Table of Contents

Executive Summary

This report focuses on website analytics from IMdb film reviews and aims to give recommendations about a company's business metrics to suggest future ideas for its market capitalisation. This report is supported by a literature review in the second section of traditional methods and the proposed methods while incorporating data analysis. In the last two sections, there is an analysis of the results and a discussion that includes managerial recommendations for the company.

1.0 Introduction

The current evolution of the US indie film sector has developed to the tune of film streaming technologies like Amazon and Netflix. Which has broken down entrenched notions of indie culture that have established brand identities, redefined conventional hallmarks of 'art house films', and promoted amateur filmmaking in commercial venues (Briggs, 2022). The trajectory of *A24* – an American independent entertainment company – has displayed an accolade-rich decade of filmmaking that has brought an incredible rise to its brand. Despite its success, independent film companies need assistance to compete against the monopoly of Hollywood. Any new production's financial success depends on consumers' reviews or

experiences of products/services as they act as candid sentiments that influence a consumer's propensity to make a purchase (Kim, Trimi and Lee, 2021; Pentheny, 2015). This emphasises the account of customer reviews and their importance, adoption, and leverage in future strategies to coincide with changing consumer preferences, normally associated with long-standing 'A-list' films.

2.0 Traditional and Proposed Methods

To make decision-making in a business context, the processes of filmmaking that inform business decisions need to be understood. The practical approach to filmmaking involves four phases of film production which include pre-production, production, post-production and distribution (Landry, 2018). From a business analytical perspective, the most pertinent methods that can be used to inform decisions about a film's value include the most former and latter stages: pre-production and distribution. Which relate to the creation and planning for a film and its decision on different distribution channels, respectively.

2.1 Traditional Methods

At the heart of traditional filmmaking methods, are creative ideas and marketing to the public. The art of creating successful films relies heavily on the director's vision, performances, the chemistry between actors, cinematography, and music (Hennig-Thurau, Ravid and Sorenson, 2021). Analytical processes in the film's pre-production value chain rely on past Box office numbers. Their performance dictates the product's life cycle, as when successful, subsequent films in the genre, production size, or certification will be created. Where turnout reflects consumer demand.

However, independent films are different from big-budget 'A-list' films in Hollywood, especially in terms of budget and marketing support. Typically, variations of critical reception in niche media channels, newspapers, magazines, or websites are used to identify changing customer preferences for independent films (Rai, 2012). Film festival screenings can also act as a vehicle for drawing exposure and Word of Mouth (WOM) marketing like fandom and buzz marketing. Which is an advertising technique that aims to create intriguing issues or events to elicit social attention, generally used on social media platforms (Carl, 2006). The distribution of films is a complex and ever-changing assessment of a film's potential, schedule, and market to ensure position does not consume each other. Film budgets need strategic management to allocate marketing expenses to maximise the film's potential

revenue (Vitkauskaitė, 2017). Indie films have limited theatrical releases and filming accrues fixed costs. This stresses the importance of using data analytics to guide these companies to success.

2.2 Proposed Method

Where past methods have concentrated on marketing principles, a proposed method could offer an opportunity to create strategic management with data analytics. Accessibility of the Internet has facilitated convergence in film culture and extended the reach and capture of people's opinions (Kim, Trimi and Lee, 2021). Despite the traditional means of watching movies in theatres, the accelerated popularity of the streaming services segment, notably during COVID-19, has been cemented in home entertainment. Films distributed to consumers through streaming services doubled between 2000 and 2010 (Waldfogel, 2017). This has reduced barriers to entry which in turn has relaxed the pre-production and the distribution bottleneck, making it easier for many consumers to find independent films (Waldfogel, 2017). Moreover, data analytics creates the ability to reform and shift the normal paradigm of just looking at Box Office numbers and investment in marketing.

Although, there is a niche audience for indie films, harnessing data from these customers can determine the appeal for these films. Predictive analytics is a major tool that can be used to forecast how budgets can be spent, dates of release and genres. A study by Pentheny (2015) discovered that modifications in the origin and content of movie reviews impact how consumers think. However, the extent of this impact varies depending on the review source and the type of information contained in the review. This type of information is broken down into three behavioural KPIs for this report. The modern entertainment industry with data analytics relies on these different Key Performance Indicators (KPIs) posited to make greater financial gains.

1. **Customer Reactions** and customer reviews alike can proxy and predict the box office numbers of films and in turn the performance of an entertainment company. Christoforou *et el.,* (2017) uses customer reactions from movie trailers to predict KPIs like Box-Office performance. Likewise, in the past, buzz and guerrilla marketing have used this to elicit reactions which facilitate reviews in media.

2. **Customer Satisfaction** is normally the subject contained in a review and is a key aspect that can be used to gain insights to improve customer experience and increase loyalty (image).
3. **Audience Engagement** is the summarisation of website traffic and social media attention that can be used to determine which exhibits are most popular and which need to be changed or curtailed.

Following Pentheny (2015) and Waldfogel (2017) the researchers used the IMDb database to determine customer thought processes around films. Which can be used as a tool to track the KPIs. The Internet Movie DataBase (IMDb) contains information related to films where users can rate media on a 10-point scale which can be accessed to build a dataset. This study will employ sentiment analysis to analyse Customer Satisfaction and Topic Modelling to analyse Customer Reactions and Audience Engagement surrounding A24 films. As understanding what exhibits are successful in these three areas can help managers create a framework for better strategic decision-making. Therefore, this study will use a comprehensive listing of reviews from IMdb users as it has been an effective precedent in research and reviews can be used as an indicative KPI towards A24 products.

3.0 Method

3.1 Data Collection and Pre-processing

This study applies *BeautifulSoup* to scrape and collect data from the A24 film profile on *IMDb* and parse information about A24 films released between 2019 – 2022 (Appendix II, p. 2). By reading HTML tags, the hyperlink, rating, and information for every film were scraped from the webpage. Following this, a deeper analysis was conducted by scraping customer reviews for every film by altering the hyperlink address to the corresponding review webpage. Mitchell's (2018) framework for extracting reviews, involved creating a loop for every film in the list, that utilised web drivers from the *Selenium* library to create an automated environment to successfully load more reviews and inspect 'review-containers.' Where *BeautifulSoup* and the *Scrapy* libraries could extract the relevant information. The process of scraping was appended to a raw dataset file named "A24_reviews.csv" which consisted of 1,766 unique reviews from 40 films, Diagram 1.

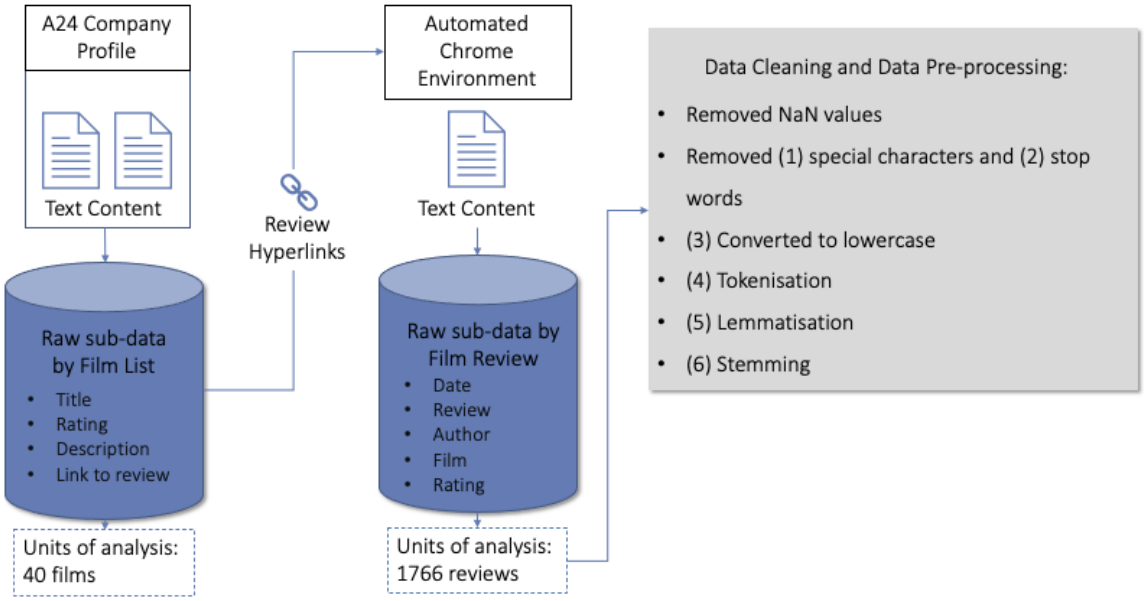


*Diagram 1: Data Collection and Processing**

**Please see Appendix II, p. 2-10 attached for more information*

Data cleaning entailed nulling '78' reviews for descriptive statistics. Likewise, columns like 'ratings' and 'duration' were converted to integers from None-Type formats. For text analysis, words were parsed into their purest forms by six tasks; (1) special characters of "[~.,%/:;?_&+*=!-]" and (2) stop words were removed, (3) converted to lower case and (4) tokenisation, (5) lemmatisation and (6) stemming was performed. The *Sklearn TF-IDFVectorizer* identified the most frequent unigrams which are then extracted to a term-frequency matrix (bag of words).

3.2 Sentiment Analysis

Sentiment analysis is used to categorise reviews into a binary variable compared to the IMDB scale. Prebuilt sentiment analysis models can be effective in different applications, dependent on the duration of training the model and batches of data that have been used to train it (Wang *et al.*, 2019; Liu *et al.*, 2019). With this consideration, to enrich the analysis, two sentiment models are incorporated to determine the best application for IMdb reviews. VADER from *NLTK* and a transformer called RoBERTa from *Hugging Face* are used to create sentiment scores for every review (See Appendix I). To test their effectiveness, an unsupervised Naïve Bayes classification, using a logistic regression was trained against 30% of the TF-IDF corpus. The classifier is plotted below with the results of VADER and RoBERTa analysis. The errors: false negatives of each model; discrepancies when determining a binary sentiment value, can be observed along the decision boundary.

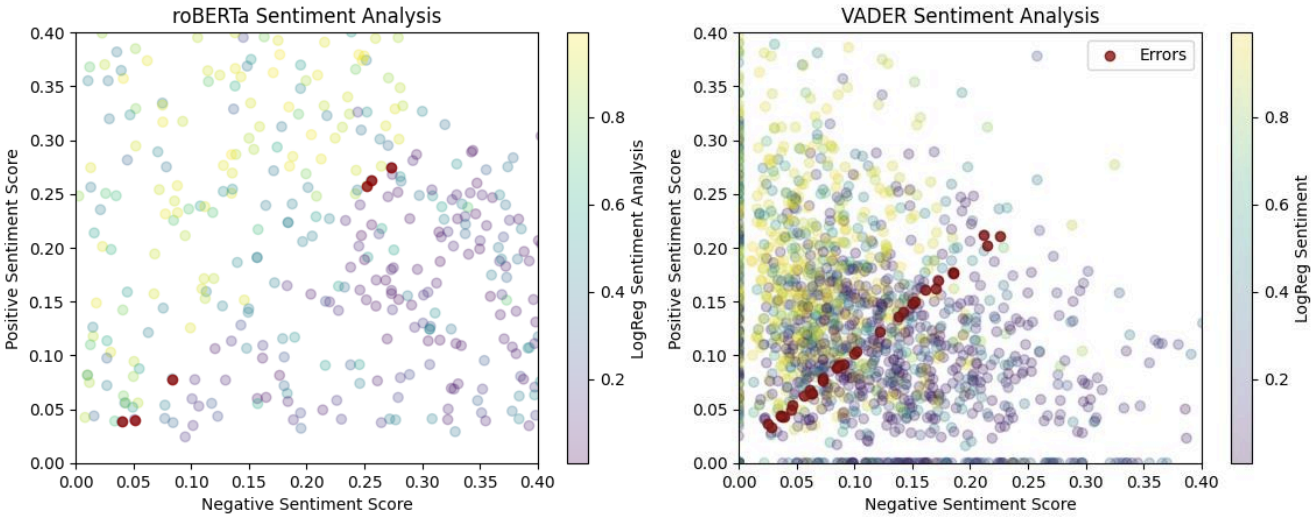


Figure 1: Sentiment Analysis machines plotted against Logistic Regression

Figure 1 can be observed are used to find differences in negative and positive ratings. VADER has more false values than RoBERTa and RoBERTa exhibits confidence in separating positives and negatives, compared to VADER which is more randomly dispersed and grouped. From this analysis, RoBERTa classifications are more precise and appear to perform better when allocating higher ratings with positive sentiment as it possesses less dispersion and fewer errors. Therefore, the following averages from the sentiment analysis in Figure 1 are implemented from the RoBERTA model with their respective 25th and 75th percentiles. Interestingly, there is a higher mean value of negative (M=0.38), with the second lowest standard deviation (STD=0.36) compared to positive sentiment (M=0.35), with the lowest variance (STD=0.36). However, both are stronger than neutral (M=0.26), with the highest variance (STD=0.21), in Figure 2.

Figure 2: Sentiment Analysis of A24 Films

Before introducing further analysis, a word cloud is generated below from the TF-IDF scores as a preliminary indicator of frequent terms. Negative and positive sentiment word clouds were generated from both the negative and positive scores. However, there was negligible difference between them. Figure 2 shows the overall frequency of qualitative terms with the summarisation of words with more frequent words appearing larger. Observing the words located in the diagram it can be found that there are very popular terms like ‘movi’, ‘like’, ‘film’; followed by positive phrases ‘much’, love’, ‘great’; with subsequent negative terms like ‘bore’ ‘noth’ ‘bad’. To create accurate topics and clustering for LDA, the popular tokens were removed from the analysis as stop words.

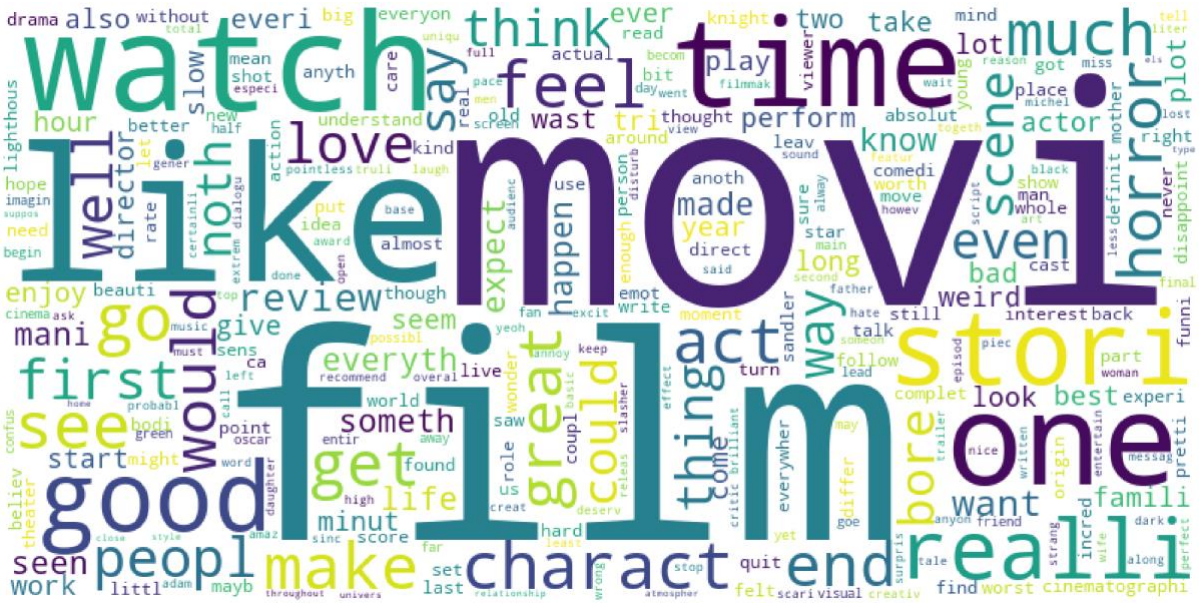


Figure 3: Word Cloud of TF-IDF values

3.4 Topic Modelling and Hierarchical Clustering

This report first employs Latent Dirichlet Allocation (LDA), which is an approach that reveals hidden semantic structures in a collection of text documents (Blei, Ng and Jordan, 2002). For this analysis clustering analysis is used to extract latent topics in the review narratives to understand better what is positive and negative about the films and give recommendations. Once reading research by (Bastani, Namavari and Shaffer, 2019; Mehrotra *et al.*, 2013) it was proven to be successful in identifying topics in lengthy text corpus like reviews, compared to smaller texts like *Twitter* tweets.

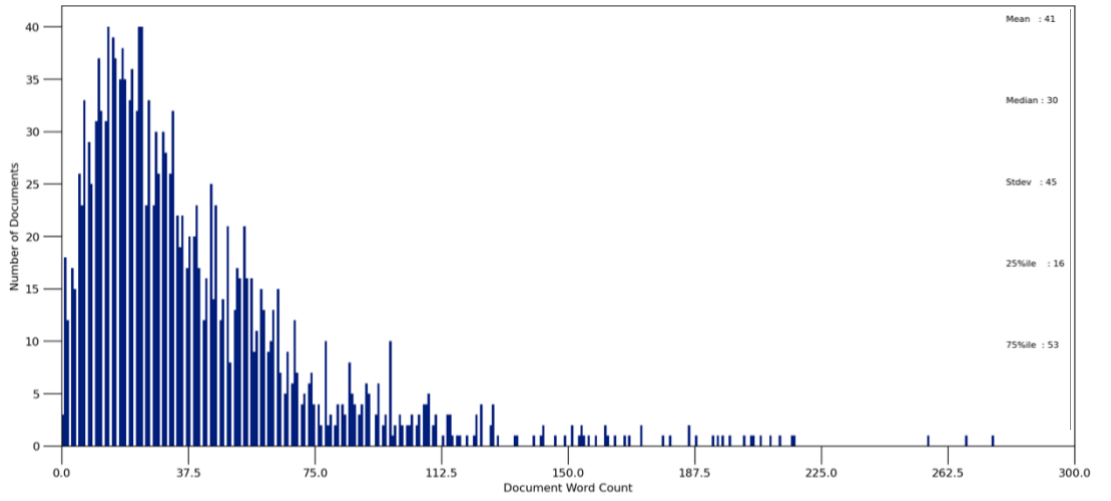


Figure 4: Distribution of Document Word Count

When preparing the data for sentiment analysis 10 lengthy reviews were removed (reviews of around > 215 words in length) that resulted in a RunTime error when executing (Appendix II). This is important to note for LDA, as it is effective with larger bodies of text. In comparison with the rest of the document of text, the average length for reviews can be observed in Figure 4. As mentioned for better performance, non-topic distinguishing words, like film titles and actors and stop words, were removed from the data. In the tradition of (Bastani, Namavari and Shaffer, 2019) the number of clusters for input was determined as K=6 by trial and error, as it provided the most meaningful topics, compared to the traditional Elbow method (Appendix II, p. 24) which generated overlapping and similar clusters.

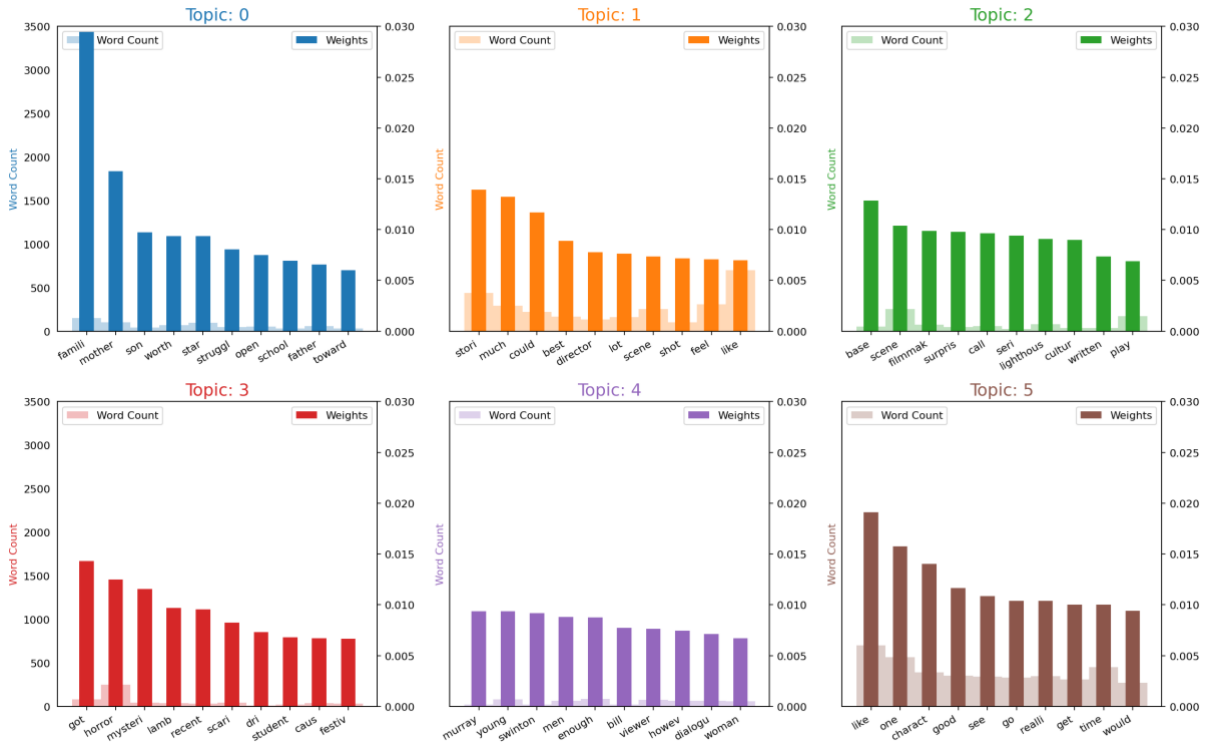


Figure 5: Word Count and Importance of Topic Keywords

Each topic in the narrative has relevant words that are highlighted in the corresponding colour to their associated topics in Figure 5. The largest topic cluster Topic 5 relates to customers liking characters, and stories and watching the films. Topic 3 explicitly denotes the appraisal of the horror genre and art house style in terms of ‘scari’, ‘mysteri’, and ‘atmosphere’. It is interesting to observe that the word ‘festiv’ is clustered with these words. This indicates that films in this genre were viewed or known for significant film festivals. Topic 0 appears to infer that words like ‘star’ relate to family-orientated words. Meaning films with ‘stars’ or

familiar actors appeal to family demographics. Where Topic 1 and Topic 2 are similar in their subtext, where there are praises for the director, stories, writing and cinematography. Topic 4 displays some negative words like 'however' and 'enough' but it is difficult to interpret the context. For consistency, a hierarchy of clusters can be constructed by using words that tended to appear together with the TF-IDF values to further develop these negative clusters. Second-order clustering is employed for its ability to group synonyms into the same cluster (Lee *et al.*, 2019).

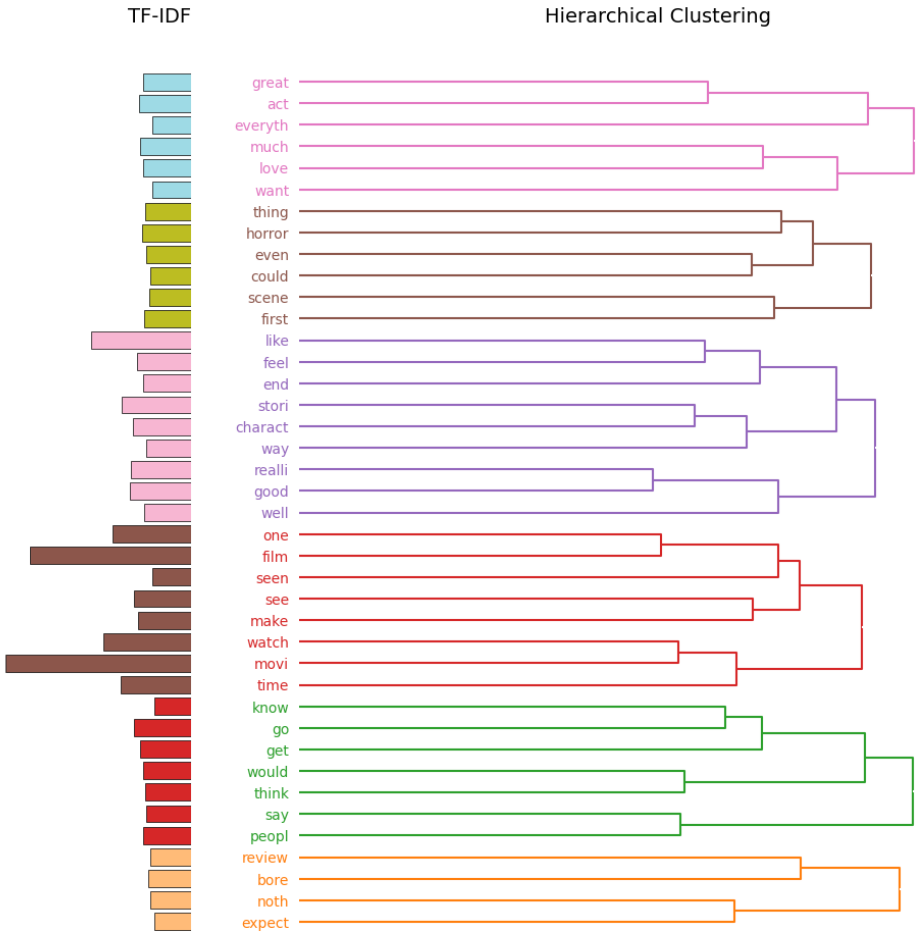


Figure 6: TF-IDF Hierarchical Clustering Dendogram

From the 40 words generated into six clusters in Figure 6, there is still little negative feedback. Analysis confirms the existence of Topic 0 in that people like the film and manages to cluster words that were loosely categorised in LDA. The first cluster indicates that reviewers 'love' the acting. The fourth cluster exhibits the highest TF-IDF in the dendogram, there is a recommendation from the reviewer that they 'think,' 'people' should 'see' films. The use of the terms 'one' 'film' and 'seen' indicates that viewers have seen at least one of the films before in terms of A24 films characteristics e.g. same director, actors. There is also

a negative prevalence of the word ‘bore’, ‘noth’ and ‘expect’ that indicates the films were not what they were expecting. While this clustering analysis confirms the connection of ‘scene’(s) in ‘horror’.

4.0 Managerial Policies

Firstly, satisfaction with A24 films occurs with reviewers who like how the films are shot, which includes the cinematography, the dialogue, and the narrative. A24 films are made in an art style as there is a high word frequency of terms like filmmaking, scene, and play, which has created an individualistic style. This was argued by Hennig-Thurau, Ravid and Sorenson (2021) to create value for films. In the reviews it was also inferred that certain recommendations that relate to the frequency of words are indicative of reviewers recommending the films to peers, this fulfils the e-WOM marketing that is outlined as one of the marketing methods used for independent film marketing.

Conversely, A24 horror films are a double-edged sword. Horror film distinctive terms were integrated into topics with Film Festivals and were popular among the frequency of words. Which suggests the reliance on the marketing of A24 Film festivals. The film industry is rapidly developing so a company’s present and future should be accounted for (Vitkauskaitė, 2017). A24 can expand its reach beyond traditional film festivals by exploring digital channels such as digital film festivals and streaming platforms. This not only shifts the risk of failure but also allows for potential success to be shared with the platform (Hennig-Thurau, Ravid and Sorenson, 2021). Making horror films because they are performing well may not be ideal for the development of A24 in the future. Going forward, A24 films should appeal to more family-orientated audiences. The hierarchical clustering analysis recommends having at least one famous actor to draw exposure and Topic 0 involves numerous family-related words which can aid viewers to see more than one film. Subsequently, conventional genres may undermine A24's potential. Genre overlapping, insofar defined as a film consisting of multiple genres opens opportunities, appeals to different spectators beyond just the Horror genre and is proven to boost box office numbers (Sim, 2023; Hennig-Thurau, Ravid and Sorenson, 2021).

5.0 Limitations

Despite the innate limitation of irony and sarcasm mistakes in sentiment analysis and topic modelling methods. Regarding this, there are 4 main limitations of this study:

Box Office Numbers

Box office numbers could not be scraped from IMdb due to the terms and conditions of the website. This is a limiting factor to the study as it is one of the industry's main KPIs and could have been used to recommend a budget for films.

Disconnect with films and the brand.

There is a disconnect between A24 films and its brand. This disconnection is related to the inherent nature of creative work as any form of creative work can be perceived as being idiosyncratic of a director rather than a product of A24 (Christopherson, 2008)

LDA is less effective with smaller texts.

75% of the data is above 16 words (mean (41) and Median (30)). The length of the texts is probably not large enough to be analysed by LDA (Figure 4). Consequently, the narratives with a total number of words less than 30 may have distorted the analysis (Bastani, Namavari and Shaffer, 2019).

Negativity Bias

According to negativity bias in Wu (2013), with review narratives, negative sentiment tends to hold more weight and attention than positive ones, even if a movie has received an equal amount of both (Kim, Trimi and Lee, 2021). There is also a greater propensity to review a film if it was below expectations.

As mentioned in the literature review, to gain meaningful insights performance needs to compare to industry benchmarks. It would also be interesting to see how films perform in a time series and look at some sort of survival analysis like Kim, Trimi and Lee (2021).

6.0 Conclusion

In response to the findings, the objective of this study was designed to find sentiment and topics in IMdb reviews of A24 films that can help suggest strategic managerial decisions. By comparing two sentiment analysis tools compared with Naïve Bayes, the sentiment towards A24 was observed as being more negative than positive. However, the KPIs from this study

that proxy box office numbers through behavioural qualities of Audience Engagement, Customer Satisfaction, and Customer Reaction could not ascertain why. Although, website analytical data methods have proven to be efficient and sophisticated in analysing the production and distribution stages of new films. The methods of this research can be also used to predict how future films will perform and provide recommendations. The future for this field of research should encompass a boarder dataset over a time scale to determine how film reviews fair over time to achieve this.

References

Bastani, K., Namavari, H. and Shaffer, J. (2019) 'Latent Dirichlet allocation (LDA) for topic modeling of the CFPB consumer complaints', *Expert Systems with Applications,* Volume 127.

Blei, D. M., Ng, A. Y. and Jordan, M. I. (2002) 'Latent Dirichlet Allocation', *Journal of Machine Research*.

Briggs, R. D. (2022) 'A Tale of Two Indies: Amazon Studios and A24 in the Streaming Age', *The Velvet Light Trap,* Number 90, pp. pp. 3-16.

Carl, W. J. (2006) 'What's all the Buzz About? Everyday Communication and the Relational Basis of Word-of-Mouth and Buzz Marketing Practices', *Northeastern University,* Vol. 19, No 4.

Christoforou, C., Papadopoulos, T. C., Constantinidou, F. and Theodorou, M. (2017) 'Your Brain on the Movies: A Computational Approach for Predicting Box-office Performance from Viewer's Brain Responses to Movie Trailers. *Frontiers in Neuroinformatics,* 11.

Christopherson, S. (2008) 'Beyond the Self-expressive Creative Worker', *Theory, Culture & Society,,* 25(7-8), pp. 73–95.

Hennig-Thurau, T., Ravid, S. A. and Sorenson, O. (2021) 'The Economics of Filmed Entertainment in the Digital Era', *Journal of Cultural Economics,* 45(2), pp. 157-170.

Kim, A., Trimi, S. and Lee, S.-G. (2021) 'Exploring the key success factors of films: a survival analysis approach. ', *Serv Bus.*

Landry, P. (2018) 'The Business of Film: A Practical Introduction', *Routledge*.

Lee, C. K. H., Tse, Y. K., Zhang, M. and Ma, J. (2019) 'Analysing online reviews to investigate customer behaviour in the sharing economy: the case of Airbnb', *Information Technology & People,* 33(3), pp. 945-961.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019) 'RoBERTa: A Robustly Optimized BERT Pretraining Approach.'.

Mehrotra, R., Sanner, S., Buntime, W. and Xie, L. (2013) 'Improving LDA topic models for microblogs via tweet pooling and automatic labeling', *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*.

Mitchell, R. (2018) *Web scraping with Python: Collecting more data from the modern web.*: O'Reilly Media, Inc.

Pentheny, J. R. (2015) 'The Influence of Movie Reviews on Consumers', *University of New Hampshire Scholars' Repository,* Honors Theses and Capstones. 265.

Rai, R. (2012) 'IDENTIFYING KEY PRODUCT ATTRIBUTES AND THEIR IMPORTANCE LEVELS FROM ONLINE CUSTOMER REVIEWS', *Department of Mechanical and Aerospace Engineering,* University at Buffalo.

Sim, G. (2023) 'The Idea of Genre in the Algorithmic Cinema. Television & New Media', *0(0)*.

Vitkauskaitė, I. (2017) 'Strategic Management of Independent Film Production Companies', *Chinese Business Review,* 16(6).

Waldfogel, J. (2017) 'How Digitization Has Created a Golden Age of Music, Movies, Books, and Television', *Journal of Economic Perspectives,* Volume 31, Number 3.

Wang, H., Gan, Z., Liu, X., Liu, J., Gao, J. and Wang, H. (2019) 'Adversarial Domain Adaptation for Machine Reading Comprehension ', *Association for Computational Linguistics*, pp. 2510–2520.

Wu, P. F. (2013) 'In Search of Negativity Bias: An Empirical Study of Perceived Helpfulness of Online Reviews', *Psychol. Mark.,* 30, pp. 971-984.

Appendix I – Abbreviations and Sentiment source links

VADER	- Valence Aware Dictionary for Sentiment Reasoning Found from: https://www.nltk.org/_modules/nltk/sentiment/vader.html
RoBERTA	- Bidirectional Encoder Representations from Transformers (BERT) - Robustly Optimized BERT Pre-training Found from: https://huggingface.co/docs/transformers/model_doc/roberta

Appendix II – Code

Importation and Installation of Libraries

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import os
import nltk
import numpy as np
from scrapy.selector import Selector
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time
from tqdm import tqdm
import warnings
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
from wordcloud import WordCloud, STOPWORDS
import nltk
from textblob import TextBlob, Word, Blobber
from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from scipy.special import softmax
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score, confusion_matrix,
from sklearn import metrics
from sklearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import IncrementalPCA
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import CountVectorizer

vec = CountVectorizer(ngram_range=(2,2), max_features=500000)
lmtsr = WordNetLemmatizer()

import sklearn.linear_model

import json
import glob

# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# spacy
import spacy
from nltk.corpus import stopwords

# vis
import pyLDAvis
import pyLDAvis.gensim

import ast

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

nltk.download('vader_lexicon')
warnings.filterwarnings("ignore")
print('Set up complete...')
```

```
/Users/matthewbaber/Library/Python/3.9/lib/python/site-packages/gensim/similarities/__init__.py:15: UserW
arning: The gensim.similarities.levenshtein submodule is disabled, because the optional Levenshtein packa
ge <https://pypi.org/project/python-Levenshtein/> is unavailable. Install Levenhstein (e.g. `pip install
python-Levenshtein`) to suppress this warning.
  warnings.warn(msg)
Set up complete...
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/matthewbaber/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Data Scraping

In [ ]:

```
import requests
from bs4 import BeautifulSoup
from time import sleep
from random import randint

HEADERS = {'User-Agent': 'Mozilla/5.0 (iPad; CPU OS 12_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, li

main = "https://www.imdb.com/search/title/?companies=co0390816"
page = requests.get(main, headers=HEADERS)

sleep(randint(8,15))

soup = BeautifulSoup(page.content, "html.parser")

urls = ["https://www.imdb.com/search/title/?companies=co0390816",    "https://www.imdb.com/search/title/

published_dates = []
titles = []
links = []
certs = []
genres = []
descriptions = []
ratings = []

for url in urls:
    page = requests.get(url, headers=HEADERS)

    sleep(randint(8, 15))

    soup = BeautifulSoup(page.content, "html.parser")


    movie_container = soup.find_all('div', class_='lister-item mode-advanced')

    for container in movie_container:
        published_date = container.find("span", class_="lister-item-year text-muted unbold").text.strip(
        if '(2019)' in published_date or '(2020)' in published_date or '(2021)' in published_date or '(2
            published_dates.append(published_date)

            title = container.find("h3", class_="lister-item-header").a.text.strip()
            titles.append(title)

            link = container.find("h3", class_="lister-item-header").a['href']
            links.append(link)

            certificate = container.p.find('span', class_ = 'certificate')
            certs.append(certificate)

            genre = container.p.find('span', class_ = 'genre').text.replace("\n", "").rstrip().split(',
            genres.append(genre)

            description = container.find("p", class_="text-muted").text.replace('|', '').replace('/n',
            descriptions.append(description)

            rating = container.find("div", class_="ratings-bar").find("strong").text.strip()
            rating_value = round(float(rating), 1)
            ratings.append(rating)

filmlist = pd.DataFrame({'title': titles,
                         'link': ['https://www.imdb.com' + link for link in links],
                         'certificate': certs,
                         'genre': genres,
                         'description': descriptions,
                         'rating': ratings,
                         'published_date': published_dates})
```

```
filmlist['review_link'] = filmlist['link'].apply(lambda x: x + 'reviews')
filmlist['published_date'] = filmlist['published_date'].apply(lambda x: x.replace('(I)', '').replace('(
filmlist['description'] = filmlist['description'].apply(lambda x: x.replace('\n', ''))


filmlist.to_csv('filmlist.csv', index=False)
```

Out[ ]:
```
'\n\nHEADERS = {\'User-Agent\': \'Mozilla/5.0 (iPad; CPU OS 12_2 like Mac OS X) AppleWebKit/605.1.15 (K
HTML, like Gecko) Mobile/15E148\'}\n\nmain = "https://www.imdb.com/search/title/?companies=co0390816"\n
page = requests.get(main, headers=HEADERS)\n\nsleep(randint(8,15))\n\nsoup = BeautifulSoup(page.conten
t, "html.parser")\n\nHEADERS = {\n    \'User-Agent\': \'Mozilla/5.0 (Windows NT 10.0; Win64; x64) Apple
WebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3\'}\n\nurls = ["https://www.imdb.co
m/search/title/?companies=co0390816",    "https://www.imdb.com/search/title/?companies=co0390816&start=
51&ref_=adv_nxt"]\n\npublished_dates = []\ntitles = []\nlinks = []\ncerts = []\ngenres = []\ndescriptio
ns = []\nratings = []\n\nfor url in urls:\n    page = requests.get(url, headers=HEADERS)\n\n    sleep(r
andint(8, 15))\n\n    soup = BeautifulSoup(page.content, "html.parser")\n\n\n    movie_container = sou
p.find_all(\'div\', class_=\'lister-item mode-advanced\')\n    \n    for container in movie_containe
r:\n        published_date = container.find("span", class_="lister-item-year text-muted unbold").text.s
trip()\n        if \'(2019)\' in published_date or \'(2020)\' in published_date or \'(2021)\' in publis
hed_date or \'(2022)\' in published_date:\n            published_dates.append(published_date)\n\n
title = container.find("h3", class_="lister-item-header").a.text.strip()\n            titles.append(tit
le)\n\n            link = container.find("h3", class_="lister-item-header").a[\'href\']\n            li
nks.append(link)\n\n            certificate = container.p.find(\'span\', class_ = \'certificate\')\n
certs.append(certificate)\n\n            genre = container.p.find(\'span\', class_ = \'genre\').text.re
place("\n", "").rstrip().split(\',\')\n            genres.append(genre)\n\n            description = co
ntainer.find("p", class_="text-muted").text.replace(\'|\', \'\').replace(\'/n\', \'\').replace(\',\',
\'\').strip()\n            descriptions.append(description)\n\n            rating = container.find("di
v", class_="ratings-bar").find("strong").text.strip()\n            rating_value = round(float(rating),
1)\n            ratings.append(rating)\n\nfilmlist = pd.DataFrame({\'title\': titles,\n
\'link\': [\'https://www.imdb.com\' + link for link in links],\n                        \'certificate
\': certs,\n                        \'genre\': genres,\n                        \'description\': desc
riptions,\n                        \'rating\': ratings,\n                        \'published_date\':
published_dates})\n\nfilmlist[\'review_link\'] = filmlist[\'link\'].apply(lambda x: x + \'reviews\')\nf
ilmlist[\'published_date\'] = filmlist[\'published_date\'].apply(lambda x: x.replace(\'(I)\', \'\').rep
lace(\'(II)\', \'\'))\nfilmlist[\'description\'] = filmlist[\'description\'].apply(lambda x: x.replace
(\'\n\', \'\'))\n'
```

In [ ]:
```
fl = pd.read_csv('filmlist.csv')
fl['review_link'] = fl['review_link'].apply(lambda x: x.replace('?ref_=adv_li_tt', ''))
fl.head()
```

Out[ ]:

	title	link	certificate	genre	description	rating	pu
0	Everything Everywhere All at Once	https://www.imdb.com/title/tt6710474/?ref_=adv...	15	['Action', ' Adventure', ' Comedy']	15139 minAction Adventure Comedy	7.9	
1	The Whale	https://www.imdb.com/title/tt13833688/?ref_=ad...	15	['Drama']	15117 minDrama	7.7	
2	X	https://www.imdb.com/title/tt13560574/?ref_=ad...	18	['Horror', ' Mystery', ' Thriller']	18105 minHorror Mystery Thriller	6.6	
3	Midsommar	https://www.imdb.com/title/tt8772262/?ref_=adv...	18	['Drama', ' Horror', ' Mystery']	18148 minDrama Horror Mystery	7.1	
4	Aftersun	https://www.imdb.com/title/tt19770238/?ref_=ad...	12A	['Drama']	12A102 minDrama	7.7	

In [ ]:
```
data = pd.DataFrame(columns=['review_date', 'author', 'rating', 'review_title', 'review', 'review_url',

for url in fl['review_link']:
    page = requests.get(url)

    soup = BeautifulSoup(page.content, "html.parser")

    film_title = soup.title.string.split(' - ')[0]

    review_titles = soup.find_all('a', class_='title')

    #//

    driver = webdriver.Chrome('chromedriver.exe')
    time.sleep(1)
    driver.get(url)
    time.sleep(1)
```

```
print(driver.title)
time.sleep(1)
body = driver.find_element(By.CSS_SELECTOR, 'body')
body.send_keys(Keys.PAGE_DOWN)
time.sleep(1)
body.send_keys(Keys.PAGE_DOWN)
time.sleep(1)
body.send_keys(Keys.PAGE_DOWN)

#//

sel = Selector(text = driver.page_source)
review_counts = sel.css('.lister .header span::text').extract_first().replace(',','').split(' ')[0]
more_review_pages = int(int(review_counts)/25)

#//

for i in tqdm(range(more_review_pages)):
    try:
        css_selector = 'load-more-trigger'
        driver.find_element(By.ID, css_selector).click()
    except:
        pass

#//

rating_list = []
review_date_list = []
review_title_list = []
author_list = []
review_list = []
review_url_list = []
error_url_list = []
error_msg_list = []
reviews = driver.find_elements(By.CSS_SELECTOR, 'div.review-container')

for d in tqdm(reviews):
    try:
        sel2 = Selector(text = d.get_attribute('innerHTML'))
        try:
            rating = sel2.css('.rating-other-user-rating span::text').extract_first()
        except:
            rating = np.NaN
        try:
            review = sel2.css('.text.show-more__control::text').extract_first()
        except:
            review = np.NaN
        try:
            review_date = sel2.css('.review-date::text').extract_first()
        except:
            review_date = np.NaN
        try:
            author = sel2.css('.display-name-link a::text').extract_first()
        except:
            author = np.NaN
        try:
            review_title = sel2.css('a.title::text').extract_first()
        except:
            review_title = np.NaN
        try:
            review_url = sel2.css('a.title::attr(href)').extract_first()
        except:
            review_url = np.NaN


        rating_list.append(rating)
        review_date_list.append(review_date)
        review_title_list.append(review_title)
        author_list.append(author)
        review_list.append(review)
        review_url_list.append(review_url)
        film_title.append(film_title)
    except Exception as e:
        error_url_list.append(url)
        error_msg_list.append(e)

reviews = pd.DataFrame({
    'review_date':review_date_list,
    'author':author_list,
    'rating':rating_list,
    'review_title':review_title_list,
```

```
        'review':review_list,
        'review_url':review_url,
        'film_title':(film_title)
        })

    data = data.append(reviews, ignore_index=True)
    data.to_csv('data.csv', index=False)
```

```
Everything Everywhere All at Once (2022) - Everything Everywhere All at Once (2022) - User Reviews - IMDb
100%|██████████| 135/135 [00:16<00:00,  8.20it/s]
100%|██████████| 275/275 [00:08<00:00, 33.57it/s]
The Whale (2022) - The Whale (2022) - User Reviews - IMDb
100%|██████████| 28/28 [00:01<00:00, 18.21it/s]
100%|██████████| 50/50 [00:01<00:00, 35.62it/s]
X (2022) - X (2022) - User Reviews - IMDb
100%|██████████| 33/33 [00:02<00:00, 12.29it/s]
100%|██████████| 75/75 [00:02<00:00, 33.27it/s]
Midsommar (2019) - Midsommar (2019) - User Reviews - IMDb
100%|██████████| 154/154 [00:10<00:00, 15.18it/s]
100%|██████████| 175/175 [00:04<00:00, 38.27it/s]
Aftersun (2022) - Aftersun (2022) - User Reviews - IMDb
100%|██████████| 10/10 [00:00<00:00, 19.83it/s]
100%|██████████| 25/25 [00:00<00:00, 25.82it/s]
Pearl (2022) - Pearl (2022) - User Reviews - IMDb
100%|██████████| 12/12 [00:00<00:00, 21.18it/s]
100%|██████████| 25/25 [00:00<00:00, 28.78it/s]
Bodies Bodies Bodies (2022) - Bodies Bodies Bodies (2022) - User Reviews - IMDb
100%|██████████| 12/12 [00:01<00:00,  9.45it/s]
100%|██████████| 50/50 [00:01<00:00, 33.10it/s]
Close (2022) - Close (2022) - User Reviews - IMDb
100%|██████████| 3/3 [00:00<00:00,  5.24it/s]
100%|██████████| 25/25 [00:01<00:00, 15.01it/s]
Uncut Gems (2019) - Uncut Gems (2019) - User Reviews - IMDb
100%|██████████| 102/102 [00:04<00:00, 23.38it/s]
100%|██████████| 100/100 [00:02<00:00, 42.58it/s]
White Noise (2022) - White Noise (2022) - User Reviews - IMDb
100%|██████████| 16/16 [00:00<00:00, 25.80it/s]
100%|██████████| 25/25 [00:01<00:00, 24.59it/s]
The Green Knight (2021) - The Green Knight (2021) - User Reviews - IMDb
100%|██████████| 75/75 [00:03<00:00, 24.48it/s]
100%|██████████| 100/100 [00:02<00:00, 46.04it/s]
Men (2022) - Men (2022) - User Reviews - IMDb
100%|██████████| 25/25 [00:01<00:00, 14.74it/s]
100%|██████████| 50/50 [00:01<00:00, 30.37it/s]
Showing Up (2022) - Showing Up (2022) - User Reviews - IMDb
0it [00:00, ?it/s]
0it [00:00, ?it/s]
The Lighthouse (2019) - The Lighthouse (2019) - User Reviews - IMDb
100%|██████████| 62/62 [00:05<00:00, 10.42it/s]
100%|██████████| 125/125 [00:03<00:00, 41.39it/s]
Marcel the Shell with Shoes On (2021) - Marcel the Shell with Shoes On (2021) - User Reviews - IMDb
100%|██████████| 4/4 [00:00<00:00, 13.23it/s]
100%|██████████| 25/25 [00:01<00:00, 22.68it/s]
Causeway (2022) - Causeway (2022) - User Reviews - IMDb
100%|██████████| 4/4 [00:00<00:00,  8.04it/s]
100%|██████████| 25/25 [00:00<00:00, 26.75it/s]
God's Creatures (2022) - God's Creatures (2022) - User Reviews - IMDb
100%|██████████| 1/1 [00:00<00:00,  6.23it/s]
100%|██████████| 25/25 [00:00<00:00, 38.56it/s]
Minari (2020) - Minari (2020) - User Reviews - IMDb
100%|██████████| 17/17 [00:00<00:00, 24.26it/s]
100%|██████████| 25/25 [00:01<00:00, 23.13it/s]
The Tragedy of Macbeth (2021) - The Tragedy of Macbeth (2021) - User Reviews - IMDb
100%|██████████| 12/12 [00:00<00:00, 18.75it/s]
100%|██████████| 25/25 [00:01<00:00, 19.55it/s]
Red Rocket (2021) - Red Rocket (2021) - User Reviews - IMDb
100%|██████████| 5/5 [00:00<00:00, 11.46it/s]
100%|██████████| 25/25 [00:01<00:00, 21.49it/s]
Irma Vep (TV Series 2022) - Irma Vep (TV Series 2022) - User Reviews - IMDb
100%|██████████| 4/4 [00:00<00:00,  8.02it/s]
100%|██████████| 25/25 [00:01<00:00, 17.47it/s]
Lamb (2021) - Lamb (2021) - User Reviews - IMDb
100%|██████████| 12/12 [00:01<00:00,  9.06it/s]
100%|██████████| 50/50 [00:01<00:00, 36.14it/s]
Talk to Me (2022) - Talk to Me (2022) - User Reviews - IMDb
```

```
0it [00:00, ?it/s]
100%|██████████| 6/6 [00:00<00:00, 21.88it/s]
Waves (2019) - Waves (2019) - User Reviews - IMDb
100%|██████████| 8/8 [00:00<00:00, 15.20it/s]
100%|██████████| 25/25 [00:00<00:00, 28.21it/s]
After Yang (2021) - After Yang (2021) - User Reviews - IMDb
100%|██████████| 5/5 [00:00<00:00, 11.10it/s]
100%|██████████| 25/25 [00:01<00:00, 21.70it/s]
Saint Maud (2019) - Saint Maud (2019) - User Reviews - IMDb
100%|██████████| 18/18 [00:00<00:00, 22.64it/s]
100%|██████████| 25/25 [00:01<00:00, 22.37it/s]
C'mon C'mon (2021) - C'mon C'mon (2021) - User Reviews - IMDb
100%|██████████| 6/6 [00:00<00:00, 12.20it/s]
100%|██████████| 25/25 [00:01<00:00, 23.27it/s]
Zola (2020) - Zola (2020) - User Reviews - IMDb
100%|██████████| 5/5 [00:00<00:00, 12.72it/s]
100%|██████████| 25/25 [00:01<00:00, 24.94it/s]
The Inspection (2022) - The Inspection (2022) - User Reviews - IMDb
0it [00:00, ?it/s]
100%|██████████| 15/15 [00:00<00:00, 35.23it/s]
Stars at Noon (2022) - Stars at Noon (2022) - User Reviews - IMDb
0it [00:00, ?it/s]
100%|██████████| 21/21 [00:00<00:00, 37.74it/s]
The Death of Dick Long (2019) - The Death of Dick Long (2019) - User Reviews - IMDb
100%|██████████| 2/2 [00:00<00:00,  6.43it/s]
100%|██████████| 25/25 [00:01<00:00, 23.62it/s]
The Eternal Daughter (2022) - The Eternal Daughter (2022) - User Reviews - IMDb
100%|██████████| 1/1 [00:00<00:00,  4.47it/s]
100%|██████████| 25/25 [00:00<00:00, 30.97it/s]
When You Finish Saving the World (2022) - When You Finish Saving the World (2022) - User Reviews - IMDb
0it [00:00, ?it/s]
100%|██████████| 24/24 [00:00<00:00, 38.48it/s]
First Cow (2019) - First Cow (2019) - User Reviews - IMDb
100%|██████████| 6/6 [00:00<00:00, 13.91it/s]
100%|██████████| 25/25 [00:00<00:00, 27.86it/s]
The Last Black Man in San Francisco (2019) - The Last Black Man in San Francisco (2019) - User Reviews - IMDb
100%|██████████| 6/6 [00:00<00:00, 14.24it/s]
100%|██████████| 25/25 [00:01<00:00, 22.19it/s]
The Hole in the Ground (2019) - The Hole in the Ground (2019) - User Reviews - IMDb
100%|██████████| 10/10 [00:00<00:00, 14.92it/s]
100%|██████████| 25/25 [00:01<00:00, 24.84it/s]
The Farewell (2019) - The Farewell (2019) - User Reviews - IMDb
100%|██████████| 13/13 [00:00<00:00, 16.73it/s]
100%|██████████| 25/25 [00:01<00:00, 19.67it/s]
The Humans (2021) - The Humans (2021) - User Reviews - IMDb
100%|██████████| 7/7 [00:00<00:00, 14.74it/s]
100%|██████████| 25/25 [00:01<00:00, 17.55it/s]
On the Rocks (2020) - On the Rocks (2020) - User Reviews - IMDb
100%|██████████| 13/13 [00:00<00:00, 30.49it/s]
100%|██████████| 25/25 [00:00<00:00, 29.70it/s]
Mr. Corman (TV Series 2021) - Mr. Corman (TV Series 2021) - User Reviews - IMDb
100%|██████████| 7/7 [00:00<00:00, 24.85it/s]
100%|██████████| 25/25 [00:00<00:00, 25.55it/s]
The Souvenir (2019) - The Souvenir (2019) - User Reviews - IMDb
100%|██████████| 6/6 [00:00<00:00, 21.81it/s]
100%|██████████| 25/25 [00:00<00:00, 34.95it/s]
```

In [ ]:
```
data = pd.read_csv('data.csv')
data.head()
```

Out[ ]:

	review_date	author	rating	review_title	review	review_url	film_title
0	2 May 2022	movieman_kev	9.0	best film of 2022\n	Profoundly deep, genuinely moving, utterly hil...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)
1	24 May 2022	evanston_dad	9.0	Felt Like I Was Seeing the Inside of My Own M...	I have trouble turning off my brain. Anxieties...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)
2	8 April 2022	AfricanBro	9.0	Don't do drugs, watch this instead.\n	If you take drugs for the first time and imagi...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)
3	20 April 2022	gbill-74877	10.0	Fantastic\n	"Be kind, especially when you don't know what'...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)
4	31 March 2022	benjaminskylerhill	10.0	The most original film ever made. Period.\n	Everything Everywhere All At Once is even craz...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)

Cleaning

In [ ]:
```
print(data.isna().sum())
print(fl.isna().sum())
```

```
review_date      0
author           0
rating          78
review_title     0
review           0
review_url       0
film_title       0
dtype: int64
title            0
link             0
certificate      0
genre            0
description      0
rating           0
published_date    0
review_link      0
dtype: int64
```

Pre-Process

In [ ]:
```
#preprocess steps and cleaning


#clean the text in review column by removing any [~.,%/:;?_&+*=!-]
def preprocess_text(text):
    cleaned = re.sub(r'[~.,%/:;?_&+*=!-]', '', text)
    return cleaned

def data_cleaning(df):

    df['review'] = df['review'].apply(preprocess_text)

    #tokenization
    from nltk import word_tokenize,sent_tokenize
    df['text_1'] = df['review'].str.lower().apply(word_tokenize)

    #stopwords removal
    stop_words = set(stopwords.words('english'))
    df['text_1'] = df['text_1'].apply(lambda x: [word for word in x if word.isalpha() and word not in st

    #lemmatization
    df["text_lmt"] = df["text_1"].apply(lambda x: [lmtsr.lemmatize(word) for word in x])

    #stemming
    ps = PorterStemmer()
    df["text_stm"] = df["text_1"].apply(lambda x: [ps.stem(word) for word in x])

    df['text_stm'] = df['text_stm'].astype(str)
```

```
    df[".len_text"] = df["text_lmt"].str.len()
    return df

cleaned_df = data_cleaning(data)
cleaned_df
```

Out[ ]:

	review_date	author	rating	review_title	review	review_url	film_title	text_
0	2 May 2022	movieman_kev	9.0	best film of 2022\n	Profoundly deep genuinely moving utterly hilar...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)	[profoundly deep genuinely moving utterly,..
1	24 May 2022	evanston_dad	9.0	Felt Like I Was Seeing the Inside of My Own M...	I have trouble turning off my brain Anxieties ...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)	[trouble turning, brain anxieties worries, ..
2	8 April 2022	AfricanBro	9.0	Don't do drugs, watch this instead.\n	If you take drugs for the first time and imagi...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)	[take, drugs first, time imagined jackie, c..
3	20 April 2022	gbill-74877	10.0	Fantastic\n	"Be kind especially when you don't know what's...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)	[kind especially know, going
4	31 March 2022	benjaminskylerhill	10.0	The most original film ever made. Period.\n	Everything Everywhere All At Once is even craz...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)	[everything everywhere even, crazier traile..
...	...	...	...	...	...	...	...	..
1761	24 May 2019	bastille-852-731547	6.0	Okay but Somewhat Disappointing Drama\n	As much as I enjoy the betterquality summer bl...	/review/rw5365158/?ref_=tt_urv	The Souvenir (2019)	[much, enjoy betterquality summer blockbust..
1762	8 June 2019	mamlukman	1.0	Warning: Do NOT see!\n	My wife and I see about 8090 movies (in theate...	/review/rw5365158/?ref_=tt_urv	The Souvenir (2019)	[wife, see movies theatersnot dvd, tv, ever..
1763	16 May 2019	ferguson-6	8.0	welcome another talented Swinton\n	Greetings again from the darkness Viewers of w...	/review/rw5365158/?ref_=tt_urv	The Souvenir (2019)	[greetings darkness viewers writerdirector,..
1764	22 February 2020	Go6565	1.0	Pointless\n	I wish I had read the user reviews (instead of...	/review/rw5365158/?ref_=tt_urv	The Souvenir (2019)	[wish, read user, reviews instead professio..
1765	29 December 2019	SnoopyStyle	6.0	slow big reveal steady decline\n	Julie (Honor Swinton Byrne) is a film student ...	/review/rw5365158/?ref_=tt_urv	The Souvenir (2019)	[julie, honor swinton, byrne film, student, ..

1766 rows × 11 columns

TF-IDF

In [ ]:
```
cleaned_df_gp  = cleaned_df.groupby('film_title').count()
cleaned_df_gp
```

Out[ ]:

film_title	review_date	author	rating	review_title	review	review_url	text_1	text_lmt	text_stm	.len_text
After Yang (2021)	25	25	24	25	25	25	25	25	25	25
Aftersun (2022)	25	25	25	25	25	25	25	25	25	25
Bodies Bodies Bodies (2022)	50	50	49	50	50	50	50	50	50	50
C'mon C'mon (2021)	25	25	24	25	25	25	25	25	25	25
Causeway (2022)	25	25	25	25	25	25	25	25	25	25
Close (2022)	25	25	25	25	25	25	25	25	25	25
Everything Everywhere All at Once (2022)	275	275	259	275	275	275	275	275	275	275
First Cow (2019)	25	25	21	25	25	25	25	25	25	25
God's Creatures (2022)	25	25	23	25	25	25	25	25	25	25
Irma Vep (TV Series 2022)	25	25	23	25	25	25	25	25	25	25
Lamb (2021)	50	50	48	50	50	50	50	50	50	50
Marcel the Shell with Shoes On (2021)	25	25	25	25	25	25	25	25	25	25
Men (2022)	50	50	49	50	50	50	50	50	50	50
Midsommar (2019)	175	175	170	175	175	175	175	175	175	175
Minari (2020)	25	25	24	25	25	25	25	25	25	25
Mr. Corman (TV Series 2021)	25	25	24	25	25	25	25	25	25	25
On the Rocks (2020)	25	25	23	25	25	25	25	25	25	25
Pearl (2022)	25	25	24	25	25	25	25	25	25	25
Red Rocket (2021)	25	25	25	25	25	25	25	25	25	25
Saint Maud (2019)	25	25	24	25	25	25	25	25	25	25
Stars at Noon (2022)	21	21	21	21	21	21	21	21	21	21
Talk to Me (2022)	6	6	6	6	6	6	6	6	6	6
The Death of Dick Long (2019)	25	25	24	25	25	25	25	25	25	25
The Eternal Daughter (2022)	25	25	25	25	25	25	25	25	25	25
The Farewell (2019)	25	25	23	25	25	25	25	25	25	25
The Green Knight (2021)	100	100	97	100	100	100	100	100	100	100
The Hole in the Ground (2019)	25	25	24	25	25	25	25	25	25	25
The Humans (2021)	25	25	24	25	25	25	25	25	25	25

film_title	review_date	author	rating	review_title	review	review_url	text_1	text_lmt	text_stm	.len_text
The Inspection (2022)	15	15	14	15	15	15	15	15	15	15
The Last Black Man in San Francisco (2019)	25	25	20	25	25	25	25	25	25	25
The Lighthouse (2019)	125	125	118	125	125	125	125	125	125	125
The Souvenir (2019)	25	25	23	25	25	25	25	25	25	25
The Tragedy of Macbeth (2021)	25	25	23	25	25	25	25	25	25	25
The Whale (2022)	50	50	46	50	50	50	50	50	50	50
Uncut Gems (2019)	100	100	95	100	100	100	100	100	100	100
Waves (2019)	25	25	24	25	25	25	25	25	25	25
When You Finish Saving the World (2022)	24	24	24	24	24	24	24	24	24	24
White Noise (2022)	25	25	25	25	25	25	25	25	25	25
X (2022)	75	75	74	75	75	75	75	75	75	75
Zola (2020)	25	25	24	25	25	25	25	25	25	25

In [ ]:
```
#bag of words and TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
bag_of_words = vectorizer.fit_transform(cleaned_df['text_stm'])

feature_names = vectorizer.get_feature_names_out()

# create a new dataframe with the bag of words representation
tf = pd.DataFrame(bag_of_words.toarray(), columns = vectorizer.get_feature_names_out())
tfidf_sum = tf.T.sum(axis=1).sort_values(ascending = False).reset_index().rename(columns={"index": "word
tfidf_transposed = tf.T

display(tf)
```

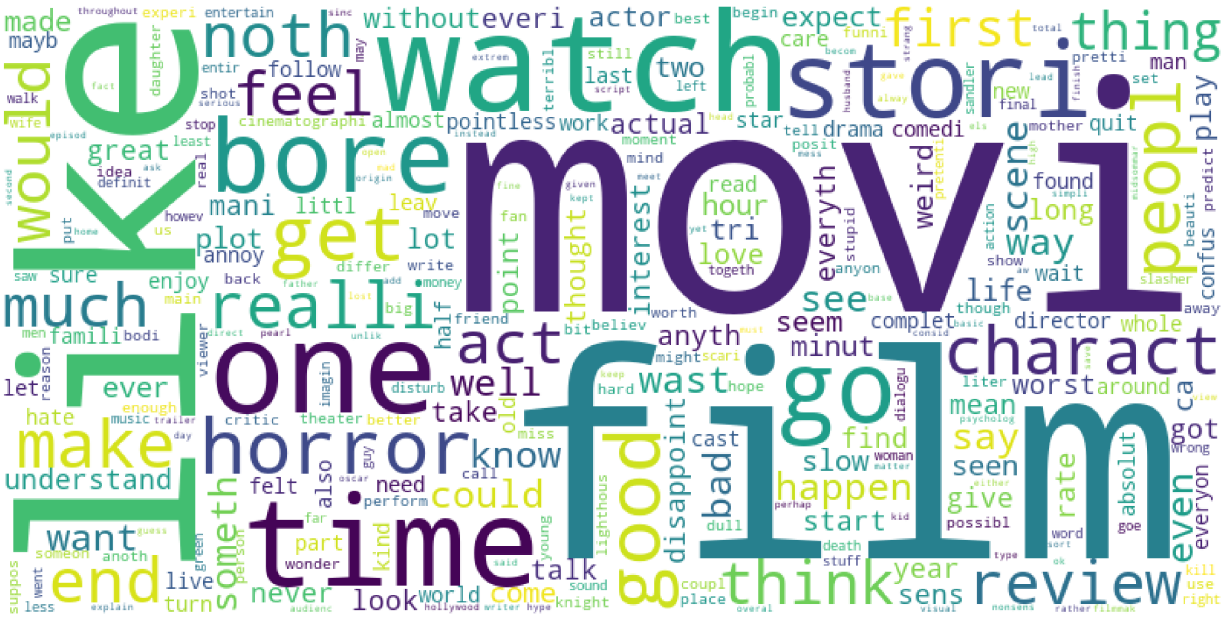
	aarrgg	ab	abandon	abba	abel	abil	abl	ablaz	abli	abnorm	...	zombi	zombieland	zone	zoo	zsolt	ätte
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1761	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1762	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1763	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1764	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1765	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	

1766 rows × 8130 columns

Sentiment Analysis

In [ ]:
```
sentiment_df = cleaned_df.copy()
#reset the index and call it id
sentiment_df.reset_index(inplace=True)
sentiment_df.rename(columns={'index': 'id'}, inplace=True)
```

In [ ]:
```
def wordcloud(df, max_words):
    wordcloud = WordCloud(width=800, height=400, background_color="white", max_words=max_words, random_s
    plt.figure(figsize = (30, 30), facecolor = 'white', edgecolor='blue')
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.show()

wordcloud(tf, 300)
```



Basic NLTK stuff

In [ ]:
```
sia = SentimentIntensityAnalyzer()
```

In [ ]:
```
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

In [ ]:
```
def polarity_scores_roberta(example):
    encoded_text = tokenizer(example, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
        'roberta_neg' : scores[0],
        'roberta_neu' : scores[1],
        'roberta_pos' : scores[2]
}
    return scores_dict
```

In [ ]:
```
res2 = {}
for i, row in tqdm(sentiment_df.iterrows(), total=len(sentiment_df)):
    try:
        text = row['review']
        myid = row['id']

        vader_result = sia.polarity_scores(text)
        vader_result_rename = {}
        for key, value in vader_result.items():
            vader_result_rename[f"vader_{key}"] = value

        roberta_results = polarity_scores_roberta(text)

        both = {**vader_result_rename, **roberta_results}
```

```
        res2[myid] = both
    except RuntimeError:
        print(f"RuntimeError on {myid}")
```

```
  0%|          | 0/1766 [00:00<?, ?it/s]
RuntimeError on 104
RuntimeError on 184
RuntimeError on 412
RuntimeError on 543
RuntimeError on 647
RuntimeError on 954
RuntimeError on 1051
RuntimeError on 1079
RuntimeError on 1095
RuntimeError on 1745
```

In [ ]:
```
query = sentiment_df.query('index == 104 or index == 184 or index == 412 or index == 543 or index == 647
query
```

Out[ ]:

	id	review_date	author	rating	review_title	review	review_url	film_title	te
104	104	18 September 2022	Pjtaylor-96-138044	8.0	Open your third (googly) eye...\n	Overwhelming in scale yet intimate in scope 'E...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)	[overwhelr scale intir scope,
184	184	12 March 2023	jboothmillard	8.0	Everything Everywhere All at Once\n	I knew this film was a big deal during its tim...	/review/rw8029183/?ref_=tt_urv	Everything Everywhere All at Once (2022)	[knew, big, deal, cinema, or
412	412	7 July 2019	Pjtaylor-96-138044	4.0	One bad trip.\n	'Midsommar (2019)' takes a very long time to g...	/review/rw5561163/?ref_=tt_urv	Midsommar (2019)	[takes, time, predict franl
543	543	14 July 2019	jboothmillard	5.0	Midsommar\n	This scary movie was becoming something of a t...	/review/rw5561163/?ref_=tt_urv	Midsommar (2019)	[scary, m becor somet talking
647	647	17 September 2022	jboothmillard	5.0	Bodies Bodies Bodies\n	I saw the trailer a short time before this fil...	/review/rw8629009/?ref_=tt_urv	Bodies Bodies Bodies (2022)	[saw, tr short, film, rele
954	954	15 June 2022	jboothmillard	4.0	Men\n	I thought maybe the film was a dark twisted ta...	/review/rw8173612/?ref_=tt_urv	Men (2022)	[thou maybe, dark, twi take,
1051	1051	17 July 2020	ironhorse_iv	8.0	The Lighthouse is a shining example of a semi...	It's time to shed some light into this A24 mot...	/review/rw5927914/?ref_=tt_urv	The Lighthouse (2019)	[time, s light, mo pic direct
1079	1079	20 October 2019	grantpaulsen	10.0	The Best Film of 2019 So Far\n	This film is a psychological horror masterpiec...	/review/rw5927914/?ref_=tt_urv	The Lighthouse (2019)	psycholog ho masterp
1095	1095	6 November 2019	theredsky	10.0	The Lighthouse Review\n	Not only is The Lighthouse a masterpiece it's ...	/review/rw5927914/?ref_=tt_urv	The Lighthouse (2019)	[lightho masterp favorite, y
1745	1745	24 October 2019	Gilly-13	8.0	I had to "solve" this film, but it was worth ...	Like many I was bored and confused after viewi...	/review/rw5365158/?ref_=tt_urv	The Souvenir (2019)	[like, n b confu viev souv

In [ ]:
```
res2 = pd.DataFrame.from_dict(res2).T
res2
```

Out[ ]:

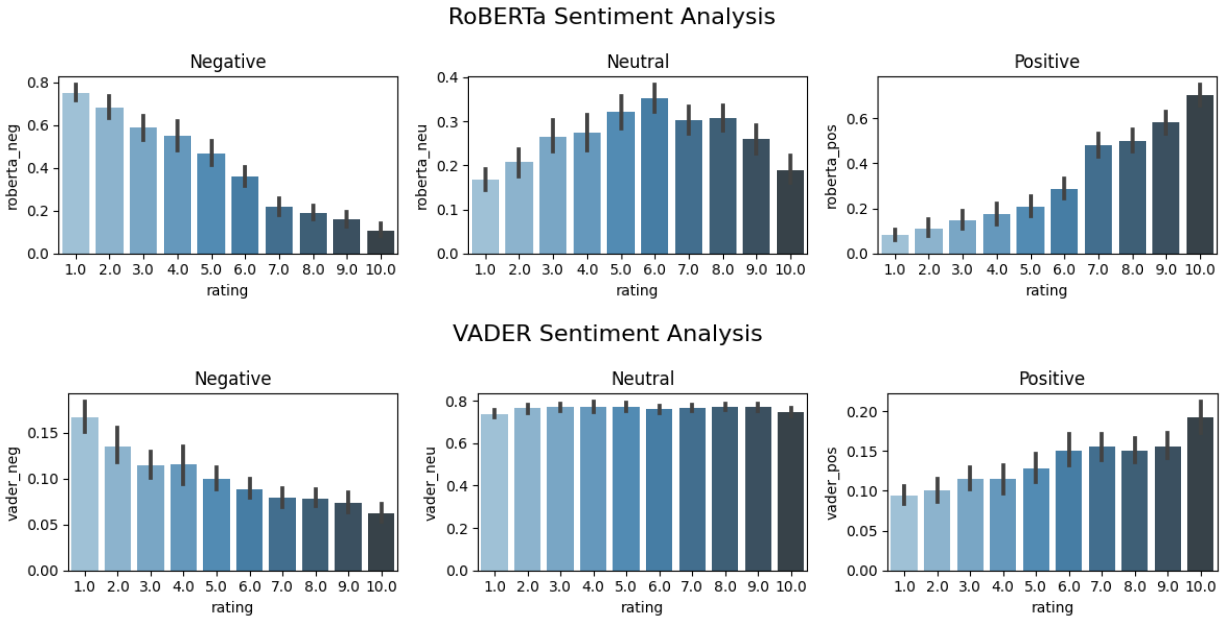
	vader_neg	vader_neu	vader_pos	vader_compound	roberta_neg	roberta_neu	roberta_pos
0	0.138	0.799	0.063	-0.3792	0.005127	0.024844	0.970029
1	0.215	0.583	0.202	-0.1531	0.854076	0.129466	0.016459
2	0.086	0.751	0.162	0.9640	0.021825	0.089965	0.888211
3	0.000	0.726	0.274	0.5267	0.307210	0.565948	0.126843
4	0.092	0.804	0.104	0.5658	0.024468	0.091965	0.883567
...	...	...	...	...	...	...	...
1761	0.049	0.639	0.312	0.9742	0.154799	0.288335	0.556866
1762	0.111	0.707	0.182	0.1531	0.938957	0.053684	0.007358
1763	0.047	0.740	0.213	0.9584	0.007885	0.088672	0.903443
1764	0.076	0.780	0.144	0.7469	0.762345	0.185355	0.052300
1765	0.135	0.865	0.000	-0.6584	0.702192	0.278444	0.019364

1756 rows × 7 columns

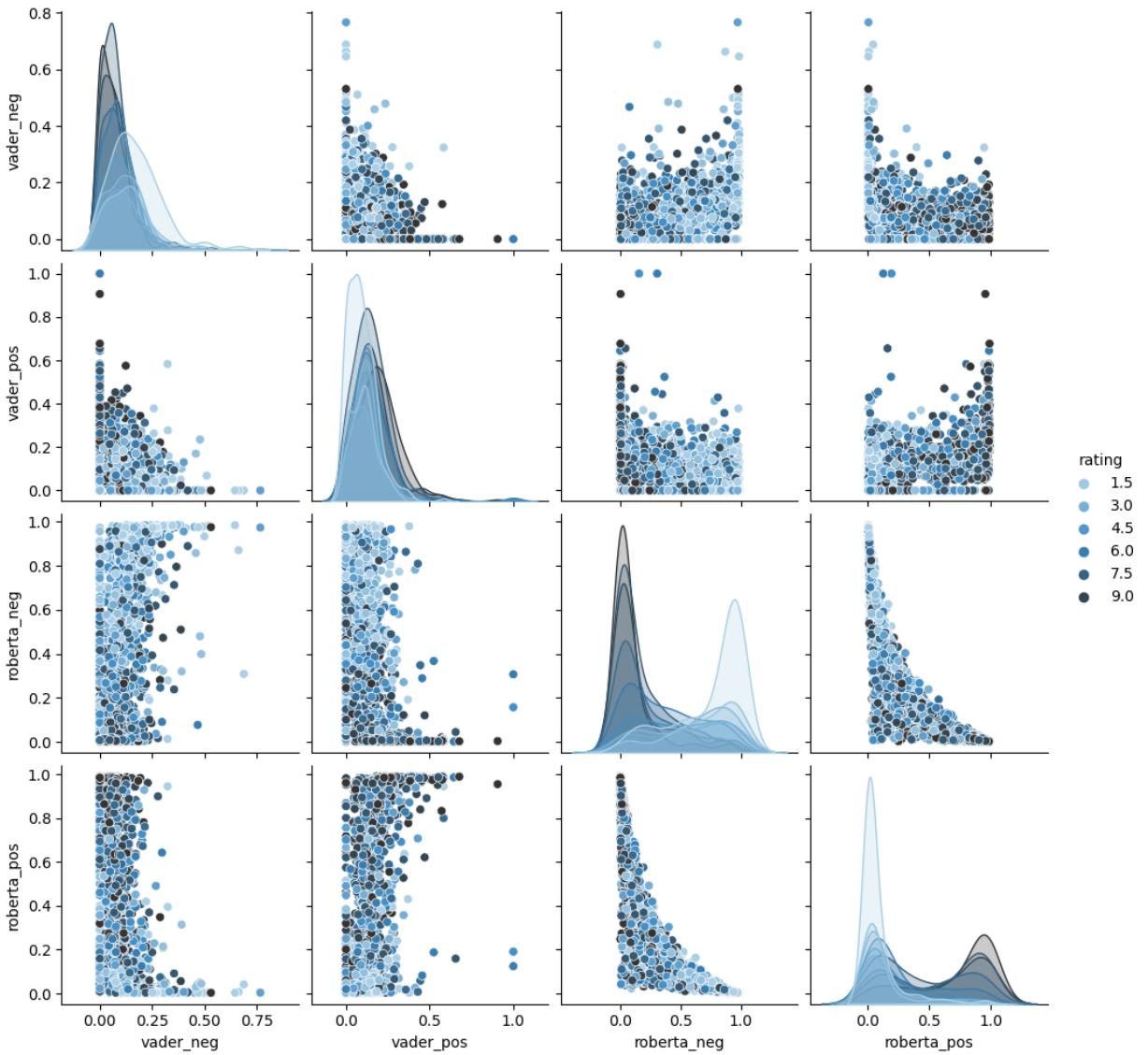
In [ ]:
```
res2 = res2.reset_index().rename(columns={'index': 'id'})
res2 = res2.merge(sentiment_df, how='left', on='id')

res.to_csv('res.csv', index=False)
```

In [ ]:
```
res = pd.read_csv('res.csv')
```

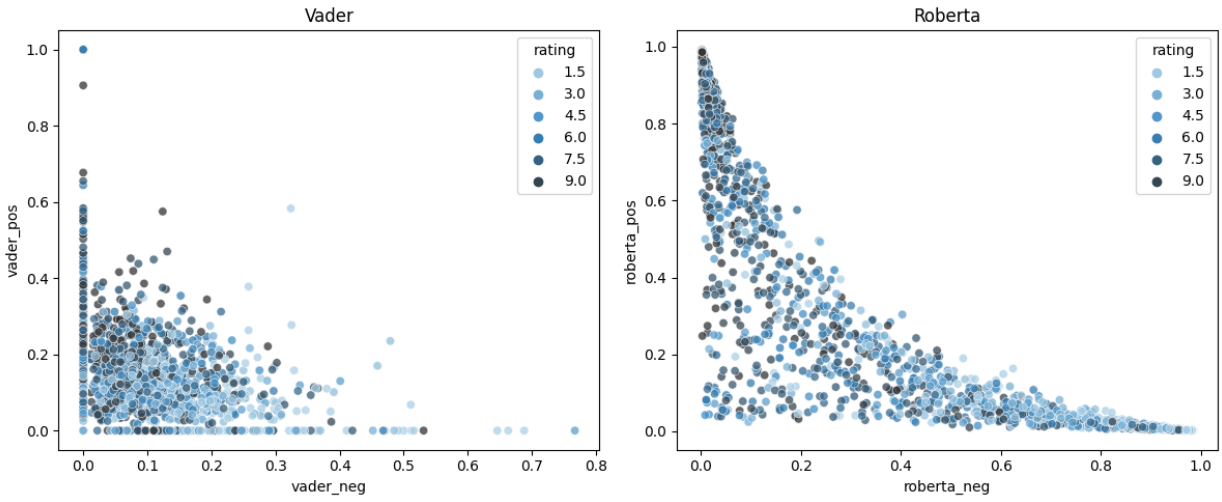
In [ ]:
```
fig, axs = plt.subplots(1, 3, figsize=(12, 3))
fig.suptitle('RoBERTa Sentiment Analysis', fontsize=16)
sns.barplot(data=res, x='rating', y='roberta_neg', palette='Blues_d', ax=axs[0])
sns.barplot(data=res, x='rating', y='roberta_neu', palette='Blues_d', ax=axs[1])
sns.barplot(data=res, x='rating', y='roberta_pos', palette='Blues_d', ax=axs[2])
axs[0].set_title('Negative')
axs[1].set_title('Neutral')
axs[2].set_title('Positive')
plt.tight_layout()
plt.show()

fig, axs = plt.subplots(1, 3, figsize=(12, 3))
fig.suptitle('VADER Sentiment Analysis', fontsize=16)
sns.barplot(data=res, x='rating', y='vader_neg', palette='Blues_d', ax=axs[0])
sns.barplot(data=res, x='rating', y='vader_neu', palette='Blues_d', ax=axs[1])
sns.barplot(data=res, x='rating', y='vader_pos', palette='Blues_d', ax=axs[2])
axs[0].set_title('Negative')
axs[1].set_title('Neutral')
axs[2].set_title('Positive')
plt.tight_layout()
plt.show()
```



In [ ]:
```
sns.pairplot(data=res, vars=['vader_neg', 'vader_pos',
       'roberta_neg', 'roberta_pos'], hue='rating', palette='Blues_d')
plt.show()
```



In [ ]:
```
fig, axs = plt.subplots(1, 2, figsize=(12, 5))
sns.scatterplot(data=res, x='vader_neg', y='vader_pos', hue='rating', alpha=0.75, palette='Blues_d',ax=a
sns.scatterplot(data=res, x='roberta_neg', y='roberta_pos', hue='rating', alpha=0.75, palette='Blues_d',
axs[0].set_title('Vader')
axs[1].set_title('Roberta')
plt.tight_layout()
plt.show()
#plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```



In [ ]:
```
sentiment_dict = {
    "negative": res["roberta_neg"].mean(),
    "neutral": res["roberta_neu"].mean(),
    "positive": res["roberta_pos"].mean()
}

sentiment_dff = pd.DataFrame(sentiment_dict, index=[0]).T

# get sentiment summary statistics
sentiment_stats = res[["roberta_neg", "roberta_neu", "roberta_pos"]].describe().loc[["25%", "50%", "75%"

# plot mean sentiment scores with error bars showing 25th and 75th percentiles
colors = ["crimson", "orange", "green"]
ax = sentiment_dff.plot(kind="bar", y=0, title="roBERTa Sentiment Analysis of A24 Film Reviews", legend=

# add axis labels
ax.set_xlabel("Sentiment")
ax.set_ylabel("Mean Sentiment Score")


plt.show()
print(sentiment_stats)
#find the standard deviation of the sentiment scores
print(res["roberta_neg"].std())
print(res["roberta_neu"].std())
print(res["roberta_pos"].std())
```


```
     roberta_neg  roberta_neu  roberta_pos
25%     0.027360     0.063319     0.029809
50%     0.272569     0.231409     0.164054
75%     0.728434     0.410902     0.723695
0.3601126222401771
0.21880914199131524
0.36764727261538527
```

Training

In [ ]:
```
res = pd.read_csv('res.csv')
```

In [ ]:
```
res['text_prepared'] = res['text_stm'].str.replace('[', '').str.replace(']', '').str.replace("'", "").st
res['roberta_sentiment'] = np.where(res['roberta_pos'] > res['roberta_neg'], 'positive', 'negative')
res['vader_sentiment'] = np.where(res['vader_pos'] > res['vader_neg'], 'positive', 'negative')
print(res['vader_sentiment'].value_counts())
print(res['roberta_sentiment'].value_counts())
res['const'] = 1
```

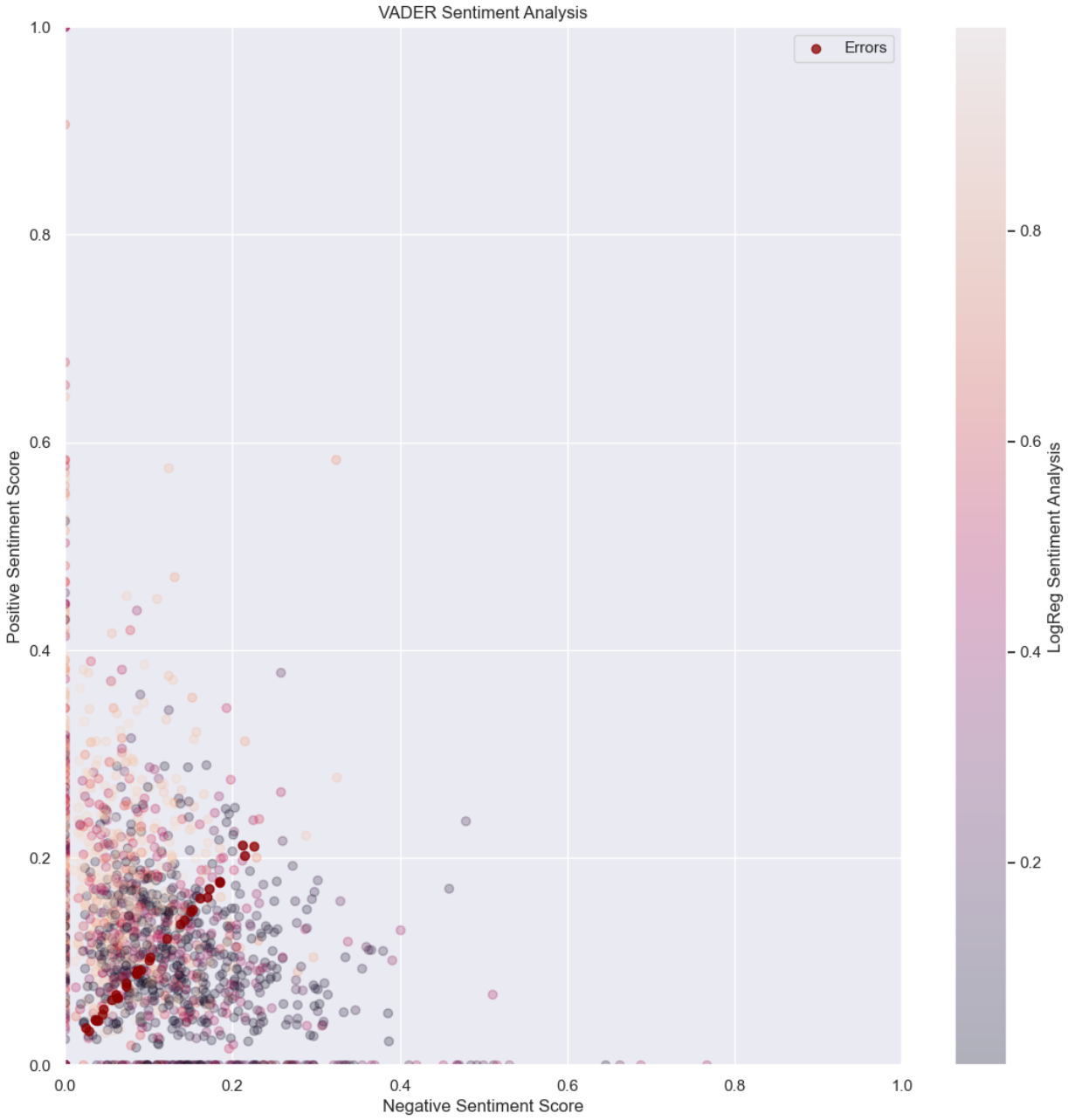
```
positive    1045
negative     711
Name: vader_sentiment, dtype: int64
negative    941
positive    815
Name: roberta_sentiment, dtype: int64
```

In [ ]:
```
X = res['text_prepared']
y = res['roberta_sentiment']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)

X_train = vec.fit_transform(X_train)
X_test = vec.transform(X_test)

log_regression = LogisticRegression()

log_regression.fit(X_train,y_train)

X = vec.transform(X)

res['predict'] = log_regression.predict_proba(X)[::,1]
```

In [ ]:
```
cdf = res[['vader_neg', 'vader_pos', 'const', 'vader_sentiment']]
cdf['vader_sentiment'] = cdf['vader_sentiment'].map({'negative':0, 'positive':1})
cdf

clf_pipeline = make_pipeline(StandardScaler(),  LogisticRegression())

X = cdf.drop(columns=['vader_sentiment'])
y = cdf['vader_sentiment']

clf_pipeline.fit(X, y)

X['y_pred'] = clf_pipeline.predict(X)
X['y_true'] = y
```

In [ ]:
```
X['predict'] = res['predict']
```

In [ ]:
```
#create a scatter plot of the vader_neg and vader_pos columns
X['relation'] = np.where(X['y_pred'] == X['y_true'], 'correct', 'incorrect')
plt.scatter(X['vader_neg'], X['vader_pos'], c=X['predict'], alpha=0.25)

plt.colorbar(label='LogReg Sentiment Analysis')
plt.scatter(X[X['relation'] == 'incorrect']['vader_neg'], X[X['relation'] == 'incorrect']['vader_pos'],

# create axis labels
plt.title("VADER Sentiment Analysis")
plt.xlabel("Negative Sentiment Score")
plt.ylabel("Positive Sentiment Score")
plt.show

plt.xlim(0.0, 1)
plt.ylim(0.0, 1)

plt.legend()

plt.show()
```



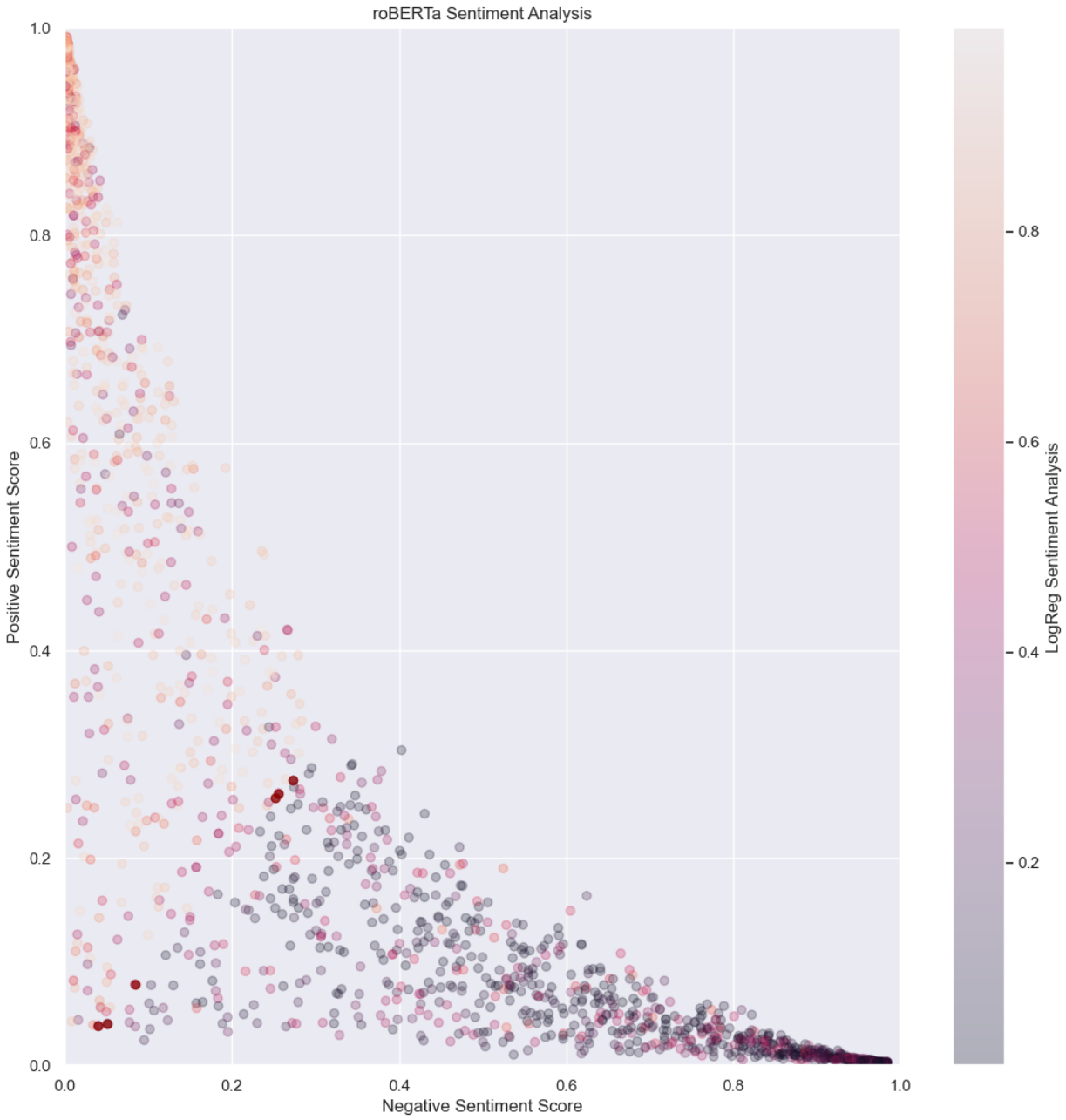
In [ ]:
```
cdf = res[['roberta_neg', 'roberta_pos', 'const', 'roberta_sentiment']]
cdf['roberta_sentiment'] = cdf['roberta_sentiment'].map({'negative':0, 'positive':1})
cdf

clf_pipeline = make_pipeline(StandardScaler(),  LogisticRegression())

X = cdf.drop(columns=['roberta_sentiment'])
y = cdf['roberta_sentiment']


clf_pipeline.fit(X, y)

X['y_pred'] = clf_pipeline.predict(X)
X['y_true'] = y
```

In [ ]:
```
X['predict'] = res['predict']
```

In [ ]:
```
X['relation'] = np.where(X['y_pred'] == X['y_true'], 'correct', 'incorrect')
plt.scatter(X['roberta_neg'], X['roberta_pos'], c=X['predict'], alpha=0.25)
plt.colorbar(label='LogReg Sentiment Analysis')
plt.scatter(X[X['relation'] == 'incorrect']['roberta_neg'], X[X['relation'] == 'incorrect']['roberta_pos

# create axis labels
plt.title('roBERTa Sentiment Analysis')
plt.xlabel("Negative Sentiment Score")
plt.ylabel("Positive Sentiment Score")

# scale the axis from just 0.4 to 0.4
```

```
plt.xlim(0.0, 1)
plt.ylim(0.0, 1)

plt.show
```

Out[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



roBERTA

In [ ]:
```
trainsize = 0.7
X = res['text_prepared']
y = res['roberta_sentiment']
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=1-trainsize,
                                                    random_state=123)
```

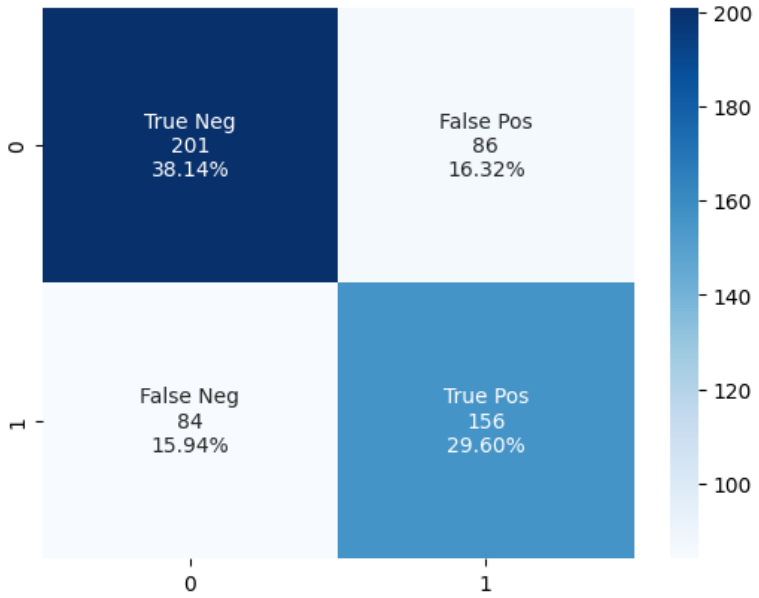
In [ ]:
```
X_train = vec.fit_transform(X_train)
X_test = vec.transform(X_test)
```

In [ ]:
```
#model = LogisticRegression(max_iter=10000)
model = MultinomialNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Training Accuracy :", model.score(X_train, y_train))
print("Test Accuracy :", model.score(X_test, y_test))

# confusion matrix
```

```
cm = confusion_matrix(y_test, y_pred)

group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                cm.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cm.flatten()/np.sum(cm)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cm, annot=labels, fmt='', cmap='Blues')
```

```
Training Accuracy : 0.9902359641985354
Test Accuracy : 0.6774193548387096
```

Out[ ]: <Axes: >



In [ ]:
```
X = res['text_prepared']
y = res['roberta_sentiment']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)

X_train = vec.fit_transform(X_train)
X_test = vec.transform(X_test)

log_regression = LogisticRegression()

log_regression.fit(X_train,y_train)

log_regression.predict_proba(X_test)[::,1]
```

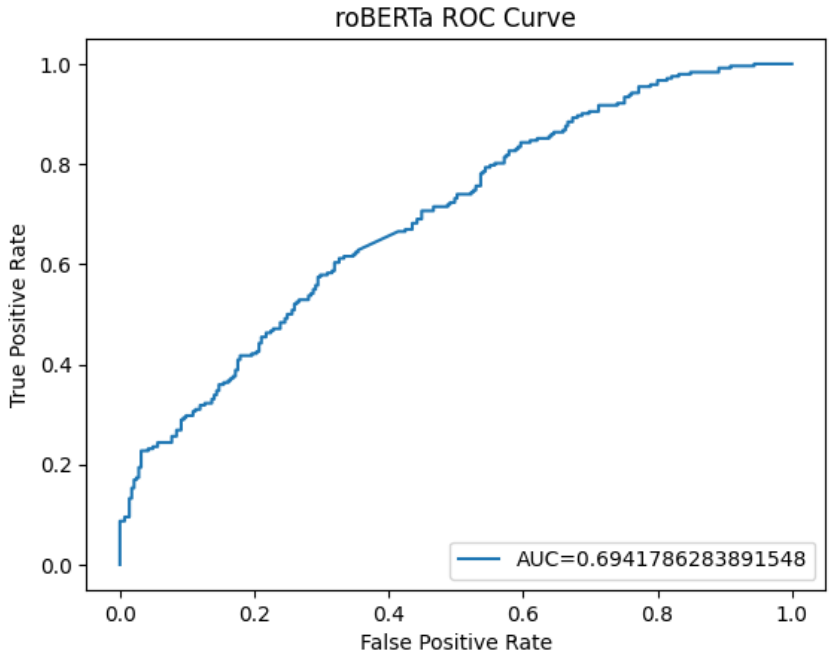
Out[ ]:
```
array([0.30791693, 0.34286801, 0.41091058, 0.4181123 , 0.24401498,
       0.72977265, 0.32903539, 0.27853409, 0.39573389, 0.25381013,
       0.4287754 , 0.4154897 , 0.38163156, 0.34707751, 0.74961531,
       0.13978363, 0.43718176, 0.0867919 , 0.64074144, 0.42408947,
       0.52446974, 0.46755129, 0.60420658, 0.45676897, 0.37580366,
       0.41803702, 0.52372435, 0.53389151, 0.34238605, 0.44158145,
       0.42086541, 0.31935608, 0.41245901, 0.754841  , 0.43130144,
       0.4461579 , 0.46292848, 0.44096424, 0.37321059, 0.70131013,
       0.92667445, 0.44197558, 0.63838248, 0.31647675, 0.39285661,
       0.44158514, 0.44977489, 0.29826361, 0.4287754 , 0.43572189,
       0.36005732, 0.38406564, 0.10067625, 0.66713328, 0.28092883,
       0.14650766, 0.62671298, 0.5771173 , 0.37997143, 0.48919753,
       0.54922479, 0.38228908, 0.6303655 , 0.61419095, 0.19962482,
       0.43681498, 0.42608985, 0.70774935, 0.55498423, 0.66588839,
       0.13684098, 0.31878085, 0.21847193, 0.28707492, 0.61902889,
       0.40299983, 0.35937077, 0.64543951, 0.4287754 , 0.40029962,
       0.51288618, 0.46629289, 0.50820652, 0.22756311, 0.4287754 ,
       0.55411426, 0.30335848, 0.57936591, 0.09272799, 0.59127197,
       0.61619819, 0.332228  , 0.43829678, 0.91021153, 0.45587343,
       0.4029765 , 0.58131765, 0.70786307, 0.5604549 , 0.93887149,
       0.61505688, 0.37007018, 0.4902411 , 0.48286305, 0.42356995,
       0.55635159, 0.53912053, 0.40273034, 0.70310717, 0.53328061,
       0.41564603, 0.55991008, 0.25044053, 0.61044986, 0.50512856,
       0.44740333, 0.38030413, 0.4287754 , 0.77039402, 0.14020916,
       0.52142432, 0.27019045, 0.4527892 , 0.56508947, 0.60760248,
       0.15181435, 0.39565337, 0.47254012, 0.37050844, 0.44616493,
       0.51649284, 0.31689018, 0.4287754 , 0.32750507, 0.42734512,
       0.4287754 , 0.40673398, 0.49244301, 0.49471998, 0.30617594,
       0.59594843, 0.50899508, 0.30822735, 0.52850211, 0.50564087,
       0.17857354, 0.45822784, 0.85797708, 0.35503154, 0.47138395,
       0.45341138, 0.34827894, 0.30960313, 0.96553908, 0.55506413,
       0.60096627, 0.45936895, 0.47562123, 0.43906985, 0.54037007,
       0.46607021, 0.18819026, 0.26162135, 0.22371915, 0.74956551,
       0.43942621, 0.37424486, 0.81711504, 0.46831553, 0.44266585,
       0.48989557, 0.4287754 , 0.50125979, 0.38504981, 0.13364457,
       0.56577867, 0.39586477, 0.46729356, 0.38491623, 0.31920951,
       0.51197647, 0.45204733, 0.34530167, 0.29717424, 0.4287754 ,
       0.42487938, 0.27058696, 0.48975584, 0.45946732, 0.42296428,
       0.39947534, 0.4287754 , 0.4287754 , 0.90103367, 0.45829494,
       0.65401684, 0.17891309, 0.43600201, 0.34405447, 0.4287754 ,
       0.49803971, 0.5387224 , 0.39567975, 0.36315118, 0.29152256,
       0.50653159, 0.61508447, 0.50707692, 0.4098383 , 0.33894232,
       0.29871132, 0.465441  , 0.55572811, 0.28707981, 0.72407733,
       0.42809738, 0.43928384, 0.29853309, 0.41700244, 0.421139  ,
       0.41786484, 0.25965247, 0.51072475, 0.12591922, 0.39366186,
       0.2852002 , 0.44830137, 0.91469244, 0.4287754 , 0.25781758,
       0.39715369, 0.50400988, 0.62510493, 0.39395915, 0.36892403,
       0.41849139, 0.37045795, 0.0867919 , 0.52799136, 0.44786296,
       0.66323325, 0.38575181, 0.57311945, 0.56166063, 0.45474843,
       0.43116798, 0.41873312, 0.2817662 , 0.23443244, 0.42105268,
       0.36908207, 0.38953382, 0.58051727, 0.41343316, 0.37111846,
       0.33059046, 0.37522811, 0.42186118, 0.1776675 , 0.436626  ,
       0.58137271, 0.2811675 , 0.46776315, 0.5343459 , 0.52101619,
       0.45230893, 0.42647328, 0.43058174, 0.08472278, 0.53868869,
       0.31753228, 0.56099447, 0.40280658, 0.6295022 , 0.64337405,
       0.7450493 , 0.26850419, 0.60575913, 0.76670933, 0.54820906,
       0.54867419, 0.52718152, 0.4287754 , 0.41682453, 0.52750533,
       0.514403  , 0.4394276 , 0.69717298, 0.46515213, 0.13751026,
       0.32855725, 0.4287754 , 0.41647431, 0.3498673 , 0.38962108,
       0.52008324, 0.48269732, 0.3847591 , 0.52971311, 0.52256831,
       0.51640984, 0.4287754 , 0.36696648, 0.43222006, 0.43864953,
       0.45972327, 0.4687387 , 0.21223039, 0.20968094, 0.4287754 ,
       0.50221127, 0.44337099, 0.47076897, 0.33133755, 0.34658643,
       0.41645482, 0.36433267, 0.67302254, 0.30038305, 0.4821444 ,
       0.4313153 , 0.34764144, 0.20021779, 0.405599  , 0.36784366,
       0.43106433, 0.40391899, 0.66413821, 0.4500627 , 0.34007301,
       0.32281749, 0.46341425, 0.31267598, 0.44188192, 0.51003775,
       0.53351637, 0.41153028, 0.35200518, 0.49886247, 0.48849166,
       0.3630679 , 0.32296163, 0.506106  , 0.44451795, 0.47597429,
       0.71744976, 0.4287754 , 0.33243938, 0.34964604, 0.4190585 ,
       0.3746119 , 0.91965303, 0.40392993, 0.42774502, 0.50679518,
       0.41379355, 0.4287754 , 0.07812923, 0.34144486, 0.72938613,
       0.64946631, 0.39523359, 0.37579189, 0.42494268, 0.42610879,
       0.54173096, 0.38078859, 0.39602858, 0.55072234, 0.22499044,
       0.46032221, 0.4112861 , 0.40702226, 0.90607821, 0.56498419,
       0.37774716, 0.35907693, 0.20419195, 0.62283409, 0.49246638,
       0.41437919, 0.44328834, 0.48024735, 0.45203668, 0.70545045,
       0.31552811, 0.52808163, 0.45954019, 0.18270097, 0.14706022,
       0.49824573, 0.57244813, 0.61317371, 0.42827574, 0.4287754 ,
       0.69966024, 0.4195225 , 0.5017635 , 0.62773921, 0.41465397,
```

```
0.35582572, 0.8074929 , 0.24455084, 0.376041  , 0.39560672,
0.34043422, 0.40660688, 0.42326804, 0.16261467, 0.47093066,
0.42705883, 0.4287754 , 0.65487643, 0.24117058, 0.5162344 ,
0.04096148, 0.35405989, 0.51775226, 0.17897653, 0.48618392,
0.4386911 , 0.37025603, 0.41341657, 0.47584991, 0.4287754 ,
0.41904886, 0.32189606, 0.38997824, 0.4287754 , 0.36385295,
0.43334898, 0.30056276, 0.58186814, 0.29461536, 0.16286728,
0.56753107, 0.36145888, 0.67994281, 0.38319939, 0.72492584,
0.2902002 , 0.31957618, 0.45084469, 0.54350172, 0.4077265 ,
0.51641736, 0.42825577, 0.49228988, 0.52893571, 0.38798012,
0.38229783, 0.60243068, 0.25277528, 0.54597333, 0.44970091,
0.44282636, 0.42772977, 0.4105906 , 0.39241138, 0.31680129,
0.74613865, 0.35182827, 0.14675691, 0.48973045, 0.4287754 ,
0.40749519, 0.28608   , 0.34350081, 0.31499611, 0.58614645,
0.49448155, 0.34096836, 0.60084384, 0.36194375, 0.50217941,
0.4287754 , 0.44561618, 0.48350876, 0.42314548, 0.69691987,
0.51988293, 0.46674979, 0.25903937, 0.44589577, 0.37490196,
0.42190095, 0.5034113 , 0.6918064 , 0.57705653, 0.49981366,
0.54722854, 0.42006532, 0.40600618, 0.4435057 , 0.3797111 ,
0.49081785, 0.17106391, 0.4554921 , 0.30377549, 0.43263306,
0.42427816, 0.21847965, 0.18901456, 0.17525491, 0.50086095,
0.5206535 , 0.40928732, 0.39523359, 0.43970845, 0.71752748,
0.47413611, 0.34159651, 0.54288724, 0.41398427, 0.20274506,
0.40324804, 0.68709039, 0.47747333, 0.40483459, 0.16056583,
0.34292681, 0.9030475 , 0.15010505, 0.68463739, 0.05634044,
0.47772797, 0.4287754 ])
```

In [ ]:
```
y_test = y_test.map({'positive': 1, 'negative': 0}).astype(int)
```

In [ ]:
```
y_pred_proba = log_regression.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)

auc= metrics.roc_auc_score(y_test, y_pred_proba)

#create ROC curve

plt.plot(fpr, tpr,label="AUC="+str(auc))
plt.title('roBERTa ROC Curve')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x13bbd6a60>
```



VADERS

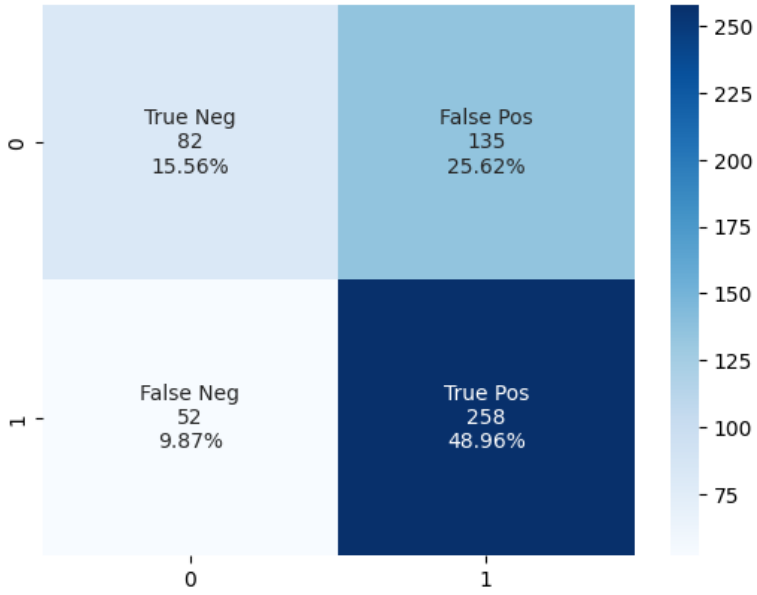
In [ ]:
```
trainsize = 0.7
X = res['text_prepared']
y = res['vader_sentiment']
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=1-trainsize,
                                                    random_state=123)
```

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer
vec = CountVectorizer(ngram_range=(2,2), max_features=500000)
X_train = vec.fit_transform(X_train)
X_test = vec.transform(X_test)
```

In [ ]:
```
#model = LogisticRegression(max_iter=10000)
model = MultinomialNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Training Accuracy :", model.score(X_train, y_train))
print("Test Accuracy :", model.score(X_test, y_test))

# confusion matrix
cm = confusion_matrix(y_test, y_pred)

group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                cm.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cm.flatten()/np.sum(cm)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cm, annot=labels, fmt='', cmap='Blues')
```

```
Training Accuracy : 0.9869812855980472
Test Accuracy : 0.6451612903225806
```

Out[ ]: <Axes: >



In [ ]:
```
X = res['text_prepared']
y = res['vader_sentiment']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)

X_train = vec.fit_transform(X_train)
X_test = vec.transform(X_test)

log_regression = LogisticRegression()

log_regression.fit(X_train,y_train)
```

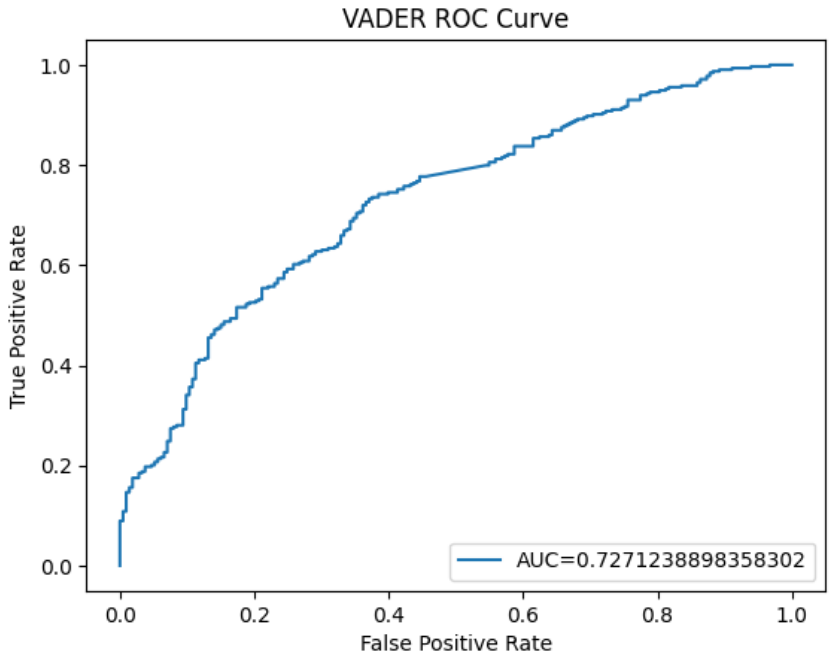
Out[ ]:
```
▾ LogisticRegression

LogisticRegression()
```

In [ ]:
```
y_test = y_test.map({'positive': 1, 'negative': 0}).astype(int)
```

In [ ]:
```
y_pred_proba = log_regression.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)

auc= metrics.roc_auc_score(y_test, y_pred_proba)
```

```
#create ROC curve

plt.plot(fpr, tpr,label="AUC="+str(auc))
plt.title('VADER ROC Curve')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
```

Out[ ]: <matplotlib.legend.Legend at 0x12f6fefa0>

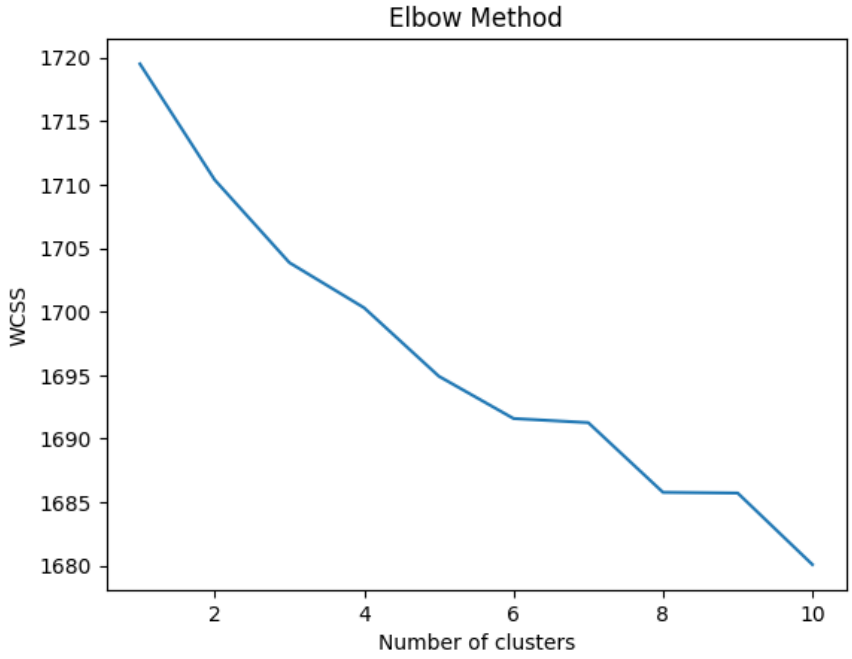


CLUSTERING AND LDA

In [ ]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

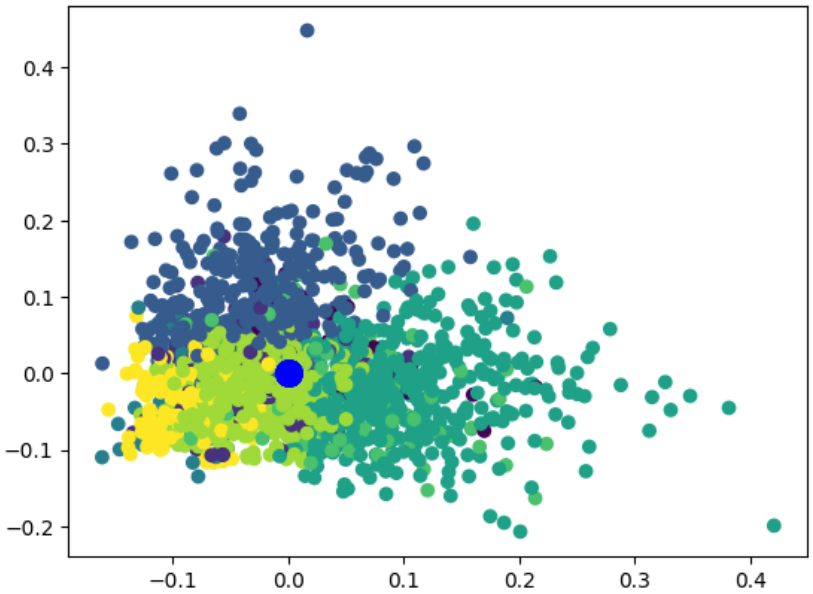
In [ ]:
```
lda = cleaned_df
```

In [ ]:
```
X = vectorizer.fit_transform(lda['text_stm'])

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Compute the first differences and the difference ratios
d1 = np.diff(wcss)
d2 = np.diff(d1) / d1[:-1]

# Find the index of the last difference ratio that is less than the 90th percentile of all difference ra
idx = np.where(d2 < np.percentile(d2, 90))[0][-1]

plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

print(f"Best number of clusters: {idx+2}")
```




```
Best number of clusters: 8
```

In [ ]:
```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
k = 8
kmeans = KMeans(n_clusters=k)
kmeans.fit(tf)

labels = kmeans.predict(tf)
centroids = kmeans.cluster_centers_

pca = PCA(n_components=2)
pca_result = pca.fit_transform(tf)

# Plot the data points colored by their cluster labels
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=kmeans.labels_)
plt.scatter(centroids[:, 0], centroids[:, 1], s=150, c='b', label='Centroids')

plt.show()
```



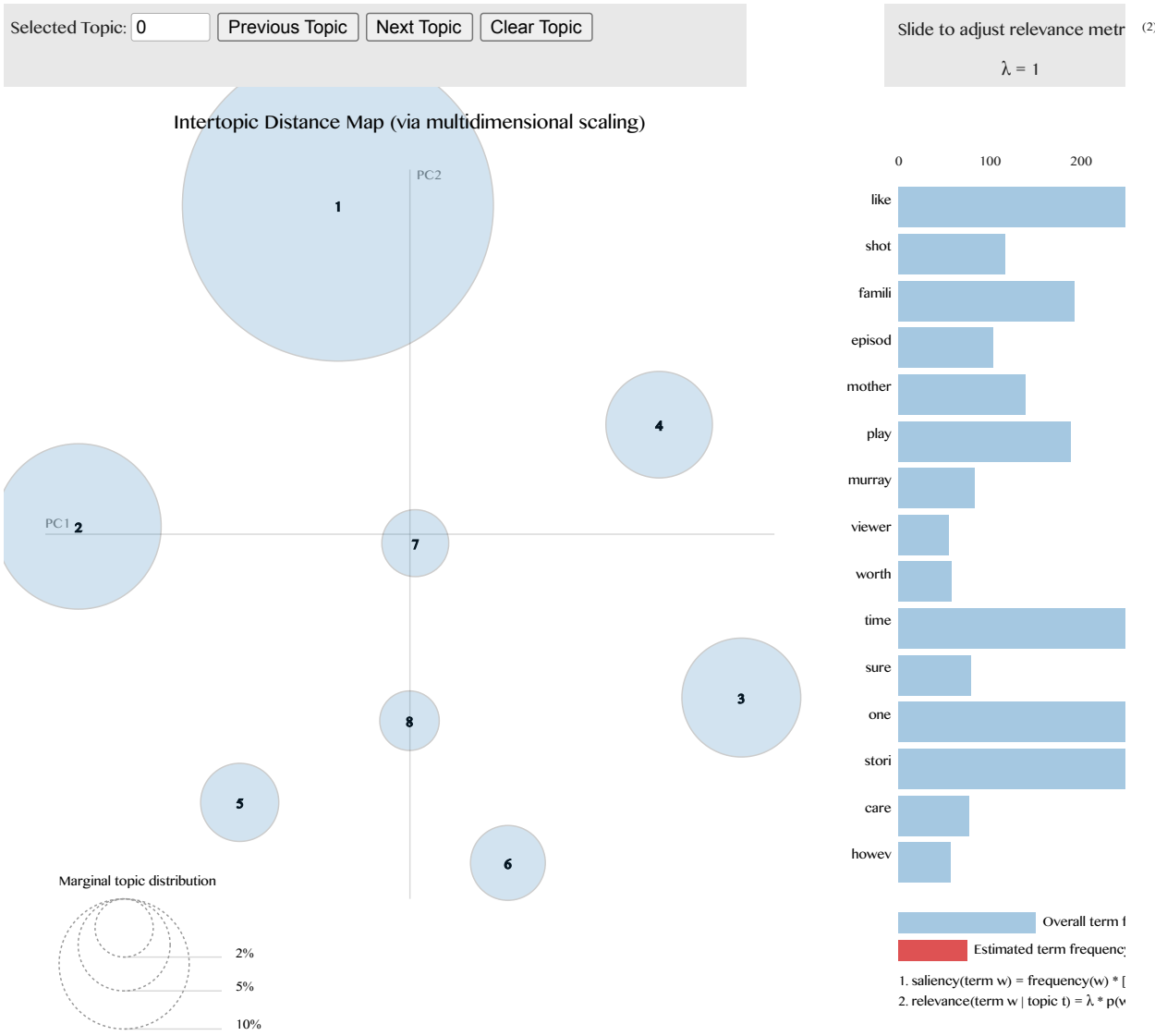
In [ ]:
```
preprocessed_texts = lda['text_stm']

preprocessed_texts = preprocessed_texts[~preprocessed_texts.isna()]

# Convert string representation of list to actual list of tokens
data_words = [ast.literal_eval(text) for text in preprocessed_texts]

# Step 2: Create a dictionary
id2word = corpora.Dictionary(data_words)
```

```
stop_words = ['film', 'movi', 'watch', 'saint', 'maud', 'knight', 'san', 'francisco', 'black', 'man', 'u


unwanted_word_ids = [id2word.token2id[word] for word in stop_words if word in id2word.token2id]
id2word.filter_tokens(bad_ids=unwanted_word_ids)


# Step 3: Convert text data to bag-of-words format
corpus = [id2word.doc2bow(text) for text in data_words]

# Step 4: Save corpus and dictionary to disk
corpora.MmCorpus.serialize('corpus.mm', corpus)
id2word.save('dictionary.gensim')
```

In [ ]:
```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=6,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto')
```

In [ ]:
```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word, mds='mmds', R=15)
vis
```

Out[ ]:

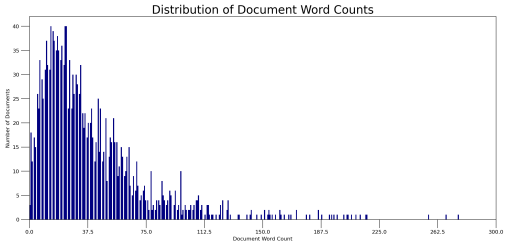


In [ ]:
```
def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), t
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data_words)

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords', 'Text']
df_dominant_topic.head(10)
```

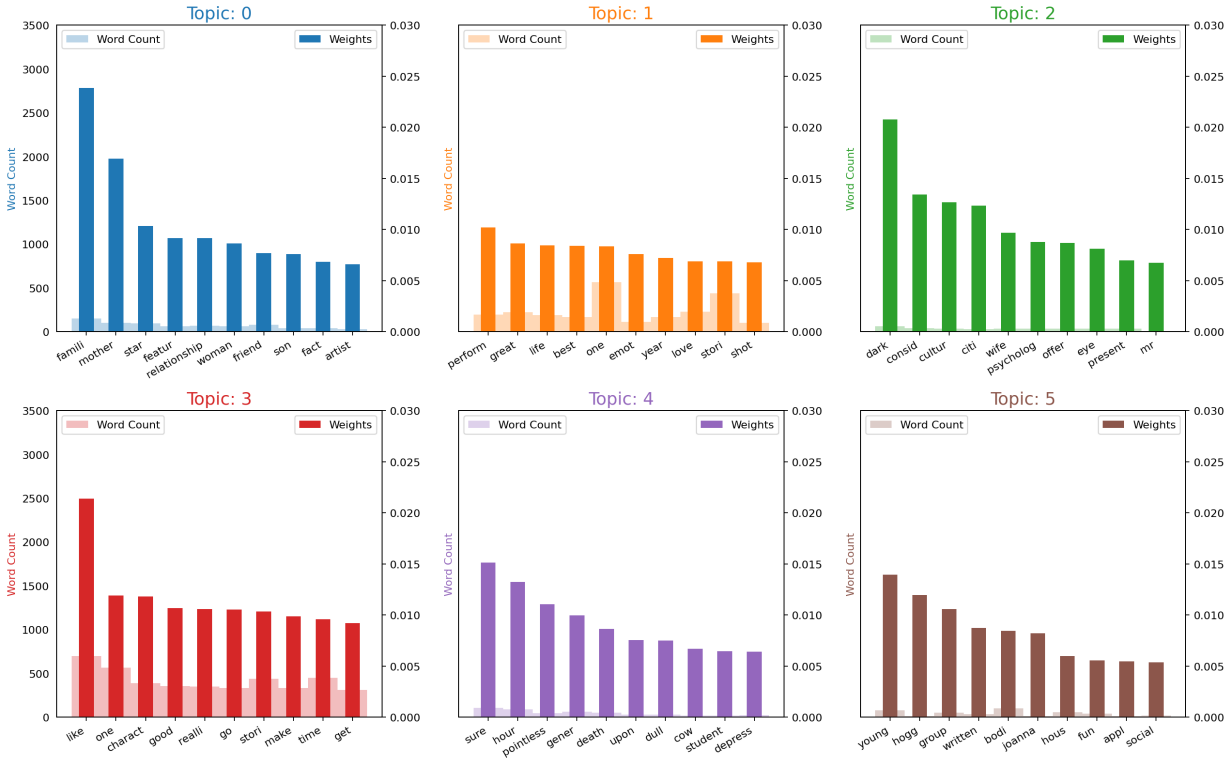
Out[ ]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	3	0.4775	like, one, charact, good, realli, go, stori, m...	[profoundli, deep, genuin, move, utterli, hila...
1	1	3	0.4558	like, one, charact, good, realli, go, stori, m...	[troubl, turn, brain, anxieti, worri, mundan, ...
2	2	3	0.7283	like, one, charact, good, realli, go, stori, m...	[take, drug, first, time, imagin, jacki, chan,...
3	3	3	0.5700	like, one, charact, good, realli, go, stori, m...	[kind, especi, know, go]
4	4	1	0.4903	perform, great, life, best, one, emot, year, l...	[everyth, everywher, even, crazier, trailer, w...
5	5	3	0.6388	like, one, charact, good, realli, go, stori, m...	[mind, weird, film, truli, film, pointlessli, ...
6	6	3	0.4026	like, one, charact, good, realli, go, stori, m...	[us, mind, film, agre, one, thing, everyth, ev...
7	7	3	0.6071	like, one, charact, good, realli, go, stori, m...	[everyth, everywher, sometim, mani, thing, man...
8	8	3	0.4888	like, one, charact, good, realli, go, stori, m...	[point, swept, sensori, overload, much, stuff,...
9	9	3	0.5078	like, one, charact, good, realli, go, stori, m...	[rare, movi, hit, hard, go, along, ride, everi...

In [ ]:
```
doc_lens = [len(d) for d in df_dominant_topic.Text]

# Plot
plt.figure(figsize=(16,7), dpi=160)
plt.hist(doc_lens, bins = 1000, color='navy')
plt.text(750, 100, "Mean   : " + str(round(np.mean(doc_lens))))
plt.text(750,  90, "Median : " + str(round(np.median(doc_lens))))
plt.text(750,  80, "Stdev   : " + str(round(np.std(doc_lens))))
plt.text(750,  70, "25%ile    : " + str(round(np.quantile(doc_lens, q=0.25))))
plt.text(750,  60, "75%ile  : " + str(round(np.quantile(doc_lens, q=0.75))))

plt.gca().set(xlim=(0, 300), ylabel='Number of Documents', xlabel='Document Word Count')
plt.tick_params(size=16)
plt.xticks(np.linspace(0,300,9))
plt.title('Distribution of Document Word Counts', fontdict=dict(size=22))
plt.show()
```



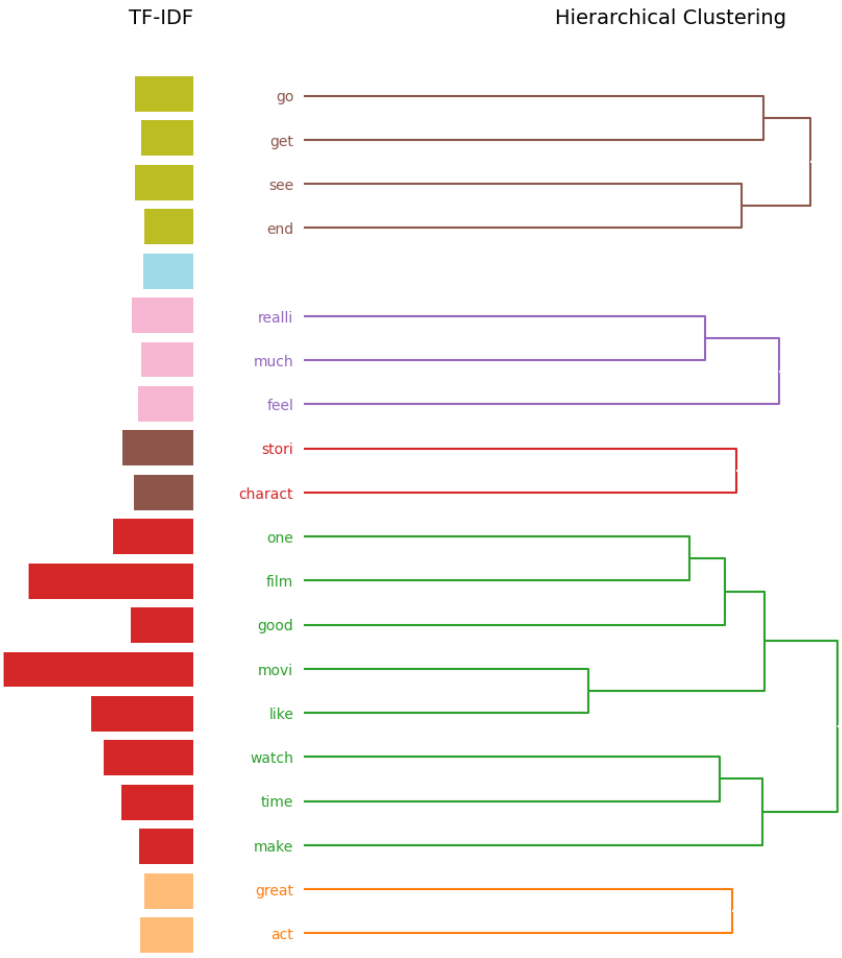
In [ ]:
```
from collections import Counter
import matplotlib.colors as mcolors
topics = lda_model.show_topics(formatted=False)
data_flat = [w for w_list in data_words for w in w_list]
counter = Counter(data_flat)

out = []
for i, topic in topics:
    for word, weight in topic:
        out.append([word, i , weight, counter[word]])

df = pd.DataFrame(out, columns=['word', 'topic_id', 'importance', 'word_count'])

# Plot Word Count and Weights of Topic Keywords
fig, axes = plt.subplots(2, 3, figsize=(16,10), sharey=True, dpi=160)
cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]
for i, ax in enumerate(axes.flatten()):
    ax.bar(x='word', height="word_count", data=df.loc[df.topic_id==i, :], color=cols[i], width=1, alpha=
    ax_twin = ax.twinx()
    ax_twin.bar(x='word', height="importance", data=df.loc[df.topic_id==i, :], color=cols[i], width=0.5,
    ax.set_ylabel('Word Count', color=cols[i])
    ax_twin.set_ylim(0, 0.030); ax.set_ylim(0, 3500)
    ax.set_title('Topic: ' + str(i), color=cols[i], fontsize=16)
    ax.tick_params(axis='y', left=False)
    ax.set_xticklabels(df.loc[df.topic_id==i, 'word'], rotation=30, horizontalalignment= 'right')
    ax.legend(loc='upper left'); ax_twin.legend(loc='upper right')

fig.tight_layout(w_pad=2)
fig.suptitle('Word Count and Importance of Topic Keywords', fontsize=22, y=1.05)
plt.show()
```



In [ ]:
```
from sklearn.metrics.pairwise import cosine_similarity
from scipy.cluster.hierarchy import ward, dendrogram
import scipy.cluster.hierarchy as sch

def clustering(df, number_words, number_clusters):

    # Get top important words (higest TF-IDF)
    important_words = tfidf_transposed[tfidf_transposed.index.isin(tfidf_sum.head(number_words).word)].
    important_words_tfidf = tfidf_sum.head(number_words)
    n = len(important_words)

    # Calculate distance between words
    dist = 1 - cosine_similarity(tfidf_transposed[tfidf_transposed.index.isin(tfidf_sum.head(number_word

    #define the linkage_matrix using ward clustering pre-computed distances
    Z = ward(dist)

    # Number of desired clusters
    T = sch.fcluster(Z, number_clusters, 'maxclust')

    # calculate labels
    labels=list('' for i in range(n))
    for i in range(n):
      labels[i]=str(i)+ ',' + str(T[i])

    # calculate color threshold
    ct=Z[-(number_clusters-1),2]

    # create the colour dictionary
    colors = {}
    for i in range(n):
      colors[important_words[i]] = plt.get_cmap('tab20')(T[i]/number_clusters)


    fig = plt.figure(figsize=(15, 12))
    gs = fig.add_gridspec(nrows=5, ncols=2, width_ratios=[1, 2], height_ratios=[1,1,100,1,1])
    ax0 = fig.add_subplot(gs[0, 1])

    # Plot dendrogram - Hierarchical Clustering
    ax2 = fig.add_subplot(gs[1:4, 1])
    ax = dendrogram(Z=Z,
                orientation="right",
                labels=important_words,
                color_threshold=ct,
                above_threshold_color="white",
                ax=ax2,
                leaf_font_size=10
```

```
                );
    ax2.set_xlim(0.5, 1.6)

    for leaf, leaf_color in zip(plt.gca().get_yticklabels(), ax["leaves_color_list"]):
      leaf.set_color(leaf_color)

    important_words_tfidf['word'] = pd.Categorical(important_words_tfidf['word'], categories=ax["ivl"],
    important_words_tfidf = important_words_tfidf.sort_values('word')



    # Plot bar chart - TF-IDF
    ax1 = fig.add_subplot(gs[:, 0])
    x = important_words_tfidf["word"]
    y = important_words_tfidf["tfidf_sum"]
    ax1.barh(x, y, align='edge', color=[colors[word] for word in x])
    ax1.set_yticks(x)
    ax1.set_yticklabels(x)
    ax1.set_xlim(175,0)
    ax1.tick_params(axis='y', colors='black')

    # Edit the plots
    ax0.spines['right'].set_visible(True)
    ax0.spines['top'].set_visible(False)
    ax0.set_axis_off()
    ax1.spines['top'].set_visible(False)
    ax1.set_axis_off()
    ax2.spines['left'].set_visible(False)
    ax2.spines['right'].set_visible(False)
    ax2.spines['top'].set_visible(False)
    ax2.spines['bottom'].set_visible(False)
    ax2.get_xaxis().set_visible(False)

    ax1.set_title("TF-IDF",loc="right", fontsize=14)
    ax0.set_title("Hierarchical Clustering", fontsize=14)

    #plt.tick_params(axis="y", labelsize=12)
    plt.show()
```

In [ ]:
```
clustering(ntf, 20, 6)
```


TF-IDF
Hierarchical Clustering
go
get
see
end
realli
much
feel
stori
charact
one
film
good
movi
like
watch
time
make
great
act