Genre Identification on (a sub-set of) Gutenberg Corpus

Divya Sasidharan
M.Sc. Digital Engineering
Otto von Guericke University
Magdeburg, Germany
divya.sasidharan@st.ovgu.de

Mondher Hentati
M.Sc. Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
mondher.hentati@st.ovgu.de

Poornima Venkatesha
M.Sc. Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
poornima.venkatesha@st.ovgu.de

Saied Akbar Ahmady
M.Sc. Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
saied.ahmady@st.ovgu.de

Sinchana Eshwarappa Prameela
M.Sc. Data and Knowledge Engineering
Otto von Guericke University
Magdeburg, Germany
sinchana.eshwarappa@st.ovgu.de

***Abstract*—This paper presents a model that classifies a fiction book by its genre, using selected 19th century books from 'Project Gutenberg' [1]. The task is a supervised multi-class text classification problem for the detection of genre of a book. This subset consists of 1079 books belonging to 9 different genres [2]. Hand-crafted feature representation was performed for this task. Each book that belongs to a particular genre belongs to that genre as it has some distinguishing characteristics. This idea was exploited to extract features to make the model understand the style, setting, sentiment, and plot of each book. In simple words, to understand, and answer the question, *what features make each book belong to that genre?*. We followed the usual machine learning pipeline of data pre-processing, modeling, and evaluation to solve this task. This project is implemented using Python 3, latest NLTK (v3.5) library, scikit-learn v0.23, and the imbalanced-learn API.**

***Index Terms*—English Fiction, Text classification, Feature extraction, Gutenberg**

I. Motivation and Problem Statement

A book should never be judged by its cover. The first thing that will grab the attention of many book readers who are looking for a book to buy or to borrow is the title or the author. However even if the book does not look interesting, what is inside may surprise you. In fact, classification of books by their genre may lead the readers to explore new books and authors.

We seek in this paper to find some patterns between the genres of books to classify them using machine learning models. We will explore and extract the features that allow us to create the corresponding model to detect and search for a new book given its genre.

We answer the following research questions in this project: How to identify genres of a book? What makes each genre different? What distinguishes them? What features do we need to extract to classify a book to its correct genre? How can we solve the class imbalance problem present in the data set? Given a new unknown book as input to the model, how well can it predict its genre?

This paper proceeds as follows: we first explain how the data set is structured in section two. Then, in section three, we explain the overall concept of our project and the process behind the choice of features and their extraction. In section four, we discuss the implementation of the concepts and what assumptions are being made and how could they affect the results. In the fifth section, we evaluate the results of our study. Finally, the sixth section concludes with our contributions and results for takeaway.

II. Data set

The data set is obtained from the OvGU website [2]. This data set contains selected 19th Century English fiction books, ignoring short stories, and books with multiple authors. In total, the data set is composed of 1079 HTML files out of which only 996 files(present in the CSV file) are labeled. The other files in the data set are ignored. All the 996 documents are extracted from 1079 HTML files using the book_id as the key given in the *master996.csv* file.

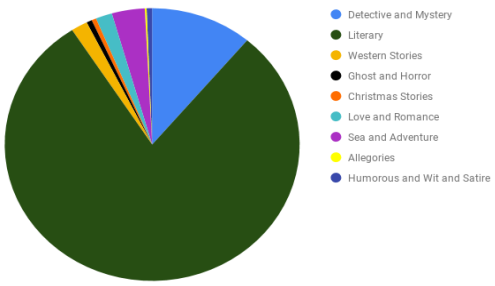

Figure 1: Genre Distribution

From Figure 1 and Figure 2, we can clearly see that there is a huge class imbalance issue with our data set. The class *Literary* dominates over the other classes with 794 instances, and the class *Allegories* is on the other end of the spectrum with only 2 instances. Class imbalance is an issue because it gives a low predictive accuracy over the imbalanced classes. Using the imbalanced data set will likely predict every instance as the majority class and give a high accuracy result, which is not useful for our task at all. This problem is addressed during the implementation stage by using the correct evaluation metrics and over-sampling strategies.

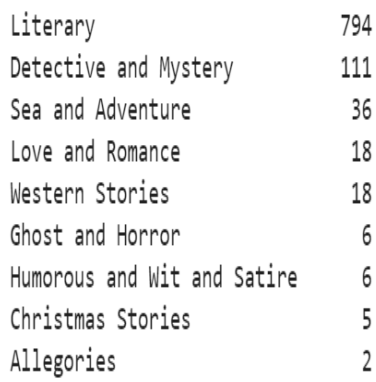

Figure 2: Count of books in each genre

III. Concept

For our data set we want to use methods that give us more information on syntactic and semantic similarity for each book in the corpus. Lexical similarity is not of our interest, as there can be matching words in two more books. Semantic similarity help in classification of each books based on the 'meaning' rather than matching words. Syntactic similarity plays a very important role to distinguish books based on the writing style of the author, complexity of sentences, sentiment of the book and so on.
Based on the above mentioned similarity measures, eight features where chosen for feature representation of each book in the corpus. The features extracted where female proper noun count, male proper noun count, count of paragraphs, average length of sentences, part of speech tags, semi-colon count, named entities and sentiment. The table I is a description of the hand-crafted features.

Writing Style: Every book has a unique style of writing, which will define the feature for that particular book. The style of writing can be defined by part of speech patterns, common usage of quotes and paragraphs, average sentence length, as well as comparing personal pronoun used will help distinguishing between different styles. In our experiment we have tried to include majority of these features to help the model learn to classify book for each genre.

Female or Male oriented: Books that belong to genres such as *Detective & Mystery* or *Sea & Adventure* are more male-oriented. *Love & Romance* books have equal number of male and female pronouns. We have leveraged this feature to classify books based on genre.

Sentence complexity: For writing style analysis, POS tags provide information regarding the usage of conjunctions, interjections, punctuation, preposition etc. These are useful in distinguishing each writing styles on the complexity level.

Setting: Another useful feature to differentiate between the genres is to understand the general background and plot of the books. Genres such as *Western stories* have a distinct rural setting and plot. These stories have a lot of conversations and usually, a high number of characters.

Sentiment: Genres such as *Love & Romance* and *Ghost & Horror* have widely different sentiments. We wanted to exploit this idea by taking the overall sentiment of a book.

Plot Complexity: Each genre has different levels of complexity. For example, *Detective & Mystery* stories have a complex plot with constant twists and turns with a lot of supporting characters. *Sea Adventure* stories also have a complex plot with span over many locations. *Literary* stories, in contrast, is more character-driven than plot and revolves around one central character.

TABLE I
Features

Feature type	Feature
1. Writing style	Paragraph count, Female pronoun count, Male pronoun count, Semi colon count, Average sentence length, All POS tags
2. Sentence Complexity	Average sentence length, POS tags (Comma, Period, Punctuation, Conjunction)
3. Female oriented	Female pronoun count
4. Male oriented	Male pronoun count
5. Setting	POS tags (Quotes), Persons count, Locations count
6. Sentiment	Compound score, Negative, Neutral, Positive
7. Plot complexity	Persons count, Locations count

IV. Implementation

The implementation was carried out using Python 3, the latest NLTK (v3.5) library, scikit-learn v0.23, and the imbalanced-learn API.

A. Data Cleaning

The given CSV file is first read into a dictionary and the *.epub* extension is removed from all the book-id values in order to match it with the HTML files. The books which are in the form of HTML files are then extracted. The data set had 1079 books, out of which only 996 are mentioned in the CSV file, and hence only those books will be read. There were also two empty files of *Literary* genre in the dataset, which were present in the CSV file, but are ignored. We observed that the raw data contained ¡p¿ tags, which were then removed for the entire corpus. We then extract word and punctuation tokens for the corpus using the NLTK

word tokenizer. Lemmas for each token is also extracted using NLTK wordnet lemmatizer. We do not do stopword removal, since we need to extract features such as female pronoun count, punctuation symbols count, etc and hence, all the features are important. Features were not represented in the usual BOW or TF-IDF form, since we do a hand-crafted feature representation and extraction.

B. Feature Extraction

The features where extracted for each book in the corpus. Table I shows the list of all hand-crafted features used. The entire feature extraction part took 8 hours on a Mac Book Pro system with i9 processor and 16 GB RAM.

Female pronoun count: We get the count of *'she'* mentioned in the corpus for each book. This feature can help us in understanding whether the book is female or male oriented. Books that belong to genres such as *Love & Romance*, and some selected *Literary* works are more female oriented. This also tells a lot about the overall writing style of a book.

Male pronoun count: The number of times the pronoun *'he'* is mentioned is also extracted. Genres such as *Sea & Adventure*, *Detective & Mystery*, and *Western stories* are primarily male-oriented and usually have one central male character. Certain works from *Sea & Adventure*, for example, *Omoo Adventures in the South Seas* by Herman Melville revolves around a primary male character.

Paragraph count: The paragraph count also helps in understanding the overall style of the book. Books from *Literary* and *Western stories* have a different style and flow of ideas and plot.

Average sentence length: The average sentence length also sheds light on the style and complexity of a book. More serious works such as those from the *Literary* genre have longer sentences. We get the average sentence length using the following formula:

$$AverageSentenceLength = \frac{TotalLengthOfAllSentences}{NumberOfSentences}$$

Semi-colon count: The count of ':' is also extracted to understand the style of the book.

POS tags: POS tags are used to learn the context of the word. It is used as part of setting, writing style and sentence complexity feature type for genre classification. We have considered the count of each tag present in each book to help the model learn the context for each genre using *nltk.pos_tag*. For Genre such as Sea & Adventure there might more statistical information like sea depth measure or the distance measures or dates. The tags corresponding to numeral values will be 'CD' and this count will be more in *Sea & Adventure* genre compared to *Love & Romance*.

Named-entity recognition: *nltk.ne_chunk* and *nltk.pos_tag* library is used to identify the named entities in the text like person and location. The count of person or location in a book can give the complexity of plot. We use pos tag 'B-PERSON' and 'B-GPE' for person and location respectively.

Sentiment analysis: *nltk.sentiment.vader* is used to perform sentiment analysis over text .The compound, positive, negative and neutral score is obtained for each book in corpus. "The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1 (most extreme negative) and +1 (most extreme positive)" [3]. This information is further used for genre classification.

C. Model Building

1) Before Over-sampling: The extracted feature set was used to train the models after a split of the feature set to train(80%) and test(20%). Stratified sampling strategy was used to split the data to address class imbalance problem and to ensure that even the minority classes are equally represneted in both train and test set. The following models where used for the experiment: Naive Bayes, Decision Tree, SVC, Bernoulli Naive Bayes, and Random Forest Classifiers. We got an accuracy value of 81% with Random forest classifier. Due to the class imbalance, we cannot rely on just the accuracy value of the classifiers. We observed that, in this case, (almost)all the test instances were predicted as belonging to *Literary* genre, and hence we got the high accuracy result, but low Precision and Recall values.

2) After Over-sampling: Over-sampling minority class was done in order to deal with this problem. Over-Sampling was preferred over under-sampling due to our small data set, as under-sampling might remove some important features that might be useful for our prediction. To overcome the class imbalance issue we have taken a different approach. Before the data split, we have used over-sampling method (SMOTE), to resolve class imbalance. "SMOTE is an oversampling technique that generates synthetic samples from the minority class. It is used to obtain a synthetically class-balanced or nearly class-balanced training set, which is then used to train the classifier" [4]. Now the data set has equal numbers for all the classes (792 which was the total count for majority genre *'Literary'*). The new feature set was used to train the models after a split of the feature set to train(80%) and test(20%), without stratified sampling. It was observed that, the model performances had improved to a great extent. The best fit was Random forest model which gave 99% accuracy. The evaluation section shows the complete report for all the models.

V. Evaluation

Machine learning models such as Naive Bayes, Decision Tree, SVC, BernoulliNB and Random Forest Classifiers are built using the extracted feature sets. The data set is split into train set consisting of 80% of features and test set consisting of 20% of features. The model was trained using the train set and evaluated on the test set.

The results have been evaluated using the conventional metrics for classification tasks - Accuracy, Precision, and Recall. Accuracy(A) is the ratio of number of correct predictions to the number of total predictions. Precision(P) being the ratio of number of positive observations predicted correctly to the total predicted positive observations and Recall R is the ratio of number of positive observations predicted correctly to the all observations in actual class.

A = (TP+TN)/(TP+TN+FP+FN)
P = TP/(TP+FP)
R = TP/(TP+FN)
where,
TP - correctly predicted positive values
TN - correctly predicted negative values
FP and FN - predicted values contradicts the actual values

However, the above mentioned models gave satisfactory accuracy, but low precision and recall on the existing data set. This problem was addressed by Over-sampling minority class in order to get good precision and recall scores. After over-sampling, three models were chosen which provided good results- Naive Bayes, Decision Tree and Random Forest Classifiers. The Classification Report for three classifiers trained upon the extracted feature set after over-sampling is shown in Figure 3, Figure 4, and Figure 5.

	precision	recall	f1-score	support
Allegories	0.97	1.00	0.99	152
Christmas Stories	0.68	0.32	0.44	161
Detective and Mystery	0.40	0.34	0.37	160
Ghost and Horror	0.66	0.82	0.73	166
Humorous and Wit and Satire	0.77	0.70	0.73	183
Literary	0.32	0.30	0.31	159
Love and Romance	0.49	0.52	0.51	145
Sea and Adventure	0.35	0.43	0.38	143
Western Stories	0.44	0.58	0.50	157
accuracy			0.56	1426
macro avg	0.57	0.56	0.55	1426
weighted avg	0.57	0.56	0.55	1426

Figure 3: Naive Bayes Classifier

	precision	recall	f1-score	support
Allegories	1.00	1.00	1.00	152
Christmas Stories	0.98	0.99	0.98	161
Detective and Mystery	0.91	0.94	0.93	160
Ghost and Horror	0.99	1.00	1.00	166
Humorous and Wit and Satire	0.98	1.00	0.99	183
Literary	0.90	0.70	0.79	159
Love and Romance	0.92	0.98	0.95	145
Sea and Adventure	0.93	0.99	0.96	143
Western Stories	0.98	1.00	0.99	157
accuracy			0.96	1426
macro avg	0.95	0.96	0.95	1426
weighted avg	0.96	0.96	0.95	1426

Figure 4: Decision Tree Classifier

	precision	recall	f1-score	support
Allegories	1.00	1.00	1.00	152
Christmas Stories	1.00	1.00	1.00	161
Detective and Mystery	0.97	1.00	0.98	160
Ghost and Horror	1.00	1.00	1.00	166
Humorous and Wit and Satire	1.00	1.00	1.00	183
Literary	1.00	0.95	0.97	159
Love and Romance	0.99	1.00	1.00	145
Sea and Adventure	0.99	1.00	0.99	143
Western Stories	1.00	1.00	1.00	157
accuracy			0.99	1426
macro avg	0.99	0.99	0.99	1426
weighted avg	0.99	0.99	0.99	1426

Figure 5: Random Forest Classifier

Among the three models, Random Forest ensemble approach gives best results. A confusion matrix for Random Forest Classifier is presented below depicting the number of correct and incorrect classification

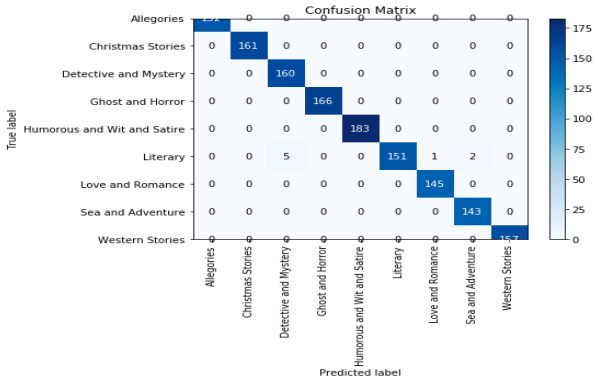

Figure 6: Confusion Matrix for Random Forest Classifier

VI. Conclusion

This Paper has presented the genre identification problem of books that requires both syntactic and thematic analysis in order to classify books from the corpus to their respective genre. This classification was challenging since the documents were longer than other normal text classification data set. In particular, a lot of time(approx.8 hours) was spent on feature set extraction which was then used to build various machine learning models. An improvement that could be made over our model is to take the sentiment over chunks of text, rather than on the whole book. Since the sentiment will vary throughout the flow of a book, it was not very beneficial to take extract sentiment at the book level. Different machine learning models are presented and evaluated for the assignment of the genre. Naive Bayes Classifier, Decision Tree Classifier and Random Forest Classifier are considered and we found out that this classification was best solved using an ensemble approach, specifically, an ensemble of random forest classifiers.

References

[1] https://www.gutenberg.org/

[2] http://dke.ovgu.de/findke/en/Research/Data+Sets-p-1140.html

[3] https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f

[4] https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-106: :text=SMOTE