Genre Classification

Sravani Dhara
Digital Engineering
Otto von Guericke University
Magdeburg, Germany
sravani.dhara@st.ovgu.de

Peer Ahamad Shaik
Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
peer.shaik@st.ovgu.de

Ashish Soni
Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
ashish.soni@st.ovgu.de

Dheeraj Tippani
Digital Engineering
Otto von Guericke University
Magdeburg, Germany
sai.tippani@st.ovgu.de

Renukadevi Krishnaraj
Electrical Engineering and Information Technology
Otto von Guericke University
Magdeburg, Germany
renukadevi.krishnaraj@st.ovgu.de

I. Motivation and Problem Statement

The Project explains in detail, the steps undertaken while working on the Gutenberg dataset of fiction books, to solve the problem of assigning a (label)genre from a set of multiple genres; to the appropriate book by using the text within the book; written by the author. The Problem statement comprises of a Classification task on text data with various challenges. In this task, every data-point is a book of fiction with a genre(label). We will be using Machine Learning and Deep Learning approaches to tackle the challenges in this classification task. Since the books contain large texts that could have different themes throughout the entire book and the Literature usually is not structured, like for e.g. few lines of data from a tweet[1, 3] expressing a opinion or a fact can be considered structured but text within the books does not follow one particular structure throughout the whole book; the task poses difficult challenges to achieve the goal of genre classification. The following are some of the challenges/problems for the task, that we need to overcome:

- Choosing appropriate data preprocessing techniques to obtain data, that is useful for our task.
- Extraction of features that are relevant with respect to fiction. for e.g. : character, plot, theme, sentiment etc.[2]
- Making sensible assumptions to collect those features.
- Selecting the appropriate model poses a challenge as given the complicated nature of the documents(books) classic approaches like bag-of-words or n-grams are not equipped to handle the information spread within paragraphs and chapters.
- Comparing the results of different classification techniques on test data through different evaluation metrics.

II. Data Set

The Gutenberg 19th century English Fiction Dataset is a subset of books that belong to the 19th Century English Fiction.

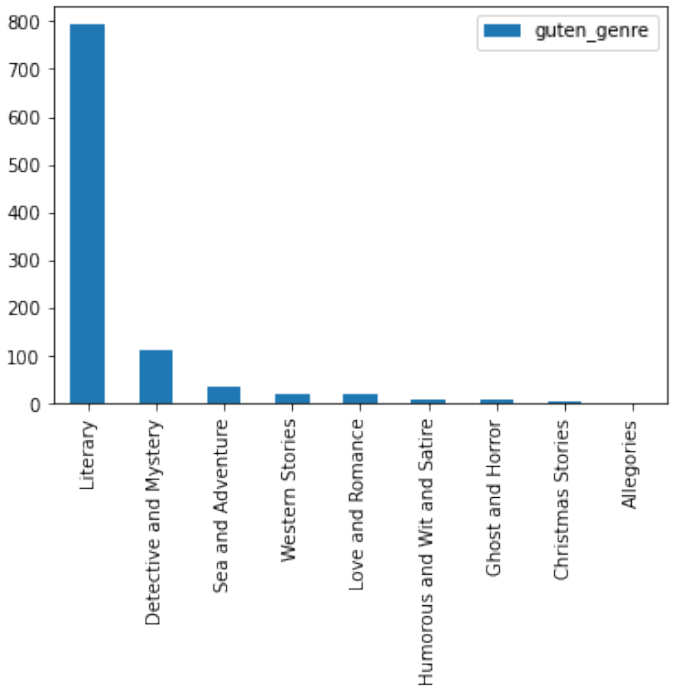

Fig. 1. Genre category and count

It was generated from Project Gutenberg[1], which consists of over 60,000 eBooks, published in the 19th century. For our Dataset, we were given 1079 html files represented as 1079 books that contain the text and a csv file that contained names of 996 books with the following columns:

- Book Name
- Book Id (Key)
- Guten Genre
- Author Name

[2]Fictional Elements

[1]Project Gutenberg

We have ignored those (1079-996) 83 extra books, for consistency. We have 9 unique genres in the Guten Genre column, namely: 'Detective and Mystery', 'Literary', 'Western Stories', 'Ghost and Horror', 'Christmas Stories' 'Love and Romance', 'Sea and Adventure', 'Allegories', 'Humorous and Wit and Satire' to work with and solve the Genre Classification problem, posed to us. Therefore, Guten Genre will be our target attribute. We have noticed class imbalance in our dataset, depicted in Fig.1. Although, there are no missing values, for any of the above mentioned columns. For the books, the 'Literary' genre, has the majority count of 794 and 'Allegories' has the least count of 2. These are the initial observations of the dataset.

III. Concept

A. Naive Bayes:

One of the most commonly used algorithm for text classification. It is based on conditional probabilities of the concurrences of a feature and respective label. We have used the model with features extracted in multiple techniques. Bag of words is one of such techniques used. We used the performances of the model features extracted from 'bag of words', as a baseline to compare other feature vectors with. We have trained the model individually using 'bag of words', 'tf-idf' and hand crafted features.

B. Multinomial Naive Bayes:

In this variation of Bayes algorithms, the term (words) frequency is also taken into account in training. The only difference between the Naive Bayes and Multinomial Naive Bayes is the difference in distribution of feature assumes. Naive Bayes assumes binomial distribution (Hence, naive), while Multinomial assumes Multinomial distribution. Features generated using 'bag of words', 'tf-idf' and hand crafted features will be used to train the model.

C. Multi Layered Perceptrons:

Another traditional text classification machine learning model, MLPs, they are simplistic in nature compared to its deep learning[7] counterparts. The input will be 'bag of words' and 'tf-idf'. This is one of our baseline model.

D. Linear SVM:

Under the assumption that most of the text classification problems are linearly separable we implement linear SVM. The features extraction techniques would be bag of words and handcrafted features. There will be more about the hand crafted features in the later sections. The advantages of using linear SVM is SVMs are fast in general and require lesser computation power compared to deep learning models, since fewer parameters to learn, linear kernels are especially faster compared to other kernels.

E. CNN:

We implemented a CNN [2] deep learning model, where it can be used to automatically generate complex features which can better classify the books. The convolution layers are followed by max pooling layers to prevent learnable parameters count from blowing up retaining the information. This is followed by a fully connected MLP with the 9 nodes representing the genres.

F. Feature extraction:

1) Bag of Words::

We used the bag of words for of representation of the documents. The purpose of this feature extraction technique is to use it as a baseline[4] for comparison with the other forms of features.

2) Term frequency-Inverse document frequency::

Using this technique to measure the importance of a term in a document. This done by multiplying the how many time the term appears with inverse of its frequency across all the documents. We can choose the top k number of terms which define the document which we can add to the features.

3) Data Preprocessing::

Basic preprocessing is done to remove the noise from the text dataset and make the data predictable and analyzable.

- Lowercase
- Tokenization
- Stopword removal
- Stemming
- numbers and special characters removal

4) Hand crafted features::

The features are hand crafted [5, 6] to catch the essence of the semantic information.During the extraction of features using Tf-idf and BOW, the semantic information is completely lost. Only statistical information remains. For a task like genre identification it is important to maintain the semantic information. The features below represent better semantic information.

- Lexical richness
- Word count
- Word density
- Punctuation count

Lexical richness: This feature represents the score computed by ratio of unique lexical items and the total number of words in the document. For every book this unique score is a feature. fig.1 shows the distribution of lexical richness score across the dataset.

Word count: The Word count represents the total number of words in every document.

Inverse word density: Inverse Word density represents Word count divided by Character count.

Punctuation count: The Punctuation count represents the count of the punctuation in every document.

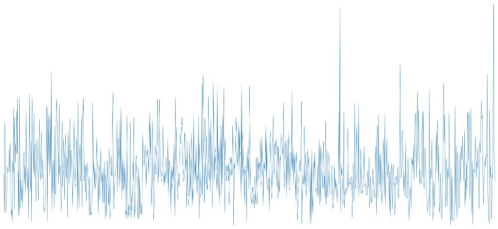

Fig. 2. Lexical richness across the entire dataset

IV. Implementation

We have implemented the said models using the tools, mentioned in table I.

Tools and Libraries			
Sr.no.	Library Name or Tool	Version	Purpose
1.	Codecs	1.3.2	To parse content from an HTML file
2.	Natural Language Toolkit	3.2.5	To tokenize and stem words
3.	Pandas	1.0.5	Data analysis and manipulation
4.	Numpy	1.18.5	To process arrays
5.	Scikit-learn	0.22.2.post1	a) To vectorize text content. (TF-IDF/TF) b) To import classifiers. c) To split the data in K folds. d) To find the evaluation metrics
6.	Google Colab	—-	Exploit faster processing speed
7.	keras	2.3.1	for deep learning model layers
8.	Pickle	4.0	Used to serialize and de-serialize Python object structures, also called marshalling or flattening.
9.	Tensorflow	2.2.0	for deep learning models

TABLE I
Tools and Libraries

The Source code consists of data preprocessing, and use of different models like Gaussian Naive Bayes, Multinomial Naive Bayes, Bernoulli naive Bayes, Linear Support Vector Machine from Scikit-learn library and Mutli-Layer Perceptron and Convolutional Neural Network from keras library. You can find below a couple of snippets from the source code. We concatenated the hand features with the Tf-idf vectors and feed the model for only machine learning techniques. We are using MLP and Convulational Neural Networks in Deep Learning portion for the classification task.

V. Evaluation

A. Evaluation Metrics:

We used Scikit-learn for training data and test data split.
75 percentage of corpus data used as training set, and 25 percentage as test set. We used 1000 iterations for traditional models and 20 epochs for deep learning metrics of batch size 32. We used 3 metrics for model evaluation. Accuracy, Precision, and F1-score. We are getting around 78 percentage accuracy on different models. All models are performing better. The difference between training accuracy and testing accuracy is very low, thus we can understand model is generalizing well. Overall Problem goal is accomplished. Although we can get better by tweaking the model, still we are getting good results.

Evaluation Metrics			
Model	Accuracy	F1-Score	Precision
Gaussian Naive Bayes	0.787	0.7692	0.7545
Multinomial Naive Bayes	0.811	0.726	0.658
Bernoulli Naive Bayes	0.6746	0.7192	0.808
Linear Support Vector machine	0.8514	0.8081	0.823
Multi Layer Perceptron	0.8795	—-	—-
Convolutional Neural Network	0.7871	—-	—-

TABLE II
EVALUATION METRICS

VI. CONCLUSION

Challenges: Reading and storing the data from HTML files all at once is complicated task, since it is eating all the memory. When we tried to use Entity Recognition for feature extraction, the run time is high for the whole corpus. Similarly, also to retrieve Parts-of-speech tagging, and porter stemming the run time is high. Our results can be considered as base model results and can be used to perform improvements and hyper parameter tuning.

Results: We could extract more topic specific hand crafted features, by fine tuning the model. We could Also augment the data, instead of considering a single documents as multiple documents, to remove class imbalance.

REFERENCES

[1] Yogesh Garg and Niladri Chatterjee. “Sentiment analysis of twitter feeds”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8883 (2014), pp. 33–52. ISSN: 16113349. DOI: 10.1007/978-3-319-13820-6.

[2] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), pp. 2261–2269. DOI: 10.1109/CVPR.2017.243. arXiv: 1608.06993.

[3] Alexander Pak and Patrick Paroubek. “Twitter as a corpus for sentiment analysis and opinion mining”. In: *Proceedings of the 7th International Conference on Language Resources and Evaluation, LREC 2010* (2010), pp. 1320–1326. DOI: 10.17148/ijarcce.2016.51274.

[4] Mathijs Pieters and Marco Wiering. “Comparison of machine learning techniques for multi-label genre classification”. In: *Communications in Computer and Information Science* 823 (2018), pp. 131–144. ISSN: 18650929. DOI: 10.1007/978-3-319-76892-2_10.

[5] Sayantan Polley and Suhita Ghosh. “Comparing the qualitative impact of different features and similarities on fictional text using SIMFIC”. In: ().

[6] Sayantan Polley et al. “SIMFIC : An Explainable Book Search Companion”. In: ().

[7] Joseph Worsham and Jugal Kalita. “Genre Identification and the Compositional Effect of Genre in Literature”. In: (2018), pp. 1963–1973.