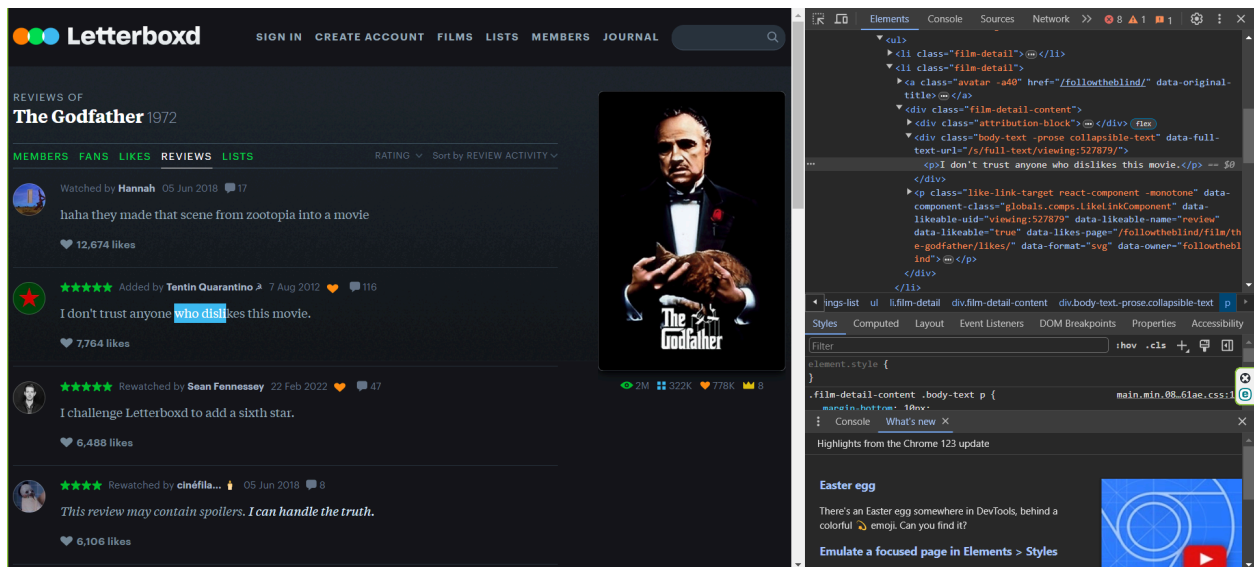


در این تمرین می‌خواهیم review های فیلم های The Godfather قسمت های 1 و 2 و 3 را از سایت Letterboxd، بدست بیاوریم و یا عبارتی Crawl کنیم. دقت کنید که توضیحات و گزارش هر بخش را به طور مبسوط در نوت بوک تمرین که به همراه این فایل آپلود میکنم آورده ام، به همین دلیل از زیاده گویی و توضیحات بی جا در این فایل امتناع میکنم. دقت کنید که تسکی که برای این تمرین روی داده های crawl شده تعریف کردیم، بررسی و رتبه بندی نظرات و احساسات مردم نسبت به این سه فیلم است، که به وسیله ی TF-IDF scores و Sentiment Analysis انجام میشود. چند نکته قابل توجه است، اول این که هر صفحه نظرات Letterboxd شامل 12 نظر میشود، برای هر فیلم نظرات 250 صفحه معادل با 3000 ریویو را crawl میکنیم و تمامی آنها را print میکنیم، دقت کنید که نتایج print شده را در یک فایل سیو کرده ایم که بعداً محتویات این فایل را preprocess میکنیم. در قسمت اول تعدادی از کتابخانه های مورد نیاز را import میکنیم. در قسمت بعدی ریویو ها را crawl میکنیم، برای این کار از لینک های زیر استفاده میکنیم.

```
godfather_url =
"/"https://letterboxd.com/film/the-godfather/reviews/by/activity
godfather_url_pattern =
"/"https://letterboxd.com/film/the-godfather/reviews/by/activity/page/250
godfather2_url =
"/"https://letterboxd.com/film/the-godfather-part-ii/reviews/by/activity
godfather2_url_pattern =
"https://letterboxd.com/film/the-godfather-part-ii/reviews/by/activity/pag
"/e/250
godfather3_url =
"/"https://letterboxd.com/film/the-godfather-part-iii/reviews/by/activity
godfather3_url_pattern =
"https://letterboxd.com/film/the-godfather-part-iii/reviews/by/activity/pa
"/ge/250
```

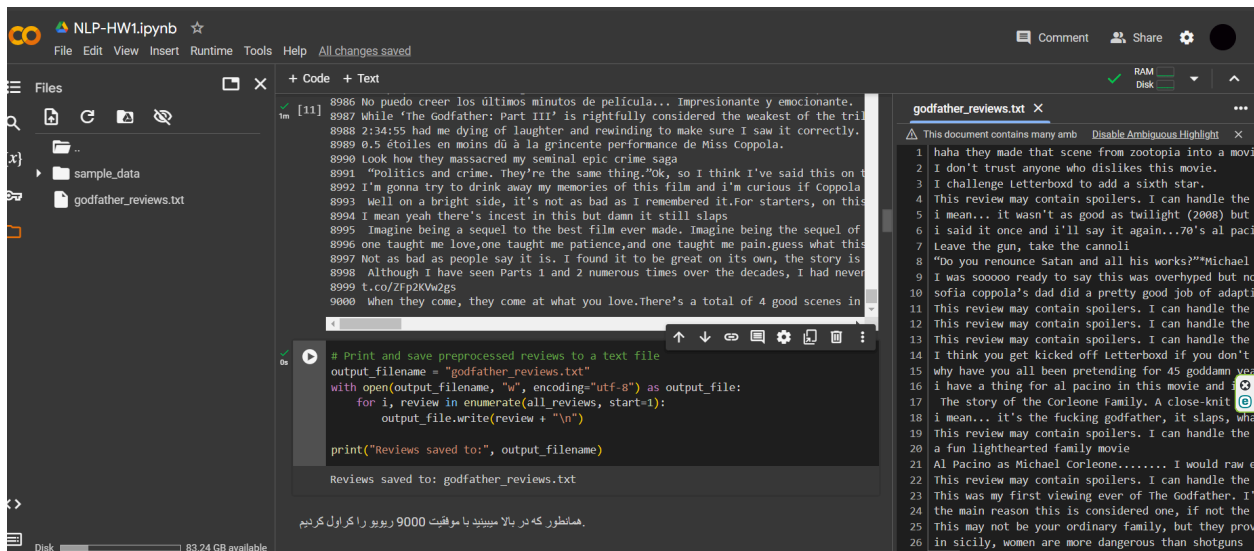
دقت کنید که چون لینک صفحه ی اول نظرات با لینک سایر صفحات متفاوت است باید برای هر فیلم از دو لینک استفاده کنیم، یکی برای صفحه ی اول و دیگری بعنوان یک الگو برای تولید همه ی آدرس های 250 صفحه ی فیلم توسط تابع generate\_url که در ادامه به آن خواهیم رسید. باید ریویو های تمام 250 صفحه را برای هر فیلم crawl کنیم، به همین دلیل نیاز به 3 تابع داریم، یکی برای تولید آدرس های تمام صفحات، دیگری برای crawl کردن اطلاعات فقط یک صفحه، و در نهایت تابع سومی که با استفاده از این دو تمام صفحات را crawl میکند.



همانطور که در تصویر بالا قابل مشاهده است برای crawl کردن ریویو های یک صفحه باید به سراغ تگ هایی با کلاس `'class_='body-text -prose collapsible-text'`

برویم و همه ی آنها را برای هر صفحه استخراج کنیم.

در نهایت نتایج استخراج شده را برای همه ی فیلم ها پرینت کرده و در یک فایل ذخیره میکنیم، دقت کنید همانطور که توضیح دادیم 3000 ریویو ابتدایی آن مربوط به قسمت اول فیلم، 3000 تای بعدی مربوط به قسمت دوم فیلم و 3000 تای اخر مربوط به قسمت سوم فیلم میشوند.



بعد از `import` و نصب تعدادی از کتابخانه های مورد نیاز برای `preprocessing` نوبت به پیاده سازی `preprocessing` میشود، با بررسی ریویو هایی که `crawl` کردم متوجه شدم که چون همه ی آنها عامیانه نوشته شده اند تعداد خیلی زیادی از آنها شامل `emoji` هستند، و همچنین اکثر آنها دارای خطای تایپی و یا اشتباه نوشتن اختصارات هستند (مثلا `did not` را به شکل `didnt` نوشته اند)، همچنین تعداد قابل توجهی از آنها با زبانی غیر از اینگلیسی نوشته شده اند که باید حذف شوند. در کلاس `preprocess` ای که پیاده سازی کردم علاوه بر پیش پردازش های معمول (`lemmatize, word tokenize,`) `remove punctuations` به این موارد هم رسیدگی کردم. همچنین چون تسکی که انتخاب کردیم بررسی احساس و نظر مردم است باید در حذف `stopwords` حواسمان باشد که کلمات نفی کننده را حذف نکنیم، فرض کنید کلمه ی `not` بعنوان `stopword` حذف شود در این صورت ریویو `this movie is not good` به `this movie is good` تغییر پیدا میکند که کاملاً مخالف چیزی است که مدنظر نویسنده بوده است، به همین دلیل یک لیست از کلمات نفی کننده در توابع `stopwords` و `lemmatize` این کلاس تعریف میکنیم. از آنجا که بعضی از ریویو ها شامل لینک میشوند، اهمیت دارد که این لینک ها را هم حذف کنیم. این کلاس شامل توابع زیر است:

`get_instance():`

از آنجا که این کلاس را به روش `singleton` پیاده سازی کردیم و فقط باید یک `instance` از آن ساخته شده باشد، این تابع را تعریف میکنیم.

`init():`

این تابع یک مسیر برای `stopwords` میگیرد، در ابتدا یک `attribute` برای `stopwords` کلاس تعریف میشود، سپس یک `attribute` برای `lemmatization` و یک `attribute` برای تصحیح املا از `Speller` تعریف میکنیم.

`preprocess():`

این تابع از توابعی که در همین کلاس تعریف کردیم برای `preprocess` های عمومی استفاده میکند.

`normalize():`

در این تابع ابتدا کل متن را به `lowercase` تبدیل کرده سپس `punctuation` ها را (مانند `;` , `...`) حذف میکنیم، مخفف سازی هارا بزرگ میکنیم (مثلا `didn't` و `didnt` را به `did not` تبدیل میکنیم) و در نهایت `lemmatization` را انجام میدهیم.

`lemmatize():`

هر کلمه ی متن را به لما ی آن تبدیل میکند.

`remove_links():`

که لینک های متن را حذف میکند.

`expand_contractions():`

که مخفف سازی هایی که گفته شد را به فرم صحیح تبدیل میکند.

`remove_emojis():`

ایموجی های متن را حذف میکند.

`remove_non_english_text():`

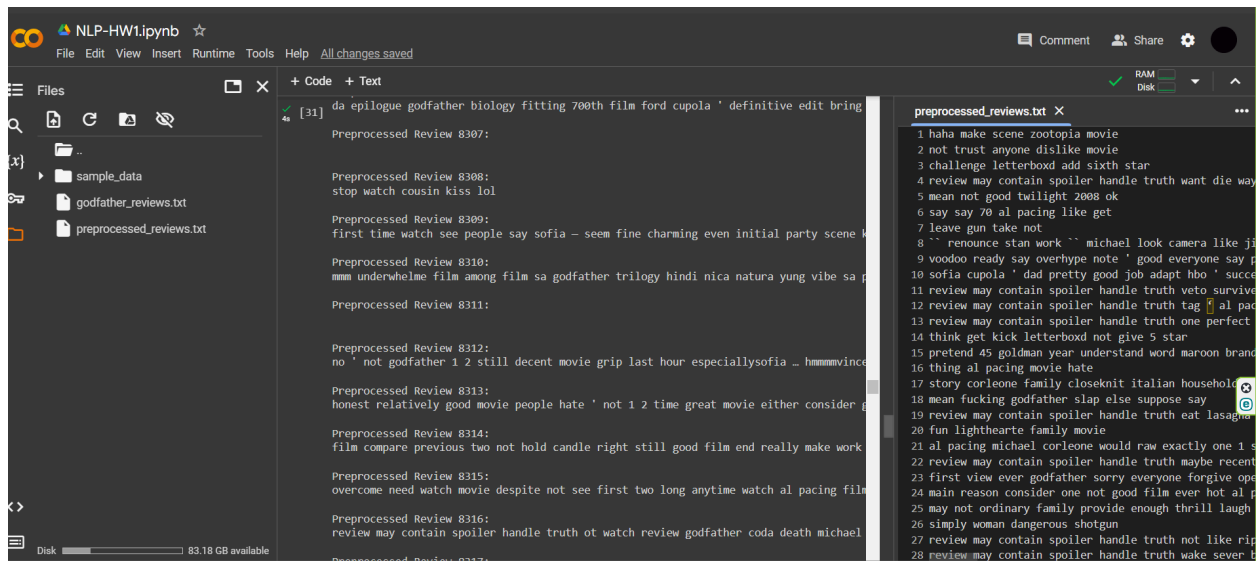
ریویو هایی که به زبانی غیر از اینگلیسی نوشته شده باشند را حذف میکند.

`correct_spelling():`

ایرادات تایپی را برطرف میکند.

در ادامه عملیات های توابع این کلاس را روی چند ورودی نمونه تست میکنیم و نتیجه را پرینت میکنیم. در ادامه فایلی که از ریویو ها سیو کرده بودیم را لود میکنیم و پیش پردازش هایی که پیاده سازی کردیم را روی آن اجرا میکنیم. در ادامه ریویو های `preprocess` شده را پرینت میکنیم و در فایل تکست دیگری سیو میکنیم. دقت کنید که پیش پردازش کردن متن

ممکن است مقداری طول بکشد. برای من بصورت لوکال 42 دقیقه و 18 ثانیه طول کشید.



The screenshot shows a Jupyter Notebook titled 'NLP-HW1.ipynb'. The left sidebar displays a file explorer with a folder named 'sample\_data' containing two files: 'godfather\_reviews.txt' and 'preprocessed\_reviews.txt'. The main area is split into two panes. The left pane shows a code cell with the following text:

```
[31] da epilogue godfather biology fitting 700th film ford cupola ' definitive edit bring  
Preprocessed Review 8307:  
  
Preprocessed Review 8308:  
stop watch cousin kiss lol  
  
Preprocessed Review 8309:  
first time watch see people say sofia - seem fine charming even initial party scene k  
  
Preprocessed Review 8310:  
mmmm underwhelme film among film sa godfather trilogy hindi nica natura yung vibe sa p  
  
Preprocessed Review 8311:  
  
Preprocessed Review 8312:  
no ' not godfather 1 2 still decent movie grip last hour especiallysofia ... hmmmvince  
  
Preprocessed Review 8313:  
honest relatively good movie people hate ' not 1 2 time great movie either consider g  
  
Preprocessed Review 8314:  
film compare previous two not hold candle right still good film end really make work  
  
Preprocessed Review 8315:  
overcome need watch movie despite not see first two long anytime watch al pacing film  
  
Preprocessed Review 8316:  
review may contain spoiler handle truth ot watch review godfather coda death michael  
Preprocessed Review 8317:
```

The right pane shows a text file named 'preprocessed\_reviews.txt' with 28 lines of preprocessed review text, including phrases like 'haha make scene zootopia movie', 'not trust anyone dislike movie', 'challenge letterboxd add sixth star', 'review may contain spoiler handle truth want die way', 'mean not good twilight 2008 ok', 'say say 70 al pacing like get', 'leave gun take not', 'renounce stan work ' michael look camera like ji', 'voodoo ready say overhype note ' good everyone say p', 'sofia cupola ' dad pretty good job adapt hbo ' succe', 'review may contain spoiler handle truth veto survive', 'review may contain spoiler handle truth tag al pac', 'review may contain spoiler handle truth one perfect', 'think get kick letterboxd not give 5 star', 'pretend 45 goldman year understand word maroon brand', 'thing al pacing movie hate', 'story corleone family closeknit italian household', 'mean fucking godfather slap else suppose say', 'review may contain spoiler handle truth eat lasagne', 'fun lighthearte family movie', 'al pacing michael corleone would raw exactly one 1 s', 'review may contain spoiler handle truth maybe recent', 'first view ever godfather sorry everyone forgive ops', 'main reason consider one not good film ever hot al f', 'may not ordinary family provide enough thrill laugh', 'simply woman dangerous shotgun', 'review may contain spoiler handle truth not like rip', and 'review may contain spoiler handle truth wake sever t'.

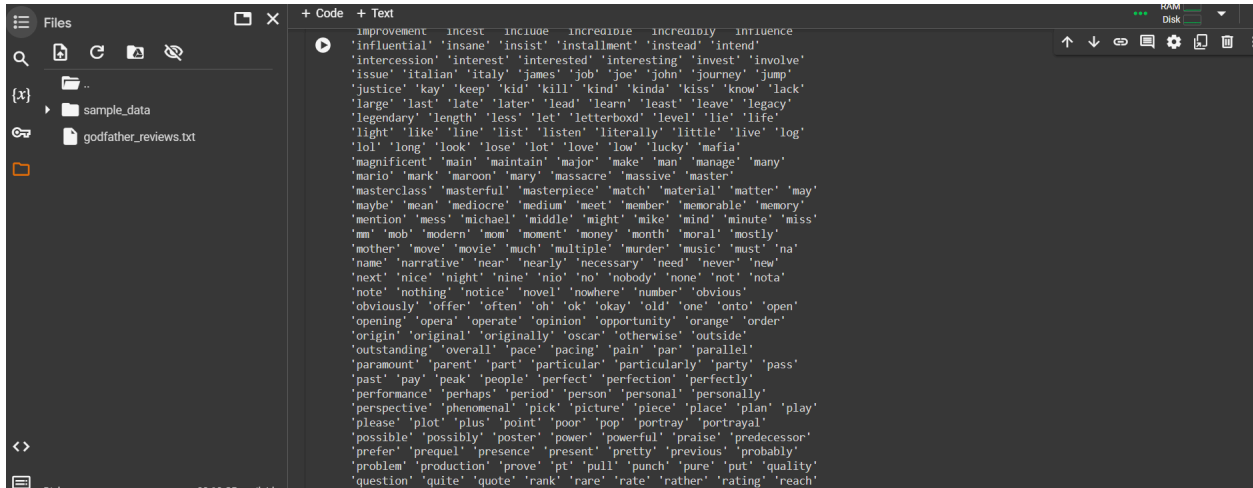
در این قسمت باید به بررسی این متن پیش پردازش شده برسیم.  
این کار را در دو قسمت انجام میدهیم. TF-IDF scores و Sentiment Analysis

### **TF-IDF scores**

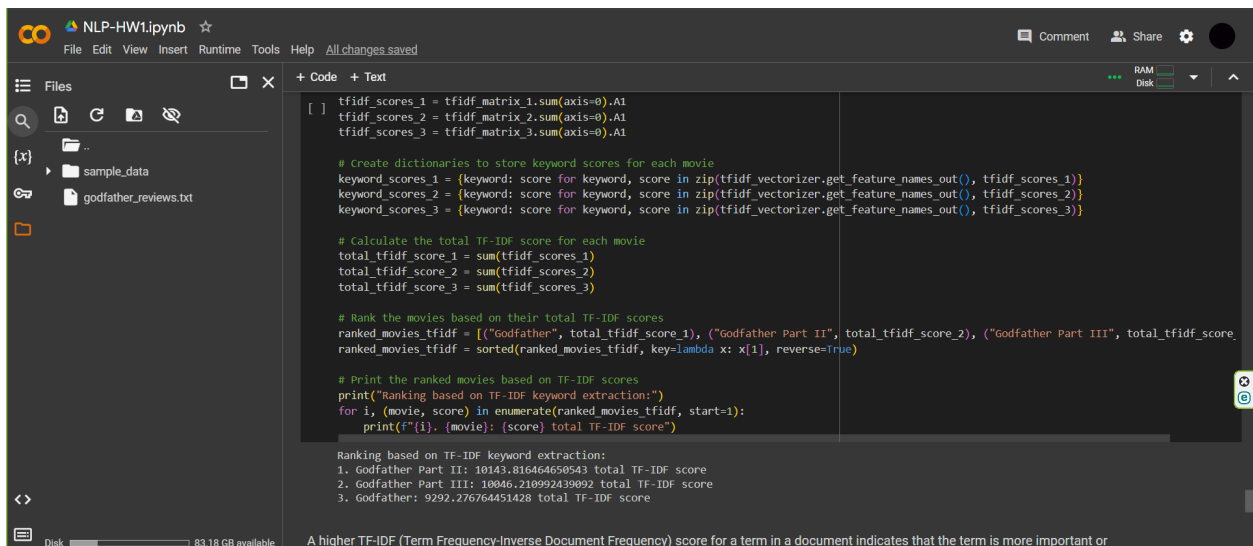
TF-IDF, short for Term Frequency-Inverse Document Frequency, is a statistical measure used to evaluate the importance of a word within a document relative to a collection of documents (corpus). It is commonly used in information retrieval and text mining to determine the significance of a term in a document.

---

Now we use TF-IDF method to extract Keywords from the preprocessed reviews.



As you can see above not the keywords are extracted based on their TF-IDF score, meaning they are selected based on their relevance in the reviews. Notice that not every extracted keyword is a positive concept, so the score that we will calculate in the next cell will tell us how passionate people were about each movie, note that not all passion is good passion :)

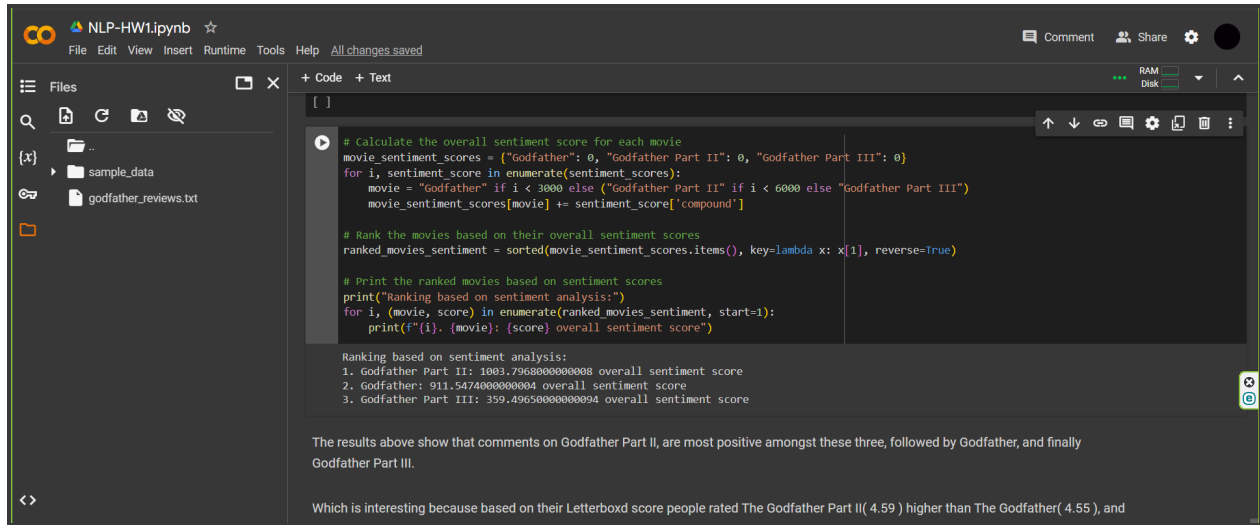


A higher TF-IDF (Term Frequency-Inverse Document Frequency) score for a term in a document indicates that the term is more important or relevant to that specific document compared to other documents in the corpus. TF-IDF is a statistical measure used in natural language processing to evaluate the significance of a term in a document relative to a collection of documents.

## Sentiment Analysis

Now we use sentiment analysis method to rank the three movies based on their popularity.

We perform sentiment analysis with the help of SentimentIntensityAnalyzer module from `nltk.sentiment.vader`



The screenshot shows a Jupyter Notebook interface with a file explorer on the left containing 'sample\_data' and 'godfather\_reviews.txt'. The main area displays Python code for sentiment analysis using NLTK's Vader module. The code calculates overall sentiment scores for 'Godfather', 'Godfather Part II', and 'Godfather Part III', ranks them, and prints the results. Below the code, the output shows the ranking: Godfather Part II (1003.7968000000008), Godfather (911.5474000000004), and Godfather Part III (359.49650000000094). A text box explains that Godfather Part II has the most positive comments, followed by Godfather, and then Godfather Part III. A final note mentions that based on Letterboxd scores, Godfather Part II (4.59) is rated higher than Godfather (4.55), and Godfather is rated higher than Godfather Part III (3.45).

```
[ ]

# Calculate the overall sentiment score for each movie
movie_sentiment_scores = {"Godfather": 0, "Godfather Part II": 0, "Godfather Part III": 0}
for i, sentiment_score in enumerate(sentiment_scores):
    movie = "Godfather" if i < 3000 else ("Godfather Part II" if i < 6000 else "Godfather Part III")
    movie_sentiment_scores[movie] += sentiment_score["compound"]

# Rank the movies based on their overall sentiment scores
ranked_movies_sentiment = sorted(movie_sentiment_scores.items(), key=lambda x: x[1], reverse=True)

# Print the ranked movies based on sentiment scores
print("Ranking based on sentiment analysis:")
for i, (movie, score) in enumerate(ranked_movies_sentiment, start=1):
    print(f"{i}. {movie}: {score} overall sentiment score")

Ranking based on sentiment analysis:
1. Godfather Part II: 1003.7968000000008 overall sentiment score
2. Godfather: 911.5474000000004 overall sentiment score
3. Godfather Part III: 359.49650000000094 overall sentiment score

The results above show that comments on Godfather Part II, are most positive amongst these three, followed by Godfather, and finally Godfather Part III.

Which is interesting because based on their Letterboxd score people rated The Godfather Part II( 4.59 ) higher than The Godfather( 4.55 ), and
```

The results above show that comments on Godfather Part II, are most positive amongst these three, followed by Godfather, and finally Godfather Part III.

This is interesting because based on their Letterboxd score people rated The Godfather Part II( 4.59 ) higher than The Godfather( 4.55 ), and they rated The Godfather higher than The Godfather Part III( 3.45 ).