Media Engineering and Technology Faculty
German University in Cairo

Classifying Movie Scripts by Genre

Bachelor Thesis

Author:	Hadeel Adel Abbas Mostafa
Supervisors:	Dr. Mohamed Ahmed Elmahdy
	Eng. Maged Shalaby
Submission Date:	15 May, 2017

Media Engineering and Technology Faculty
German University in Cairo

Classifying Movie Scripts by Genre

Bachelor Thesis

Author:	Hadeel Adel Abbas Mostafa
Supervisors:	Dr. Mohamed Ahmed Elmahdy
	Eng. Maged Shalaby
Submission Date:	15 May, 2017

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Hadeel Adel Abbas Mostafa
15 May, 2017

Acknowledgments

First of all, I want to express my deepest thanks and gratitude to my supervisor Dr. Mohamed Elmahdy for his guidance, his direction which was extremely important and productive, his great support and motivation. I would like to extend my gratitude to Eng. Maged Shalaby for his help, ideas, technical advices and support. Last but not least, I would like to thank my family and friends for their support and encouragement especially my brother Haitham for his continuous motivation, his help in time of need and patience.

Abstract

Text Classification is a supervised Machine learning task which classify documents by topics. This thesis explores assigning genres to movie scripts that is taken as text. The text taken consists of movie dialogues as movie scripts consists mainly of dialogues. Different classifiers are used namely: Naïve Bayes, Support Vector Machine and Random Forest Classifier. The problem is a multi-class and multi-label classification as a movie can have more than one genre. Thus, one-vs-all classification approach is used. Words existence, Term Frequency, Term Frequency Inverse Document Frequency and Latent Dirichlet Allocation topic distribution values are tried as features values for different classifiers. Evaluating the different classifiers, choosing the best one and discovering best technique for the classification is the aim of this thesis.

Contents

Chapter 1

Introduction

1.1 Motivation

Genres in simple words are categories for movies. In order to let people find a particular type they like and enjoy, movies are classified by genres and movies can have more than one genre. On choosing a movie to watch, people tend to search for movies having the genres they like or consider movie recommendations. IMDB, which is the source of movie scripts genres of dataset used in this thesis, is a good reference for movie genres and TasteKid is good for movie recommendations. An expert determines the genres of a movie or people use their own interpretation to vote for them. As movies having same genres have common words and dialogues, it is feasible to develop algorithms to classify movies by genres which can help to classify obscure movies whose genres are undetermined.

1.2 Main Objective

It is not easy to declare the genres of a movie. Someone watching a movie might classify it as thriller while another watching the same movie might classify it as horror, there is no standard for such a classification. It may be useful to develop algorithms that do these classifications and these algorithms need machine learning techniques to do the job. In this thesis different classifiers will be tried to assign the genres to movie scripts using a dataset of already classified movies. Moreover, testing the different techniques for such classification and choosing the best one will be done. This can be done as movies having same genre, have words in common.

1.3 Structure of the Thesis

In Chapter 2, we discuss some definitions and the classifiers that are used in the implementation. Then we move on to the steps of the implementation in Chapter 3. Chapter 4 shows the results of our implementation and discussion about them. Concluding the thesis in Chapter 5. Finally, our recommendation for future work in Chapter 6.

Chapter 2

Background

2.1 Machine Learning

Machine learning is used in various things like recommendation systems for You-tube and Amazon, auto-text completion like in Google search, email Spam filter, PC playing vs. a player in games like Go, etc. The work of machine learning is not hard coded, as conditional statements, since it requires experts to put the rules. Also, changing a statement or a block will lead to changing enormous other things in the code. So, machine learning is basically doing statistical calculations, processing of data and extracting features from the data to get the results. There are two types of machine learning: supervised and unsupervised learning.

2.1.1 Supervised Machine Learning

Supervised machine learning, which is addressed in this thesis, is considered when the data with their predictions are available on training. For example, In sentiment analysis for movie reviews, two different review-classes are predefined; either positive or negative. Accordingly, a new movie's review will follow any of the predefined classes. Another example is email Spam filter which checks whether an email is Spam or not [2] [3].

2.1.2 Unsupervised Machine Learning

Unsupervised machine learning is considered when the data is available in the absence of their predictions. It is harder than the supervised because the types of classes are not predetermined and their number is also unknown. For example, dividing customers into groups that have similar preferences, such groups can be "gamers", "computer geeks", "students" or others. [2] [3].

2.2 Natural Language Processing

Natural Language Processing (NLP), which is part of Artificial Intelligence (AI), is to make computer understand human spoken language (Like English). Machine Learning makes that possible as it analyzes the data and tries to catch the pattern and understand it. NLP may take as inputs both speech and written text, but another field is responsible for the speech processing part in case of speech input. The two main goals of NLP are Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLG aims at ordering a computer to generate the Natural Language. Unlike NLG, NLU is creating representation like sparse trees from the input Natural language. Therefore, NLG can be considered as the opposite operation of NLU [11].

2.3 Tools

2.3.1 Natural Language Toolkit

In order to deal with Natural Language processing with python on different operating systems like Windows, Linux and Mac, Natural Language Toolkit (NLTK) is one of the most famous easy to use platforms for such a task. At the beginning, NLTK started with a computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania and later many people added and contributed in its development. NLTK can do different tasks including: word and sentence tokenization, stemming, classifying, identifying named entities, etc. Also, NLTK has over 50 corpora and lexical resources [4].

2.3.2 Scikit-learn

Scikit-learn (sklearn) is an open source free library for machine learning that is mostly written in python. Originally, it started by David Cournapeau in a Google summer code and later contributions were made by others [1]. NumPy, SciPy, and matplotlib are used to build the sklearn.

2.4 Naïve Bayes Classifier

One of the most common techniques for classification, that we will be using, is Naïve Bayes (NB) which is based on Bayes Theorem. From its name NB is a naïve approach that is known for that it treats the features as if they are independent of each other (Conditional independence). Also, the bag of words assumption which assumes that the position does not matter. For example, a word at the middle of a movie script and another at the end from NB point of view their position will not affect the probability.

According to Bayes Rule

$$p(A \mid B) = \frac{p(B \mid A) * p(A)}{p(B)}$$

where $p(A \mid B)$ is the posterior probability, $p(B \mid A)$ is the likelihood and $p(A)$ is the prior probability.

Which can be translated to

$$p(G_k \mid d_1, d_2, ..., d_n) = \frac{p(d_1, d_2, ..., d_n \mid G_k) * p(G_k)}{p(d_1, d_2, ..., d_n)}$$

where G represents the genre and d represents the document

Since $p(d_1, d_2, ..., d_n)$ is independent of G and could be considered as a constant

$$p(G_k \mid d_1, d_2, ..., d_n) \propto p(d_1, d_2, ..., d_n \mid G_k) * p(G_k) \tag{2.1}$$

From NB Conditional independence

$$p(d_1, d_2, ..., d_n \mid G_k) = \prod_{i=1}^{n} p(d_i \mid G_k)$$

Therefore equation 2.1 can be written as

$$p(G_k \mid d_1, d_2, ..., d_n) \propto \prod_{i=1}^{n} p(d_i \mid G_k) * p(G_k)$$

$$\Downarrow$$

$$\hat{G_k} = \arg\max_{G} P(G_k) \prod_{i=1}^{n} P(d_i \mid G_k)$$

2.5 Random Forest Classifier

Random Forest can be described as a number of correlated decision trees and the word forest is used to symbolize the number of trees. The Random Forest classifier is based on the bagging technique which creates a model with low variance by averaging the noise and unbiased models. Subsets are created from the features set superset (subsets can overlap, may have different sizes and are chosen randomly thus the word Random) to create a number of decision trees. Suppose we want to predict some sample, in that case we make each tree of the decision trees predict which class the sample should belong to and check the voting for each class and the class with maximum vote is the class which the sample should belong to [9].

2.6 Support Vector Machine

Support Vector Machine (SVM) goal is to draw a hyperplane (also known as decision boundary) that divide the training vectors into two classes like shown in Figure 2.1 where there are two classes the circles and triangles and the hyperplane separates both of them.

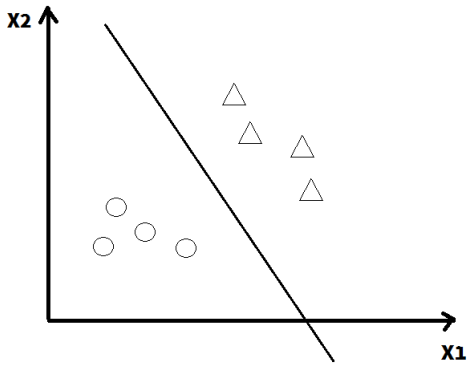

Figure 2.1: Hyperplane separating the two classes

So, if there can be more than one hyperplane, the best hyperplane to choose from is the one that provides the maximum margin from both classes which can be observed by the perpendicular distance between points of the class and the hyperplane. In Figure 2.2 there are two hyperplanes L1 and L2 ,however, it is clear that L1 has more margin thus making it the chosen hyperplane.

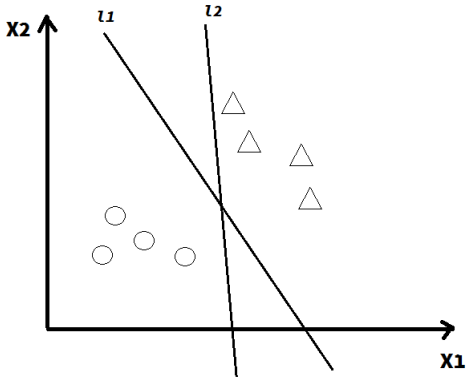

Figure 2.2: L1 is the better hyper plane as it leaves more margin between the two classes

2.7 N-Grams

Probabilistic language model goal nowadays is to assign probability to a sentence or a sequence of words so that for example in Machine Learning we could differentiate between good and bad translations, spelling correction, speech recognition, etc.

If we have the a sequence of words (history) and want to get the probability of a certain word (w) occurring after this sequence, then find the number of times the sequence happen followed by this word and the number of times without it.

$$P(w \mid history) = \frac{count(history \text{ followed by } word)}{count(history)}$$

However, this way is not that smart as the English language could have many new sentences that are created very often. So, we should get the probability in a smarter way.

Using chain Rule

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)...P(w_n \mid w_1^{n-1}) \\ &= \prod_{k=1}^{n} P(w_k \mid w_1^{k-1}) \end{aligned} \tag{2.2}$$

Where probability of sequence of words N is either $w_1..w_n$ or w_1^n

From Equation 2.2, the joint probability can be calculated by multiplying conditional probabilities, however, it is still the whole sentence. So, instead of doing so for the whole sentence, we do it for the last N number of words and that is the N-Grams [12].

In case of bi-gram (N=2), it is enough to check the preceding word only and thus the conditional probability is approximated to

$$P(w_n \mid w_1^{n-1}) \approx P(w_1 \mid w_{n-1})$$

Table 2.1 shows an example of a sentence and different values of N for N-Grams

N	**Type**	**#1**	**#2**	**#3**
1	Unigram	Keep	moving	forward
2	Bigram	Keep moving	moving forward	
3	Trigram	keep moving forward		

Table 2.1: Unigram, Bigram and Trigram of the English sentence "Keep moving forward"

2.8 Latent Dirichlet Allocation

People can guess the topics of a document by reading it and understand what it is about and that is possible as there is enough knowledge about the world. We need a similar function for machines and here comes the role of LDA.

LDA is a three-level hierarchical Bayesian unsupervised learning model which considers K set of topics. It considers each document as a collection of topics with percentages of how much of the document belong to this topic. For example: document1 is 20% topic1, 30% topic2 and 50% topic3, where each topic is a collection of words with probabilities for them to belong to that topic and these words are the features that carry the information.

In LDA, one word may belong to more than one topic as it has different meaning. For example: the words lead could be considered as noun (metal) and could be considered as verb (lead the organization). Also, the order of words does not matter as LDA follows the bag-of-words assumption. For example: consider words like (heart, vessels, blood, arteries, cells, organisms and bacteria), they indicate a biology topic and there was no need of order for the words or structure to infer that [6].

2.9 Related and Previous work

There are different approaches for identifying the movie genres: approaches that rely on pictorial data and video analysis like "Movie Genre Classification with Convolutional Neural Networks". They used the movie trailers in order to classify the movies by genre using CNN-MoTion (Convolution Neural Networks for Movie Trailer Classification) a method that encapsulate convolution neural network architecture [13]. Another is Movie Genre Classification via Scene Categorization which also uses movie trailers to classify the movie by genre and they considered only four main genres: action, comedy, drama and horror using state-of-art scene feature detectors and descriptors namely: GIST, CENTRIST and W-CENTRIST [14].

Other approaches rely on textual data such as Movies Genres Classification by Synopsis [10] in which four different methods for determining the genre was used: One-Vs-All approach using SVM, Multi-label K-nearest neighbor (KNN), Parametric mixture model (PMM) and Neural network (NN). The genres that they considered with percentage of movies in them are action (13.2%), adventure (9.1%), comedy (30.0%), crime (13.1%), documentary (16.3%), drama (49.1%), family (11.0%), romance (15.65%), short lms (33.4%) and thriller (13.6%). Table 2.2 shows the results.

Another textual approach is Classifying Movie Scripts by Genre with a MEMM Using NLP-Based Features in which similar to our case classify movie scripts by genre. In this approach, two classifiers are used: NB Classifier and a Maximum Entropy Markov Model (MEMM) Classifier. The movie scripts were obtained online from websites like

	Precision	Recall	F-score
SVM(Default)	0.66141	0.39572	0.47151
SVM(weighted)	0.47689	0.62533	0.53785
SVM(1:1)	0.51205	0.61631	0.54999
KNN(k=97)(MLE)	0.40987	0.73400	0.51580
KNN(k=88)(MAP)	0.64093	0.33261	0.40060
PMM(MLE)	0.46371	0.54234	0.48550
PMM(MAP)	0.53624	0.45876	0.47375
NN(λ = 1,PC = 1000)	0.67630	0.41513	0.49896
NN(λ = 1,PC = 4000)	0.65444	0.43225	0.50849
NN(λ = 1,PC = 8000)	0.62493	0.45708	0.52044
NN(λ = 1, No PCA)	0.64453	0.43666	0.50938

Table 2.2: Results for Movies Genres Classification by Synopsis [10]

Genre	Count
Drama	236
Thriller	172
Comedy	119
Action	103
Crime	102
Romance	76
Sci-Fi	72
Horror	71
Adventure	62
Fantasy	48
Mystery	45

Genre	Count
War	17
Biography	14
Animation	13
Music	12
Short	11
Family	10
History	6
Western	5
Sport	5
Musical	4
Film-Noir	3

Table 2.3: The genres and their count for Classifying Movie Scripts by Genre with a MEMM Using NLP-Based Features [5]

dailyscript.com and the number of scripts used for both train and test is 399 which was divided 90% and 10% for training and testing respectively. Moreover, they had 22 different genres [5]. Table 2.3 represent the genres and their count.

And the four main components they used in their experiment are: Non-NLP, Basix-NLP, Part of Speech tagging (POS) and Named Entity recognition (NER) and as for the evaluation metric, they used relaxed version of F1 score called Partial Credit (PC) Score. As for the results for the MEMM classifier it is found in Table 2.4.

Another paper published July last year Automated Movie Genre Classification with LDA-based Topic Modeling also tackles the same problem of classifying movie scripts by genre [7]. They used Latent Dirichlet allocation (LDA) to train their model. They considered the following 9 genres: Action, Adventure, Comedy, Crime, Drama, Horror,

Feature Set	PC Score	F1 Score
all	0.52588	0.47244
all-no-speaker	0.52588	0.47244
base-nlp-and-ner	0.52388	0.45669
base-nlp-and-ner-no-speaker	0.52943	0.46457
base-nlp-and-pos	0.53327	0.48031
base-nlp-and-pos-no-speaker	0.60133	0.55147
base-nlp-and-pos-and-ner	0.53777	0.46457
base-nlp-and-pos-and-ner-no-speaker	0.54388	0.47244
just-base-nlp	0.56310	0.51969
just-ner	0.50371	0.43307
just-ner-no-speaker	0.50371	0.43307
just-non-nlp	0.54960	0.48819
just-pos	0.4882	0.56498
non-nlp-and-base-nlp	0.55861	0.48031
non-nlp-and-ner	0.52922	0.47244
non-nlp-and-ner-no-speaker	0.52922	0.47244
non-nlp-and-pos-and-ner	0.53269	0.46457

Table 2.4: Results for Classifying Movie Scripts by Genre with a MEMM [5]

Mystery,Romance and Sci-Fi. The number of movie scripts used is 1094 and are obtained from Internet Movie Script Database (IMSDB). Figure 2.3 shows the training set Genre distribution.

The training and test sets were divided 80% and 20% respectively. As a baseline, they used the top 5 and 6 genres as the output of their predictive system (ML5 and ML6). They also tested random forest and SVM and Table 2.5 shows the results.

Algorithm	Precision	Recall	F-measure
ML5	0.3242	0.6672	0.4167
ML6	0.3066	0.7346	0.4160
SVM	0.5028	0.3028	0.3516
Random Forest	0.5563	0.3530	0.4068
LDA5	0.3183	0.7206	0.4415
LDA6	0.3549	0.8036	0.4923

Table 2.5: Results for Automated Movie Genre Classification with LDA-based Topic Modeling [7]

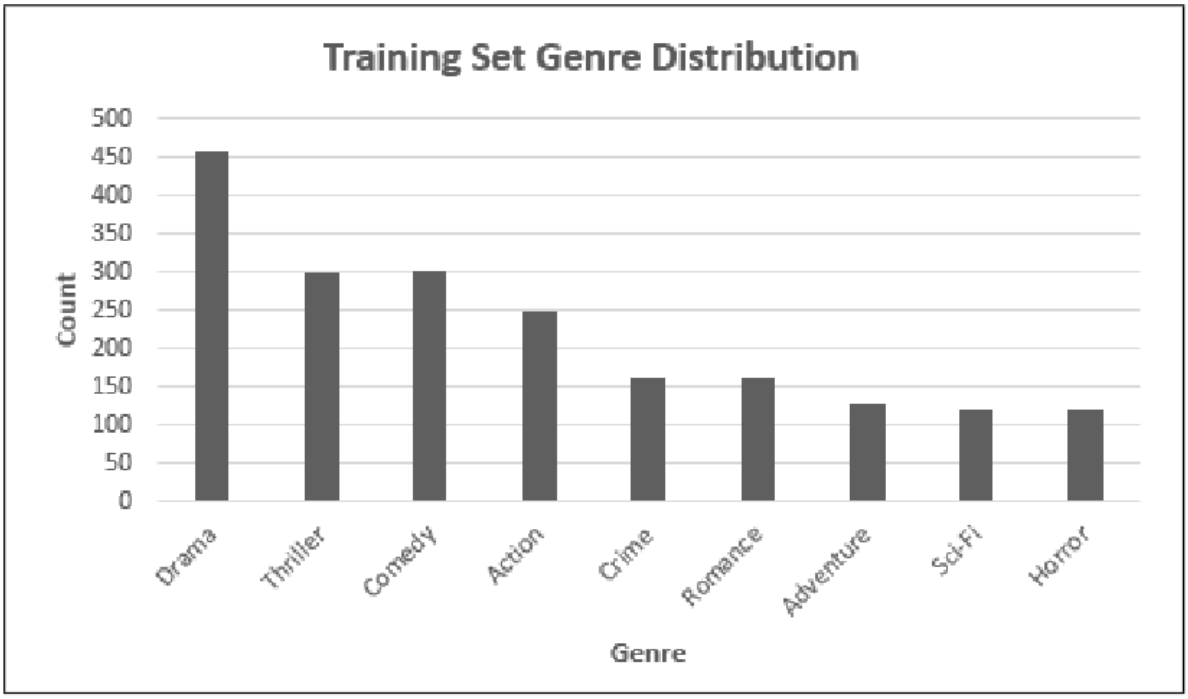

Figure 2.3: Training set Genre Distribution for Automated Movie Genre Classification with LDA-based Topic Modeling [7]

Chapter 3

Methodology

In this chapter the dataset and how the classification took place will be discussed. Figure 3.1 shows a flowchart showing the proposed movie genre classification pipeline. As shown in the figure, after reading the dialogue, there is preprocessing the data, choosing feature type, then choosing feature value, followed by feature selection, classification and finally evaluation. Choosing feature type is choosing from LDA, top(N) Inverse Document Frequency (IDF), frequent words or N-Grams. On the other hand, choosing feature values involve choosing boolean, Term Frequency (TF), Term Frequency Inverse Document Frequency (TF-IDF) or LDA topic distribution values feature value.

3.1 Dataset

Movie scripts is mostly dialogues so movie dialogues are obtained from Cornell Movie–Dialogs Corpus [8]. They obtained the movie scripts from dailyscripts and the genres from IMDB. There are 24 genres in these datasets from which 14 are eliminated to have the genres: 'crime', 'comedy', 'thriller', 'horror', 'romance', 'adventure', 'action', 'drama', 'sci-fi and 'mystery'. Subsequently, there are 610 movies remaining as movies with no genre are eliminated and Figure 3.2 shows distribution of genres in the movies after elimination.

The average number of genres per movie is equal to 2.623, where minimum number of genres is 1 and maximum is 6. Figure 3.3 shows number of movies containing the count of genres.

The movie dataset is split 80% and 20% for training and testing respectively.

As for the preprocessing of the data, some movies had repeated genres for example: movie1 had genres [g1,g1,g3] the extra g1 is removed and as for movies missing their genres, we got their genres from IMDB. We got the movie dialogues of each movie alone by checking the id of the movie which is in the format of 'm'+number as dialogues of different movies were together and could be differentiated by the id of the movie associated with the dialogue. After that, tokening the dialogue to words using NLTK and getting the frequency of each word across all the training set for later usage is done.

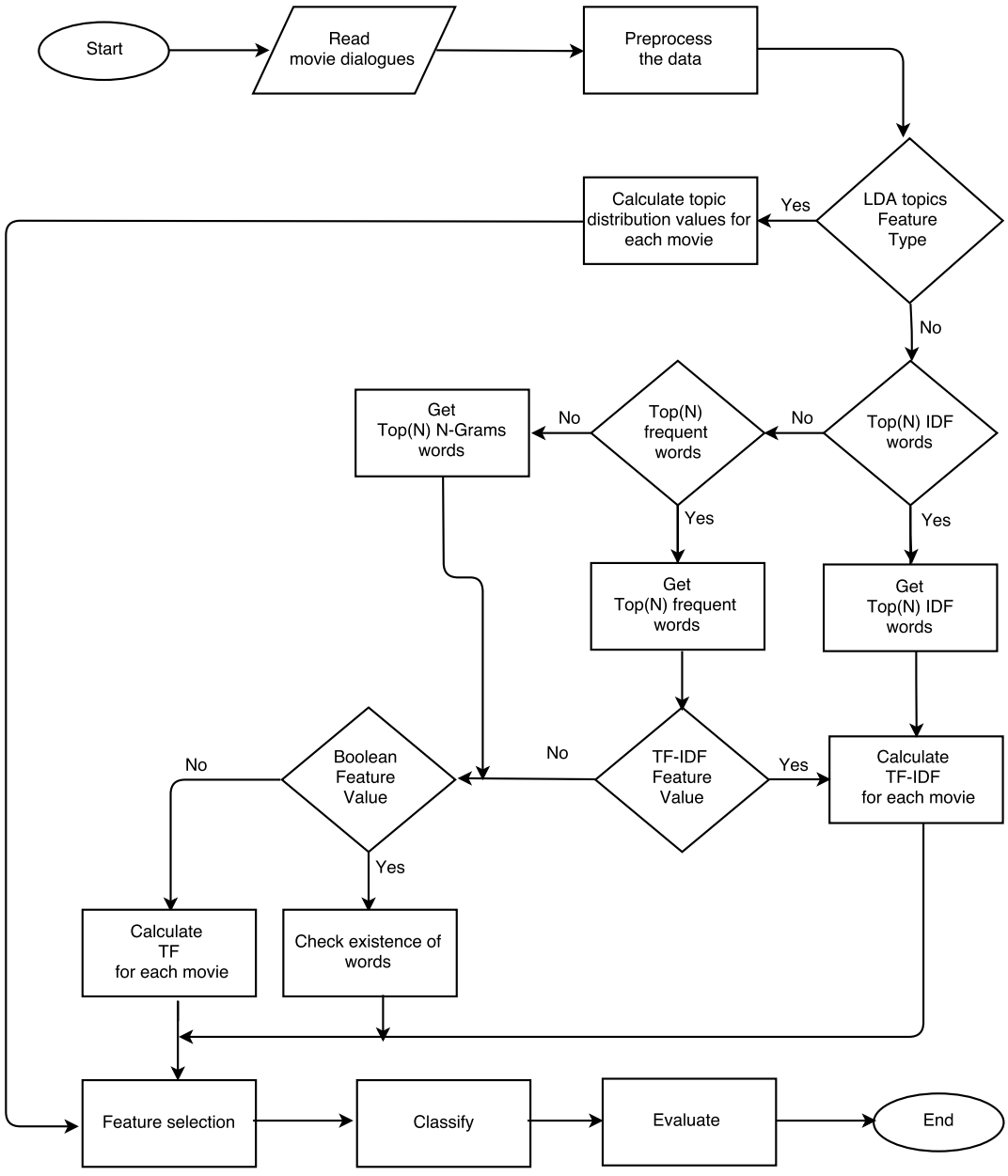

Figure 3.1: Flowchart showing the proposed movie genre classification pipeline

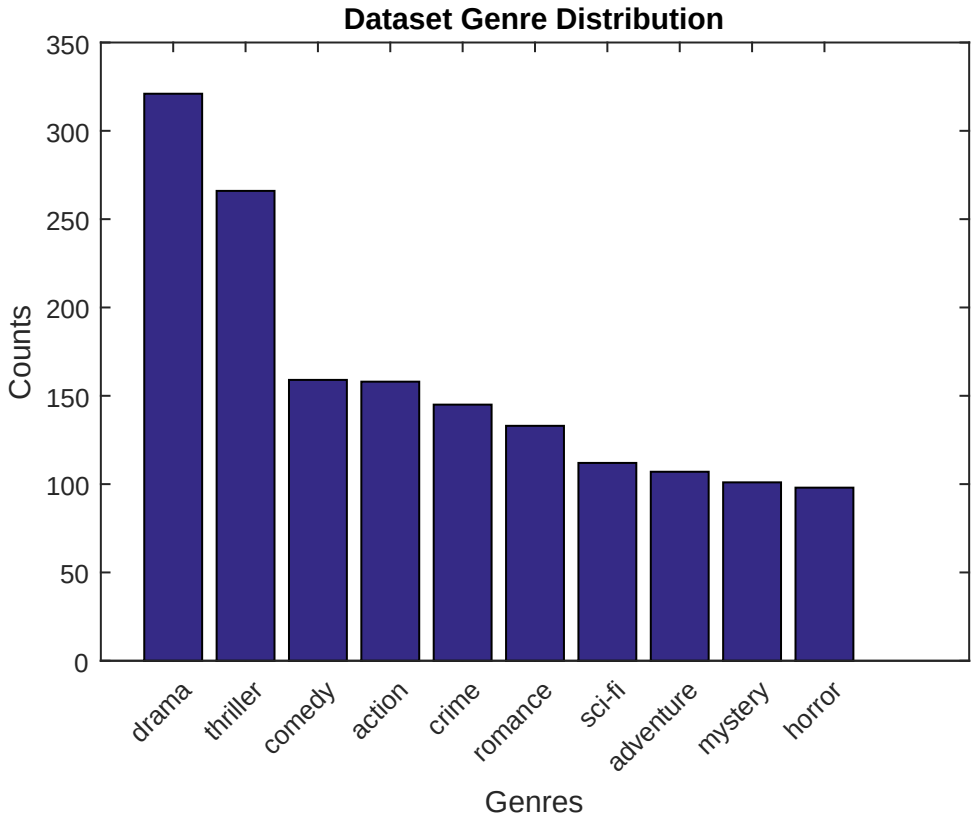

Figure 3.2: Dataset genre distribution: showing the genres and how many movies having a specific genre

3.2 Features

Word-tokenizer is used to convert the movie dialogue into separate words and then the frequency of each word is calculated among the words in the training set. The words is then sorted by frequency and top(N) frequent words (most frequent words) are chosen to be the features as a start. Features are the useful information extracted from the data given (movie scripts in our case) to feed it to the classifiers. The values chosen for N are 2,000, 3,000 and 4,000. In some runs, stop words (which are the extremely commonly used words that search engine usually ignores for example stop words in English language: The, a, he, she, etc.) are removed while experimenting because they do not add genre related information to the model.

3.2.1 Boolean

After having top(N) frequent words from the training set, for each movie it is checked whether each word in the top(N) frequent words exists in the movie or not (features have Boolean value and the value is: if found then True else if not found then False) which we will refer to later in this thesis as Boolean feature value. This operation is repeated after removing the stop words.

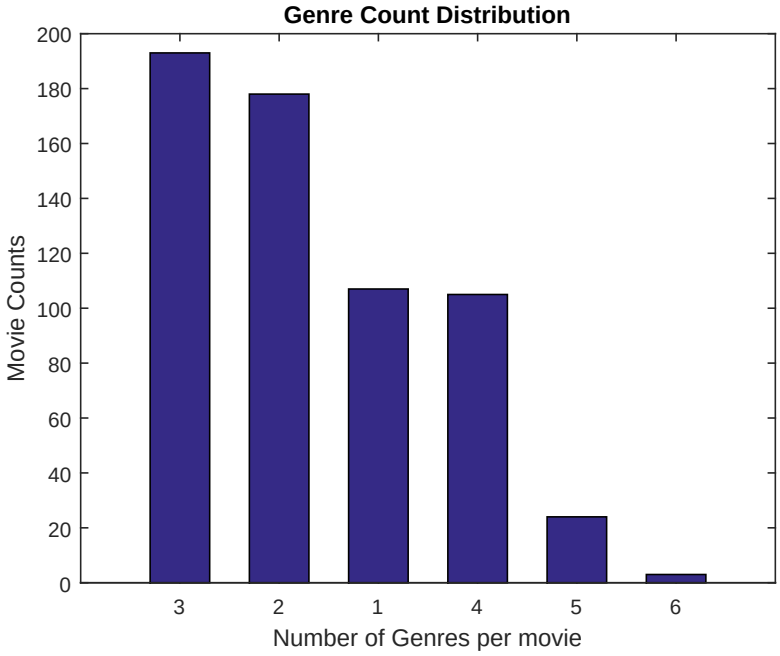

Figure 3.3: The distribution of number of genres per movie

3.2.2 Term Frequency and TF-IDF

As a next step, we tried the TF instead of Boolean feature value for the feature value on the most frequent words. This step is done because frequency would be a better indicator of the weight and importance of the word as a feature for it adds more information instead of just saying whether the word exist or not. We will refer to it as TF feature value. Also, TF-IDF, which is the combination of TF and IDF, is experimented.

TF is the number of times a word is repeated in a document and IDF indicates the uniqueness of the word in the document together they are used to calculate TF-IDF.

$$TF_w = count(w)$$

$$IDF_w = \log \frac{D_t}{D_w}$$

where D_t is the total number of documents.

D_w is the number of documents containing this word and sometimes one is added to avoid division by zero. and as for the value of TF-IDF

$$TF\text{-}IDF_w = TF_w * IDF_w$$

As for how IDF determines the uniqueness of a word: if a word is repeated in all documents then $IDF = \log(1) = 0$.

Going back to TF feature values, some classifiers need to take discrete values. So, it is better to use ranges for the TF feature values where each range is represented by one value. We put 4 ranges other than the zero as shown in Table 3.1.

Ranges	[1 , 58 [	[58 , 116 [	[116 , 174 [	[174 , ∞ [
Values	1	2	3	4

Table 3.1: The ranges and their corresponding values for TF when used in classifiers that take discrete value features

3.2.3 Extra Dialogue Features

The features at first were only the top(N) frequent words in the training set but later some features were added: The number of characters in the movie, Th number of digits (written in numbers not letters), the number of Named Entity Recognition (NER) which is the task of identifying the named entity in the text including names of people, organizations, locations, etc. and the average length of the lines that a character said at a time per movie.

3.2.4 N-Grams

After that, we tried the N-Grams (discussed in background chapter) as features and used the frequency as value instead of taking one word frequency into consideration. Using the N-Grams may be more effective for it considers more than one word at a time (sequence if words) as feature. For example:'A lot' can have different meaning from 'lot' and in case of N-Grams 'A lot' can be considered as one feature. We took N words frequency and N value we tried are 2, 3 and 4.

3.2.5 LDA

The last feature values we tried is the LDA (which was discussed in the Introduction chapter) topic distribution values. First step is training the LDA model by TF scores where words occurring in only one movie and words occurring in 95% of the movies are removed and the chosen number of topics is 10 and after getting the percentages of document of training set they were added them as features instead of the top(N) frequent and top(N) TF-IDF.

3.3 Classification

In order to compare between the different classifiers. same random seed and state are used to have the same training set and testing set every run and some other parameters. Without specifying random seed and random state, different training and test set will be generated every run as random seed is like starting value used to generate random values. So, specifying numbers for them grantee same split.

At the beginning the training and test was done on the traditional Naïve Bayes found in the NLTK. The start was with NB as it is fast, requires low storage and is a good starting baseline for classification and as for the features:

As a start we took the most frequent 2,000 words with Boolean feature followed by 3,000 and 4,000 most frequent. The operation is repeated after removing the stop words (which are the extremely commonly used words that search engine usually ignores for example stop words in English language: The, a, he, she, his, etc.) because they do not add genre related information to the model.

As a next step we tried the TF feature value instead of boolean feature value value with the ranges mentioned in the features section in Table 3.1 as NB is one of the classifiers that takes discrete. The following step is trying the TF-IDF with ranges.

The features at first was top(N) frequent words in the training set but later some features, which were mentioned in the feature section, were added: The number of characters in the movie, Th number of digits, the number of NER and the average length

of the lines that a character said at a time per movie and again in case of NB classifier ranges was determined for these features.

Then instead of top(N) frequent words as features, top(N) in IDF is used. Both are tried with different classifiers.

Moving on from the traditional NB, The next used classifier is Gaussian NB in which the features likelihood is considered as Gaussian.

$$P(d_i \mid G_k) = \frac{1}{\sqrt{2\pi\sigma_G^2}} \exp\left(-\frac{(d_i - \mu_G)^2}{2\sigma_G^2}\right)$$

Where from using maximum likelihood, σ_G and μ_G are estimated.

Also, since Gaussian can take continuous values, there is no need for the ranges. Therefore, the values are given as it is in the features. Moreover, the extra dialogue features (average line length, number of characters,number of digits and number of NER) also can be taken as it is but it is better to have these features normalized to have their values between zero and one. So, we divide these extra dialogue features by the number of words in the associated movie.

The next type of NB classifier used is the Multinomial NB. It requires that the features have integer values, however, using TF-IDF values is also acceptable. Both boolean features value and TF feature values with ranges are used.

Moving on from NB, we tried the Random Forest Classifier and tried it for different number of trees namely: 10, and 100. SVM Classifier (SVC) and Linear SVC classifiers also were used. Again different number of features (1,000, 2,000 and 3,000) and different features values (top frequent and top TF-IDF) were tried for the different classifiers.

Instead of top(N) frequent words and top(N) IDF, N-Grams is used. As for feature values: frequency and word existence are experimented. We tried the different classifiers mentioned before: Multinomial NB, Gaussian, SVC, Linear SVC and Random forest with the N-Grams.

In the later stages we introduced LDA. We got the LDA topic distribution of each document which is the percentage of each topic in that document individually. Applying this LDA model on both training and testing documents in order to obtain the features to classify them is done as well.

3.4 Feature Selection

As the number of features is huge ranging from 2,000 to 4,000, there must have been features that add no information, is redundant and just consumes time for classification. For this reason, we decided to discard some features using feature selection.

- Selection by percentage (SelectPercentile): where a certain percentage is kept from the features. The features that are kept are the one with highest scores and the rest of the features are removed. There are different scoring functions that can be used for SelectPercentile. The scoring functions used are chi2 (χ^2), f-classif and mutual-info-classif

- Selection by variance (VarianceThreshold): where low variance features are removed. By default, VarianceThreshold removes all zero variance features and the percentage (e.g. 0.8 = 80%) given removes the features whose values are repeated in this percentage (0.8) of the samples. Variance is calculated as shown in Equation 3.1.

$$Var\left[X\right] = p(1-p) \tag{3.1}$$

3.5 Evaluation Metrics

As each movie may have more than one genre the classification problem became multi-class classification and the approach used is One-Vs-All approach where the problem is divided to multiple binary classifiers, for each genre there is is classifier and for each classifier precision, recall and F-score are calculated. Precision is an indicator of how much of the predicted output is relevant, recall is indicator of how much of the relevant instances is predicted and F-score (also called F-measure) is the harmonic mean of the two.

$$precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$F - score = \frac{1.0}{\frac{0.5}{precision} + \frac{0.5}{recall}} \tag{3.4}$$

Where TP is true positives i.e. The genres correctly predicted genres as existing

TN is true negatives i.e. The genres correctly predicted as non-existing

FN is false negatives i.e. Genres incorrectly predicted as non-existing

FP is false positives i.e. Genre incorrectly predicted as existing

Also after movies genres are predicted, each movie has its original genres and predicted genres. Precision, recall and F-score is calculated for these movies by getting the true positive count which is obtained by the intersection of actual genres and predicted genres,

predicted count which is the count of all predicted genres of all movies and reference count which is the count of all actual genres of all movies.

$$recall = \frac{truePostiveCount}{refCount} \tag{3.5}$$

$$precision = \frac{truePostiveCount}{predictedCount} \tag{3.6}$$

As for the F-score it is obtained the same way as Equation 3.4

Finally, average precision, recall and F-score are calculated from the movies by calculating the precision using Equation 3.2 and recall using Equation 3.3 of each movie in the test set and divide them by the number of movies. Then F-score is calculated using Equation 3.4.

Table 3.2 shows data example to show an example of how the evaluation takes place where a-z are treated as genres.

Movie#	**Reference**	**Predicted**
1	$[a, b]$	$[a]$
2	$[b]$	$[a]$
3	$[a, c]$	$[a]$
4	$[a, d]$	$[e]$

Table 3.2: Data example to explain on it the way of evaluation

For the binary genre classifiers, consider genre (a). The movies having genre (a) are 1, 3 and 4 in reference so reference-set = {1,3,4} and movies 1, 2 and 3 in predicted so predicted-set = {1,2,3}.

$$TP = LengthOfIntersection = Length([1, 3]) = 2$$

Substituting in Equations 3.2, 3.3 and 3.4 $precision = \frac{2}{3}$, $recall = \frac{2}{3}$ and $F - score = \frac{2}{3}$.

As for the precision, recall and F-scores of the movies calculated using the same data example in Table 3.2

$$TP = SumOfIntersections = 1 + 0 + 1 + 0 = 2$$

$$reference - Count = 2 + 1 + 2 + 2 = 7$$

$$Predicted - Count = 1 + 1 + 1 + 1 = 4$$

So, From Equations 3.6 and 3.5: $precision = \frac{2}{4}$, $recall = \frac{2}{7}$ and $F - score = \frac{4}{11}$.

Movie#	**Recall**	**Precision**
1	0.5	1
2	0	0
3	0.5	1
4	0	0

Table 3.3: Precision and Recall for each movie in Data example found in Table 3.2

Finally, the average precision, recall and F-scores: The precision and recall are calculated for each movie as shown in Table 3.3.

$$\text{Avg-Recall} = \frac{0.5 + 0.5}{4} = 0.25$$

$$\text{Avg-Precision} = \frac{1 + 1}{4} = 0.5$$

$$\text{Avg-F-Score} = \frac{1}{\frac{0.5}{0.25} + \frac{0.5}{0.5}} = \frac{1}{3}$$

Chapter 4

Results and Discussion

The bold boundary found in some of the tables has the best F1 result in the corresponding table.

4.1 Results

4.1.1 Naïve Bayes

The results of the NB in the NLTK (Bernoulli NB) for top(N) frequent words without extra features is in Table 4.1 where AvgP is average-precision, AvgR is average-recall, AvgF1 is average F-score (F-measure), Prec is precision, Rec is recall, F1 is average F-measure, N is the number of features (Most frequent), SW indicates whether stop words are included (True) or not (False) and FtVal indicates whether it's Bool, TF or TF-IDF feature value and in case of TF featVal ranges are used. The NLTK NB is the closest in results to the Multinomail NB which gives the best results. The best result of the NLTK NB is $F1 = 60.28\%$ with top(4000) frequent words and boolean feature values in the presence of stop words.

4.1.2 Multinomial NB

Table 4.2 shows the results of MultinomialNB Classifier without the Extra features. In case of TF feature Value ranges are used. While Table 4.3 shows the binary classifiers result of multinomial classifier best result found which is surrounded by bold boundary in Table 4.3. MultinomialNB gives the best results and the best one is of $F1 = 63.25\%$ when stop words were removed with top(4000) frequent words as features and TF as feature value.

N	SW	FtVal	AvgP	AvgR	AvgF1	Prec	Rec	F1
2000	T	bool	53.78	68.89	60.40	50.81	69.18	58.59
2000	F	bool	53.27	68.18	59.81	50.81	68.87	58.48
3000	T	bool	55.41	67.93	61.03	52.16	68.24	59.13
3000	F	bool	55.31	68.02	61.01	52.03	68.55	59.16
4000	T	bool	57.33	68.29	62.33	53.98	68.24	**60.28**
4000	F	bool	57.07	68.90	62.43	53.69	68.55	60.22
2000	T	TF	52.52	68.85	59.59	50.11	69.18	58.12
2000	F	TF	53.43	67.71	59.73	50.58	68.24	58.10
3000	T	TF	55.41	67.91	61.03	51.29	68.87	58.79
3000	F	TF	56.19	67.48	61.32	52.40	68.55	59.40
4000	T	TF	55.68	69.02	61.64	52.62	69.50	59.89
4000	F	TF	57.21	67.48	61.92	53.48	67.61	59.72

Table 4.1: The results of NLTK NB Classifier without Extra features fot top(N) frequent words

N	SW	FtVal	AvgP	AvgR	AvgF1	Prec	Rec	F1
2000	True	TF	59.11	66.38	62.53	57.65	66.35	61.70
2000	False	TF	59.31	67.51	63.14	57.65	66.35	61.70
3000	True	TF	59.63	68.36	63.70	58.84	66.98	62.65
3000	False	TF	60.77	67.36	63.90	59.43	65.41	62.28
4000	True	TF	63.48	68.70	65.99	60.29	66.35	63.17
4000	False	TF	61.84	69.01	65.23	60.69	66.04	**63.25**
3000	True	bool	60.41	68.56	64.23	58.04	66.98	62.19
3000	False	bool	60.77	67.36	63.90	59.43	65.41	62.28
4000	True	bool	62.63	68.09	65.25	58.99	66.04	62.31
4000	False	bool	61.57	69.01	65.08	60.17	66.04	62.97
2000	True	bool	57.95	66.65	62.00	56.08	66.67	60.92
2000	False	bool	58.90	67.30	62.82	57.69	66.04	61.58

Table 4.2: The results of MultinomialNB without Extra Features and features are the top(N) frequent

4.1.3 Gaussian NB

Table 4.4 shows the results of GaussianNB Classifier where FT is the type of the features (the criteria upon which the features were chosen and represented as indices of feature vector) top(N) in frequency (freq) or top(N) IDF (idf) and E for the extra (dialogue) features where T if extra features are included and F means they are removed. Observing the results of GaussianNB, we can see high recall values. The best F-score using GaussianNB classifier is $F1 = 53.60\%$ when top(2000) frequent words with boolean feature

Genre	Prec	Rec	F1
action	67.6	71.4	69.4
adventure	45.2	66.7	53.8
comedy	66.7	62.1	64.3
crime	57.1	60.6	58.8
drama	68.2	78.9	73.2

Genre	Prec	Rec	F1
horror	40.0	40.0	40.0
mystery	20.0	18.8	19.4
romance	46.2	54.5	50.0
sci-fi	84.0	70.0	76.4
thriller	68.8	80.0	73.9

Table 4.3: The genres Binary classifiers Results of Best Multinomial Result

values are used in the absence of stop words.

N	SW	FT	FtVal	E	AvgP	AvgR	AvgF1	Prec	Rec	F1
2000	T	freq	TF	F	50.57	64.19	56.57	45.39	65.09	53.49
2000	F	freq	TF	F	51.36	64.07	57.01	45.56	64.47	53.39
4000	F	freq	TF	F	56.34	54.17	55.24	49.22	49.69	49.45
2000	T	freq	bool	F	43.49	75.22	55.11	40.91	76.42	53.29
2000	F	freq	bool	F	42.98	74.17	54.43	41.75	74.84	**53.60**
4000	F	freq	bool	F	44.92	60.23	51.46	41.32	56.92	47.88
4000	F	freq	TF-IDF	F	46.59	62.80	53.50	42.99	58.81	49.67
4000	F	idf	TF-IDF	F	64.30	25.91	36.93	52.78	23.90	32.90
2000	F	freq	bool	T	42.93	74.72	54.53	41.75	74.84	53.60

Table 4.4: The results of GaussianNB

4.1.4 Random Forest

Table 4.5 shows the results of Random Forest Classifier For top(N) frequent features, where NT is the number of trees. We can observe the high high precision values in the table. The best F-score in the table ($F1 = 44.36\%$) occurs with top(2000) frequent words and TF feature values in the absence of stop words. The extra (dialogue) features gives same results for conditions mentioned of best F-score in Random Forest, but gives different average precision, recall and F-score.

4.1.5 Support Vector Machine

Table 4.6 shows the results of SVC classifier without the extra dialogue features. The best results in SVC is $F1 = 39.18$ when top(4000) frequent words with TF-IDF feature values in the absence of stop words. On observing the table, SVC gives high precision values. On the other hand, Table 4.7 shows the result of Linear SVC classifier. Best Linear SVC result is $F1 = 55.77\%$ when top(4000) frequent words as features, TF-IDF

E	NT	N	SW	FtVal	AvgP	AvgR	AvgF1	Prec	Rec	F1
F	10	2000	T	bool	59.01	36.57	45.16	51.04	30.82	38.43
F	10	2000	F	bool	59.76	37.51	46.09	51.94	33.65	40.84
F	10	2000	T	TF	61.99	35.00	44.74	55.19	31.76	40.32
F	10	2000	F	TF	67.07	37.25	47.90	58.16	35.85	**44.36**
F	10	3000	F	TF	62.03	38.85	47.78	54.27	33.96	41.78
F	10	3000	T	TF	63.41	34.55	44.73	55.49	31.76	40.40
F	10	4000	F	TF	57.79	36.61	44.83	53.37	32.39	40.31
F	100	2000	F	TF	74.39	29.77	42.52	66.93	26.73	38.20
F	100	3000	F	TF	76.29	31.68	44.77	69.84	27.67	39.64
F	100	4000	F	TF	71.68	28.64	40.93	65.85	25.47	36.73
T	10	2000	F	TF	59.46	36.96	45.59	56.57	35.22	43.41

Table 4.5: The results of Random Forest Classifier when features are top(N) frequent

as feature value in the absence of stop words and without extra features. The F-scores values of LinearSVC table is considered to be high.

N	SW	FT	FtVal	AvgP	AvgR	AvgF1	Prec	Rec	F1
2000	T	freq	TF	48.24	23.18	31.32	46.72	17.92	25.91
2000	F	freq	TF	69.38	27.91	39.81	65.22	23.58	34.64
2000	T	freq	bool	70.73	24.34	36.21	61.39	19.50	29.59
2000	F	freq	bool	70.33	23.86	35.63	60.00	18.87	28.71
4000	F	freq	TF	76.29	29.91	42.97	71.30	25.79	37.88
4000	F	freq	TF-IDF	76.69	30.93	44.09	71.07	27.04	**39.18**
4000	F	idf	TF-IDF	46.34	22.94	30.69	46.34	17.92	25.85

Table 4.6: The results of SVC Classifier without Extra dialogue features

4.1.6 N-Grams

The results of using N-Grams alone as feature value without extra Dialogue features is in Table 4.8. Ng indicates whether it is Bi-gram (2), Tri-gram (3) or 4-Gram (4) and in classifiers RF100 mean random forest classifier with 100 trees and Multinomial is MultinomialNB. Observing the table, the Bi-gram results is better than Tri-gram and 4-Gram. Also, boolean feature values gives better result than frequency (TF). The best result $F1 = 57.90\%$ takes place with MultinomialNB with top(4000) frequent Bi-grams.

N	SW	FT	FtVal	E	AvgP	AvgR	AvgF1	Prec	Rec	F1
2000	T	freq	TF	F	54.77	59.63	57.10	50.58	54.72	52.57
2000	F	freq	TF	F	58.45	59.76	59.10	53.17	55.35	54.24
3000	F	freq	TF	F	58.37	59.66	59.01	54.41	56.29	55.33
4000	F	freq	TF	F	57.66	59.19	58.41	54.13	55.66	54.88
4000	F	freq	bool	F	64.02	48.89	55.44	61.60	45.91	52.61
4000	F	freq	TF-IDF	F	66.91	52.05	58.55	64.08	49.37	**55.77**
4000	F	idf	TF-IDF	F	50.12	30.65	38.04	45.36	27.67	34.38
4000	F	freq	TF-IDF	T	60.73	55.18	57.82	56.33	53.14	54.69

Table 4.7: The results of Linear SCV Classifier

Classifier	N	Ng	FtVal	AvgP	AvgR	AvgF1	Prec	Rec	F1
RF100	3000	2	freq	73.92	25.80	38.25	65.77	22.96	34.03
SVC	3000	2	freq	77.10	25.34	38.14	68.32	21.70	32.94
LinearSVC	3000	2	freq	54.57	50.20	52.29	49.50	46.54	47.97
GaussianNB	3000	2	freq	49.23	63.32	55.39	45.43	65.72	53.73
LinearSVC	4000	2	freq	57.72	52.44	54.95	52.92	48.43	50.57
GaussianNB	4000	2	freq	52.49	63.48	57.46	49.16	64.47	55.78
LinearSVC	4000	3	freq	49.39	43.98	46.53	47.06	40.25	43.39
LinearSVC	4000	4	freq	42.44	42.87	42.65	36.31	39.62	37.89
GaussianNB	4000	3	freq	60.30	50.95	55.23	55.12	49.06	51.91
GaussianNB	4000	4	freq	46.08	55.73	50.45	40.98	55.03	46.98
LinearSVC	4000	2	bool	65.65	47.02	54.79	59.91	42.77	49.91
GaussianNB	4000	2	bool	53.67	64.35	58.53	50.86	64.78	56.98
Multinomial	4000	2	bool	58.93	59.08	59.01	57.63	58.18	**57.90**

Table 4.8: The Results of N-Grams when used as Feature Value without Extra Features

4.1.7 LDA

The results of using LDA alone as feature value is in Table 4.9. LDA Alone as feature value does not give high scores. The best result $F1 = 36.42$ is with the GaussianNB classifier.

4.1.8 Feature Selection

Figure 4.1 shows the feature selection using select percentile using three different scoring function on MultinomialNB when stop words were removed with top(4000) frequent words as features and TF as feature value. The maximum peak of chi2 is at F-score = 63.71% when 70% of the features are kept, maximum peak of f-classif is at F-score = 63.66%

Classifier	AvgP	AvgR	AvgF1	Prec	Rec	F1
GaussianNB	25.54	66.68	36.93	24.88	67.92	**36.42**
LinearSVC	55.28	21.31	30.77	48.60	16.35	24.47
SVC	46.34	22.94	30.69	46.34	17.92	25.85
MultinomialNB	46.34	22.94	30.69	46.34	17.92	25.85

Table 4.9: The Results of LDA when used as Feature Value

when 30% of the features are kept and maximum peak of mutual-info-classif is at F-score = 64.67% when 70% of the features are kept.

Feature selection by removing low variance features (VarianceThreshold), where same conditions of MultinomialNB mentioned in feature selection by select percentile, is in Figure 4.2. This selection gives no improvement compared to before doing it with the current conditions.

4.2 Discussion

Observing the tables of the results, We can notice that

- The classifier that gives the best results is MultinomialNB Classifier with $F1 = 63.25\%$ reaching $F1 = 64.67\%$ with feature selection although it does not have the highest recall and precision values followed by NLTK NB and LinearSVC Classifier.
- For feature selection by select percentile gives better results than by variance-threshold. Observing figure 4.1, the scoring function that achieved the highest F-score is mutual-info-classif $F1 = 64.67\%$ and the scoring function f-classif performs badly but still got a higher value than without selecting features when keeping 30% of the features. On the other hand, removing low variance features (using variance-threshold for feature selection) gives no improvement.
- For Random Forest Classifier and SVC: The precision is better than other classifiers but that is not the case with the recall. So, in case we generally care more about the precision (we care more that the results have less wrong genres or predicting correctly a subset of the genres but not all of it), getting high precision values can be done by using the Random Forest Classifier or SVC.
- The highest recall values can be found with the GaussianNB classifier.
- LDA alone as features has low F-measure values even with different number of topics selected because of the very small number of topics but N-Grams has good performance in our case.

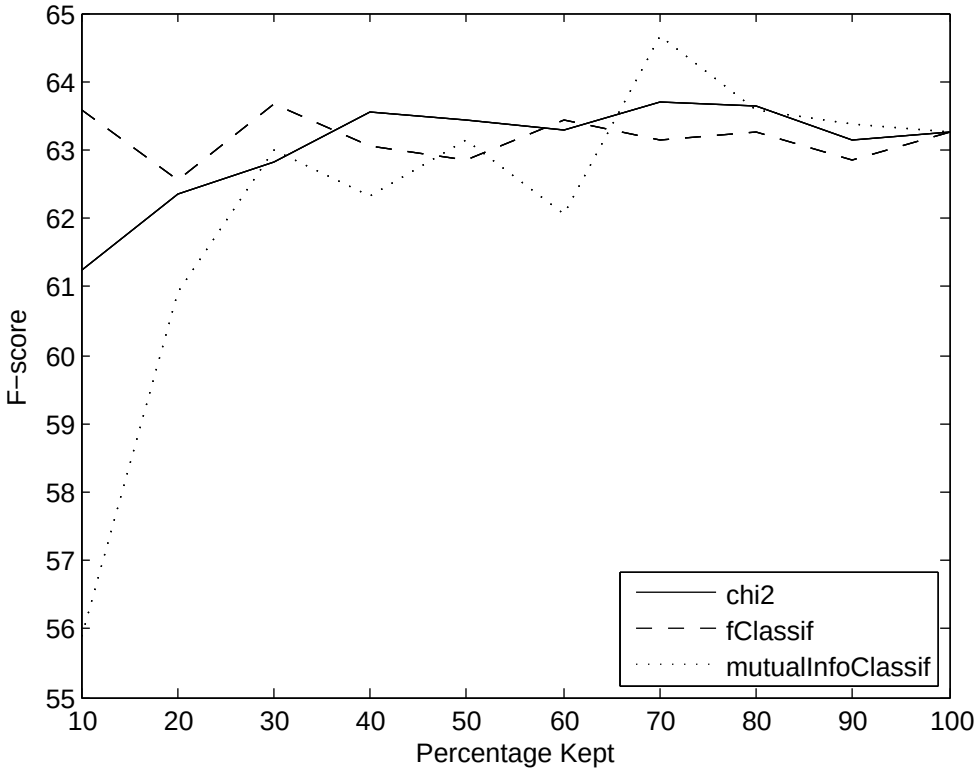

Figure 4.1: Feature selection graph using three scoring functions for select percentile with different percentage of the features kept. Feature selection is done on MultinomialNB when stop words were removed with top(4000) frequent words as features and TF as feature value

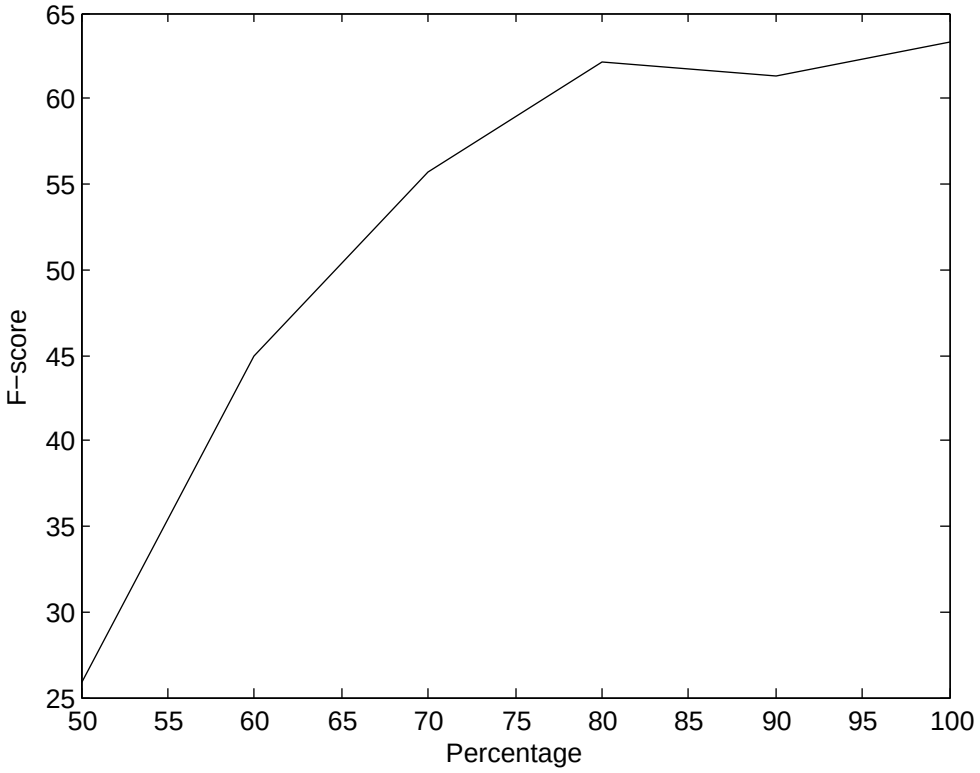

Figure 4.2: Feature selection graph using Variance-Threshold method with different percentage having same conditions of MultinomialNB mentioned in figure 4.1. The percentage describe the removal of words in case feature values (either True or False) is repeated by this percentage

- For the number of features going from 2,000 to 3,000 to 4,000, there were fluctuations in the F-score and we could not conclude that increasing the feature number improve the results.
- As expected TF feature values generally gave better results than Boolean feature values as more information is given as Boolean feature values is the mere existence of the word in the movie.

Compared to the results of Automated Movie Genre Classification with LDA-based Topic Modeling [7], our best results of F-measure are better, however, this is not a fair comparison for we have different dataset and different distribution of genres, but we came to agreement with them in that Random Forest classifier gives high precision values.

Chapter 5

Conclusion

The target of this thesis is to classify movie scripts by genre and choosing the best classification technique. Also, this thesis uses movie dialogues as input as movie scripts are mostly dialogues. The data set is from Cornell Movie Dialogs Corpus. First of all, the dataset is divided into 80% and 20% for training and testing sets respectively. Then, different classifiers are used for the classification namely: Naïve Bayes Classifier (Bernoulli Naïve Bayes, Gaussian NB and Multinomial NB), Support Vector Machine and Random Forest Classifiers. As for the feature types, either top(N) frequent words, top(N) IDF and top(N) N-Grams are chosen. Word existence, TF, TF-IDF and LDA topic distribution values are experimented for feature values.

Experimenting on the different classifiers, MultinomialNB gave the best result of $F1 = 63.25\%$ when stop words were removed with top(4000) frequent words as features and TF as feature value. Also, both SVC and Random Forest classifiers gave higher precision values compared to other classifiers but their recall values were low. On the other hand, Gaussian NB gave highest recall values but with a low precision.

Chapter 6

Future Work

As for recommendations for future work, trying to stratify the data may lead to better results and by stratifying the data it means to make each genre have the same percentage as training and testing percentages. For example: if the dataset is divided into 80% train and 20% test then 80% of movies having genre g1 should be in training set and 20% should be in test set. This task is not hard for single-label classification, however, it is a little challenging for classifying movie scripts by genre is a multi-label classification problem.

This thesis focuses on English movie scripts only. So, we suggest exploring more languages for classifying movie scripts by genre for there are movies in many other different languages. A similar model to the one proposed in this thesis can be used with different languages with slight changes. Example of changes: in case of a different language, stop words are different.

Increasing the dataset size can also improve the results. This should be considered as the dataset size used is considered to be small (only 610 movies). This can be done by:

- Crawling websites like dailyscripts as it has large number of movie scripts with the associated IMDB link for the genres.
- Including novels in the classification may improve the results as extra data is added and more data can give more information.

Appendix

Appendix A

Lists

NLP	Natural Language Processing
NLU	Natural Language Understanding
NLG	Natural Language Generation
NB	Naïve Bayes
SVM	Support Vector Machine
SVC	SVM Classifier
LDA	Latent Dirichlet Allocation
TF	Term Frequency
IDF	Inverse Document Frequency
TF-IDF	Term Frequency Inverse Document Frequency
NER	Named Entity Recognition
NLTK	Natural Language Toolkit

List of Figures

List of Tables

Bibliography

[1] Scikits - scikit-learn. https://scikits.appspot.com/scikit-learn. (Accessed on 05/14/2017).

[2] Sarah Guido Andreas C. Mller. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. OReilly Media, 1 edition, 2016.

[3] Jason Bell. *Machine Learning: Hands-On for Developers and Technical Professionals*. Wiley, 1 edition, 2014.

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

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

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

[7] Brandon Chao and Ankit Sirmorya. Automated movie genre classification with lda-based topic modeling.

[8] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

[9] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[10] Ka-Wing Ho. Movies genres classification by synopsis.

[11] Ela Kumar. *Natural language processing*. IK International Pvt Ltd, 2011.

[12] James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 710, 2000.

[13] Gabriel S Simões, Jônatas Wehrmann, Rodrigo C Barros, and Duncan D Ruiz. Movie genre classification with convolutional neural networks. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 259–266. IEEE, 2016.

[14] Howard Zhou, Tucker Hermans, Asmita V Karandikar, and James M Rehg. Movie genre classification via scene categorization. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 747–750. ACM, 2010.