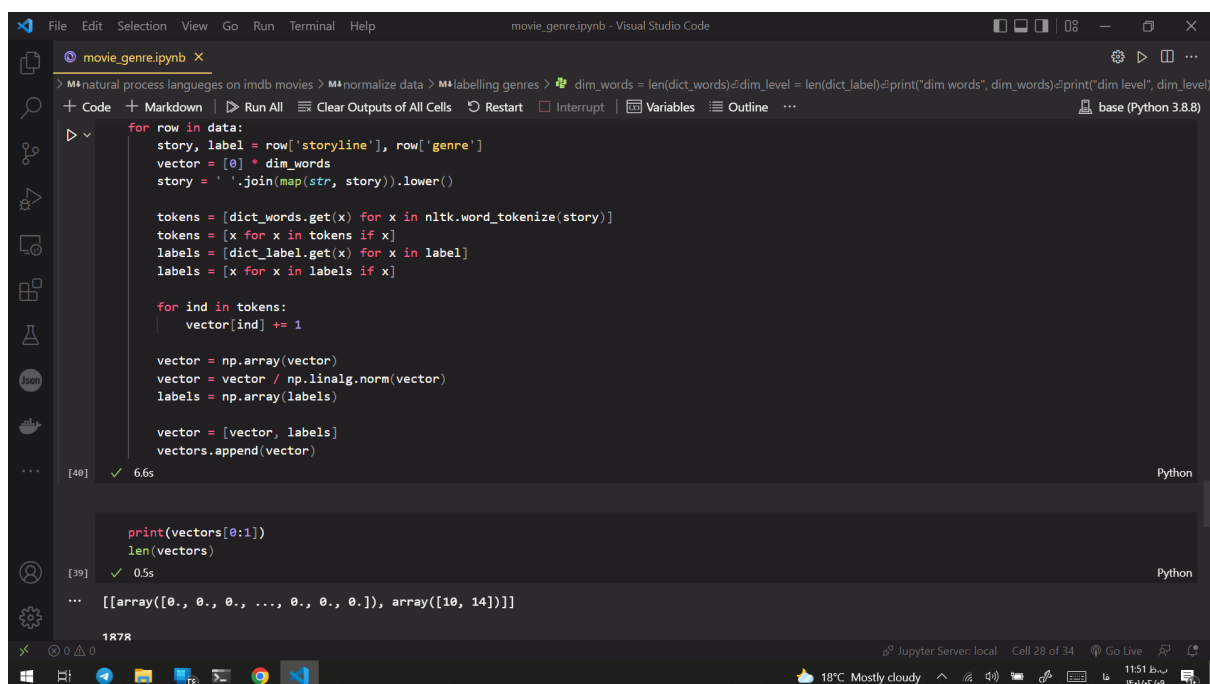
[ ] Python

[46] ✓ 0.4s Python

dim words 12717
dim level 27

create vectors
```

در مرحله بعد، برای هر استوری لاین یک وکتور از کلمات ساخته که در آن تعداد دفعات تکرار کلمات هم نشان داده میشود. اندازه وکتور را نرمال سازی کردم که به یک برسد. ساخت وکتور:



```
File Edit Selection View Go Run Terminal Help
movie_genre.ipynb - Visual Studio Code

> M*natural process languages on imdb movies > M*naturalize data > M*labelling genres > dim_words = len(dict_words); dim_level = len(dict_label); print("dim words", dim_words); print("dim level", dim_level)

+ Code + Markdown | ▶ Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ... base (Python 3.8.8)

for row in data:
    story, label = row['storyline'], row['genre']
    vector = [0] * dim_words
    story = ' '.join(map(str, story)).lower()

    tokens = [dict_words.get(x) for x in nltk.word_tokenize(story)]
    tokens = [x for x in tokens if x]
    labels = [dict_label.get(x) for x in label]
    labels = [x for x in labels if x]

    for ind in tokens:
        vector[ind] += 1

    vector = np.array(vector)
    vector = vector / np.linalg.norm(vector)
    labels = np.array(labels)

    vector = [vector, labels]
    vectors.append(vector)

[48] ✓ 6.6s Python

print(vectors[0:1])
len(vectors)

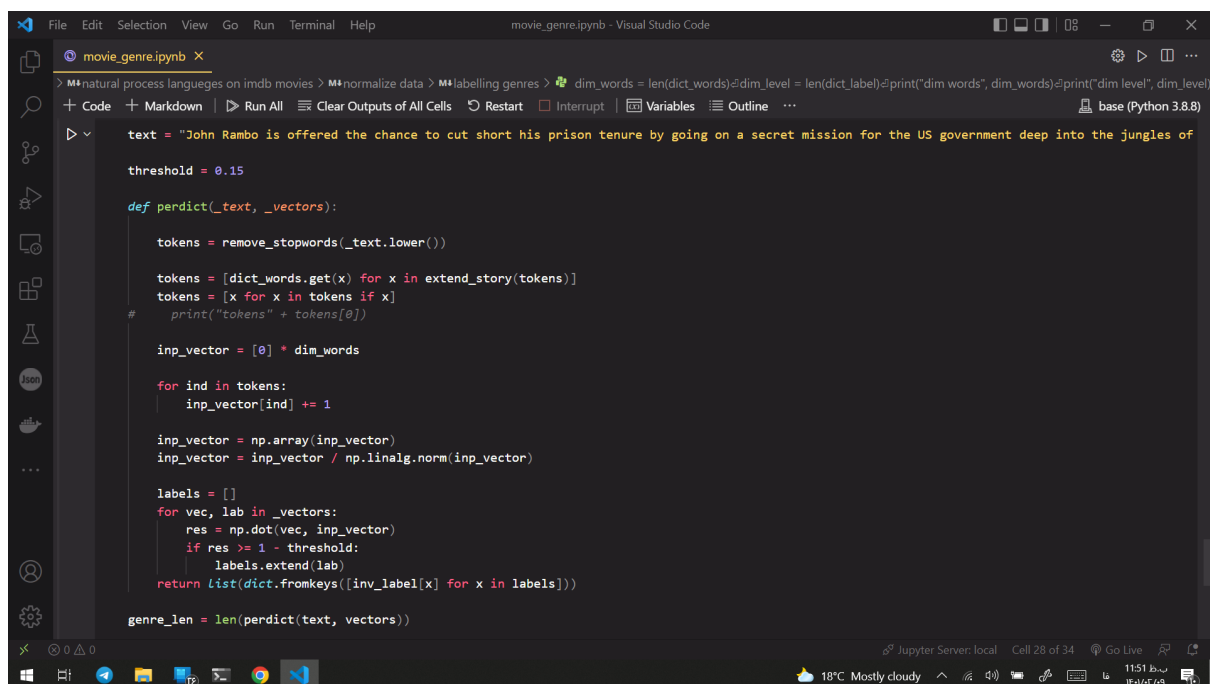
[39] ✓ 0.5s Python

[[array([0., 0., 0., ..., 0., 0., 0.]), array([10, 14])]]

1878
```

در مرحله بعد، یک استوری لاین از یک فیلم، که در فایل dataset نداریم، به عنوان نمونه دادم.

همان مراحل که برای استوری لاین های قبلی انجام دادم، برای این استوری لاین جدید تکرار کردم و وکتور ساختم. سپس زاویه وکتورها را بررسی کردم و با استفاده از کسینوس زاویه بین آنها، ژانرهای نزدیک به آن را پیدا کردم و در خروجی چاپ کردم.



```
File Edit Selection View Go Run Terminal Help
movie_genre.ipynb - Visual Studio Code

> M+natural process languages on imdb movies > M+normalize data > M+labelling genres > dim_words = len(dict_words);dim_level = len(dict_label);print("dim words", dim_words);print("dim level", dim_level)
+ Code + Markdown + Run All + Clear Outputs of All Cells + Restart + Interrupt + Variables + Outline ... base (Python 3.8.8)

text = "John Rambo is offered the chance to cut short his prison tenure by going on a secret mission for the US government deep into the jungles of

threshold = 0.15

def perdict(_text, _vectors):

    tokens = remove_stopwords(_text.lower())

    tokens = [dict_words.get(x) for x in extend_story(tokens)]
    tokens = [x for x in tokens if x]
    # print("tokens" + tokens[0])

    inp_vector = [0] * dim_words

    for ind in tokens:
        inp_vector[ind] += 1

    inp_vector = np.array(inp_vector)
    inp_vector = inp_vector / np.linalg.norm(inp_vector)

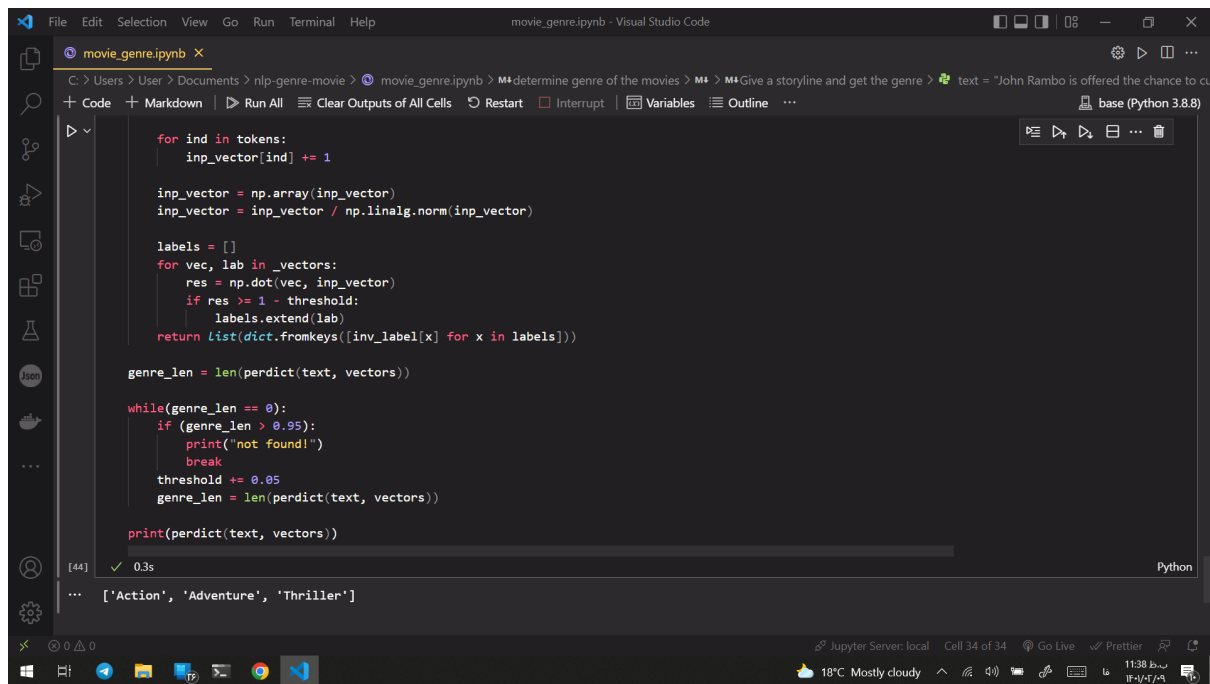
    labels = []
    for vec, lab in _vectors:
        res = np.dot(vec, inp_vector)
        if res >= 1 - threshold:
            labels.extend(lab)
    return List(dict.fromkeys([inv_label[x] for x in labels]))

genre_len = len(perdict(text, vectors))
```

برای مثال استوری لاین فیلم Rambo را قرار دادم. با سرچ در گوگل برای فیلم Rambo، استوری لاین زیر را آورد:

Having long-since abandoned his life as a lethal soldier, John Rambo (Sylvester Stallone) lives a solitary life near the Thai border. Two weeks after guiding a missionary (Julie Benz) and her comrades into Burma, he gets an urgent call for help. The missionaries have not returned and although he is reluctant to embrace violence again, Rambo sets out to rescue the captives from the Burmese army.

در قدم این استوری لاین را قرار دادم و ژانر فیلم را به شکل زیر در خروجی چاپ کرد:



```
for ind in tokens:
    inp_vector[ind] += 1

inp_vector = np.array(inp_vector)
inp_vector = inp_vector / np.linalg.norm(inp_vector)

labels = []
for vec, lab in _vectors:
    res = np.dot(vec, inp_vector)
    if res >= 1 - threshold:
        labels.extend(lab)
return List(dict.fromkeys([inv_label[x] for x in labels]))

genre_len = len(perdict(text, vectors))

while(genre_len == 0):
    if (genre_len > 0.95):
        print("not found!")
        break
    threshold += 0.05
    genre_len = len(perdict(text, vectors))

print(perdict(text, vectors))
```

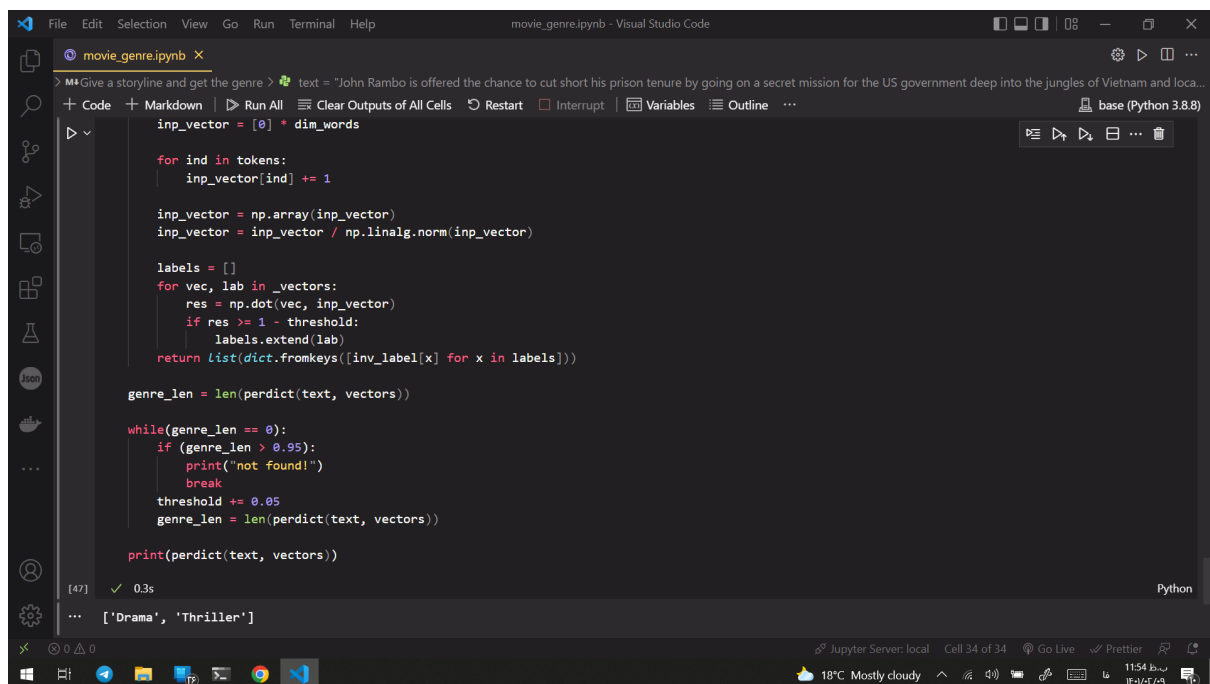
[44] ✓ 0.3s Python

... ['Action', 'Adventure', 'Thriller']

به عنوان یک نمونه دیگر، سریال برکینگ بد را امتحان کردم:

storyline: Set in Albuquerque, New Mexico, between 2008 and 2010, Breaking Bad follows Walter White, a meek high school chemistry teacher who transforms into a ruthless player in the local methamphetamine drug trade, driven by a desire to financially provide for his family after being diagnosed with terminal lung cancer

خروجی:



```
inp_vector = [0] * dim_words

for ind in tokens:
    inp_vector[ind] += 1

inp_vector = np.array(inp_vector)
inp_vector = inp_vector / np.linalg.norm(inp_vector)

labels = []
for vec, lab in _vectors:
    res = np.dot(vec, inp_vector)
    if res >= 1 - threshold:
        labels.extend(lab)
return List(dict.fromkeys([inv_label[x] for x in labels]))

genre_len = len(perdict(text, vectors))

while(genre_len == 0):
    if (genre_len > 0.95):
        print("not found!")
        break
    threshold += 0.05
    genre_len = len(perdict(text, vectors))

print(perdict(text, vectors))
```

[47] ✓ 0.3s Python

... ['Drama', 'Thriller']

در ادامه لینک صفحاتی که از آنها دیتا جمع‌آوری کردم را قرار میدهم:

1. [https://www.imdb.com/chart/top/?ref\\_=nv\\_mv\\_250](https://www.imdb.com/chart/top/?ref_=nv_mv_250)
2. <https://www.imdb.com/chart/boxoffice>
3. <https://www.imdb.com/chart/moviemeter>
4. <https://www.imdb.com/chart/tvmeter>
5. <https://www.imdb.com/chart/toptv>
6. <https://www.imdb.com/india/top-rated-indian-movies>
7. <https://www.imdb.com/chart/bottom>
8. <https://www.imdb.com/search/title/?genres=action>
9. <https://www.imdb.com/search/title/?genres=adventure>
10. <https://www.imdb.com/search/title/?genres=biography>
11. <https://www.imdb.com/search/title/?genres=crime>
12. <https://www.imdb.com/search/title/?genres=comedy>
13. <https://www.imdb.com/search/title/?genres=documentary>
14. <https://www.imdb.com/search/title/?genres=drama>
15. <https://www.imdb.com/search/title/?genres=family>
16. <https://www.imdb.com/search/title/?genres=fantasy>
17. <https://www.imdb.com/search/title/?genres=history>
18. <https://www.imdb.com/search/title/?genres=mystery>
19. <https://www.imdb.com/search/title/?genres=horror>
20. <https://www.imdb.com/search/title/?genres=sci-fi>
21. <https://www.imdb.com/search/title/?genres=documentary>
22. <https://www.imdb.com/search/title/?genres=war>
23. <https://www.imdb.com/search/title/?genres=thriller>
24. <https://www.imdb.com/search/title/?genres=war>
25. <https://www.imdb.com/search/title/?genres=western>