Universiteit Leiden
The Netherlands

Opleiding Informatica & Economie

Classifying film scripts by genre using word features and structured features: a text mining approach

Thijs van Meurs

Supervisor:
Dr. Suzan Verberne

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

21/07/2020

Abstract

In film making, scripts contain more text than the mere dialogue one sees on the big screen. All actions, visual settings and background information is provided in these scripts. To see whether those words or the structure of the documents provide information regarding its genre(s), we have proposed two methods of measuring. The first method is using the words as features. This bag of words will be a sparse matrix containing all the words as features. We have carefully filtered out words which did not contribute to the prediction of the genre. The second method is based on the structure of the film scripts. The features are mostly numeric and ratios and will answer questions such as whether a particular genre is more outdoors than indoors.

Working with word features (a bag of words model) brings more to the table than just the features themselves. The features need to be rated on their importance. This can be done in two ways. The first way is straight forward: counting the term frequency. The second way is the term frequency inverse document frequency. These weight functions are an important factor for the input of the models. These two weight functions can only be applied to the word features.

The problem at hand is a multi-label (and multi-class) classification problem: there are multiple genres for the classifier to train and choose from, and a film script can have multiple genres.

The models used for the text features are logistic regression and Support Vector Machines. Both do well on classification problems. For the engineered features we have chosen for an easier to dissect and comprehend model: the decision tree as well as logistic regression.

The results show a bias towards the genres which occur more frequently in the data set. However, this favoritism can be minimized with under-sampling. Based on the comparison per method (text features versus engineered (structured) features), it shows that the text features are the more informative ones. The measures in their recall, precision and f1 score are greater than those for the engineered features. Therefore we can conclude that text features are the better choice when classifying film scripts.

Contents

Chapter 1

Introduction

The world of film making and storytelling is growing steadily, with a box office revenue of over 40 billion dollars (approximately 35 billion euros) in 2019. Since 2005, this is an increase of 83% [Wat19]. However, film is not the only industry that uses scripts: games and theatre are two major players who use scripts on a regular basis.
Many processes are being optimized with regards to their function, using technology. Take for instance computer generated imagery (CGI). Many film studios use this technique to cut costs (and save time) of actors, locations and special effects. These are all objective processes which can be enhanced by technology. Some of the finer arts of film making are still done by hand. Script writing is such a process. Writing scripts is an iterative process on which many people work simultaneously. Many people have many different views on subjects, which are lead by experience and emotion. It is therefore hard to pin-point the reasoning and factual logic behind the choices made in scripts. This thesis will explore the subjective craft of writing scripts and try to find the objectivity in this process. We will try to see if it is possible to make a creative process more objective, by analysing the use of words and the structure of the scripts. This thesis will be a stepping stone for future work in this area.

1.1 Research objective

The goal of this research is to classify film scripts by their respected genre using text mining. The research question which is derived from this goal is:

"What feature representation or combination of features are the more informative for classifying film scripts to their corresponding genre"

Two parts will be analysed extensively: 1) word representation and 2) engineered features. In the first part we will be looking at which specific words or word combination are more or less exclusive for a specific genre. In the second part of the thesis we will be creating features which are associated with the structure of the film scripts. After these two parts are conducted thoroughly, we will examine which method fares better in

answering the research question.

1.2 Thesis Overview

The thesis will start with the background section. This section highlights the knowledge and definitions applicable for reading this thesis. It also shows the prior research which has been done around this subject or regarding the process or methods. Once the fundamental knowledge has been established, we will describe the process of collecting and pre-processing the data. This data - and the knowledge derived from the data - will in turn be visualized. In the previous section we briefly mentioned this thesis will be split in two parts: the word features and the engineered features. Section 3.4 will explain the difference between the two parts. Based on these differences we will build our model accordingly.
Chapter 4 will show the results. The results are the outcome of the models - how well the models perform, as well as the most important features per genre. These results will also be split in the two parts, as well as a comparison.
We conclude this thesis with the discussion and conclusion for future work.

Chapter 2

Background

Before we can start with analysing the data, we have to benchmark some of the processes which start beforehand. Such as background knowledge (section 2.1) and related work (section 2.2).
The background section describes the prior knowledge needed to either fabricate the results or to gain perspective on the insights of this paper. Future sections could be broken up into two subsections: word features (text based features) and engineered features (fabricated features). These two subjects are the silver lining in this paper, and will be our main focus when finding the answers to the research question.
The related works section describes the prior research that has been done on this subject. Though be it through the same methods or techniques.

2.1 Text classification

Text mining is the process of finding valuable information from words, characters and context in the natural language. In this thesis we are focusing on word features, which includes the extraction, cleansing and structuring (parsing) of text data [HNP05]. After the extraction we will converge into two domains to find the answer to the research question: using word features and using engineered features. The difference between regular data mining and text mining is that in text mining the patterns are extracted from natural language text rather than from structured databases of facts [Mie06].

Word features

When working with text data, all the features one collects from the text are included in one dataframe. The film scripts (i.e. the text which make up the film scripts) are tokenized. This means that all the words from all the film scripts are converted into features (much like regular data mining). This representation is called the *Bag of Words*. The resulting dataframe is a sparse matrix with high dimensionality. This matrix has a large vocabulary, which are the possible features for the model. All the words of all the scripts are included in this vocabulary, and are therefore (possible) features. A sparse matrix means that many of the features are not

applicable as feature for the current row in the data set, the matrix will therefore consist of zeroes (or low values) for many features. A row corresponds to a single film script (with its genre). The columns (the features) are the words retrieved from the scripts, which are included in the decision making process for the algorithm.

Engineered features

The features are engineered from the text as well. However they do not include the words. Potential engineered features include the summation of small phrases, sentences or other recurring parts. The datatype can vary and depends on the feature one wants to pursue.

Both the extraction and the cleaning, pre-processing, model-building and model-evaluating will be done using Python. The extraction of the of the data will be done using a technique called *web scraping*. The goal is to find the information (film scripts in natural language) on a website and extracting it to your local machine.

2.1.1 Web scraping

Section 3.2 will go into the depths of this process.
Web scraping is the act of retrieving information embedded into the front-end of a website. In this thesis it is used to retrieve the film scripts on which all the transformation and information is based.

2.1.2 Pre-processing

Section 3.2 will go into the depths of this process.
In text mining, the corpus is what a data set is in regular structured data mining tasks. The corpus contains all the text data on which transformations will occur. There are two types of transformations on the corpus done in this thesis: tokenization and stop words removal. These parameters will also be explained in chapter 3.

- tokenization: this method makes sure every word or group of words in the corpus is set as feature. These are the features on which the models are trained on.
- stop words removal: removing the stop words from the corpus. Words that inherently have no added meaning to the corpus, such as pronouns in this thesis, are removed. This process is done carefully, because stop words may have meaning in this classification task, whereas in other subjects it does not (e.g. pronouns, mentioned earlier). The function 'CountVectorizer' of the sklearn feature_extraction.text library, helps with the removal of stop words. In addition, you can set the parameters 'min_df' or 'max_df' to filter out any word or word group from the corpus to avoid over-saturated or desaturated corpora. Min_df means minimum document frequency, and max_df means maximum document frequency.
  - Min_df: The threshold value on which words are removed. If a word is found in less documents than this value - could either be a percentage or an absolute value, than the word is removed from the feature matrix. This word will therefore not be considered in the model.

– Max_df: This parameter is the same as min_df, but for a maximum: if the word is found in more documents than the threshold value, the word is removed.

2.1.3 Evaluation of classification tasks

There are many metrics one can evaluate their model with. Classifying film scripts is a multi-class and multi-label problem. Therefore using the accuracy metric will result in a low prediction score. This is because the model needs to predict the genres exactly right. The accuracy measure will not be used in this thesis. Other metrics which measure the feasibility of the model are the recall, precision and F1 score.

- Recall: measures the correctly predicted values (TP) over all correct values.

$$\frac{TP}{TP + FN}$$

- Precision: measures the correctly predicted values over all the predicted values.

$$\frac{TP}{TP + FP}$$

- F1 score: The harmonic mean between the recall and precision.

$$2 * \frac{Recall * Precision}{Recall + Precision}$$

2.1.4 Term weights

In text mining one can measure the weight of the words in two ways: counting the words in a document (*term frequency)* (e.g. with the help of *CountVectorizer*) or using *tf-idf*.

- Term frequency: counting each unique word across all the different classes.

$$tf(t,d) = \frac{Term\ t\ in\ document\ d}{Total\ words\ in\ d}$$

- Term frequency - inverse document: frequency (tfidf): counting each unique word across all different documents, but taking into account which other documents do or do not contain this word. The more documents contain the same word, the lower its weight. This means that the word poses a less likely identifier for the class to be considered, and vice versa.

$$tf - idf(t,d) = tf(t,d) * log(\frac{N}{df_t})$$

With N the total number of documents, df the number of documents containing term t

2.2 Related work

Prior research conducted on this subject was primarily conducted with regards to feature engineering. This paper [BS08] describes features obtained using NLP (natural language processing) methods. They have extracted features based on e.g. locations mentioned in the scripts. The amount of film scripts their data set contained were 399. Their source, among others, for the data was dailyscript.com, as they explained in their paper. Both their data set and ours had a maximum of 7 and a minimum of 1 genre per script (figure 3.3). The usage of multi-class and multi-label classification is no different than our research, with the exception of the amount of classes: 22 (we have a total of 18, but not all were used). A notable conclusion of both their paper as well as ours, is that the four most chosen genre are: Action, Drama, Comedy and Thriller. Which not entirely coincidentally are the genre with the most amount of scripts, as shown in figure 3.4. They used NLP cues to build their features, which can be seen as a combination of our text features and engineered features. The classifiers used were a selection of Naive Bayes classifiers (tree augmented Naive Bayes, forest augmented Naive bayes and more) as well as a MEMM (Maximum Entropy Markov Model). The MEMM is particularly useful in Part-of-speech tagging, which is the main approach in their research. The best classifier returned a F1 score of 0.4805 for the Naive Bayes, and 0.5196 for the MEMM classifier. Section 4 will refer to this paper more detailed.

The classification of film scripts using text features is a method which translates well into other creative processes as well, such as music and books. Both could also be used in the classification process. The following two papers will demonstrate genre classification with natural language as their core starting point.

B. Guthrie et al. [GHHW19] describes the process of classifying books in fiction or non-fiction. They have used word count and word frequency as features for their models, which yielded an accuracy of 75% for their binary classification task. Unfortunately, they have not stated any of the measures of the metrics (recall, precision, F1) we are using. Comparing their results with ours is therefore not possible.
We have also used the approach of word frequency in our word feature method. Identifying fiction from non-fiction is a binary classification task, single-labeled. This thesis focuses on the multi-class and multi-label classification. The thesis of B. Guthrie et al. is of interest because the process is fairly parallel: using the same libraries, following the same steps for pre-processing and model training. Further in our research we had to divert from their approach, because of the differences mentioned above.

Hu X. et al. [HDE09] describes the process of using both lyrical text information as well as engineered features (using methods such as Part of Speech) for the classification of moods of music. Their bag of words was the most efficient method for classifying the mood of a song. This method and classes are almost synonymous to classifying film genres. They filter out the indicative information such as ”written by ...”. In our research we will not be doing the explicit filtering. This will be handled by the vectorizer, further explained in section 3.

Chapter 3

Data and Methods

This chapter is dedicated to the process of collecting, cleaning and visualizing the data. A visual representation can be found in fig 3.1 [May17].

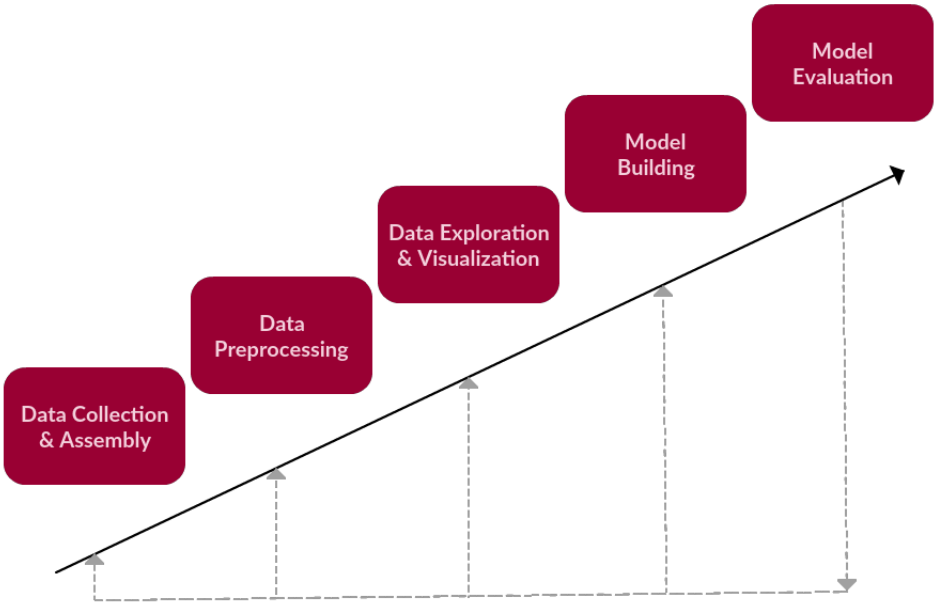

Figure 3.1: The process of collecting, cleaning and training/testing data

3.1 Data collection

The data used in this thesis has been embedded in a website (www.imsdb.com). In order for us to gain control of the scripts, we had to use a technique called *web scraping*. Web scraping is extracting information (visible on the client side of the web application) from a website and manipulation the data locally. The data (film scripts) was extracted and saved locally in a .csv file for easy use later on.
The scraping was automated using a Python library called BeautifulSoup. With this library one can select the contents within HTML 'tags': e.g. , <h1>, etc.

A few problems arose during the process:

1. The scripts were not always in the expected position of the website
2. The scripts had a different format (.pdf, .jpeg)

For the first problem in the list above we had to find a way if other scripts had this problem as well. If the positioning of the scripts on different part of the website was general enough (i.e. more scripts could be found on this part of the website), we could retrieve these data points as well. Thus establishing a bigger data set. Problem number two is beyond the scope of this thesis.
Figure 4.1 shows the relation between the successfully and the unsuccessfully retrieved scripts. Of those unsuccessfully retrieved, 31% (43 scripts) were wrongly formatted. This means that the data was either imprinted in .JPEG, encoded in .PDF or the like. 69% (95 scripts) resulted in an error during the scraping process. This mean the scripts were not in the targeted space/spaces on the website on which the scraping occurred. The name of these scripts were saved and later looked at manually.
There were a total of 1070 useful film scripts collected (figure 3.2). In total there are 18 genres, with varying scripts corresponding to those genres (figure 3.3)

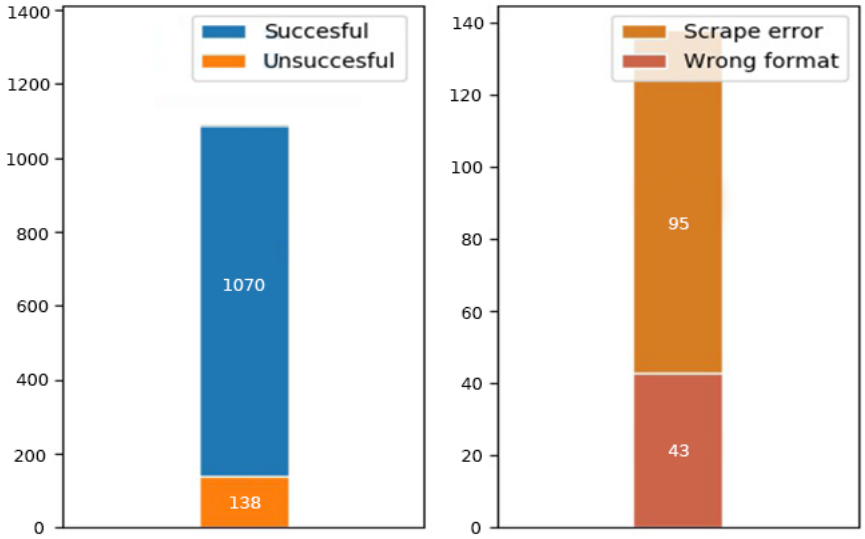

Figure 3.2: Total scripts (left), unsuccessfully retrieved scripts (right)

All scripts were saved in folders according to their corresponding genre. It is possible - more rule than exception - that the same script is found in more than one folder. This indicates that a script has multiple genres. We will go into further detail in section 3.3.

3.2 Preprocessing

3.2.1 Cleaning

Much of the data cleaning has already been done during the *data collection* (section 3.1): wrongly formatted data has been ignored, data files below the threshold of 100 bytes (indicating the tags on the site exists, but the content did not match the description (e.g. empty files, (re)moved files, etc.)) have been removed and errors adhering to different categories have been ignored or removed. This is done to ensure the quality of the extracted scripts.
The scripts are placed in folders according to their corresponding genre. In order to work with the film scripts, we had to load them into my program using the *Pandas* library.
As stated previously, there are two sections which we will be researching: word features and engineered features. The engineered features are based on the structure of the film scripts. There are no external data sets consulted. The dataframes of these two section will vary, as we shall demonstrate below in table 3.1 and 3.4. Note that the classes displayed may not be all the targets a film script has. The sample size has been reduced for visualisation purposes, thus resulting in the exclusion of certain genres and therefore fewer target classes are considered.

3.2.2 Allocating classes

Once the content of the scripts were loaded into the program, the classes needed to be determined for the scripts. In order to work with the assigned classes per script, the target (type: string) needs to be converted to a binary class. The column 'binary target' in the tables 3.4 and 3.1 shows the classes converted into binary lists. The lists are n classes big and each element is a binary representation of a script corresponding to a class. The index values correspond to the alphabetically ordered class names.

3.3 Visualization and exploration

To comprehend the scope of the data set, we need to visualise the data. Visualising the data can bring insights into the data in an easier to process manner. Figure 3.3 describes the amount of genres each script has, against the frequency of these occurrences. The mean is 2.65 genres per script.

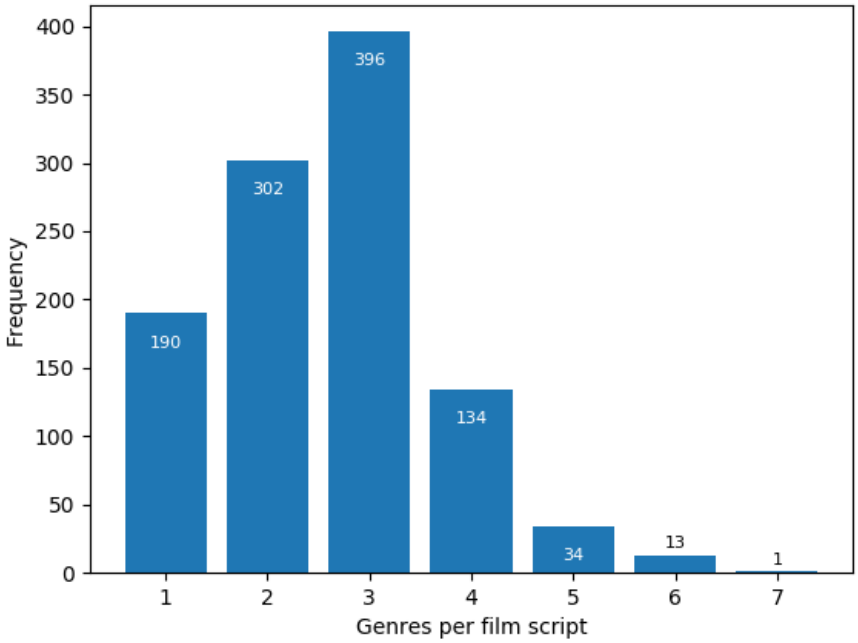

Figure 3.3: sum of the amount of genres per script

To visualize it differently, one could calculate the amount of scripts per genre. Figure 3.4 shows exactly that. This figure is not mutually exclusive: film scripts could correspond to more than one category (but at least one).

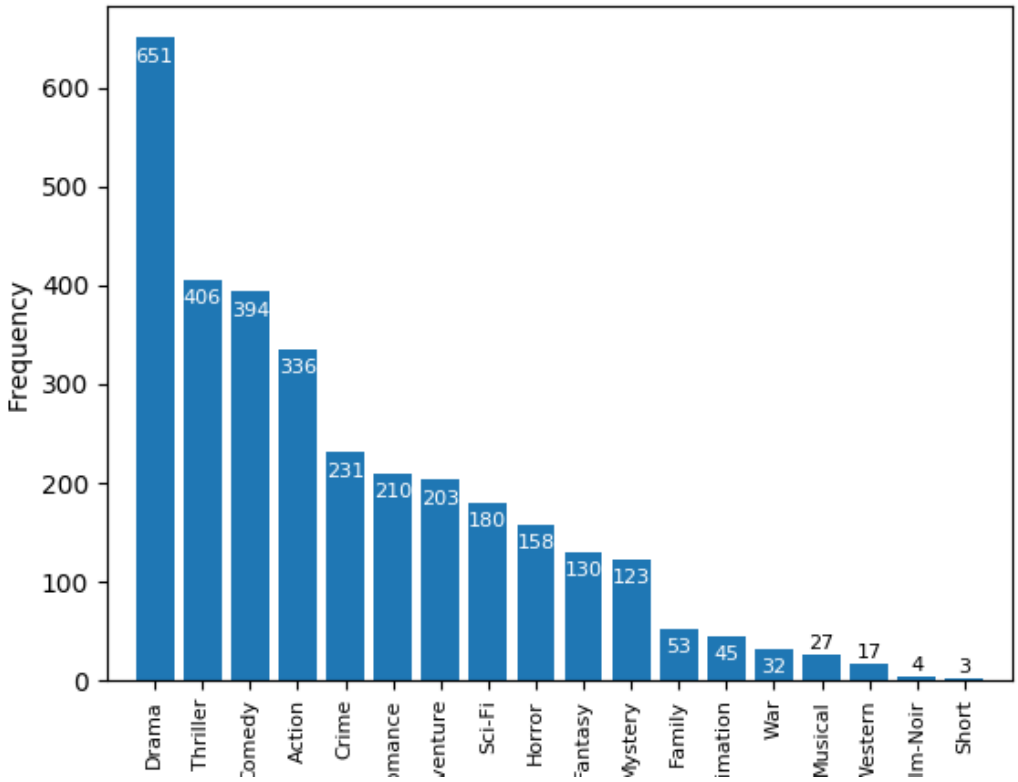

Figure 3.4: Amount of scripts which corresponds to a particular genre.

3.4 Feature extraction

3.4.1 Word features

Word features are the words in a film script. Each word could cumulatively add up to become an important predictor for a particular genre. First, we will describe the dataframe used in our research. We have used this dataframe as a base for our exploring research in this subject.

A snapshot of the final dataframe (including target and prediction) is constructed as described in table 3.1. The dataframe consists of five columns: filename, content, target, binary target and target predicted.

filename: the name of the film scripts.

content: the complete and raw filmscript.

target: the genre(s) associated with the film script.

binary target: values from 'target' column, but described in a binary list, *n* classes long.

target predicted: target predicted by the classifier displayed in string data type.

This dataframe contains the column 'content'. This is the full script of the corresponding file. In order to train the model on this data, we need to tokenize and then vectorize the words. The vectors resulting from this process are the input the model is training on. The vectorization is done by the functions CountVectorizer or tfidfVectorizer. These two functions first tokenize (using the fit() method) and then vectorize (using the transform() method) the words for the models. This will result in a sparse matrix, because the vector will

contain a lot of zeros due to the large vocabulary, created in the tokenization step. Table 3.2 shows the vectorized table of the 'content' column, with TF as the weight function. If TF-IDF were the weight function, the cells would involve fractions between 0 and ∞ (table 3.3). This vector is the input for the classifiers.

filename	content	target	binary target	target predicted
This Is 40.txt	\n\n\n THIS IS 40 \n\n...	[Comedy]	[0, 1, 0, 0]	[Comedy, Drama]
Mission Impossible.txt	\n\n\n "Mission: Imposs...	[Action, Thriller]	[1, 0, 0, 1]	[Action, Thriller]
Mission Impossible II.txt	\n\n\n "Mission: Imposs...	[Action]	[1, 0, 0, 0]	[Action]

Table 3.1: DataFrame example - feature text, before vectorization. The 'target predicted' column are the classes which the model predicts the script ought to be. It shows only a maximum of four classes, because this model has been trained on four classes.

bird	eggs	dancing	...	gun	more	waiter
2	0	0		18	6	3

Table 3.2: DataFrame example - content vectorized using TF

bird	eggs	dancing	...	gun	more	waiter
0.56	0.00	0.00		3.43	1.86	0.06

Table 3.3: DataFrame example - content vectorized using TF-IDF

Besides the different models that are trained for comparison, there are also two different weight functions: *term frequency (TF)* and *term frequency - inverse document frequency (TF-IDF)*. These two weight functions are defined in chapter 2.1. TF and TF-IDF use extraction methods for the tokenization of the word featuress. TF Uses CountVectorizer and TF-IDF uses TfidfVectorizer, from the sklearn feature extraction library. These vectorizers tokenize the features, meaning every word in the scripts (*content* column in the dataframe of table 3.1 and 3.4) are potential features for the models. TF only counts the words, which gives them a linear weight: if a word *'a'* is found ten times more frequently than word *'b'*, the weight of word *'a'* is weighted ten times more for the respective feature in the vector matrix. TF-IDF weighs differently: if a word is found in a script, but not in any of the other scripts, the vector value - or weight - of this word will be high. This weight will be greater than a word found in multiple scripts. The weight of a word is considered per document, and not per class. The matrices the vectorizers produce are sparse matrices, with the words from the film scripts as features. Many features do not produce much insight for classifying the target variable and will clutter the matrix and therefore be a burden rather than an asset for the model. In order to filter these words out, the vectorizers introduce certain parameters. The parameters are not only to filter words out, but also to find meaningful combination of words, which would be obvious for the human reader. These parameters are the following:

- stop_words: the words which will be filtered out of becoming a feature. One can provide its own list of (stop) words. We have provided the list of words from the nltk library. This list contains words such as 'the', 'a', 'and' and 'them'. After an initial run we have decided these words do not have an inherent meaning for genre classification. To avoid clustering, we have decided to filter these words out.

- max_df: max_df is short for *maximum document frequency*. This parameter is a floating point value between the 0 (zero) and 1, or an integer with an absolute number, meaning the amount of documents the vectorizer takes into consideration. The value is a threshold. If the document frequency of the word is above the threshold value (i.e. if too many film scripts contain the word), the word will be left out of the feature matrix. This is done because some words - much like the stop words in the previous bullet point - appear too frequently, they lose their value. We have found that a threshold value of .95 (i.e. if the word appears in 95% of the film scripts) yields the best results. In hindsight, tuning the parameters with a validation set would have been most effective. However, for this parameter tuning we have used a regular train and test split. The seed of the split during this phase depended on the time (i.e. using a time function to calculate the seed), which gave us a random records with the same train/test split division. We have gone through this process several times, to ensure the model was not overfit, or expressed errors or faults in other ways.

- min_df: the *minimum document frequency* is in all aspects the same as the max_df, except the words that are ignored are *below* its threshold value. The min_df value we have used is .10.

- ngram_range: word groups can also contain insights for the models to train on. However, we have found that the classification got worse when using word groups. We have tried groups of one (actual words), two, three and a combination.

3.4.2 Engineered features

The initial dataframe used for the engineered features should contain the filename, content of the film scripts, and the class. After this, the actual features can be derived. We focus mainly on the structure of the film scripts. This means we will check whether film scripts are using more punctuation in particular genres than others; this would for example indicate shouting or questions being asked. Once we were able to locate the punctuation whilst letting the structure of the scripts intact, we could start extracting structure based features from the content. The dataframe used for the model consists of thirteen features: filename, content, target, binary target, sum sentences, sum !, sum ?, sum int, sum ext, sum int/sum ext, sum !/sum sentences, sum ?/sum sentences and target predicted. The two features sum !/sum sentences and sum ?/sum sentences consider both the dialogues and monologues, this also includes thoughts of the characters. In the structure of the scripts - syntactically - there is no difference between the two.
The features that differ from the data frame in the section prior are described below.

sum sentences: the amount of sentences comprising the script, excluding scene headers and malfunctioned sentences.
sum !: amount of exclamatory sentences.
sum ?: amount of questions.
sum int: amount of scenes shot indoors.
sum ext: amount of scenes shot outdoors.
sum int/sum ext: ratio between in- and outdoor scenes.

sum !/sum sentences: ratio between exclamatory dialogue and total sentences.

sum ?/sum sentences: ratio between questions in dialogue and total sentences.

A snapshot of the final dataframe (including target and prediction) is constructed as described in table 3.4.

filename	content	target	binary target	target predicted
American Hustle.txt	\n\n\n AMERICAN H... \n\n...	[Drama]	[0, 1, 0, 0]	[Action, Drama, Thriller]
Colombiana.txt	\n\n\n COLOMBIANA \n...	[Action]	[1, 0, 0, 0]	[Drama, Thriller]
Mission Impossible II.txt	\n\n\n "Mission: Imposs...	[Action]	[1, 0, 0, 0]	[Comedy, Drama]

sum sentences	sum !	sum ?	sum int	sum ext	sum int/ext	sum!/sum sentences	sum?/sum sentences
11683	195	559	115	29	3.97	0.0167	0.0478
5398	162	265	198	80	2.48	0.0300	0.0491
4307	57	239	93	51	1.82	0.0132	0.0555

Table 3.4: DataFrame example - engineered features. The 'target predicted' column are the classes which the model predicts the script ought to be. It shows only a maximum of four classes, because this model has been trained on four classes.

3.5 Model building

Models are being trained and evaluated based on the (input) data provided. Before the model can be build, the data needs to be collected, sanitized, and features need to be extracted (sections 3.1, 3.2, and 3.4 respectively). The output of these processes are the input for the model.

The logistic regression classifier is a binary classifier. For our multi-class problem we had to use a one-vs-rest approach in order to have the logistic regression classifier function for our goal. This means that for every class the classifier will evaluate if the script fits into a genre, based on the features of the script at hand and on the ones it has learned per class. Furthermore, there can be made a distinction between the process of how the data is ranked on importance (the weight functions) and the model itself. There are two feature weight functions this thesis studies: *Term frequency* and *term frequency - inverse document frequency*. These functions are explained in chapter 2 and in section 3.4. The models we use in this thesis are *Support Vector Machines (SVM)* and *Logistic Regression* for text features, and *logistic regression* and *decision trees* for the engineered features.

For the models to perform well, and to know whether or not the results are just, we will also demonstrate the models using under-sampling. We have shown in the data visualization section that some genres have more scripts than others. To tackle this problem, we will be normalizing the input for the models. The scripts are normalized to a minimum and a maximum of two hundred scripts per genre. The minimum ensures that only genres who have two hundred scripts or more are selected for the model. The maximum ensures that only two hundred scripts will be used to train on.

3.6 Model evaluation

As described in previous sections, we have evaluated the models intuitively and based on the metrics we hold as standard (recall, precision and F1). We have used a random seed (using the time function) for the train-test

split. The test size was 0.50 (which leaves 0.50 for the training size) for the word features, and 0.25 for testing (0.75 for training) for feature engineering. The seed on which the split was made varies with a time function.

In the previous section we have mentioned the models we use. Two models stand out: the SVM and logistic regression. The SVM model has a hyperparameter C. We have tuned this soft margin (the support vectors) during the train phase. As in a previous section, we did this by training and testing the model on a random seed, to validate the performance of our model. However, the hyperparameter C showed little to no effect on the result of our model. Thus we used the default value $C = 1$. Logistic regression uses also the C hyper parameter. Same as with the SVM, after evaluating the results of this parameter, we have concluded this did not affect the model substantially.
We have used a threshold value to indicate whether a predicted genre should fit the script or not. This value was set to 0.33. This threshold value mean that if the classifier identifies a genre with a 0.33 or higher, the genre will be assigned to the script. This is the prediction. A higher threshold value would result in a peak in the recall metric, and a possible trough in precision, and vice versa. When evaluating the models, we kept both the metrics and the confusion matrices under due consideration.

Chapter 4

Results

After evaluating the models to find the best possible results, without over- or underfitting the model, we arrive at the results. Here we will showcase the results from our research. The output of the models will be divided into three sections:

- Word feature results
- Engineered feature results
- Comparison

Firstly, we will demonstrate the measures explained in chapter 2 for each of the different approaches. The approaches are the two methods of classifying: word features and engineered features. These are the first two result sections. Secondly - the third section -, we will compare the results to find the answer to our research question.

4.1 Word feature results

We will show the results in comparison to the two models used for this multi-label multi-class classification problem: Support Vector Machine (SVM) and logistic regression. We will also compare the two weight functions (TF and TF-IDF). The weight functions are used to weigh the importance of the terms (words) of the film scripts. This is where models are trained on. Both weight functions are applied to both models, and are compared. To serve the interest of this research, we will only discuss the best model. Table 4.1 shows the best model using its best weight functions: logistic regression with TF-IDF weight function. It should not be surprising that the TF-IDF function performs better than the TF weight function, since this is a more advanced version of the latter.

Details for logistic regression model:

- Minimum scripts: 30

- Maxmimum scripts: ∞
- Test size: 0.50
- Threshold value: 0.33

	Precision	Recall	F1	Support
Action	0.62	0.57	0.59	73
Adventure	0.71	0.39	0.50	39
Animation	1.00	0.10	0.17	10
Comedy	0.60	0.55	0.57	85
Crime	0.50	0.17	0.25	53
Drama	0.69	0.72	0.70	154
Family	0.00	0.00	0.00	12
Fantasy	0.04	0.40	0.08	23
Horror	0.59	0.21	0.31	32
Mystery	0.75	0.06	0.11	21
Romance	0.41	0.20	0.27	53
Sci-Fi	0.77	0.26	0.39	25
Thriller	0.52	0.63	0.57	87
War	1.00	0.08	0.14	9
Micro-average	0.63	0.43	0.51	676

Table 4.1: Measures per genre and the micro-average of the logistic regression model. Weight function used: TF-IDF.

This table shows all the measures of the metrics per genre. When both this table and confusion matrix 4.1 are compared, one can derive some interesting facts. When the spread between the recall and precision is relatively large, the confusion matrix shows that these genres are not often picked. One can deduce that the model is not trained well on these genres. The support metric shows on how many scripts the model is trained, and can confirm the previous statement. More on this in section 4.3

The runner-up was the SVM with TF-IDF weight function. Both the TF models performed the least well. The models are biased towards four genres: Action, Comedy, Drama and Thriller (note the similarities of the paper of A. Blackstock and M. Spitz ([BS08]). Since the film scripts of the genres included in this analysis have a lower limit of thirty, and an upper limit of infinity, the models are skewed and biased due to the input (training) data. The bias is towards the genres with more film scripts to train on, because there are simply more references for these film scripts. Figure 3.4 shows the genres with the most films scripts. Not surprisingly the four genres with the most data are the ones towards the models are skewed. To fix the bias, we have normalized the minimum and maximum amount of film scripts on which a genre can train on. In this case we had to under-sample the data set, because over-sampling would create extra features. These features are the words in a scripts. We want to see which words are most characteristic per genre, over-sampling would defeat that purpose. The lower limit and upper limit are both fixed on two hundred. This means that all genres have

two hundred (randomly sampled) scripts on which the models will train. Figure 4.2 show the results of this experiment. We only show logistic regression with weight function TF-IDF because this is the best classifier.

Table 4.6 will show the metrics in a clear overview, including the metrics of the other models and weight functions. These metrics are micro-averages, since there is a class imbalance. The accuracy metric is excluded from this research, because when calculating the accuracy for a multi-label classification problem, the predicted value and actual target must match. If they do not match perfectly, this will then contribute negatively to the accuracy metric, resulting in a low accuracy measure. The recall and precision metric are more useful in such scenarios.

4.1.1 Important features

Due to the weight function, a series of prominent features can be extracted. This extraction is based on the coefficient of the models. The following table will show the features of a small sample of classes, with model parameters being held constant (as described in the start of section 4.1). The features are displayed in descending order with regards to its importance.

	Drama	**Action**	**Horror**	**Romance**
1	eyes	gun	door	love
2	eve	fire	creature	star
3	cereal	bridge	blood	aunt
4	hospital	weapons	something	buddy
5	later	evil	dead	crash

Table 4.2: Logistic regression, TF-IDF. Best character words for identifying a genre, scored by the classifier

Table 4.2 shows the most prominent features for the classification of the genres. The genres are specified in the columns. These words are for the most part very self explanatory: 'gun' for the action genre, and 'blood' for horror. But some may be more dubious than others. In the same table, in the 'Romance' column, one can find the word 'aunt'. Evidently, the classifier links the word to the 'romance' genre more than to any other genre. Therefore it is possible that this word has an overbearing weight for the romance genre. Possible explanations could be that the training sample size was little, and this word characterizes the romance genre. This is possible because of the weight function: this word's frequency could be large in a script in the romance genre (TF), whilst not in any other genre (IDF).
The TF-IDF weight function fares quite well when looking at the important features. The TF functions leaves one wondering a while longer: Logistic regression, TF, 'action' genre, shows the "words" 'un' and 'de' as a few dominant features.
Another interesting, yet consistent, feature is that all the possible combination of classifier and weight function, it shows the word 'love' for the romance genre is the most dominant feature.

As seen in the tables above most of the words are logical correlations to the genres. Associating the word 'love'

with the 'romance' genre is rather obvious. However, there are a few anomalies which make for an odd 'most important features': 'Action' genre, the words 'un' and 'de'. Perhaps there were some multilingual scripts in the data, on which is heavily trained. We have not used any k-fold training techniques and since the program has been set up with a static seed when determining results, the model ought to be deterministic.

4.2 Engineered feature results

The engineered features have different parameters and models as the text based features. Instead of using a SVM and logistic regression, we have used a decision tree and logistic regression. The reason for this is the SVM is better at high dimension classification problems and also decision trees are easier to interpret than support vector machines.
Table 4.3 shows the three measures discussed in chapter 2 for the engineered feature models. The logistic regression classifier shows a more promising result in the recall metric, however this model is less reliable than the decision tree. We will be using the decision tree classifier as example from now on until the end of this chapter. Reason for this is, when we evaluated the under-sampled models, logistic regression diverted 59% (downwards, from 0.61 to 0.25) on the recall metric in comparison to the regular sampled model. The decision tree only shifted 7% downwards.

	Precision	Recall	F1
Decision tree	0.22	0.31	0.26
Logistic regression	0.16	0.61	0.25

Table 4.3: The measures of the decision tree and the logistic regression classifiers. Minimum of 30 scripts per class: fourteen classes are trained on.

Details for decision tree model:

- Minimum scripts: 30
- Maxmimum scripts: ∞
- Test size: 0.25
- Threshold value: 0.33

4.2.1 Important features

Table 4.4 shows the important features of the decision tree model.

Features	Feature importance (in %)
Sum sentences	7.53
Sum exclamatory sentence	6.69
Sum questioning sentence	7.21
Total interior shots	3.61
Total exterior shots	12.58
Interior exterior ratio	15.34
Exclamation in dialogue ratio	25.30
Questions in dialogue ratio	21.73

Table 4.4: The feature importance of the engineered features for the decision tree model.

The rows of this table are in the same order as in section 3.4.2, but the names are more suggestive. The feature which shows the most promising results in this list, is 'Exclamation in dialogue ratio'. This means that genres with the most (or the least) exclamations in dialogue as percentage of the total sentences, are the most distinctive for our model. Exclamations in scripts are usually in a dialogue, because the text surrounding dialogue are only clues for the scene's setting (such as props and clothing). Therefore scripts where characters yell or shout (or the exact opposite) are more likely to be predicted well. We have ensured to only use the dialogue of the scripts when extracting this feature. The same holds true for questions. The decision tree also shows that the most prominent features are the ones on which the split occur the earliest: exclamation in dialogue ratio and questions in dialogue ratio were the earliest and the information gain difference between the current node and its children were the greatest for these features. Unfortunately, the decision tree is too grand and too clustered (due to the multi-class and multi-label approach) to display.

4.3 Comparison

The logistic regression with TF-IDF is the best word feature classifier, and the decision tree is the best engineered feature classifier. Table 4.5 shows these models side by side. It is clear to see that the text feature model is performing better than the engineered feature model. We can conclude that the word features are more suggestive of the genre than the structure of a script. Tables 4.6 and 4.7 show all the metrics of the models. The bold text are the best performing measures in their respective category.

		Precision	Recall	F1
Word features	Logistic regression - TF-IDF	0.63	0.43	0.51
Engineered features	Decision tree	0.22	0.31	0.26

Table 4.5: Comparison of the metrics between the best classifier of the bag-of-words model and the engineered features model

Tables 4.6 and 4.7 show the results of the metrics of all the classifiers used in this research. The bold text indicate the best measure in its performance metric.

	Precision	Recall	F1
SVM TF	0.45	0.49	0.47
Log. regr. TF	0.21	**0.56**	0.31
SVM TF-IDF	**0.67**	0.34	0.45
Log. regr. TF-IDF	0.63	0.43	**0.51**

Table 4.6: All word feature metrics per model

	Precision	Recall	F1
Log. regr.	0.61	0.16	0.25
Decision tree	**0.22**	**0.31**	**0.26**

Table 4.7: All engineered feature metrics per model

Now that we know that the logistic regression with TF-IDF weight function is the best model for classifying film scripts by genre, we can compare the genres to each other. A confusion matrix shows the true value (row) of a class, against the predicted value (column) by the model. Matrices 4.1 and 4.2 show the two confusion matrices plotted for the logistic regression classifier from the bag of words model, the latter being under-sampled. The diagonal from top left to bottom right is the true positive (TP) line: which are the correctly predicted values by the model. The purpose of the confusion matrix is not to calculate the recall and precision, but it is a nice verification measure. The purpose is to show which classes are predicted for each actual class, to show which classes share features. It shows for instance in matrix 4.1 that at the true value of comedy, 122 times drama is predicted (along with some other genres). Vice versa for a true value of drama, 122 predictions of comedy are also predicted by the model. This means that even though the model is wrong in its prediction (prediction =/= true value, or - alternatively - not on the TP diagonal), the structure of both genres are similar enough to the model that it picked the other genre(s) as well.

Again, the four major genres are highly represented, therefore we will also look at the under-sampled confusion

matrix: matrix 4.2. The confusion matrix of figure 4.2 shows that once the input is normalized (under-sampled) for the model, i.e. the same amount of scripts are trained on, the matrix shows more promising results. One can then compare the genres with another, and deduct which genres are alike and which are not. The shape of the matrix is rather symmetrical to the TP line (especially the under-sampled matrix), which indicates the genres - based on frequency of words - are alike and the models are not over- or underfit. The genres which are the most symmetrical, are the genres that are most alike in terms of vocabulary. Both matrices should be considered when making these deductions. Action and thriller are both well represented in the matrices, and so are crime and thriller. Both these genres share common features. It also shows that drama and adventure should be highly correlated, according to matrix 4.1, but not according to 4.2.

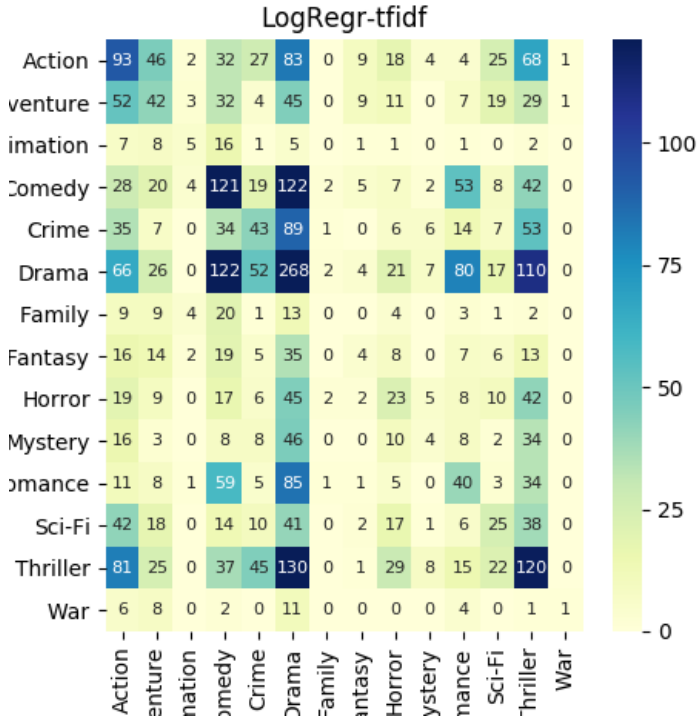

Figure 4.1: Word feature - Confusion matrix of genres: Logistic regression TF-IDF

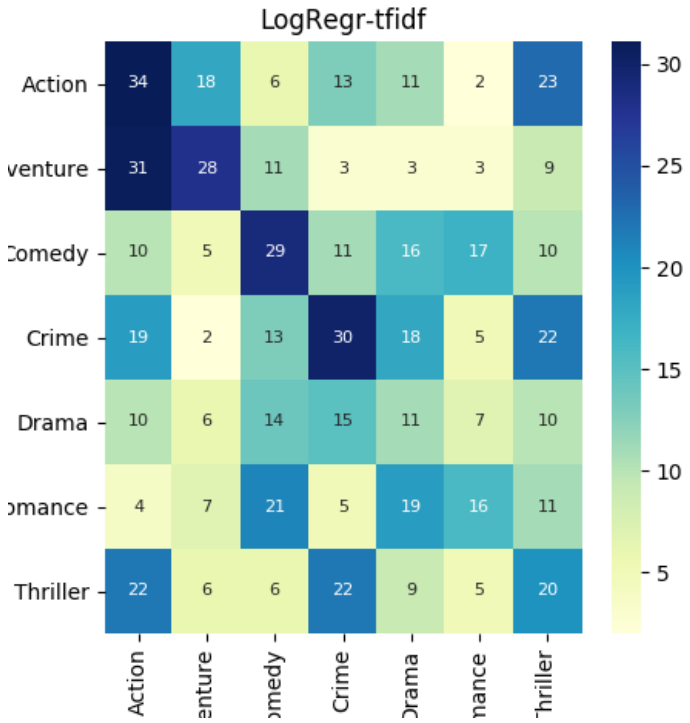

Figure 4.2: Word feature - Confusion matrix of genres: Logistic regression TF-IDF, under-sampled

Chapter 5

Discussion

We started this research with a sample size of approximately 1200 film scripts. These scripts are divided into a total of eighteen genres. The scripts are not equally distributed over the genres, which made the models less accurate because of the skewedness of this data in the training of the models. We have come to the conclusion that the models of both the text features and the engineered features worked as intended, since there were similarities to the work of A. Blackstock and M. Spitz [BS08], albeit with a more populated sample space. Also, the confusion matrices show symmetrical results, which indicates that prediction 'A' finds actual class 'B' and vice versa; the features of the genres are similar to each other. About half of the scripts belong to the 'Drama' genre, and around a third to 'Thriller', 'Comedy' and 'Action'. With a completely random guess the classifier would still be able to predict the genre reasonably well. This will have consequences for recall and precision. To counter this problem we have tried to normalize (under-sample) the input to two hundred scripts per genre, shown in section 4.1 and 4.2. It shows a smaller recall and precision, reason being the 'guess-factor' is smaller and there are fewer scripts to train on. As mentioned before, half of the scripts fall into the 'Drama' class. Randomly predicting - i.e. guessing - 'Drama' will yield true 50% of the time.
This 'guess-factor' is also present in the engineered feature predictions. Since both models suffer from the 'guess-factor' bias, we can conclude that the text features are performing better than the engineered features. Even when taking the 'guess-factor' into account.

Chapter 4 shows both the text feature results and the engineered feature results. Table 4.2 shows the most prominent words per class. If we combine these sample of words with the confusion matrices, we can see why some genres are more related to each other than others. When we look at the 'action' genre, matrix 4.1 shows this genre is both predicted and the correct class quite frequently. It is also highly correlated with the 'adventure' genre, as the matrices tell us. When we delve deeper into the coefficients of the model, it tells us that in the top twenty-five most characterizing words for both genres, six match: 'bridge', 'sword', 'desert', 'drew' (e.g. sword drawing, past tense), 'guards', 'chuck' (synonymous for throwing).
We can also take the least correlated genres and come to the same conclusions. If we compare 'crime' and 'adventure' we see in their models that their top twenty-five words have far fewer features (words) in common, only one: 'guys'. The take-away from these analyses are, some genres are more correlated than others, because

of their inherent similarities in their overlapping story subjects.

It shows the text engineered models are performing better than the feature engineered models. The conclusion we can draw from this, and thereby answering the research question, the feature representation using the bag of words approach is superior to the features regarding the structure of the film scripts.
Our paper is a mere exploratory research. An example and motivation for future endeavours for finding objectivity in a subjective discipline.

Chapter 6

Conclusions

The future research in and around the subject of classifying film scripts can be extended into the realms of its economic value. This research has been the stepping stone for machine learning into the creative process that is writing films. Future work could focus on the objectivity of film scripts: which parts work well with the audience, and what does not work well. These parts could be the words in a dialogue, the length of a sentence or interior-exterior ratio (as described in chapter 3). Research should look for the underlying structures and principles, with regards to the audience: what is the flow of the film that connects with the audience. Such structures date back to the time of Aristole [AriBC]. He has described the dramatic structure for theatre play, which is still the foundation of modern storytelling. The dramatic structure is the foundation (or skeleton) of a theatre play, but adopted by filmmakers and writers. This structure is a mere guideline, but if this structure together with the audience preference can be mapped, one can anticipate which films will or will not be a success.
This paper can also be extended to other media, such as books or even music. As described in the paper of Xiao Hu, J et al. [HDE09], lyrics can be easily interpreted by NLP tools. If songs are extracted by their lyrics or structure (text/word features and engineered features, respectively), bound together with a target value such as 'top 100 song' (binary), one can predict which songs will be a hit.

There are limitations to the research conducted in this paper. In section 4.1 we have discussed the most important words of the models. In order to minimize the outlier probability, a k-fold cross validation could be conducted. This way the training sets get shuffled each iteration, minimizing the risk of anomalies (i.e. less overfitting and less bias). An other point may be to get an equal amount of data per genre. You will reduce the amount of bias or favoritism towards one or more classes, thus reducing the need for over- or under-sampling. Cleaning the data is an important factor on determining the class(es). To benefit more from the strength of the weight functions, one could use stemming and lemmatization on the word features. By doing so, more words that are semantically the same but syntactically are not, are categorized as the same feature. This increases the weight of the words.
The engineered features are less represented than the text features in this thesis. In order to compete fairly,

there need to be more features describing the structure of the film scripts. In future work more time should be devoted towards this aspect of the research.

Bibliography

[AriBC] Aristotle. Poetics. c 335 BC.

[BS08] Alex Blackstock and Matt Spitz. Classifying movie scripts by genre with a memm using nlp-based features. 2008.

[GHHW19] Ben Guthrie, Jordan Henstrom, Ben Horrocks, and Ryan West. Determining genre of classical literature with machine learning. 2019.

[HDE09] Xiao Hu, J. Stephen Downie, and Andreas F. Ehmann. Lyric text mining in music mood classification. 2009.

[HNP05] A. Hotho., A. Nürnberger, and G. Paaß. A brief survey of text mining. 2005.

[May17] Matthew Mayo. A general approach to preprocessing text data, 2017.

[Mie06] A. Mieczyslaw. Intelligent information processing and web mining. 2006.

[Wat19] Amy Watson. Global box office revenue from 2005 to 2019, 2019.