Hybrid Content-based Movie Recommender System

Project Group 2
Lihua Pei
Xiao Han
Siqi Guan
Github Link: https://github.com/hanxiao0607/Movie-Recommendation-System

***Abstract*—Recommender Systems play a more and more important role in our lives. Currently, they are not optimal. In our project, we believe the visual feeling of movies can be extracted out as great features that can make a difference for different movies. Therefore, we try to build the hybrid content-based movie recommender system by adding information from movie trailers and movie synopsis.**

Keywords—recommender system, RGB, TF-IDF, OpenCV, cosine similarity

I. Introduction

A recommendation system is a kind of information filtering system which is trying to predict the preferences of a user, and make suggestions based on these preferences.

There is a wide use for recommendation systems. These have become explode developed over the last few decades and are currently optimized in most online applications that we use like Google, Facebook, Amazon, Netflix. The content of such platforms covers various fields like movies, videos, music, books, and products. Social network from social media platforms, products recommendation on e-commerce websites, the match couples on dating websites and search engine results returned on Google.

The movie recommender system can be divided into two major groups, which shows in fig. 1, the content-based filtering system known as cold start recommender system and collaborative filtering system known as warm start system.

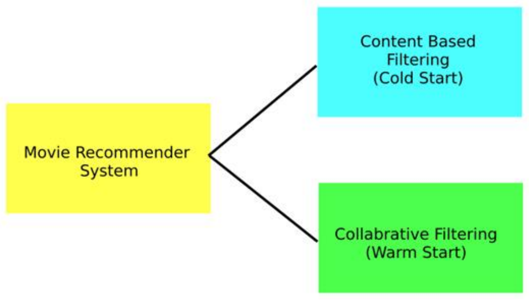
Movie Recommender System
Content Based Filtering (Cold Start)
Collabrative Filtering (Warm Start)

Fig. 1 Two filtering system

Fig. 2 shows Collaborative Filter system, which maintains a database of many users' rating of a variety of items which makes use of the user data and ignoring content. Almost all existing commercial recommender use this approach because collaborative filter system gives an appropriate suggestion based on users.

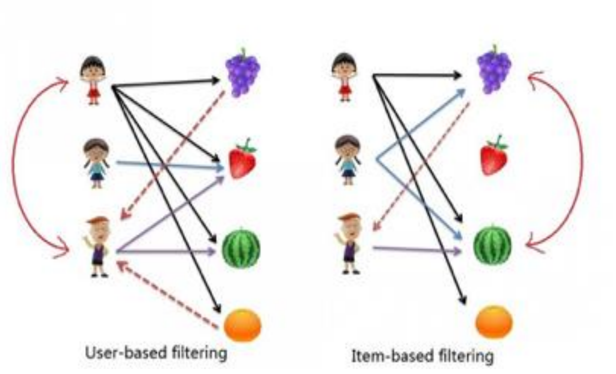

Fig. 2 An example of collaborative filtering [1]

However, the collaborative filter system has the limitation when it is lack of user data such as a new movie website have no or not enough users data, a new movie comes out which have no any user previously rated and a new user's first click.

Content-based Filtering system can be a solution when there is not enough user data because Content-based Filtering system only use the data from each movie and assumed to operate user independently.

Content Based: The recommendation system recommends other movies which are similar to what selected movie.

$$f(\text{movie}) \rightarrow \{\text{similar movies}\}$$

Collaborative: The recommendation system recommends movies which are rated highly by similar users.

$$f(\text{movies, user}) \rightarrow \{\text{similar movies based on users}\}$$

Currently most commonly used content of a movie are genres, director, actors etc. Therefore, we believe adding movie trailer RGB information and IMDB movie synopsis to build a hybrid content-based movie recommender system can make a difference and give a better recommendation.

In section II, we will explain data collection and processing. In section III, we will show feasibility test before we build our model. The details about our model shows in section IV. For the last two section, we will give a conclusion and talk about future works.

II. DATA COLLECTION AND PROCESSING

In our project, we use movie trailers RGB information to improve the result of content-based recommender system. The structure of our dataset shows in fig. 3. There is no existing dataset contains movie trailer RGB information. We collect our own dataset by using YouTube API and IMDB API. In "MMTF-14K: A Multifaceted Movie Trailer Dataset for Recommendation and Retrieval" [2], Y. Deldjoo, M. G. Constantin, B. Ionescu, M. Schedl, and P. Cremonesi create a list of 25,623 movies dataset. There are three columns "youtubeId", "movieId", and "title" in this dataset. "youtubeId" is a string contains part of movie trailer's URL. We concatenate 'https://www.youtube.com/watch?v=' with youtubeId in this dataset to get the full URL for each movie trailer. "movieId" is a unique number for each movie in this dataset. The movieId is generated by the author of "MMTF-14K". We don't use this column in our project. "title" is string which contains title and year of movie. We use title to search on IMDB to get other information of the movie.

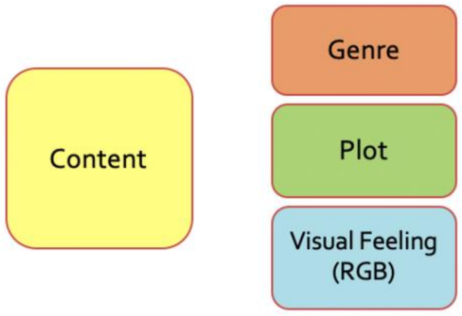

Fig. 3 Dataset structure

Fig. 5 shows the process of data collection. The first step of data collection, we use YouTube API (pytube) to download the movie trailer with the full URL for each movie. With the movie trailer, we use OpenCV (cv2) to capture video and get the total number of frames in the movie trailer. We have tried four different methods to get the RGB information. First, using only one image, such as movie poster, for each movie. In this method, we only get a little information about the movie. The variance is not significant when we labelled the movie with genres and try to show the difference by using one image RGB information. Then, we come up with the other two methods getting a frame every 15 seconds or getting a frame every 5 frames. Both these two methods have a problem that there would be lots of black frame and the frame we get probably not important. At last, we extract all the frame except the first 5% and the last 5% of the movie trailer. The reason why we skip the first and last parts is that the first part of the movie trailer only contains producer information which is not related to the movie. The last 5% of the movie trailer usually contains the time when the movie will be released. With the rest of the frames, we use Python Imaging Library (PIL) to get every frame RGB mean, median, and variance and store as list. We create 27 new features which show in Appendix I.

The second step of data collection, we use IMDB API (imdb) to collect information, such as genres, director, actors, runtime, and country. For each movie, we only extract the first director, top 3 actors. In IMDB, some movies will have more than one synopsis written by different people. Our code will collect all the synopsis and discard the name of the writer. After collection, we create dummy variables for genres, actors, director, and country and transforming synopsis into sparse matrix by using tf-idf. Our pilot dataset has 7,275 movies information with 55 columns.

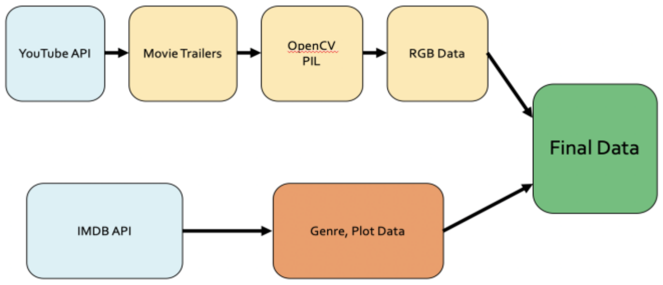

Fig. 4 Data collection process

III. PRE-TEST

We do a pre-test to find out whether RGB information is useful or not. We randomly select 500 movies from our dataset. In this dataset, there are 95 movies have "Action" label while not having "Drama" label. There are 163 movies have "Drama" label but not have "Action" label. We include all RGB information, actors, director, runtime, and country in this dataset. Then transforming actors, director, and country into dummy variables. The dimensionality become high due to actors' name and director name. We do PCA to check how many principal components will efficiently discriminate the variance of these two labelled groups of movies. Fig. 5 shows the result of scree plot for all the principal components.

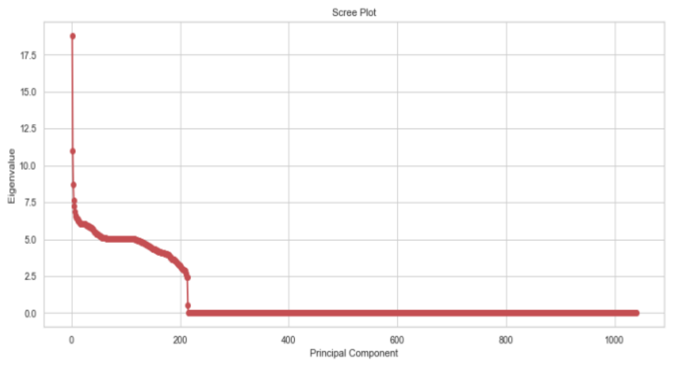

Fig. 5 PCA scree plot

According to elbow method, we select 220 principal components for the next step. The cumulative variance explained shows in fig. 6.

We sort the principal components according to variance explained. The first 3 principal components show in appendix II. RGB information have high weights in first 3 principal components in different ways. It means RGB information is useful to discriminate movies.

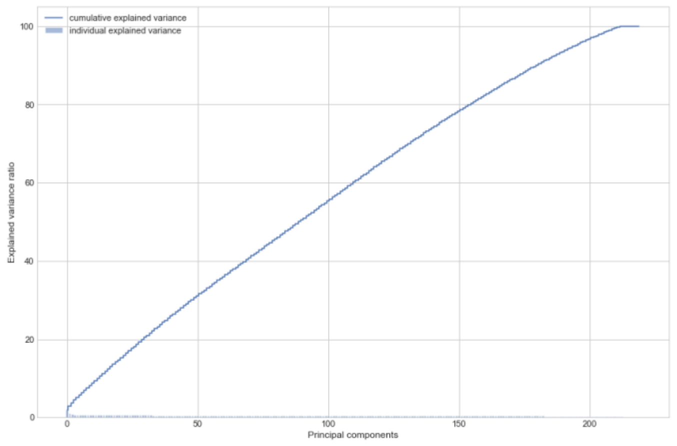

Fig. 6 Cumulative variance explained

IV. Model

The process of our model shows in fig. 7. We do data preprocessing after we collect our dataset. We give the processed data as input for our model. The model will return different cosine similarity matrix as output. User will give input a movie name and number of how many similar movies do they want to return to the filter. The filter will return multi lists of movies based on different similarity matrix.

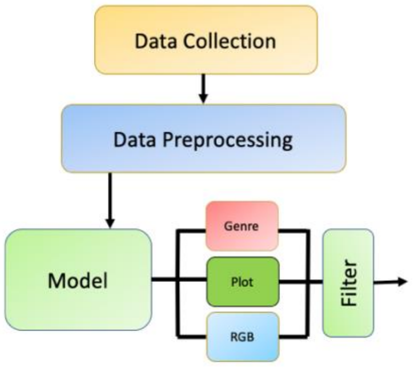

Fig. 7 Model flow diagram

We use cosine similarity to compute our similarity matrix. The function of cosine similarity shows below. The reason why we want to use cosine similarity is that it will return similarity value between two movies from 1 to -1.

$$similarity = cos(\theta) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

A. *Movie synopsis model*

For movie synopsis model, we only use movie synopsis as the only variable. The similarity matrix is calculated by using tfidf. The result shows in section V. There is a big problem for this model. If two movies have the same main character name, the similarity will be high regardless of other conditions.

B. *Genres model*

For genres model, we only include genres, director, actors, runtime, and country. This model works better than synopsis model, while still has a problem. For example, two movies will get high similarity when director, actors, runtime, and country are similar, even if the genres are not the same.

C. *Movie synopsis plus genres model*

For movie synopsis plus genres model, we combine genres and synopsis together. The result of this model is better than the first two models. However, this model still has the same problems as the first two models.

D. *Hybrid mode*

Finally, we combine synopsis, genres, and RGB into one model. This model has a big improvement than the others. RGB information will improve the recommendation with same color saturation, brightness, and hue. For instance, two movies with the same main character name and the same genres, but one is darkness style and one is brightness style, our hybrid model will return lower similarity value than the others.

V. Result

Fig. 8 represent the performance of various content-based models' recommendation for movie Toy Story 1. Fig. 9 shows the RGB information of movies Sherk, Hook and Malice with movie Toy Story 1. The left histogram generally shows a feeling of movies. Shrek and Toy story 1 are much lighter than movie Hook and Malice. The Right histogram present the fluctuate through the movie trailers. The movie Malice have much low fluctuate than other movies.

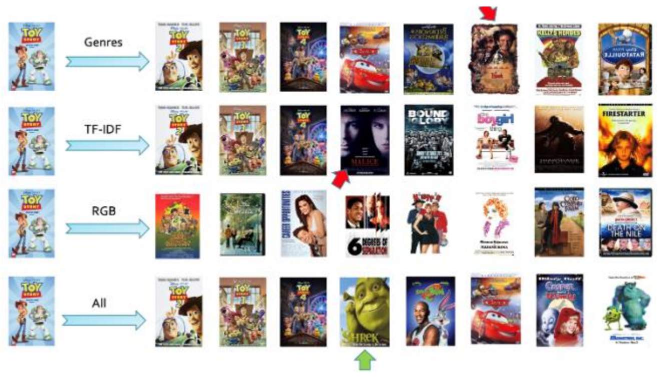

Fig. 8 Recommended movies from different models

By inspecting into the result, the RGB model's recommendations seems very random because movies may land in the same Hue, Temperature, and Lightness range but they are totally unrelated. The model solely depend on RGB information is very bias, but we find out RGB information can be treated as a great regularizer. The genres model gives most reasonable result but there is one bias Hook. Hook is the adventure movies which is same as the Toy Story 1, but the visual feeling of Hook is dark and depressed which is opposite with Toy Story 1's visual feeling which is illustrious and relax. It is very difficult for genre model to make difference between Toy Story 1 and Hook. However, we add RGB information as the regularizer to decrease the similarity between Toy Story 1 and Hook. Then, our hybrid content-based model replaces movie hook by the

Shrek which have more bright and relaxed visual feeling. The TF-IDF model is very sensitive by the character name so the TF-IDF model give Malice and Toy story very high similarity. However, by adding RGB information as regularizer, Malice will be taken out recommend list.

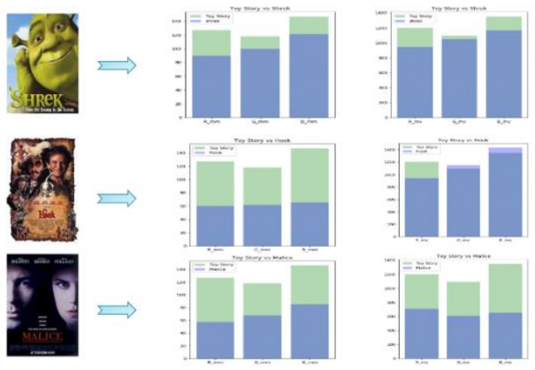

Fig. 9 RGB information for 3 movies vs Toy Story

VI. Conclusion

A. Summary

We collect our own dataset with movie trailers RGB information. In section II, we proved RGB information will improve the performance of normal content-based movie recommendation system. We build the hybrid contented-based movie recommendation system with genre, synopsis and RGB information. We find out that the RGB information can be a great regularizer to improve the result from color hue, brightness, and saturation.

B. Future Work

For further study, we will try to improve our model in several methods. First, we will collect more movies data by using our code. We will build a database to store movies' details, such as poster, trailer, genres, and other information. With more data points in our database, our model will give a better result. Second, we will improve the method of extracting RGB features for movie trailers. For now, we plan to use OpenCV with deep learning, such as R-CNN, to extracting object in movie trailers. Then using extracted object to compute RGB information. At last, combine collaborative filtering recommender into our model. For the movie with enough users information, collaborative filtering recommender will give a better result. We want to use users data to generative labels to re-weight different similarity matrix in our hybrid model by using boosting.

References

[1] J. Le, "The 4 Recommendation Engines That Can Predict Your Movie Tastes," Medium, 11-Jun-2018. [Online]. Available: https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-109dc4e10c52. [Accessed: 12-Dec-2019].

[2] Y. Deldjoo, M. G. Constantin, B. Ionescu, M. Schedl, and P. Cremonesi, "MMTF-14K: A Multifaceted Movie Trailer Dataset for Recommendation and Retrieval," Proceedings of the 9th ACM Multimedia Systems Conference on - MMSys 18, 2018.

APPENDIX I - CREATED RGB FEATURES

Feature	Meaning	Feature	Meaning
r_mm	Mean of all frames' mean of all pixels on each frame of red color	r_mem	Mean of all frames' median of all pixels on each frame of red color
g_mm	Mean of all frames' mean of all pixels on each frame of green color	g_mem	Mean of all frames' median of all pixels on each frame of green color
b_mm	Mean of all frames' mean of all pixels on each frame of blue color	b_mem	Mean of all frames' median of all pixels on each frame of blue color
r_mme	Median of all frames' mean of all pixels on each frame of red color	r_meme	Median of all frames' median of all pixels on each frame of red color
g_mme	Median of all frames' mean of all pixels on each frame of green color	g_meme	Median of all frames' median of all pixels on each frame of green color
b_mme	Median of all frames' mean of all pixels on each frame of blue color	b_meme	Median of all frames' median of all pixels on each frame of blue color
r_mv	Variance of all frames' mean of all pixels on each frame of red color	r_mev	Variance of all frames' median of all pixels on each frame of red color
g_mv	Variance of all frames' mean of all pixels on each frame of green color	g_mev	Variance of all frames' median of all pixels on each frame of green color
b_mv	Variance of all frames' mean of all pixels on each frame of blue color	b_mev	Variance of all frames' median of all pixels on each frame of blue color
r_vm	Mean of all frames' variance of all pixels on each frame of red color	b_vme	Median of all frames' variance of all pixels on each frame of blue color
g_vm	Mean of all frames' variance of all pixels on each frame of green color	r_vv	Variance of all frames' variance of all pixels on each frame of red color
b_vm	Mean of all frames' variance of all pixels on each frame of blue color	g_vv	Variance of all frames' variance of all pixels on each frame of green color
r_vme	Median of all frames' variance of all pixels on each frame of red color	b_vv	Variance of all frames' variance of all pixels on each frame of blue color
g_vme	Median of all frames' variance of all pixels on each frame of green color		

Appendix II - Top 3 Principal Components for Pre-test

	PC1	**PC2**	**PC3**
duration	(-0.0037980820661461545+0j)	(-0.006723295910071406+0j)	(-0.04682691902544083+0j)
facenumber_in_poster	(0.009070246210905141+0j)	(-0.05260873897971741+0j)	(-0.04225263094762165+0j)
budget	(-0.0017440264462187933+0j)	(0.03534807406036156+0j)	(0.03696577222371753+0j)
imdb_score	(-0.012414518898765596+0j)	(-0.010855175531016116+0j)	(-0.07098145865650343+0j)
r_mm	(0.21134296322833135+0j)	(-0.05231254051673768+0j)	(0.04605874925861751+0j)
g_mm	(0.2123524986032918+0j)	(-0.09367292567243639+0j)	(0.031115914109731177+0j)
b_mm	(0.1968440118553205+0j)	(-0.1145727402338734+0j)	(-0.01181347479398449+0j)
r_mme	(0.2070665777338289+0j)	(-0.08675455938849035+0j)	(0.011601381256747381+0j)
g_mme	(0.2039312301065728+0j)	(-0.12197368529792096+0j)	(-0.0060660509841182+0j)
b_mme	(0.19062969614888667+0j)	(-0.13359576970196388+0j)	(-0.037871138199143416+0j)
r_mv	(0.12008129670421146+0j)	(0.16130717021588378+0j)	(0.18458803535352866+0j)
g_mv	(0.12531592355903545+0j)	(0.15038395093818485+0j)	(0.203487862848336+0j)
b_mv	(0.11479553132056429+0j)	(0.1461683531803104+0j)	(0.1864978023920877+0j)
r_mem	(0.20101045635301157+0j)	(-0.08344671491049621+0j)	(0.072814081658296+0j)

g_mem	(0.20020899970208483+0j)	(-0.11758130776104173+0j)	(0.06333572185586944+0j)
b_mem	(0.18968883271170392+0j)	(-0.13225592850445467+0j)	(0.012123898224387043+0j)
r_meme	(0.18660803820201846+0j)	(-0.13885505896894398+0j)	(0.03697330985151612+0j)
g_meme	(0.1834358260441276+0j)	(-0.16094403662109943+0j)	(0.026350383790370503+0j)
b_meme	(0.17492183797052407+0j)	(-0.1646493242529338+0j)	(-0.015657630668147977+0j)
r_mev	(0.14009968032216+0j)	(0.16595590334598556+0j)	(0.13777849708356515+0j)
g_mev	(0.15023300350749536+0j)	(0.1623506490156173+0j)	(0.15241905429223354+0j)
b_mev	(0.13695752055872007+0j)	(0.16506799448626128+0j)	(0.12918523522262731+0j)
r_vm	(0.16265747964480978+0j)	(0.1400083238949634+0j)	(-0.12879938834254442+0j)
g_vm	(0.16494299284389116+0j)	(0.12466103896871128+0j)	(-0.1757675753749053+0j)
b_vm	(0.1351101632251054+0j)	(0.1112168309744505+0j)	(-0.2119376536789359+0j)
r_vme	(0.16994436935820537+0j)	(0.09326195328020566+0j)	(-0.13798175176857744+0j)
g_vme	(0.17025173397473276+0j)	(0.07991424298032127+0j)	(-0.1810610734667323+0j)
b_vme	(0.13738896883059504+0j)	(0.07802952583130836+0j)	(-0.21337739255508062+0j)
r_vv	(0.09144618079051502+0j)	(0.2133266229678286+0j)	(-0.026000394163430227+0j)
g_vv	(0.09939880701222144+0j)	(0.22015198605510822+0j)	(-0.035469321517285504+0j)

actor_3_name_Steven Weber	(0.049297118069184895+0j)	(0.0075328581333670335+0j)	(-0.03595137876468093+0j)
actor_3_name_Stockard Channing	(0.005077461918340552+0j)	(-0.02458116987431363+0j)	(-0.005464206350798445+0j)
actor_3_name_Tatyana Ali	(-0.008491055463441971+0j)	(0.0030936541010074207+0j)	(0.02444881223958326+0j)
actor_3_name_Theodore Bikel	(-0.0014882221189250913+0j)	(-0.002209076315856099+0j)	(-0.03843794914914283+0j)
actor_3_name_Thomas Mitchell	(-0.024656613528766306+0j)	(-0.001983127307041592+0j)	(-0.050640875061938796+0j)
actor_3_name_Tom Atkins	(-0.023661560090547772+0j)	(0.0345660250621482+0j)	(-0.00453714513176161+0j)
actor_3_name_Troy Garity	(-0.022962105641040058+0j)	(-0.004941716689144223+0j)	(-0.005762311436197472+0j)
actor_3_name_Vanessa Martinez	(0.0026674845278602836+0j)	(0.015594809197971926+0j)	(-0.006767416508139398+0j)
actor_3_name_Victor Buono	(0.022100928304923894+0j)	(0.02762132516213242+0j)	(-0.06454004967372096+0j)
actor_3_name_Victor Wong	(0.01888436825403213+0j)	(0.003796355826469826+0j)	(-0.002085486452609015+0j)
actor_3_name_Vincent Pastore	(0.008838786645132367+0j)	(-0.03266973509362572+0j)	(-0.02833287363092719+0j)
actor_3_name_Vincent Schiavelli	(0.001321066296802+0j)	(-0.004745922535754577+0j)	(-0.028443268506956068+0j)
actor_3_name_Will Patton	(-0.0029476477402538333+0j)	(0.021809270967332758+0j)	(0.010393100839534718+0j)

actor_3_name_William Devane	(-0.007259367599492985+0j)	(0.006180516626018515+0j)	(-0.008412129474436312+0j)
actor_3_name_William Holden	(-0.01472079118478584+0j)	(0.002733021169483535+0j)	(0.010394829911982545+0j)
actor_3_name_William Hootkins	(-0.00024413706311280379+0j)	(0.010651758622743045+0j)	(0.014289762865415822+0j)
actor_3_name_William Windom	(-0.001079532765774481+0j)	(-0.026597608775051445+0j)	(-0.034750324441917634+0j)
country_Aruba	(0.01153276392962419+0j)	(-0.003046591893696859+0j)	(0.03403642262545592+0j)
country_Australia	(0.011999820680477547+0j)	(-0.07647733075467038+0j)	(0.047009154191341676+0j)
country_Canada	(-0.007213592165833629+0j)	(0.032060357570045335+0j)	(0.021807094197453268+0j)
country_France	(0.010227617680776466+0j)	(-0.04362003941533519+0j)	(-0.0007122363127119183+0j)
country_Germany	(0.022545609641595493+0j)	(-0.029195610019711892+0j)	(-0.012722120701604722+0j)
country_Italy	(-0.002613534862544999+0j)	(-0.04686591569830055+0j)	(0.004055395627841391+0j)
country_Japan	(-0.01944839772444475+0j)	(-0.01714284042993241+0j)	(0.009071547512192324+0j)
country_Netherlands	(-0.01056453651193511+0j)	(0.015889097459020864+0j)	(0.02574257944699737+0j)
country_New Zealand	(0.005722550680480128+0j)	(-0.012222258670128962+0j)	(0.06227392185926041+0j)
country_Norway	(-0.020781738985339027+0j)	(-0.008912294218565125+0j)	(-0.014708360110967834+0j)
country_Spain	(-0.008468534865951527+0j)	(0.01028436975937792+0j)	(-0.007402942705266409+0j)

country_UK	(-0.017721876647460888+0j)	(0.004338533615724356+0j)	(-0.015035793046688435+0j)
country_USA	(0.008263852252596942+0j)	(0.04741427153498175+0j)	(-0.031385622560226305+0j)