In [1]:
```
import pandas as pd
import numpy as np

# For counting the frequency of words
from sklearn.feature_extraction.text import CountVectorizer

# For finding the difference between angles of the count vectors
from sklearn.metrics.pairwise import cosine_similarity

df = pd.read_csv("movie_dataset.csv")
```

In [4]:
```
# Lets have a quick glance of the movies dataset
df.head(3)
```

Out[4]:

	index	budget	genres	homepage	id	keywords	origina
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...	
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6	

3 rows × 24 columns

In [5]:
```
# Just taking the columns that will be used for finding the cosine similarity.
features = ['keywords','cast','genres','director']
```

In [6]:
```
# Combining the relevant features before count vectorization
def combine_features(row):
    return row['keywords']+" "+row['cast']+" "+row['genres']+" "+row['director']
```

In [20]:
```
for feature in features:
    df[feature] = df[feature].fillna('') #filling all NaNs with blank string

df["combined_features"] = df.apply(combine_features,axis=1)
```

In [8]:
```
# Ensuring everything is combined properly
df.iloc[0].combined_features
```

Out[8]:
```
'culture clash future space war space colony society Sam Worthington Zoe Saldan
a Sigourney Weaver Stephen Lang Michelle Rodriguez Action Adventure Fantasy Sci
ence Fiction James Cameron'
```

In [15]:
```
cv = CountVectorizer() #creating new CountVectorizer() object
count_matrix = cv.fit_transform(df["combined_features"]) # Count matrix consistir
count_matrix
```

Out[15]:
```
<4803x14845 sparse matrix of type '<class 'numpy.int64'>'
        with 97547 stored elements in Compressed Sparse Row format>
```

In [18]:
```
# Finding the angular difference between the count vectors of each of the movies.
cosine_sim = cosine_similarity(count_matrix)
cosine_sim
```

Out[18]:
```
array([[1.        , 0.10540926, 0.12038585, ..., 0.        , 0.        ,
        0.        ],
       [0.10540926, 1.        , 0.0761387 , ..., 0.03651484, 0.        ,
        0.        ],
       [0.12038585, 0.0761387 , 1.        , ..., 0.        , 0.11145564,
        0.        ],
       ...,
       [0.        , 0.03651484, 0.        , ..., 1.        , 0.        ,
        0.04264014],
       [0.        , 0.        , 0.11145564, ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.04264014, 0.        ,
        1.        ]])
```

In [11]:
```
# Helper functions
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]

def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
```

In [21]:
```
movie_user_likes = "Avatar" # Movie on the basis of which recommendation will be
movie_index = get_index_from_title(movie_user_likes)
similar_movies = list(enumerate(cosine_sim[movie_index])) # Cosine similarity of
similar_movies
```

Out[21]:
```
[(0, 1.0000000000000004),
 (1, 0.10540925533894599),
 (2, 0.12038585308576921),
 (3, 0.03774256780481986),
 (4, 0.23094010767585033),
 (5, 0.1924500897298753),
 (6, 0.0),
 (7, 0.1405456737852613),
 (8, 0.08206099398622181),
 (9, 0.11785113019775793),
 (10, 0.23094010767585035),
 (11, 0.07698003589195011),
 (12, 0.12038585308576921),
 (13, 0.11547005383792516),
 (14, 0.181848241863327),
 (15, 0.07548513560963972),
 (16, 0.1382602259640567),
 (17, 0.12309149097933272),
 (18, 0.1405456737852613),
```

In [22]:
```
# Lets sort the similar movies based on cosine similarity in descending order
sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)[1:]
sorted_similar_movies
```

Out[22]:
```
[(94, 0.42339019740572564),
 (2403, 0.3774256780481986),
 (3208, 0.3464101615137755),
 (47, 0.34426518632954817),
 (56, 0.33596842045264647),
 (3158, 0.3333333333333337),
 (2198, 0.31426968052735443),
 (2696, 0.30792014356780045),
 (4401, 0.28867513459481287),
 (1531, 0.2858966759567453),
 (278, 0.2810913475705226),
 (1053, 0.2809003238667948),
 (239, 0.2765204519281134),
 (838, 0.2749859704614352),
 (61, 0.27498597046143514),
 (232, 0.2694301256218254),
 (4332, 0.2694301256218254),
 (661, 0.264197974633739),
 (4593, 0.264197974633739),
```

In [14]:
```
# Finally, lets print the top 10 similar movies to Avatar
# We are hoping for movies with genre action, adventure, fantasy, science-fictior
i=0
print("Top 5 similar movies to "+movie_user_likes+" are:\n")
for element in sorted_similar_movies:
    print(get_title_from_index(element[0]))
    i=i+1
    if i>5:
        break
```

```
Top 5 similar movies to Avatar are:

Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
Alien
```

In [ ]: