Exploratory Analysis on Movie Summaries

AI Data Science Internship Evaluation 2019

Xiaoxuan Yang

Synopsis

Problem Statement

A central question in text mining and natural language processing is how to quantitatively understand texts. In this particular project, the objective is to explore the `Wikipedia movie summaries`. Following the given guidelines, I use natural language processing (NLP) tools with R and conduct univariate and multi-variate exploration of the dataset, summarized below:

- find the most produced and the most profitable movie genres
- identify common characteristics in movies summaries
- compare the word usage in the top five produced genres with Zipf's law

> Note: Zipf's law suggests that empirically, the frequency of any word is inversely proptional to its rank in the frequency table. In other words, word frequency decreases very rapidly for lesser used words in a document.

Solution Overview

In this project, I use the following packages to solve for the topics below:

1. **Genre Frequency**: classic `tm` package to create corpus and calculate word frequency.
2. **Automatic Annotations**: `spaCy` Python library through an R wrapper,`cleanNLP`, to convert raw texts of the movie summaries into feature-rich data frames; provides basic annotators to do part of speech tagging, named entity recognition (like person or places), etc; `tidytext` to calculate term frequency and combine it with inverse document frequency (not plotted in the summary).
3. **Topic Modeling**: `topicmodels`, coupled with `LDA` to divide summaries into topics and match the topics with genres.
4. **Zipf's Law Examination**: `tidytext` to calculate term frequency and ranks after tokenization; `lm` to find the exponent of the power law curve.

Major Insights

- The most produced movie genres are drama, comedy, romance film, and thriller.
- Adventure movies attract the most box office revenue, even though there are fewer adventure movies overall.
- Movie summaries of top five produced genres share similar common nouns, such as*who*, *father* and *time*.
- Preliminary result from topic modeling shows that the top five produced movie genres use distinct nouns in their movie summaries overall, which may result from the usage of film-specific names and places.
- The movie summaries use fewer common words as compared to other texts (demonstrated by Zipf's law) but still mostly follow Zipf's law curve.

Exploratory Data Analysis

Data Preparation and Preview

The dataset contains a dataframe of around 42k rows and 7 columns. For the**Genre** column, I first remove the space for two-word categories and manually remove terms like "Dogme95". The **Date** column consists of cells with various data types and leves of details. Since I do not need the exact release day of the movies in this particular project, I convert the date into a new **Year** column. After removing rows with missing values in the **Year** column, let's explore the remaining data with a basic plot of the numbers of movies released across all years.

```
  ggplot(movie_data, aes(year)) +
    geom_bar() +
    labs(x = "Released Year", y = "Movie Count") +
    theme(plot.title = element_text(hjust = 0.5))
```

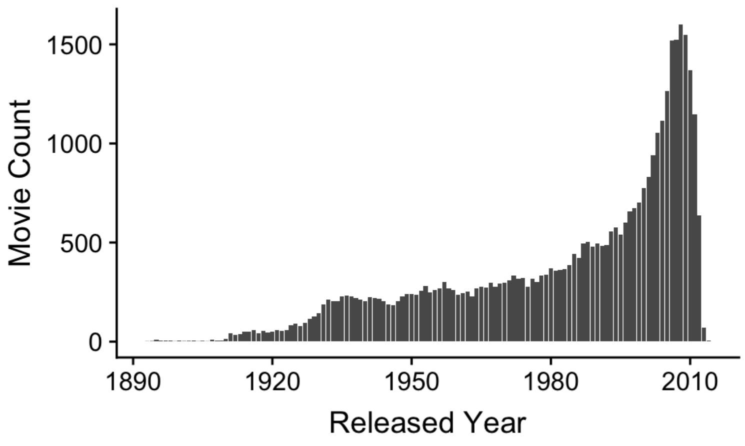

Figure 1. Histogram of Movies Released Over the Years.

As we can visually observe from the plot, Fig. 1 displays an exponential increase of movie production from late 1980s to early 2010s. This exercise helps us to understand the dataset.

Movie Genres

Next, I convert genre variables to a corpus by using the classic `tm` package. My goal is to find the most produced and the most profitable movie genres. I assume that each movie is allowed to represent several genres (see `this article` for randomly assignment approach).

```
# Create Corpus for movie genres
  genre <- Corpus(VectorSource(movie_data$genres))
  genre_dtm <- DocumentTermMatrix(genre)  # create a document-term matrix
  genre_freq <- colSums(as.matrix(genre_dtm)) # calculate frequency by summing up the elements
  freq <- sort(colSums(as.matrix(genre_dtm)), decreasing=TRUE)
  genre_wf <- data.frame(Word=names(freq), Frequency=freq) # organize the new dataframe
  genre_wf$Word <- unquote(genre_wf$Word) # eliminate the quotation marks (could be done earlier)
  rownames(genre_wf) <- NULL  # remove rownames
```

Using the code above, I create a document term matrix with movies designated by rows and its associated genres by columns. By summing up the elements (counts) in each column, I generate the genre frequency, as shown in Table 1.

Table 1. Genre frequency table with top 5 genres.

Genre	Frequency
Drama	19128
Comedy	10467
Romance Film	6664
Thriller	6529
Action	5867

Similarly, I sum up all the box office revenues for movies across all given years. Since I do not know if the given dataset is an exhaustive list of released movies, I choose summation over averaging method to compare revenues between genres. In the side-by-side plots below, I notice that even though adventure movies are not the most produced genre, they attract the most box office revenue over the years–it takes ten years to make *Avatar* and who wouldn't want to watch it on a big IMAX screen more than once?

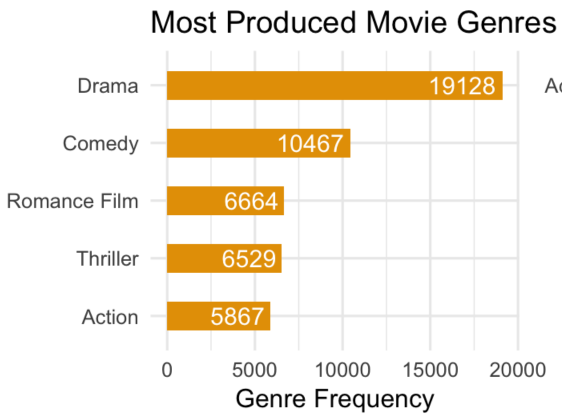

Figure 2.1 Top five most produced movie genres

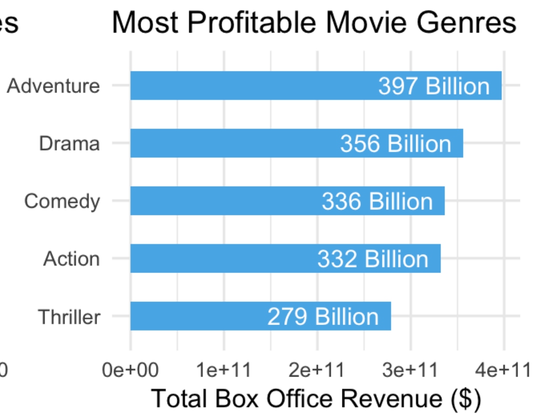

Figure 2.2 Top five most profitable movie genres

Characteristics of the Movie Summaries

Here comes the most exciting and challenging part of the project. After subsetting the dataset to only include the top produced movie genres, I dive into the texts of those movie summaries using `cleanNLP` and `tidytext`. The package `cleanNLP` (learn more) produces a data table that includes annotated objects such as lemmas and part-of-speech, as indicated in Table 2. For example, the word `Eva` is labeled as a non proper noun (pos = NNP) and the first token (tid = 1) in the first sentence (sid = 1).

> Assumption: I ignore the film level difference and compile all summaries within each genre as one document.

```
# Initialize the NLP backend
  library(reticulate)
  library(cleanNLP)
  use_python(python = '/usr/local/bin/python3') # set the right python version
  py_config() # check python configuration
  cnlp_init_spacy(model_name = "en") # initialize spaCy library (after select python version)
  object <- cnlp_annotate(dataset$summary, doc_ids = dataset$genre) #  top five genres dataset
```

Table 2. Tokens table by cleanNLP. The fields id, sid, and tid serve as a composite key for each token (adapted from Arnold 2017).

id	sid	tid	word	lemma	upos	pos	cid
Drama	1	1	Eva	eva	PROPN	NNP	0
Drama	1	2	,	,	PUNCT	,	3
Drama	1	3	an	an	DET	DT	5
Drama	1	4	upper	upper	ADJ	JJ	8
Drama	1	5	class	class	NOUN	NN	14

Since `cleanNLP` has the tidy data model built in to represent corpus annotations, I seemlessly incorporate it with the piping notation and `ggplot2` graphics. After counting the tokens (in the form of lemma) and rank them based on the frequency, I manage to analyze the most used **nouns** across five genres. As Fig. 3 suggests, those five genres share a similar set of frequently used words. The word *who* ranks the top for all genres, which makes sense since movie summaries have a descriptive language. While it's reasonable to see *love* in the romance film genre, it's interesting to see the word *police* appear in both action and thriller genres. I also use `tidytext` to verify the most frequency words (as documented at the end of my coding file). Since `tidytext` doesn't distinguish between verbs and nouns, the word *kill* makes it to the top common words in both action and thriller genres, too.

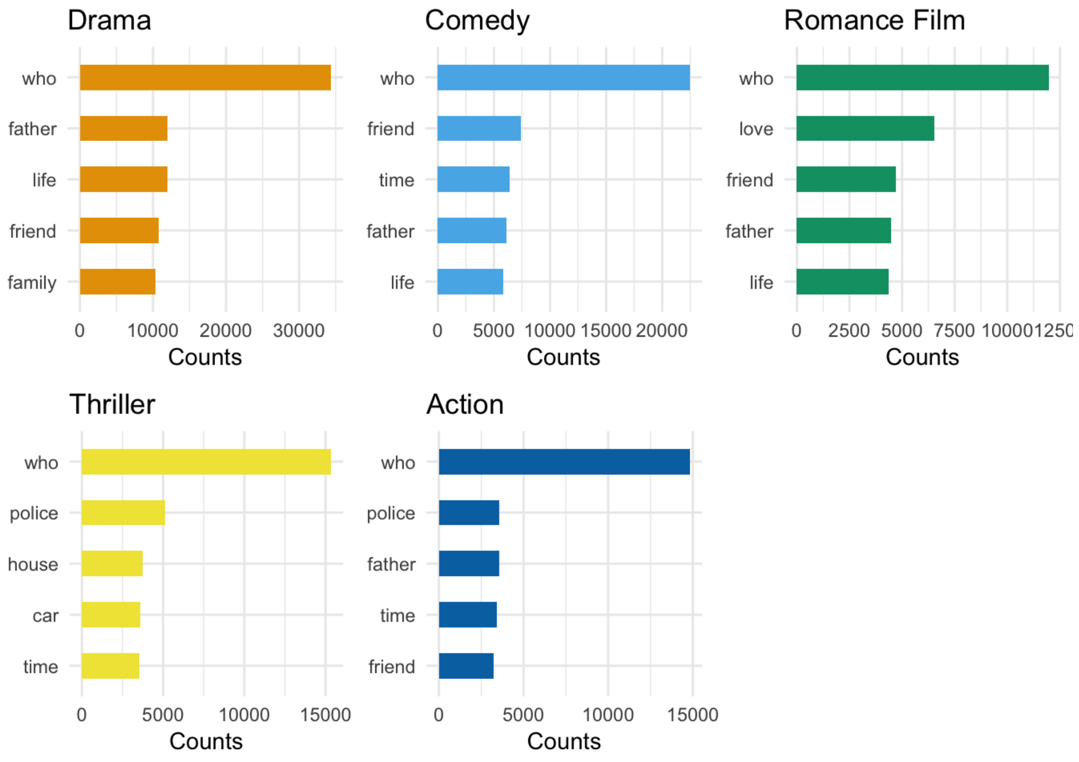

Figure 3. Top frequently used words for selected genres.

Another helpful high-level summary of the content is to apply topic models, which describe themes within a textual corpus. Each theme consists of a collection of words that co-occur. Specifically, I am using an unsupervised Bayesian machine learning model, known as Latent Dirichlet allocation (LDA), to find clusters of words (topics) that are specific to different genres. For this preliminary study, I have five different genres that can relate to five separate topics. Knowing the "right answer", I want to see if the algorithm correctly distinguish the five categories.

```
  library(topicmodels)

# Pull out the raw counts of term frequency
  dep <- cnlp_get_token(anno) %>%
    filter(pos %in% c("NN", "NNS")) %>%
    cnlp_utils_tfidf(min_df = 0.05, max_df = 0.95,
                     type = "tf", tf_weight = "raw")

# Create a five-topic model and set a seed for reproductability
  tm_movies <- LDA(dep, k = 5, control = list(seed = 1234))
```

> Latent Dirichlet allocation: each topic is a mixture of words, while each document (genre, in this case) is a mixture of topics.

After tidying model objects and extracting the per-topic-per-word and per-document-per-topic probabilities, I can now evaluate the model fit–how words are grouped into topics and how topics are assigned to documents. As shown below in Fig. 4, I select the top five words that are most common within each topic.

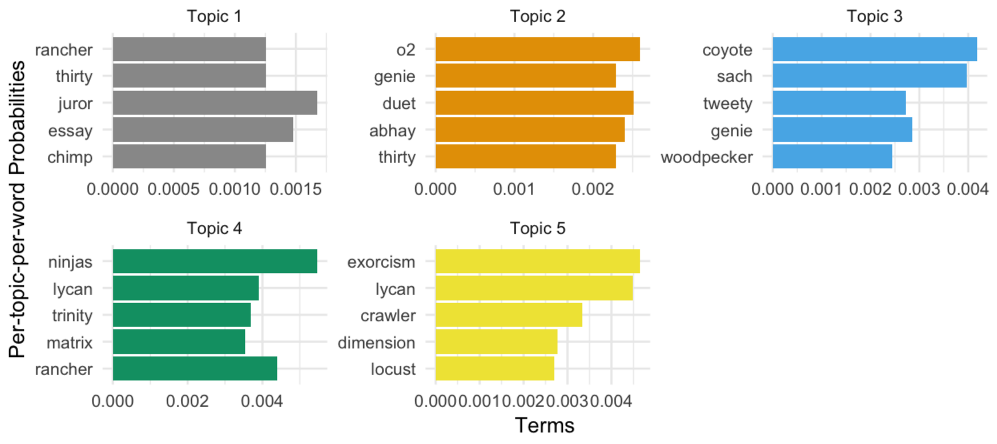

Figure 4. The terms that are most common within each topic.

I also display per-document-per-topic probabilities (Table. 3) . Each value in the table represents the estimated proportions of genre summaries generated from each topic. For example, 0.16% of the words in comedy genre summaries are generated from topic 2.

```
# Extract information on topic probabilities for each genre
  table_gamma <- tidy(tm_movies, matrix = "gamma")

# Plot
table_gamma[, 2:6] %>%
  mutate(Genre = row.names(table_gamma)) %>%
  select(Genre, everything()) %>%
  kable(align = "c", escape = F) %>%
  kable_styling(c("striped", "condensed"), full_width = F)
```

Table 3. Probability distribution of genres over topics

Genre	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Action	0.0000014	0.0000014	0.0000014	0.9999946	0.0000014
Comedy	0.0000008	0.0015768	0.9984208	0.0000008	0.0000008
Drama	0.9992839	0.0002261	0.0004888	0.0000006	0.0000006
Romance Film	0.0000021	0.9999917	0.0000021	0.0000021	0.0000021
Thriller	0.0000016	0.0000016	0.0000016	0.0000016	0.9999938

While it looks like that the model perfectly matches the genres with only one topic, we need to take a step back and recall that I treat all movies summaries the same within each genre. Thus, the large scale of analysis on genre discourages the assignment of more than one topic to one genre. In the future, to better assess the data, I need to seperate genres summaries into different film ids and run the similar approach. In addition, allowing a larger set of topics can be helpful to address more detailed patterns in language.

Nonetheless, I am happy to see *ninjas* being assigned to the action genre while*crawler* is assigned to the thriller genre (Fig. 4). The sharp difference in topic assignment implies that the movie summaries use a large amount of film-specfic non proper nouns like character names and places, which is verified by calculating inverse document frequency in `tidytext`.

Evidence of Zipf's law

Zipf's law can be understood as an inverse relationship between the word frequency and the rank (relative frequency to other words), as shown below:

\[\text{frequency} \propto \frac{1}{\text{rank}}\]

Thus, our objective is to plot frequency against rank and make the comparison. First, to plot the term frequency, I apply`tidytext` to quickly run the term frequency distribution for the five genres. The reason that I choose `tidytext` over `cleanNLP` are the following:

1. Even though `cleanNLP` has more comprehensive annotations, it takes a fairly long time to run.
2. `tidytext` works well for applications that do not need more advanced annotators. For this part of the project, I simply need to

calculate the term frequency without any additional annotations.

Thus, I plot the frequency of each word against the number of words with the same frequency. By this logic, I expect long tails to the right for all genres, because very few words occur frequently. I also expect high bars at low frequency due to random character strings that may result from incomplete cleaning of the dataset.

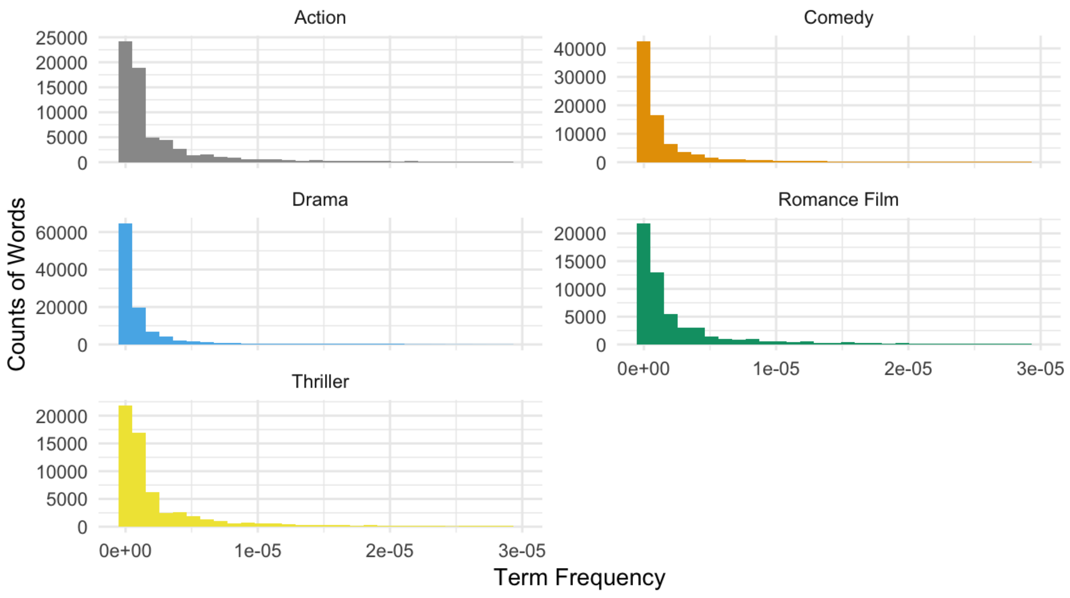

Figure 5. Total word frequency distribution in selected genres.

As shown in Fig. 5, those plots display similar distributions for those five genres. With the same dataframe I use to plot term frequency, I rank each word within the frequency table, in preparation to compare with Zipf's law.

```
# Organize term frequency by genres and add ranks
    freq_by_rank <- topfive_words %>%
      group_by(genre) %>%
      mutate(rank = row_number(), # find the rank
             "term_frequency" = n/total)
```

Zipf's law is often visualized by plotting rank on the x-axis and term frequency on the y-axis on log scales. Taking the approach fully documented `here`, I find the exponent of the power law equation by doing a linear regression to a partial rank range and plotting the fitted power law with the rank data.

```
# Select the middle portion of the rank range to find the exponent of the power law
    rank_subset <- freq_by_rank %>%
      filter(rank < 5000,
             rank > 80)

# Use linear model to find the proper coefficients
    lm <- lm(log10(term_frequency) ~ log10(rank), data = rank_subset)

# Fit a curve that follows Zipf's law
    freq_by_rank %>%
      ggplot(aes(rank, term_frequency, color = genre)) +
      geom_abline(aes(intercept = lm$coefficients[1],
                  slope = lm$coefficients[2],
                  fill = "Fitted Zipf's law"), linetype = 1) +
      geom_line(size = 1.1, alpha = 0.8, show.legend = TRUE) +
      xlab("Rank") + ylab("Term Frequency") +
      labs(colour= "Top Five Produced Movie Genres", fill = "") +
      scale_x_continuous(trans='log10') +
      scale_y_continuous(trans='log10') +
      theme(legend.title = element_text(colour="Black", size=10, face="bold"))+
      theme_minimal() +
      scale_color_manual(values=cbPalette)
```

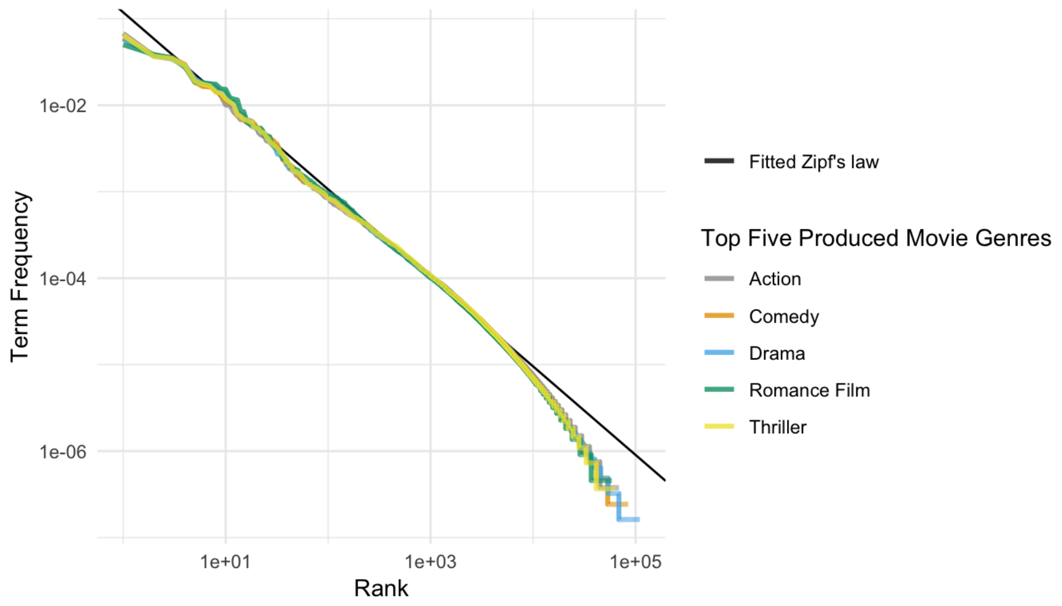

Figure 6. Fitting an exponent of Zipf’s law with word frequency.

I have found our plots of the corpus similar to the classic version of Zipf’s law (as indicated by the black straight line), with obvious deviation at low rank (e.g. rank = 80000). The deviation at low rank suggests that the language used in movie summaries consists of fewer numbers of rare words compared to other collections of language, which are not uncommon for many kinds of language. The minor deviation at high rank (e.g. rank = 5) suggests that the film summaries do not use as many common words. This may result from an extensive use of proper nouns that are case-specific and “rare” when considering its appearance in all reviews, which again echoes with our conclusion in last section.

Future Steps

1. More data cleaning: Due to the time limits, I only clean selected columns. However, for future projects, descriptors like`Runtime`, can be very helpful to understand how movie length changes over years. In addition, several genres categories have the potential to be integrated. For example, the genres “gay” and “gay interest” can be organized into the same genre category.
2. Genre association: As I notice from analyzing the genres, the majority of the movies is categorized into several different genres. It will be informative to understand how those genres overlap.
3. More accurate annotations: To increase the accuracy of the annotation process, I can choose to use an even larger size model when running the `cleanNLP` package. There are also more questions to explore with other advanced annotators, such as character names and locations mentioned in the movie summaries.
4. More exploration on topic modeling: In topic modeling, the next important step for this project is to use film as a unit for clustering and recognize each genre as a mixture of different topics. I also hope to try to specify the appropriate level of granularity for this corpus of movie summaries.

Acknowledgements

I have to credit my go-to book, “Text Mining with R”, for helping me navigate tools like LDA. I also want to thank all the developers who create those wonderful natural language processing tools.