Classifying Movie Genres With Plot Summaries

Brian Kimmig, Jesus Zarate

1 Introduction

The internet today contains an abundance of information. We see a great deal of images, videos, and somewhat more common, text. It is common that people want to summarize a topic or the content of a chunk of text. This is a challenging problem for computers and machine learning (ML) algorithms as sentences and paragraphs are highly unstructured. A majority of ML algorithms require the data to be in a consistent form, from vectors to matrices. Furthermore, they require that the data is numerical. This poses the main problem for us, how do you meaningfully turn a sentence or paragraph into numerical data?

The goal of this project was to take text data, and assign it to a category (or categories). We could try to do this with Twitter data but the categorizing of Tweets can be largely subjective and was beyond the scope of this class. We decided that a good dataset to try these methods on would be film synopses and genres. In this case the the text describing a movie can vary greatly, but there is a consistent and widely agreed on genre categorization. This makes for a great dataset to help us see if we can categorize textual data. We aim to see if we can take the text of a film synopses, build meaningful features, and learn a model to categorize the genres of other movies based on their synopses.

Building meaningful features from text sounds nice, but what does that actually mean? Go into other stuff here? What methods are we planning on using? Point to them below?

2 Data

We obtained the base of our data from a dataset published on Kaggle. The dataset contained information on ~ 5000 movies. We used this dataset for the movie list and the IMDB IDs. With IMDB IDs it is easy to automate gathering the synopses of each movie via GET requests to the OMDB API. The OMDB API allows you to search for movies, and gather information via the title or the IMDB ID. To ensure we get the correct information for every movie we performed GET requests querying with the IMDB ID.

The OMDB API allows a user to specify the length of the plot summary it returns, with either 'full' or 'short'. We chose to gather the 'full' synopses for every movie we queried.

From the Kaggle data we used the OMDB API to compile title, plot summary, and genres for all ~ 5000 movies. The data was stored in a JSON file, with each entry (or movie) containing the fields ['title', 'plot', 'genres'].

There were considerably more genres than expected, in total there were 26. Figure 1 shows every genre and its percent occurrence. In our data, there were genres that occurred less than 1% of the time, and generally they tended to be more obscure genres. Specifically, the genre 'Game-Show' occurred 0.02% of the time. Because of the rare genres we decided to limit our genre classification labels to those that occur more often. In the end we settled on a cut of 10% (shown by the red dotted line in Figure 1). This was to ensure that we had captured the major, or most common, genres. The cut of 10% also allowed us to ensure that every movie had at least 1 label, or genre, associated with it. If the cut was higher, we found that some movies did not have a genre associated with it. We also wanted to avoid throwing away data.

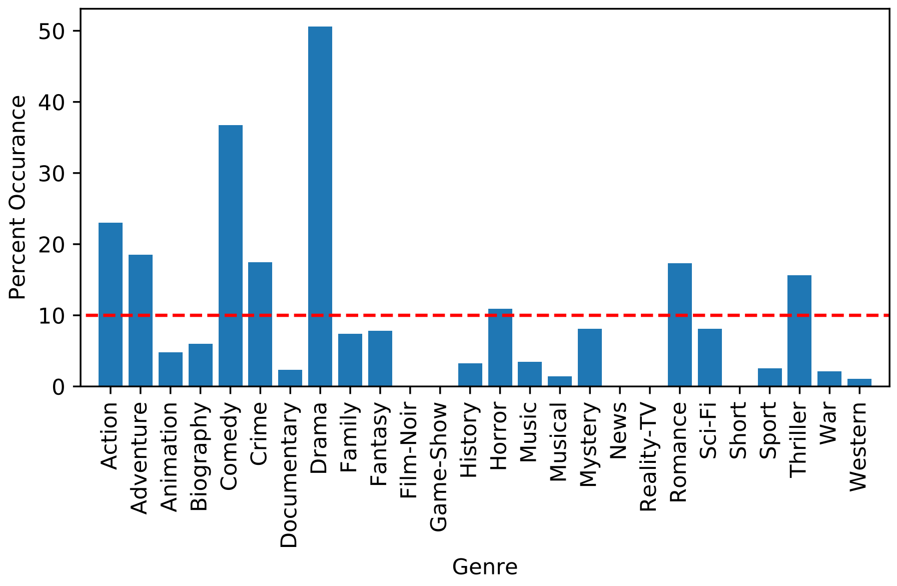

Figure 1: The percent occurrence for each genre. The red dotted line represents the cut made to get the list of genres (10%).

In the end, with our 10% cut we ended up with the genres Action, Adventure, Comedy, Crime, Drama, Horror, Romance, and Thriller.

The genre labels set up with a one hot encoding. This was done to allow us to easily classify them with either a multi label classifier, or by individually, by column, with other classifiers (discussed further in §3). Figure 2 shows the genres for every movie in the one hot encoding. A yellow line indicates the movie is associated with that genre, from this we can clearly see that a majority of the movies are associated with the 'Drama' genre.

The synopses of the movies live in long strings or documents. We have a list of these that we will processing to create the features that we will then use to build models to classify genres. Extracting features from these documents is the main part of the project and will be discussed §3.

3 Methods and Results

In this section we will discuss the methods we used to create features for classification. We will start by creating features using Term Frequency - Inverse Document Frequency (TF-IDF), then with Latent Dirichlet Allocation [1], and finally a combination of the two.

We will classify genres of movies with three different classifiers. The first will be a simple linear regression, as learned in class. The labels are binary for each genre, so some thresholding will be done to get labels using linear regression. Next, we will classify with logistic regression, which is a well known regression model for predicting binary labels. For both linear and logistic regression we will fit for each genre individually. The last classifier we use will be the random forest [2]. We chose to use Random Forests due to their popularity in data science competitions and their multi label classification capabilities; we can classify all labels simultaneously.

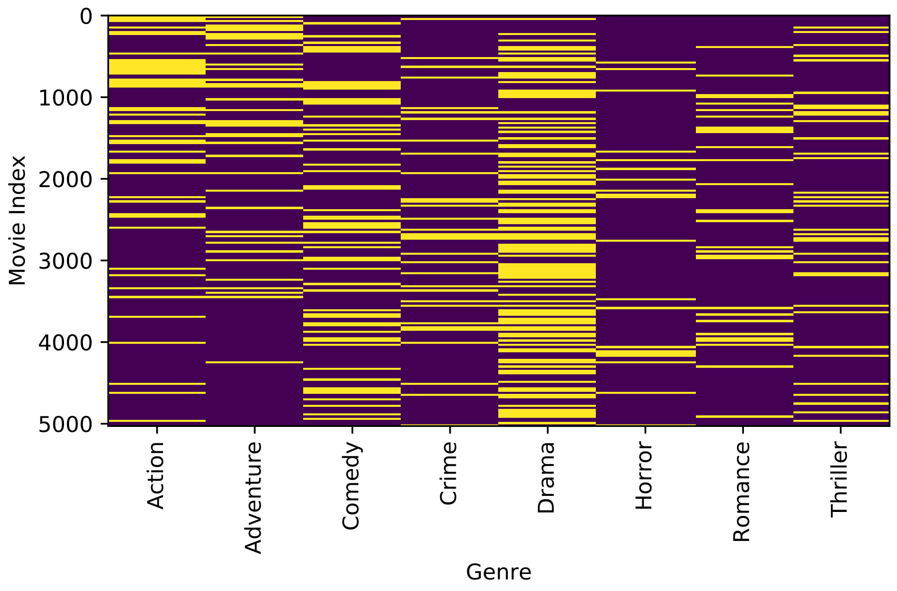

Figure 2: The one hot encoding of the genres for every movie. Each row represents a movie and a column represents a genre. Yellow indicates the movie has that genres associated with it.

For both the logistic regression, and random forest classifier we used the scikit-learn implementations [4].

3.1 Term Frequency - Inverse Document Frequency

3.1.1 Linear Regression

Classify one genre at a time.

3.1.2 Logistic Regression

Classify one genre at a time.

3.1.3 Random Forrest

Classify all genres at once.

3.2 Latent Dirichlet Allocation

Topic models are methods of taking large amounts of data and finding, understanding, and summarizing the information in it [6, 3]. We believe that hidden in these plot summaries there will be some fairly obvious groups of words, or topics, that will be seen together. After some on-line research we came across the concept of Latent Dirichlet Allocation (LDA) [1]. LDA is a fairly popular topic model.

LDA essentially looks for topics in documents, determines some topics, and gives an estimate of the proportion of that topic that is seen in the document. The general idea that documents are random mixtures of topics, and topics are characterized by words [1].

DESCRIPTION

For this portion of the project we chose to use a the LDA routine in scikit-learn [4]. This implementation allows you to pick the number of topics you would like created and lets you tweak and set some other parameters. We chose to use the default settings for everything but the number of topics. We chose the number of topics to be equal to the number of genres. We did this to see if the words in the topics found matched with those of the genres. Below you can see the topics, by number, and the top 20 words associated with that topic.

1. charlie king eddie prince charles invention cabin oil harold 1950s beverly mistaken embark chloe bennett throne royal lily princess hills
2. world harry god evil save vegas earth terrible jay thief real human radio cult team picture new stop rare annie
3. team crew money group ship nick angeles captain earth los black mike gang million alien sam new member hunter plan
4. man life family young world father town new son story help time love old home years named year finds city
5. war police murder american agent group world man secret new killer drug young case help mission bond detective fbi crime
6. film music rock documentary movie story band star early set singer dance park world director hollywood follows danny musical lee
7. life school new love family friends old friend time mother day young year just father high finds woman best daughter

We see that some of the words in the topics are kind of ambiguous, but in some cases the topics are clearly defined. For instance, in topic 2 we see words that may fit well with

FEATURE MATRIX AND TOPICS (first 20 words?)

3.2.1 Linear Regression

3.2.2 Logistic Regression

3.2.3 Random Forrest

3.3 TF-IDF + LDA

3.3.1 Linear Regression

3.3.2 Logistic Regression

3.3.3 Random Forrest

4 Discussion

4.1 Brian's Thoughts

4.2 Chuy's Thoughts

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] IIIT-Hyderabad Goutam Nair. Text mining 101: Topic modeling, 2016. [http://www.kdnuggets.com/2016/07/text-mining-101-topic-modeling.html].

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Wikipedia. List of genres — wikipedia, the free encyclopedia, 2017. [Avaliable at https://en.wikipedia.org/w/index.php?title=List_of_genres&oldid=774783669].

[6] Wikipedia. Topic model — wikipedia, the free encyclopedia, 2017. [https://en.wikipedia.org/wiki/Topic_model].