CMPT 353 – Computational Data Science

Group Project: Wikidata, Movies, and Success

Presented by:

Zihan Wang 301329429

Chen Zhao 301308092

Aug 1, 2019

Index

Problems Addressed

In this project, we are interested in these questions:

1. Do the various criteria for success (critic reviews, audience reviews, profit / loss) correlate with each other?
2. How can we measure whether a movie is successful or not? Can we predict these types of success criteria using some related information? (e.g. director, cast member, genre, production company)
3. Can we use the plot summary to predict the genre of a movie? Does plot summary predict success in any useful way?
4. Have any of the information in our data changed over time (depending on the movie's release date)?
5. (Made up by us) What is the sentiment score for each genre of movies, based on their plot summaries? Are the movies positive or negative through the plot summary?

General Data Acquisition

We are actually given 4 json zip files: "wikidata-movies", "rotten-tomatoes", "omdb-data" and "genres" as well as 3 .py code for generating "wiki" and "omdb". The first problem that we met was like, there is no way to calculate the profit of a movie. Hence, to get more data that we need:

- We added "take_first(wd, 'production_company'), wd['nbox'], wd['ncost']" to "build_wikidata_movies.py". Then we run both "transform_download.py" and updated "build_wikidata_movies.py" together on HDFS cluster and copied the finished zip file to local. Originally "wikidata-movies" has 40,430 rows. For our new data "wiki-company", it has 41,557 rows with 3 columns more. This is not bad though.
- As our instructor Greg suggests, we do not modify or try to get more data for the rest data files. They are good enough for analysis.

1. Correlations Discovery

Problem Refined

The first question could be refined to “whether these attributes (critic reviews, audience reviews, profit / loss) have a linear relation with each other”. Additionally, we broaden the question a little bit, to “how well can we use practical machine learning models to predict one attribute (say, audience ratings), through input the other attributes (e.g. critic ratings)?”

Data Cleaning

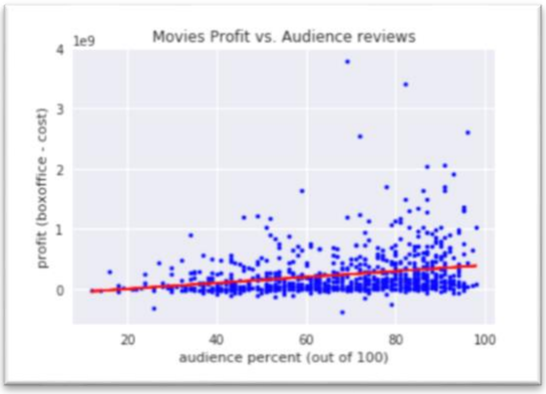
Have a look at 3 zip files, we can get “audience_average”, “audience_percent”, “audience_ratings”, “critic_average”, “critic_percent” directly after joining (merge) tables “wiki-company” and “rotten-tomatoes” together, based on matched rotten_tomatoes_id. Later, we drop all the NaN values in these columns. To filter really uncommon movies, we choose movies with >= 40 “audience_ratings”. After doing these, the joined table has 817 movies left.

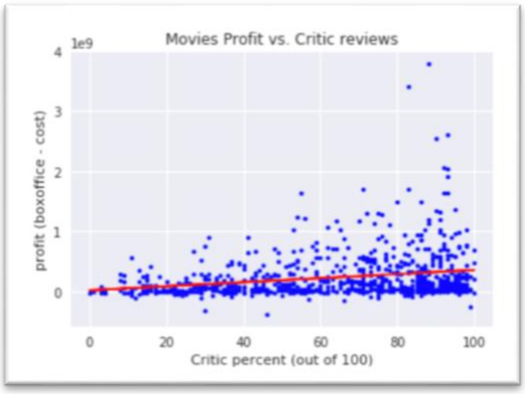
To get the profit / loss, it looks like we can just simply use the column “nbox” subtract “ncost”. That number is based on US dollars. However, then we realized a problem that, **the profit / loss we get is based on the movie’s publication year**. The money value is actually changing by inflation over the years. For example, if movie A made $10 profit in 1950 and movie B made $15 profit in 2000, actually movie A made more profit because $10 in 1950 is worth $16 in 2000. Hence, we perform a correction on profit based on the average inflation rate over the years to make the profit statistics more reliable.

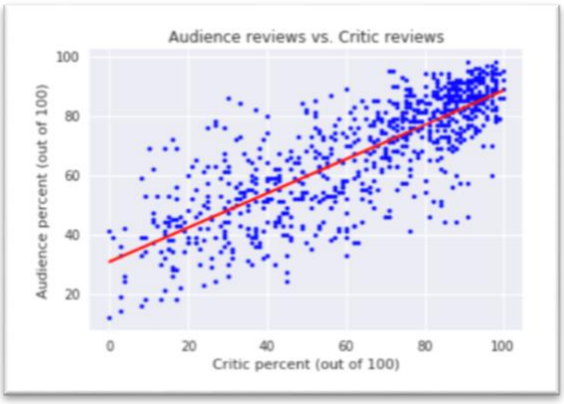
By the min() function, we find that the oldest movie is from 1927. Through the research on inflation rates, we find that from 1927 to 2019, the value of US dollars has an inflation rate of 2.97% per year on average. We neglect only a few years with very little deflation. Then we define a function to calculate the “real value of US dollar” based on year 2019. Through the money value on each year, finally we get the “real profit” for each movie, based on their publication years.

Linear Regression

We first do a linear regression test on these 3 variables: critic reviews (percent), audience reviews (percent) and profit. By calling the stats.linregress and print p.value as well as r.value, we get:





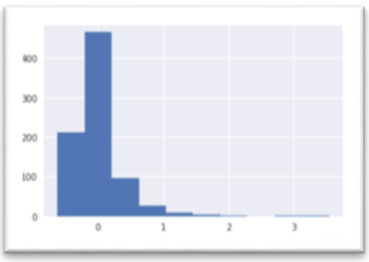


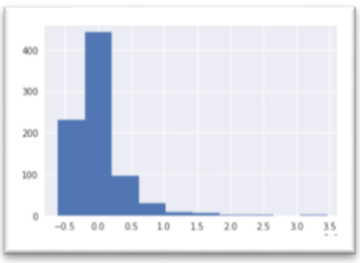
	P – Value	R – value ^ 2
Profit vs. Audience	6.931e-13	0.0613
Profit vs. Critic	1.152e-11	0.0549
Audience vs. Critic	3.268e-183	**0.64**

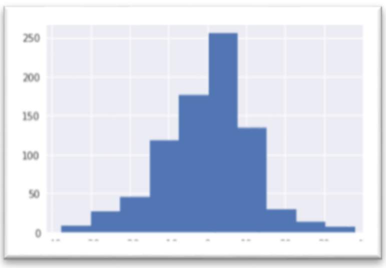
Analysis:
Although the p-value for each linear regression test is super low (<<0.05) which theoretically means that the slope is not 0 (y depend on x linearly), for the first two tests we still cannot conclude that they have a positive linear relationship with each other, because of the love r-value^2. The r-value^2 means that only 6% / 5% y-value could be explained by x-value, which is not persuasive enough to show they are correlated with each other.

For the third test, as the r-value^2 is 0.64, we could probably fairly say that there is a positive correlation between audience reviews and critic reviews. That reasonably makes sense because normally there will not be an extreme difference between audiences' taste.

Also, have a look at the histograms of residuals:

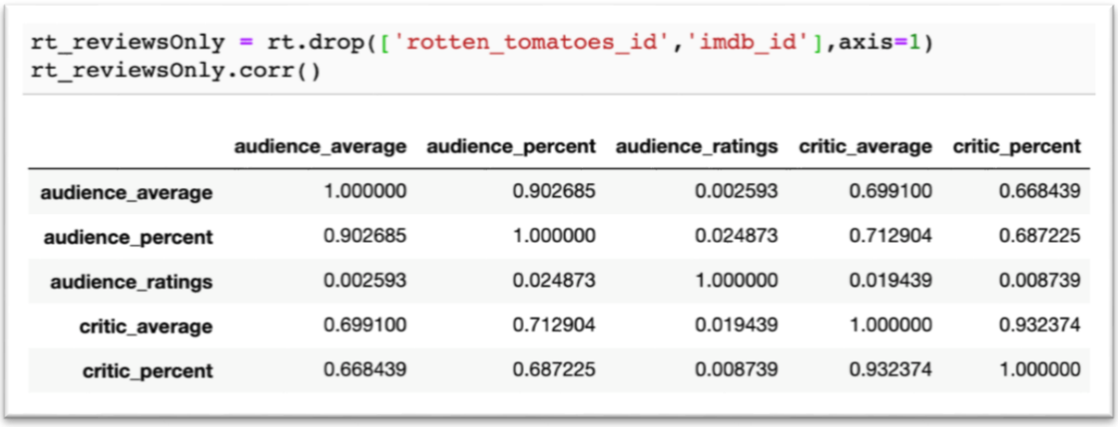






As there are more than 40 data collected and they look “fairly” normal distributed, we could apply the Central Limit Theorem to conclude that residuals are normally distributed. The linear regression test is valid.

Have a look at the correlation table of ratings:



```
rt_reviewsOnly = rt.drop(['rotten_tomatoes_id','imdb_id'],axis=1)
rt_reviewsOnly.corr()
```

	audience_average	audience_percent	audience_ratings	critic_average	critic_percent
audience_average	1.000000	0.902685	0.002593	0.699100	0.668439
audience_percent	0.902685	1.000000	0.024873	0.712904	0.687225
audience_ratings	0.002593	0.024873	1.000000	0.019439	0.008739
critic_average	0.699100	0.712904	0.019439	1.000000	0.932374
critic_percent	0.668439	0.687225	0.008739	0.932374	1.000000

The table shows that kind of, audience_average, audience_percent, critic_average, critic_percent does hold a not bad correlation with each other. Audience_ratings is totally not (all numbers are < 0.05). It makes sense because we cannot say that a movie has good ratings if it has a large number of reviews --- it could be a terrible film so that everyone wants to complain!

Another Tries: Machine Learning

Here, we redefined the question to be “how well can machine learning models do to predict these values”. We plan to try 3 basic models: Bayes_model (GaussianNB), KNN_model and SVC_model.

The first thing we do is to scale the data to a same measure --- hundred-mark %. We do a min/max scaler manually for critic_average (*10, originally out of 10) and audience_average(*20, originally out of 5). Then all data is out of 100.

As the others are kind of similar, we just give one approach: input ‘audience_percent’, ‘critic_average’ and ‘critic_percent’, can we predict ‘audience_average’ successfully?

However, there is a problem like, it is almost impossible to predict an exactly correct rating, because they are float numbers out of 100. To try to fix that, we design a new “test_score” function to give more space of the prediction from ML models. For instance, if we set a tolerance level of 4, input a critic percent (63.0 / 100) and get the predicted audience (66.5 / 100), it would also be considered as successfully predicted because 3.5 < 4.

The corrected test score is:

	Bayes_Model (GaussianNB)	**KNN Model**	**SVC Model**
Test Score	0.725490196078	0.656862745098	0.696078431372

We do not see a huge difference among these models, nor see a really different result as the correlation test gave us. We can conclude that these four attributes (audience_average, audience_percent, critic_average, critic_percent) does hold a relation with each other.

2. Movies Success

Problem Refined

How can we say that a movie is successful or not? It depends.

There are different measures from different people's views. For most people, if we see good rating scores for a movie, we will probably go to watch it and say "they are successful" if we love that movie as well. For investors of a movie, they may care about how much profit they could generate through this investment. Similarly, in a professional's view such as directors or movie judges, they care about how many awards / nominations that a movie could get a lot. They may not focus on public ratings and profits that much.

Overall, for our project, we define the success of a movie in 3 different views:

- "**Social Success**": whether the movie has a good **public evaluation** and reputation, depending on ratings from audiences and critics. This is the most intuitive measure for the public of a movie's success.
- "**Commercial Success**": whether the movie made a good **profit** (or, profit rate), depending on the cost and revenue. Some movie may not have very positive ratings from audiences, but it has an excellent rate of capital return, which is a good signal for capital investments.
- "**Profession Success**": whether the movie won any **awards** and **nominations** from the view of professional judges. Some movies may not bring huge profits for investors nor hold a super high audience rating, but they have special meanings for the development of our movies industry.

As far as we concerned, we choose to use "director", "cast member", "genre" and "production company" as inputs to predict these 3 types of success separately. The reason is because we suppose when we decide to watch a movie, we firstly care about its director and cast members. If the genre conforms to our preference, we may want to have a try. Lastly, the production company is probably a directed feature on whether a movie is successful as well.

Techniques Choice

The problem we meet is like, our input data is all in string type (or, a list of strings).
On the one hand, traditional machine learning cannot use non-numeric values as inputs. On the other hand, we could not transfer data like "director" to numeric values accordingly (for instance, "James Cameron = 1" and "Christopher Nolan = 2") because they do not have an absolute good or bad ordinal value. Each element has its own effect for a movie.

The final choice of us is to use **Neural Network**. As long as we transfer the input information to numerical matrix values reasonably, it could fit and predict the results. We let each feature be a node on input layer and let 3 types of success be the output layer separately.

Data Cleaning and Model Construction

First of all, we join "wiki", "rotten-tomatoes" and "omdb" tables together by movie's id. After merging 3 tables, we get 9612 rows. The second step is to delete useless rows with NaN values. Then we are good to construct the inputs.

We plan to construct a matrix to transfer the nominal strings to numeric values, which is able to be fit into neural network. For each feature (director, cast member, genre), firstly we need to discover how many different elements are in each feature. This makes up the horizontal axis of our matrix. Then, for each movie (vertical axis), if that element is served in this movie, we would assign a 1 to that entry, and 0 instead. For example, if the data has totally 4 different directors and 2 movies, the matrix will be 2 × 4. If director 2 served movie 1 and director 1, 4 served movie 2, the entries of the matrix will be: [[0, 1, 0, 0], [1, 0, 0, 1]]. In this artful method, we could transfer every non-numeric input to a number matrix. Then we could get totally three 2-D matrices, with same number of columns (number of movies).

However, as we try, the matrix will be super large with tens of thousands of columns. That means there are truly many different types in directors and cast members. It reminds us to extract the most useful data. Firstly, we decide to only keep the first 2 directors if there occurs more as a list, because we suppose that the main directors are essential. Secondly, we extract the first 4 cast members if there occurs more, as the movies are mostly influenced by protagonists. Finally, the same works for genre. For a movie it may not make much sense if it has more than 3 genres. As a result, we decide to only keep the first 3 genres.

Inversely, for every feature, if some value only shows once, probably that element is not that important and will not influence the whole performance. For example, a cast member only occurs once in all movies or a minor director only directs one movie. We delete the elements that shows only once.

For the measure of 3 types of success, there are some data cleaning and reasonable transformation as well. We will discuss for each later. Now we already have three 2-D matrices representing 3 features and are ready to train the model.

Neural Network Analysis for 3 Types of Success

We will talk about how we modify the output layers for each of the 3 success:

1. **Commercial Success**:

   The "real profit" is the measure of success. Similar with how we get the real profit (by calculating the money value based on years inflation), we merge the tables and extract what we need: "director, cast_member, genre, production_company, real_profit". Because actually there are just 985 rows with cost record, after data cleaning finally we get 392 eligible movie records.

   We have a look on the profits of movies. Basically, over the years, movies could generate around 10^7 to 10^8 profits. The MLPClassifier works surprisingly well for this type of success. With the tolerance level = 10^6, it shows a 0.979 testing score and 0.996 training score. The testing score is also well-shown for DecisionTreeClassifier and RandomForestClassifier models.

   Analysis: with the models we built, we could probably draw a conclusion that these features are highly connected with a movie's commercial success.

2. **Profession Success**:

   The number of "awards" and "nominations" are the measure of profession success. We think that each award and nomination contribute to different amount of its profession success. Specifically, different types of award could also contribute differently. For example, Oscar and Golden Globe are more valuable than the other awards. Hence, we define a “points correlation” function to calculate how good a movie in profession success:

- From IMDB: if a movie wins Oscar, it will show Oscar awards first. Sometimes it will hide the Golden Globe wins. Hence, we suppose that in IMDB, Oscar is valued a little bit more than Golden Globe.
- If a movie wins Oscar for “x” times, the points will add 10*x
- If a movie wins Golden Globe “x” times, the points will add 7*x
- If a movie is nominated by Oscar, it adds 5*x
- If a movie is nominated by Golden Globe, it adds 3*x
- For minor wins, the point just adds 1*x. For minor nominations, the point just adds 0.25*x.

Through this rule and by applying regular expression extraction, we could generate a kind of fair “profession success points” for each movie. Using this point value as the output layer, we are ready to train our model.

Because we set the values of each award / nomination subjectively, we also set a kind of tolerative space for our model. With the tolerance score of 5, the MLPClassifier performs not bad, with modified testing score 0.647.

Analysis: Probably (definitely!) there is a better measure of profession success scheme, the one we use is a fearless attempt. We could carefully get a conclusion that, very likely the input features are related to a movie’s profession success.

3. **Social Success**:

The “audience review” is the measure of success. As we analyzed before, there is no much difference between “audience / critic – average / percent” (they have a good linear correlation with each other). So, we choose “audience average” and “critic average” as our “y” (output layer).

To filter the data a little bit, similar with the strategy before, we select movies with over 40 reviews collected and neglect really uncommon movies. Then we do a series of data cleaning. There are 2867 rows (movies) left.

We first try the MLPClassifier and with a tolerance level, 0.15. As the ratings are out of 5 or 10, this is a medium space for mistakes. The results are frustrating on “critic average” which is 0.057, but much better in “audience average”, 0.29.

However, if we have a test on train score, the numbers are around 0.95. That totally means that there is a “p-hacking”. The MLPClassifier model does not perform very well. The reason could be like we did not input an appropriate solver or hidden_layer_sizes, although we have tried many.

Then we cast our eyes over decision trees. For the RandomForestClassifier, it shows 0.43 testing score and 0.14 training score, which is not very persuasive. For the DecisionTreeClassifier, we get 0.33 testing score and 0.29 training score. As they are closed, this model performs the best.

Analysis: There is **no** reasonable conclusion we could get for “these features could lead to ‘social success’”. All the models that we use are not that satisfactory. To avoid p-hacking, we would not try more models or parameters in order to get a higher score.

3. Genre Prediction

Problem Refined

Can we use the plot summary to train and predict genres, corresponding to each movie? If yes, combined with the conclusion of our Question 2 (genres take a feature to predict a movie's success in some way), the plot summary may be associated with predicting success.

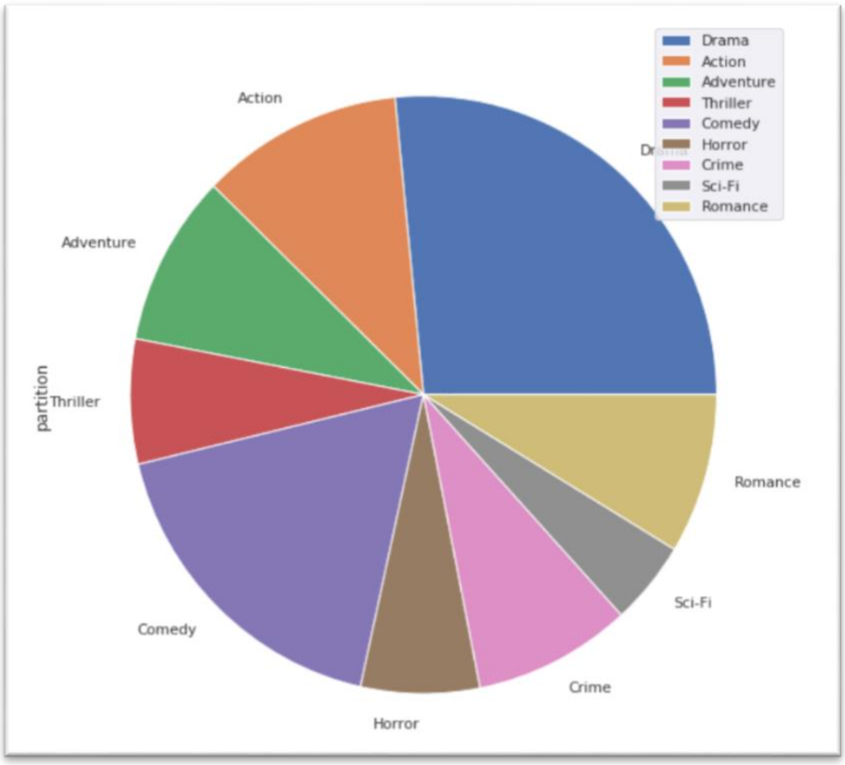
Data Cleaning

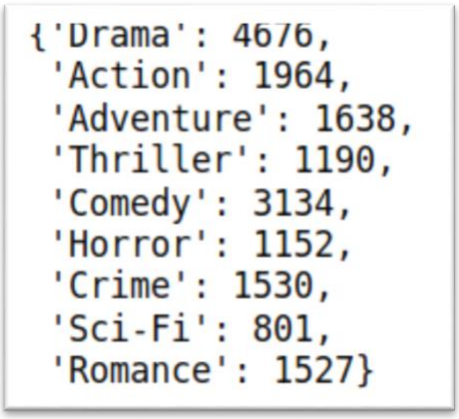
When we first look at the OMDB DataFrame, large number of genres and plots define different test cases. We count the number of movies belonging to each genre and the result strongly differs from each other in the range from minimum 1 (Talk-Show) to the maximum 4676 (Drama). Since some types of genres account for only small fraction comparing to the whole data set, we only select the genres which appears more than 800 times in the OMDB table. After I filter these genres from OMDB DataFrame, only 9 genres are selected and used to the next steps.

After we remove all the irrelevant genres and keep these nine genres in OMDB table, we create a **clean_text()** function to remove all punctuation, redundant symbols as well as white space and transform all words to lower-case.

At this current step, all words are transformed to the same case. However, the tenses of verbs, noun plurals and other linguistic differences will make the same word from different format hard to distinguish. Thus, we use **PorterStemmer** and **LancasterStemmer** to transfer every word to its stem such as this -> thi. However, even if all words have been transformed to its stem, there is still some inaccuracy because of the high frequency of general words such as 'the', 'a' and etc. that we can obviously recognize from the graph below.

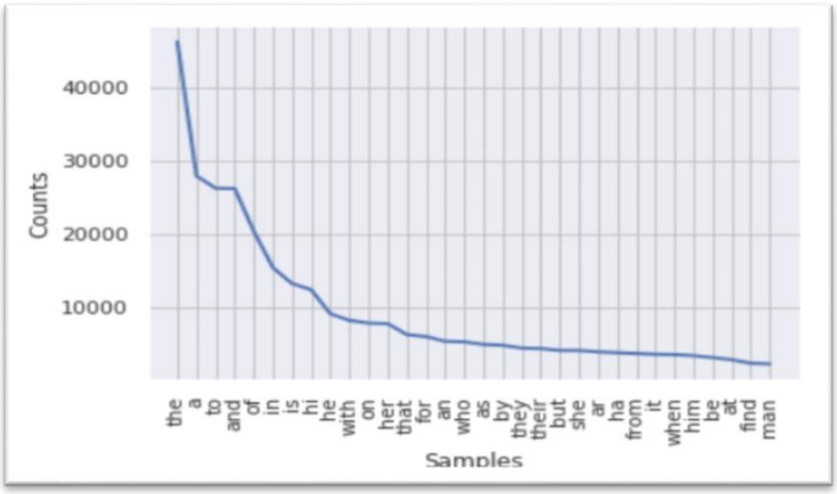
To remove all these general words with high frequency, we import the **stopwords** from corpus and filter all these stop-words from our omdb database.



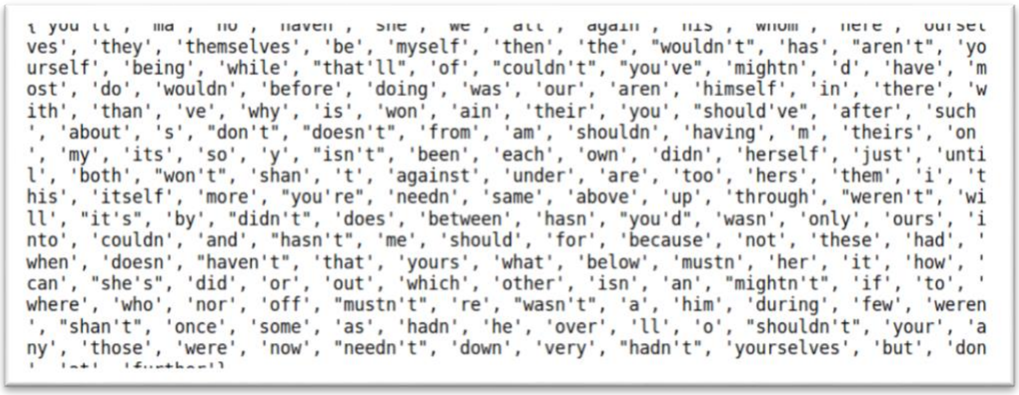


```
{'Drama': 4676,
 'Action': 1964,
 'Adventure': 1638,
 'Thriller': 1190,
 'Comedy': 3134,
 'Horror': 1152,
 'Crime': 1530,
 'Sci-Fi': 801,
 'Romance': 1527}
```

Genres Partition Pie Chart / Statistics



High Frequency Words



```
{"you'll", 'ma', 'no', 'haven', 'she', 'we', 'all', 'again', 'his', 'whom', 'here', 'oursel
ves', 'they', 'themselves', 'be', 'myself', 'then', 'the', "wouldn't", 'has', "aren't", 'yo
urself', 'being', 'while', "that'll", 'of', "couldn't", "you've", 'mightn', 'd', 'have', 'm
ost', 'do', 'wouldn', 'before', 'doing', 'was', 'our', 'aren', 'himself', 'in', 'there', 'w
ith', 'than', 've', 'why', 'is', 'won', 'ain', 'their', 'you', "should've", 'after', 'such
', 'about', 's', "don't", "doesn't", 'from', 'am', 'shouldn', 'having', 'm', 'theirs', 'on
', 'my', 'its', 'so', 'y', "isn't", 'been', 'each', 'own', 'didn', 'herself', 'just', 'unti
l', 'both', "won't", 'shan', 't', 'against', 'under', 'are', 'too', 'hers', 'them', 'i', 't
his', 'itself', 'more', "you're", 'needn', 'same', 'above', 'up', 'through', "weren't", 'wi
ll', "it's", 'by', "didn't", 'does', 'between', 'hasn', "you'd", 'wasn', 'only', 'ours', 'i
nto', 'couldn', 'and', "hasn't", 'me', 'should', 'for', 'because', 'not', 'these', 'had', '
when', 'doesn', "haven't", 'that', 'yours', 'what', 'below', 'mustn', 'her', 'it', 'how', '
can', "she's", 'did', 'or', 'out', 'which', 'other', 'isn', 'an', "mightn't", 'if', 'to', '
where', 'who', 'nor', 'off', "mustn't", 're', "wasn't", 'a', 'him', 'during', 'few', 'weren
', "shan't", 'once', 'some', 'as', 'hadn', 'he', 'over', 'll', 'o', "shouldn't", 'your', 'a
ny', 'those', 'were', 'now', "needn't", 'down', 'very', "hadn't", 'yourselves', 'but', 'don
', 'at', 'further'}
```

Stop words

Machine Learning

In this part, we focus on train and prediction on OMDB data with two methods and some related models.

One method is using **MultiLabelBinarizer** model to read list of genres and then convert from this intuitive format to the supported multilabel format. After that, we use Machine Learning with Text **TF-IDF** to fit and transform the plots filtered by **PorterStemmer**, then used **LogisticRegression()** and **OneVsRestClassifier()** to train the model, because both these models are used to handle the multiclass case and train algorithm by the one-vs-rest scheme and one-vs-all scheme. We found the accuracy score to be around 0.52 which is not high enough but acceptable. To improve this score, we use **predict_proba** function increasing the score to 0.64 from **OneVsRestClassifier()** which returns the probability estimates rather than the exact class by inspecting its corresponding classifier. We then switched to a different method of transforming the text-based plot_summary which is **CountVectorizer**, the accuracy score is slightly higher at around 0.6 and the score increasing to 0.62 after calling **predict_proba**. To compare with data without finding its stem, we use **TF-IDF** as a model to train and predict the score. Both normal score (0.5) and probability estimates score (0.625) are slightly smaller than the score of stemmed data.

Another method is to expend the omdb table based on the genres each movie corresponding to. For example, the genres of 'a cryptic message from the past sends james bo…' are action, adventure and thriller and we expend the one line to three lines. Each line will correspond to one genre. After expending our original dataframe, its rows increase from 8876 to 17612 and we used Machine Learning with Text TF-IDF to fit and transform the plot summary, then used **multinomial Naïve Bayes** and **SelectPercentile** to train the model. We found the accuracy score to be around 0.29 which is reasonably low, as the score is a reflection of all genres correctly predicted.

To discover a more reliable way to approach this question, we then build a **score_corr()** function to count the number of at least one genre predicted successfully for each movie and the score increases from 0.3 to approximately 0.8. However, although the numeric value exponentially increases, we cannot conclude that there is an obvious improvement based on the function we create, because the function merely changes our question and it is a representation like: 'at least one genre predicted successfully for each movie'.

Conclusion

Conclusively, we successfully predict all genres of each movie with accuracy at most 0.63 and at least one genre predicted successfully for each movie with accuracy 0.80.

Expectation

We have an idea to use levenshtein distance to find all formats and tenses of each unique word. For example, the distance between ‘apple’ and ‘apples’ is 1 and we can regard both of them as apple. However, we fail to do so, since words like ’bag’ and ‘bad’ also have distance 1 between each other but they are totally different. Maybe we will try to use that to build a similar model in the future.

4. Change Over Time Discovery

Problem Refined

We are interested in a simple question, which is related to what we did before:

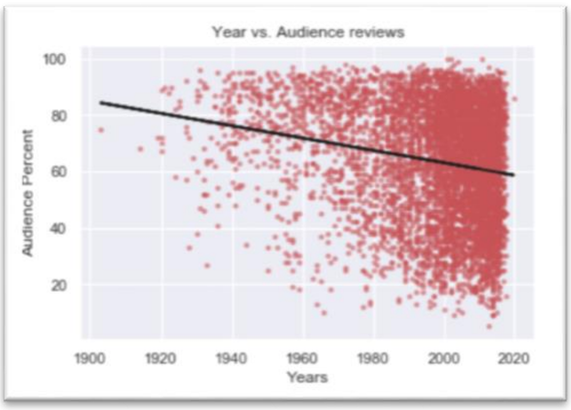
1. Is there a change in audience reviews of movies over time (a movie's publication date)? Are we becoming more and more picky and strict to evaluate a movie?

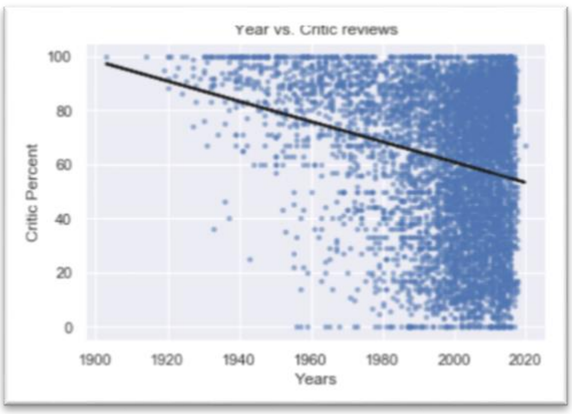
Data Cleaning

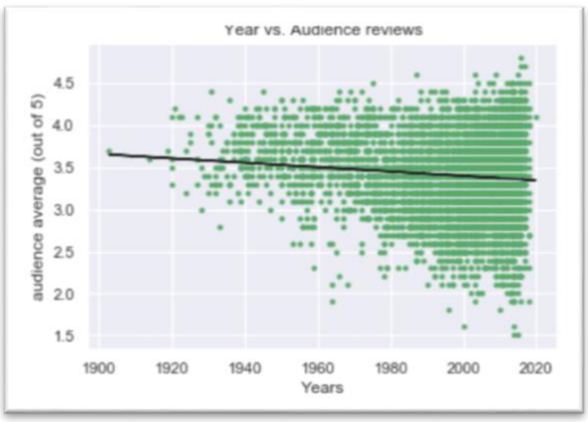
As we treat this in a simple view, there is not much work done in data cleaning. We just merge tables and extract movies with >= 40 audience ratings.

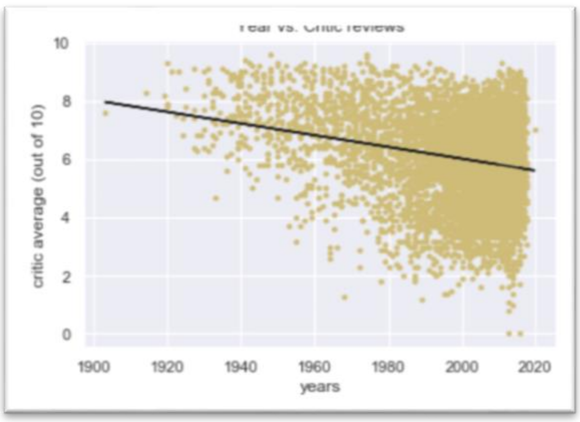
Linear Regression

We do a linear regression for each of the: "audience_average", "audience_percent", "critic_average", "critic_percent":









Analysis

There is a clearly negative slope for each graph. Can we conclude that people are actually being mean to evaluate a movie? In our opinion, the answer is hard to say, prefer to be no. There are a few limitations in our analysis:

- There are clearly fewer number of reviews for the movies before 1960. That makes sense as not everyone likes to watch old movies. Hence the samples for movies before is not that persuasive.
- Based on our question, we do not know the age of the reviewers. It could be like a child makes a good rating for an old movie and an old man makes one poor rating for the latest movie. In this scenario, teenagers are being tolerating and old people are picky, which contracts to our assumption: people are gradually being strict to evaluate movies.

5. Sentiment Score on Genres / Plot Summaries

Problem Refined

What is the sentiment score (positive / negative) for each genre of movies, based on its plot summary? Intuitively we treat comedies as "positive" and thrillers to be "negative". Is our intuition true for each genre?

Data Cleaning

As this question is only related to plot summaries and genres, we just use the IMDB table. Basically, we just filter rows without NaN values.

Sentiment-Score Model and Analysis

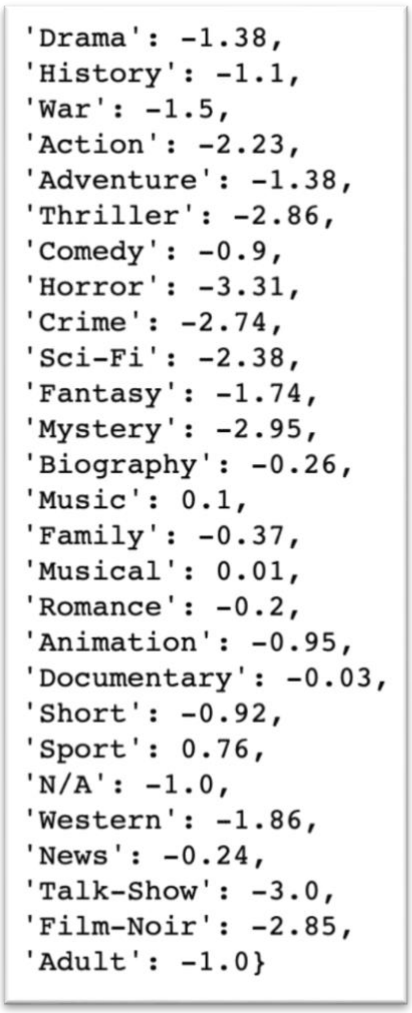
By chance, we get two text files from online called "positive-words.txt" and "negative-words.txt" (cited). The algorithm is basic:

- For each plot summary (row), we use regular expression to split the sentences into single words of lower case.
- For each plot summary, we go over the positive word text file and negative word text file. If there matches a positive word, the score +1, while for negative words the score -1. The final calculation is its sentiment score.
- By using a 2-D matrix, we integrate each movie (plot summary) to each genre type. The horizontal axis means different type of genres, and the vertical axis is the list of movies. This matrix marks the sentiment score. For instance, [[0, 0, 3], [1, -2, 0]] means there are totally 3 different type of genres and 2 movies recorded. The first movie contributes 3 positive sentiment score to genre 3. The second movie contributes 1 positive point to genre 1 and -2 negative sentiment score to genre 2.

The final result we get is kind of interesting:

There are some statistics that reasonably makes sense. For example:

- On average "crime", "thriller" and "mystery" genres are aggressive and mysterious, which may contain more negative words. Hence their sentiment score is much lower than the others.
- For a large number of genres such as "Music", "Romance", "Documentary" and "Biography", there is not much obvious emotional trend. Or, they could reasonably be either positive or negative. Hence the average score tends to be 0.



```
'Drama': -1.38,
'History': -1.1,
'War': -1.5,
'Action': -2.23,
'Adventure': -1.38,
'Thriller': -2.86,
'Comedy': -0.9,
'Horror': -3.31,
'Crime': -2.74,
'Sci-Fi': -2.38,
'Fantasy': -1.74,
'Mystery': -2.95,
'Biography': -0.26,
'Music': 0.1,
'Family': -0.37,
'Musical': 0.01,
'Romance': -0.2,
'Animation': -0.95,
'Documentary': -0.03,
'Short': -0.92,
'Sport': 0.76,
'N/A': -1.0,
'Western': -1.86,
'News': -0.24,
'Talk-Show': -3.0,
'Film-Noir': -2.85,
'Adult': -1.0}
```

However, there are some results that go against our assumption, for instance:

- The average sentiment score for "Comedy" is -0.9, which is negative.

This is worth discovering. We have a look on some sample comedy plot summaries and infer a possible reason. It is like, for some plot summaries of comedies, most protagonists will suffer an unfortunate experience first, and it takes long (takes about 2/3 plot length). Then his life will become more and more bright. Also, people will laugh at the bad luck of protagonists. Hence, the **plot summary** is more in negative. But generally, for the **atmosphere** of comedies, it may be positive.

Project Experience Summary

Zihan Wang:

Wikidata, Movies and Success -- CMPT353 **May – Aug, 2019**

- Get a team work with one teammate to build a project to investigate and analyze the relationship between movie's and success using SFU Gitlab for documentation.
- Overviewing and putting together work of teammates to make overall analysis
- Build a neural network model to predict a movie's success, based on three different types of success.
- Create a sentiment score model to define whether a movie is positive or negative by its plot summary.
- Develop a model and program to predict the regression line using stats.linear regression method, machine learning linear and polynomial method.
- Fit the best linear regression line on a positive cluster plot and degree of polynomial regression to give a best fitted polynomial regression line.

Chen Zhao:

Wikidata, Movies and Success -- CMPT353 **May – Aug, 2019**

- Initial general data cleaning and manipulating the data for easy analysis
- Collaborating, brainstorming with one teammate about how to implement the data science to find answers for questions, as well as coming up with the overall structure of program
- Utilize pandas to come up with procedure to separate data into groups based on genre and do analysis on the potentially changing factors.
- Used MultiLabelBinarizer to do multilabel classification for the plot of the movies, and make predictions to train and test data set.
- Applied both Tfidfvectorizer and Word Count Vectorizer to the plot summaries of the movies to do natural language processing and explored if it is possible to predict a movie's success from its plot summary.