# Movie Recommender Systems

# About Dataset

## Context

These files contain metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

## Content

This dataset consists of the following files: movies_metadata.csv: The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.

keywords.csv: Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.

credits.csv: Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.

links.csv: The file that contains the TMDB and IMDB IDs of all the movies featured in the Full MovieLens dataset.

links_small.csv: Contains the TMDB and IMDB IDs of a small subset of 9,000 movies of the Full Dataset.

ratings_small.csv: The subset of 100,000 ratings from 700 users on 9,000 movies.

The Full MovieLens Dataset consisting of 26 million ratings and 750,000 tag applications from 270,000 users on all the 45,000 movies in this dataset can be accessed here

## Acknowledgements

This dataset is an ensemble of data collected from TMDB and GroupLens. The Movie Details, Credits and Keywords have been collected from the TMDB Open API. This product uses the

TMDb API but is not endorsed or certified by TMDb. Their API also provides access to data on many additional movies, actors and actresses, crew members, and TV shows. You can try it for yourself here. The Movie Links and Ratings have been obtained from the Official GroupLens website. The files are a part of the dataset available here

## Inspiration

This dataset was assembled as part of my second Capstone Project for Springboard's Data Science Career Track. I wanted to perform an extensive EDA on Movie Data to narrate the history and the story of Cinema and use this metadata in combination with MovieLens ratings to build various types of Recommender Systems. Both my notebooks are available as kernels with this dataset: The Story of Film and Movie Recommender Systems

Some of the things you can do with this dataset: Predicting movie revenue and/or movie success based on a certain metric. What movies tend to get higher vote counts and vote averages on TMDB? Building Content Based and Collaborative Filtering Based Recommendation Engines.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sd
import matplotlib.pyplot as plt
```

In [2]:
```
links = pd.read_csv("links.csv")
movies = pd.read_csv("movies_metadata.csv", low_memory=False)
rating = pd.read_csv("ratings.csv")
credits = pd.read_csv("credits.csv")
keywords = pd.read_csv("keywords.csv")
```

In [3]:
```
links
```

Out[3]:

| | movieId | imdbId | tmdbId |
|---|---|---|---|
| **0** | 1 | 114709 | 862.0 |
| **1** | 2 | 113497 | 8844.0 |
| **2** | 3 | 113228 | 15602.0 |
| **3** | 4 | 114885 | 31357.0 |
| **4** | 5 | 113041 | 11862.0 |
| **...** | ... | ... | ... |
| **45838** | 176269 | 6209470 | 439050.0 |
| **45839** | 176271 | 2028550 | 111109.0 |
| **45840** | 176273 | 303758 | 67758.0 |
| **45841** | 176275 | 8536 | 227506.0 |
| **45842** | 176279 | 6980792 | 461257.0 |

45843 rows × 3 columns

In [4]:
```
movies
```

Out[4]:

| | adult | belongs_to_collection | budget | genres | homepage | id |
|---|---|---|---|---|---|---|
| **0** | False | {'id': 10194, 'name': 'Toy Story Collection', ... | 30000000 | [{'id': 16, 'name': 'Animation'}, {'id': 35, '... | http://toystory.disney.com/toy-story | 862 |
| **1** | False | NaN | 65000000 | [{'id': 12, 'name': 'Adventure'}, {'id': 14, '... | NaN | 8844 |
| **2** | False | {'id': 119050, 'name': 'Grumpy Old Men Collect... | 0 | [{'id': 10749, 'name': 'Romance'}, {'id': 35, ... | NaN | 15602 |
| **3** | False | NaN | 16000000 | [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam... | NaN | 31357 |
| **4** | False | {'id': 96871, 'name': 'Father of the Bride Col... | 0 | [{'id': 35, 'name': 'Comedy'}] | NaN | 11862 |
| **...** | ... | ... | ... | ... | ... | . |
| **45461** | False | NaN | 0 | [{'id': 18, 'name': 'Drama'}, {'id': 10751, 'n... | http://www.imdb.com/title/tt6209470/ | 439050 |
| **45462** | False | NaN | 0 | [{'id': 18, 'name': 'Drama'}] | NaN | 111109 |
| **45463** | False | NaN | 0 | [{'id': 28, 'name': 'Action'}, {'id': 18, 'nam... | NaN | 67758 |
| **45464** | False | NaN | 0 | [] | NaN | 227506 |
| **45465** | False | NaN | 0 | [] | NaN | 461257 |

45466 rows × 24 columns

In [5]:
```
credits
```

Out[5]:

| | cast | crew | id |
|---|---|---|---|
| **0** | [{'cast_id': 14, 'character': 'Woody (voice)',... | [{'credit_id': '52fe4284c3a36847f8024f49', 'de... | 862 |
| **1** | [{'cast_id': 1, 'character': 'Alan Parrish', '... | [{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de... | 8844 |
| **2** | [{'cast_id': 2, 'character': 'Max Goldman', 'c... | [{'credit_id': '52fe466a9251416c75077a89', 'de... | 15602 |
| **3** | [{'cast_id': 1, 'character': "Savannah 'Vannah... | [{'credit_id': '52fe44779251416c91011acb', 'de... | 31357 |
| **4** | [{'cast_id': 1, 'character': 'George Banks', '... | [{'credit_id': '52fe44959251416c75039ed7', 'de... | 11862 |
| **...** | ... | ... | ... |
| **45471** | [{'cast_id': 0, 'character': '', 'credit_id': ... | [{'credit_id': '5894a97d925141426c00818c', 'de... | 439050 |
| **45472** | [{'cast_id': 1002, 'character': 'Sister Angela... | [{'credit_id': '52fe4af1c3a36847f81e9b15', 'de... | 111109 |
| **45473** | [{'cast_id': 6, 'character': 'Emily Shaw', 'cr... | [{'credit_id': '52fe4776c3a368484e0c8387', 'de... | 67758 |
| **45474** | [{'cast_id': 2, 'character': '', 'credit_id': ... | [{'credit_id': '533bccebc3a36844cf0011a7', 'de... | 227506 |
| **45475** | [] | [{'credit_id': '593e676c92514105b702e68e', 'de... | 461257 |

45476 rows × 3 columns

In [6]:
```
keywords
```

Out[6]:

| | id | keywords |
|---|---|---|
| **0** | 862 | [{'id': 931, 'name': 'jealousy'}, {'id': 4290,... |
| **1** | 8844 | [{'id': 10090, 'name': 'board game'}, {'id': 1... |
| **2** | 15602 | [{'id': 1495, 'name': 'fishing'}, {'id': 12392... |
| **3** | 31357 | [{'id': 818, 'name': 'based on novel'}, {'id':... |
| **4** | 11862 | [{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n... |
| **...** | ... | ... |
| **46414** | 439050 | [{'id': 10703, 'name': 'tragic love'}] |
| **46415** | 111109 | [{'id': 2679, 'name': 'artist'}, {'id': 14531,... |
| **46416** | 67758 | [] |
| **46417** | 227506 | [] |
| **46418** | 461257 | [] |

46419 rows × 2 columns

In [7]:
```
movies_mn = movies[['id','genres','original_title','overview','production_companies']]
credits = credits[['id','cast','crew']]
keywords = keywords[['id','keywords']]
print(movies_mn.shape)
```

```
(45466, 5)
```

In [8]:
```
movies_mn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   id                    45466 non-null  object
 1   genres                45466 non-null  object
 2   original_title        45466 non-null  object
 3   overview              44512 non-null  object
 4   production_companies  45463 non-null  object
dtypes: object(5)
memory usage: 1.7+ MB
```

In [9]:
```
movies_mn.isnull().sum()
credits.isnull().sum()
keywords.isnull().sum()
```

Out[9]:
```
id          0
keywords    0
dtype: int64
```

In [10]:
```
print(movies_mn['id'].dtype)
```

```
object
```

In [11]:
```
print(movies_mn['id'][~movies_mn['id'].str.isnumeric()].unique())
```

```
['1997-08-20' '2012-09-29' '2014-01-01']
```

In [12]:
```
movies_mn = movies_mn[pd.to_numeric(movies_mn['id'], errors='coerce').notnull()]
```

In [13]:
```
movies_mn['id'] = movies_mn['id'].apply(lambda x: x if x.isnumeric() else None)
movies_mn['id'] = pd.to_numeric(movies_mn['id'], errors='coerce').fillna(0).astype('Ir
```

In [14]:
```
movies_mn['id'] = movies_mn['id'].fillna(0).astype('Int64')
```

In [15]:
```
merged_data = keywords.merge(movies_mn, on='id')
merged_data = merged_data.merge(credits, on='id')
```

In [16]:
```
import ast
```

In [17]:
```
print(merged_data['genres'].head())
```

```
0    [{'id': 16, 'name': 'Animation'}, {'id': 35, '...
1    [{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
2    [{'id': 10749, 'name': 'Romance'}, {'id': 35, ...
3    [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...
4                      [{'id': 35, 'name': 'Comedy'}]
Name: genres, dtype: object
```

In [18]:
```
def toList(text):
    if isinstance(text, str):
        try:
            parsed = ast.literal_eval(text)
        except (ValueError, SyntaxError):
            return []
    elif isinstance(text, list):
        parsed = text
    else:
        return []

    if isinstance(parsed, list):
        return [item['name'] for item in parsed if isinstance(item, dict) and 'name' i
    return []
```

In [19]:
```
def getDir(obj):
    list = []
    for i in ast.literal_eval(obj):
        if i['job'] == 'Director':
            list.append(i['name'])
            break;
    return list
```

In [20]:
```
merged_data['crew'] = merged_data['crew'].apply(getDir)
```

In [21]:
```
merged_data.head(10)
```

Out[21]:

| | id | keywords | genres | original_title | overview | production_companies | cast | c |
|---|---|---|---|---|---|---|---|---|
| **0** | 862 | [{'id': 931, 'name': 'jealousy'}, {'id': 4290,... | [{'id': 16, 'name': 'Animation'}, {'id': 35, '... | Toy Story | Led by Woody, Andy's toys live happily in his ... | [{'name': 'Pixar Animation Studios', 'id': 3}] | [{'cast_id': 14, 'character': 'Woody (voice)',... | [J Lass |
| **1** | 8844 | [{'id': 10090, 'name': 'board game'}, {'id': 1... | [{'id': 12, 'name': 'Adventure'}, {'id': 14, '... | Jumanji | When siblings Judy and Peter discover an encha... | [{'name': 'TriStar Pictures', 'id': 559}, {'na... | [{'cast_id': 1, 'character': 'Alan Parrish', '... | Johns |
| **2** | 15602 | [{'id': 1495, 'name': 'fishing'}, {'id': 12392... | [{'id': 10749, 'name': 'Romance'}, {'id': 35, ... | Grumpier Old Men | A family wedding reignites the ancient feud be... | [{'name': 'Warner Bros.', 'id': 6194}, {'name'... | [{'cast_id': 2, 'character': 'Max Goldman', 'c... | [Hov Deu |
| **3** | 31357 | [{'id': 818, 'name': 'based on novel'}, {'id':... | [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam... | Waiting to Exhale | Cheated on, mistreated and stepped on, the wom... | [{'name': 'Twentieth Century Fox Film Corporat... | [{'cast_id': 1, 'character': "Savannah 'Vannah... | [Fo Whita |
| **4** | 11862 | [{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n... | [{'id': 35, 'name': 'Comedy'}] | Father of the Bride Part II | Just when George Banks has recovered from his ... | [{'name': 'Sandollar Productions', 'id': 5842}... | [{'cast_id': 1, 'character': 'George Banks', '... | [Cha Sh |
| **5** | 949 | [{'id': 642, 'name': 'robbery'}, {'id': 703, '... | [{'id': 28, 'name': 'Action'}, {'id': 80, 'nam... | Heat | Obsessive master thief, Neil McCauley leads a ... | [{'name': 'Regency Enterprises', 'id': 508}, {... | [{'cast_id': 25, 'character': 'Lt. Vincent Han... | [Mic M |
| **6** | 11860 | [{'id': 90, 'name': 'paris'}, {'id': 380, 'nam... | [{'id': 35, 'name': 'Comedy'}, {'id': 10749, '... | Sabrina | An ugly duckling having undergone a remarkable... | [{'name': 'Paramount Pictures', 'id': 4}, {'na... | [{'cast_id': 1, 'character': 'Linus Larrabee',... | [Syd Poll |
| **7** | 45325 | [] | [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam... | Tom and Huck | A mischievous young boy, Tom Sawyer, witnesses... | [{'name': 'Walt Disney Pictures', 'id': 2}] | [{'cast_id': 2, 'character': 'Tom Sawyer', 'cr... | [P He |
| **8** | 9091 | [{'id': 949, 'name': 'terrorist'}, {'id': 1562... | [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam... | Sudden Death | International action superstar Jean Claude Van... | [{'name': 'Universal Pictures', 'id': 33}, {'n... | [{'cast_id': 1, 'character': 'Darren Francis T... | [P Hya |
| **9** | 710 | [{'id': 701, 'name': | [{'id': 12, 'name': | GoldenEye | James Bond must | [{'name': 'United Artists', 'id': 60}, | [{'cast_id': 1, | [Ma Camp |

| id | keywords | genres | original_title | overview | production_companies | cast | c |
|---|---|---|---|---|---|---|---|
| | 'cuba'}, {'id': 769, | 'Adventure'}, {'id': 28, '... | | unmask the mysterious | {'name'... | 'character': 'James | |

In [22]:
```
def castNames(text):
    list = []
    counter = 0
    for i in ast.literal_eval(text):
        if(counter<5):
            list.append(i['name'])
            counter+=1
        else:
            break
    return list
```

In [23]:
```
merged_data['cast'] = merged_data['cast'].apply(castNames)
```

In [24]:
```
merged_data.head(10)
```

Out[24]:

| | id | keywords | genres | original_title | overview | production_companies | cast | c |
|---|---|---|---|---|---|---|---|---|
| **0** | 862 | [{'id': 931, 'name': 'jealousy'}, {'id': 4290,... | [{'id': 16, 'name': 'Animation'}, {'id': 35, '... | Toy Story | Led by Woody, Andy's toys live happily in his ... | [{'name': 'Pixar Animation Studios', 'id': 3}] | [Tom Hanks, Tim Allen, Don Rickles, Jim Varney... | [. Lass |
| **1** | 8844 | [{'id': 10090, 'name': 'board game'}, {'id': 1... | [{'id': 12, 'name': 'Adventure'}, {'id': 14, '... | Jumanji | When siblings Judy and Peter discover an encha... | [{'name': 'TriStar Pictures', 'id': 559}, {'na... | [Robin Williams, Jonathan Hyde, Kirsten Dunst,... | Johns |
| **2** | 15602 | [{'id': 1495, 'name': 'fishing'}, {'id': 12392... | [{'id': 10749, 'name': 'Romance'}, {'id': 35, ... | Grumpier Old Men | A family wedding reignites the ancient feud be... | [{'name': 'Warner Bros.', 'id': 6194}, {'name'... | [Walter Matthau, Jack Lemmon, Ann-Margret, Sop... | [Hov Deu |
| **3** | 31357 | [{'id': 818, 'name': 'based on novel'}, {'id':... | [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam... | Waiting to Exhale | Cheated on, mistreated and stepped on, the wom... | [{'name': 'Twentieth Century Fox Film Corporat... | [Whitney Houston, Angela Bassett, Loretta Devi... | [Fo Whita |
| **4** | 11862 | [{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n... | [{'id': 35, 'name': 'Comedy'}] | Father of the Bride Part II | Just when George Banks has recovered from his ... | [{'name': 'Sandollar Productions', 'id': 5842}... | [Steve Martin, Diane Keaton, Martin Short, Kim... | [Ch Sh |
| **5** | 949 | [{'id': 642, 'name': 'robbery'}, {'id': 703, '... | [{'id': 28, 'name': 'Action'}, {'id': 80, 'nam... | Heat | Obsessive master thief, Neil McCauley leads a ... | [{'name': 'Regency Enterprises', 'id': 508}, {... | [Al Pacino, Robert De Niro, Val Kilmer, Jon Vo... | [Mic M |
| **6** | 11860 | [{'id': 90, 'name': 'paris'}, {'id': 380, 'nam... | [{'id': 35, 'name': 'Comedy'}, {'id': 10749, '... | Sabrina | An ugly duckling having undergone a remarkable... | [{'name': 'Paramount Pictures', 'id': 4}, {'na... | [Harrison Ford, Julia Ormond, Greg Kinnear, An... | [Syc Pol |
| **7** | 45325 | [] | [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam... | Tom and Huck | A mischievous young boy, Tom Sawyer, witnesses... | [{'name': 'Walt Disney Pictures', 'id': 2}] | [Jonathan Taylor Thomas, Brad Renfro, Rachael ... | [F He |

| | id | keywords | genres | original_title | overview | production_companies | cast | c |
|---|---|---|---|---|---|---|---|---|
| **8** | 9091 | [{'id': 949, 'name': 'terrorist'}, {'id': 1562... | [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam... | Sudden Death | International action superstar Jean Claude Van... | [{'name': 'Universal Pictures', 'id': 33}, {'n... | [Jean-Claude Van Damme, Powers Boothe, Dorian ... | [H Hy |
| **9** | 710 | [{'id': 701, 'name': 'cuba'}, {'id': 769, | [{'id': 12, 'name': 'Adventure'}, {'id': 28, ' | GoldenEye | James Bond must unmask the mysterious | [{'name': 'United Artists', 'id': 60}, {'name'... | [Pierce Brosnan, Sean Bean, Izabella | [M Camp |

In [25]:
```
merged_data['overview'] = merged_data['overview'].astype(str)
merged_data['overview'] = merged_data['overview'].apply(lambda x : x.split())
```

In [26]:
```
def collapse(list):
    collapsed_list = []
    for i in list:
        collapsed_list.append(i.replace(" ",""))
    return collapsed_list
```

In [27]:
```
merged_data['keywords'] = merged_data['keywords'].apply(collapse)
merged_data['genres'] = merged_data['genres'].apply(collapse)
merged_data['crew'] = merged_data['crew'].apply(collapse)
merged_data['cast'] = merged_data['cast'].apply(collapse)
```

In [28]:
```
merged_data.head()
```

Out[28]:

| | id | keywords | genres | original_title | overview | production_companies | cast | |
|---|---|---|---|---|---|---|---|---|
| **0** | 862 | [[, {, ', i, d, ', :, , 9, 3, 1, ,, , ', n, a,... | [[, {, ', i, d, ', :, , 1, 6, ,, , ', n, a, m,... | Toy Story | [Led, by, Woody,, Andy's, toys, live, happily,... | [{'name': 'Pixar Animation Studios', 'id': 3}] | [TomHanks, TimAllen, DonRickles, JimVarney, Wa... | [Joh |
| **1** | 8844 | [[, {, ', i, d, ', :, , 1, 0, 0, 9, 0, ,, , ',... | [[, {, ', i, d, ', :, , 1, 2, ,, , ', n, a, m,... | Jumanji | [When, siblings, Judy, and, Peter, discover, a... | [{'name': 'TriStar Pictures', 'id': 559}, {'na... | [RobinWilliams, JonathanHyde, KirstenDunst, Br... | [Joe |
| **2** | 15602 | [[, {, ', i, d, ', :, , 1, 4, 9, 5, ,, , ', n,... | [[, {, ', i, d, ', :, , 1, 0, 7, 4, 9, ,, , ',... | Grumpier Old Men | [A, family, wedding, reignites, the, ancient, ... | [{'name': 'Warner Bros.', 'id': 6194}, {'name'... | [WalterMatthau, JackLemmon, Ann-Margret, Sophi... | [Howa |
| **3** | 31357 | [[, {, ', i, d, ', :, , 8, 1, 8, ,, , ', n, a,... | [[, {, ', i, d, ', :, , 3, 5, ,, , ', n, a, m,... | Waiting to Exhale | [Cheated, on,, mistreated, and, stepped, on,, ... | [{'name': 'Twentieth Century Fox Film Corporat... | [WhitneyHouston, AngelaBassett, LorettaDevine,... | [Fores |
| **4** | 11862 | [[, {, ', i, d, ', :, , 1, 0, 0, 9, ,, , ', n,... | [[, {, ', i, d, ', :, , 3, 5, ,, , ', n, a, m,... | Father of the Bride Part II | [Just, when, George, Banks, has, recovered, fr... | [{'name': 'Sandollar Productions', 'id': 5842}... | [SteveMartin, DianeKeaton, MartinShort, Kimber... | [Cha |

In [29]:
```
merged_data['collection'] = merged_data['keywords'] + merged_data['genres'] + merged_d
```

In [30]:
```
new_data = merged_data[['id','original_title', 'collection']].head(25000)
new_data
```

Out[30]:

| | id | original_title | collection |
|---|---|---|---|
| **0** | 862 | Toy Story | [[, {, ', i, d, ', :, , 9, 3, 1, ,, , ', n, a,... |
| **1** | 8844 | Jumanji | [[, {, ', i, d, ', :, , 1, 0, 0, 9, 0, ,, , ',... |
| **2** | 15602 | Grumpier Old Men | [[, {, ', i, d, ', :, , 1, 4, 9, 5, ,, , ', n,... |
| **3** | 31357 | Waiting to Exhale | [[, {, ', i, d, ', :, , 8, 1, 8, ,, , ', n, a,... |
| **4** | 11862 | Father of the Bride Part II | [[, {, ', i, d, ', :, , 1, 0, 0, 9, ,, , ', n,... |
| **...** | ... | ... | ... |
| **24995** | 43193 | Dangerous When Wet | [[, {, ', i, d, ', :, , 4, 3, 4, 4, ,, , ', n,... |
| **24996** | 70512 | L'inferno | [[, {, ', i, d, ', :, , 6, 1, 5, 4, ,, , ', n,... |
| **24997** | 73103 | Dante's Inferno | [[, ], [, {, ', i, d, ', :, , 1, 6, ,, , ', n,... |
| **24998** | 38471 | Dark Alibi | [[, {, ', i, d, ', :, , 3, 7, 8, ,, , ', n, a,... |
| **24999** | 73997 | David Spade: Take the Hit | [[, ], [, {, ', i, d, ', :, , 3, 5, ,, , ', n,... |

25000 rows × 3 columns

In [31]:
```
new_data['collection'] = new_data['collection'].apply(lambda x: " ".join(x))

new_data['collection'] = new_data['collection'].apply(lambda x: x.lower())

new_data.head()
```

Out[31]:

| | id | original_title | collection |
|---|---|---|---|
| **0** | 862 | Toy Story | [{'id':931,'name':'jea... |
| **1** | 8844 | Jumanji | [{'id':10090,'name':'b... |
| **2** | 15602 | Grumpier Old Men | [{'id':1495,'name':'fi... |
| **3** | 31357 | Waiting to Exhale | [{'id':818,'name':'bas... |
| **4** | 11862 | Father of the Bride Part II | [{'id':1009,'name':'ba... |

In [32]:
```
import nltk
```

In [33]:
```
from nltk.stem.porter import PorterStemmer
```

In [34]:
```
ps = PorterStemmer()
def stem(text):
    y = []

    for i in text.split():
        y.append(ps.stem(i))

    return " ".join(y)
```

In [35]:
```
new_data['collection'] = new_data['collection'].apply(stem)
```

In [36]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [37]:
```
cv = CountVectorizer(max_features=5000,stop_words='english')
```

In [38]:
```
vector = cv.fit_transform(new_data['collection']).toarray()
```

In [39]:
```
vector.shape
```

Out[39]:
```
(25000, 5000)
```

In [40]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

In [41]:
```
similarity = cosine_similarity(vector)
```

In [42]:
```
def recommend(movie):
    index = new_data[new_data['original_title'] == movie].index[0]
    distances = similarity[index]
    most_similar_movies = sorted(list(enumerate(distances)),reverse=True,key = lambda

    for i in most_similar_movies:
        print(new_data.iloc[i[0]].original_title)
```

In [43]:
```
recommend('Father of the Bride Part II')
```

```
Babbitt
Father of the Bride
About Schmidt
All Night Long
Zero Bridge
```

In [44]:
```
import pickle
pickle.dump(new_data.to_dict(), open('final-movie-dict.pkl', 'wb'))
pickle.dump(similarity,open('final-similarity.pkl', 'wb'),protocol = 4)
```

In [ ]:
```

```