Semester Project: Genre Identification on Gutenberg Corpus

Group 27

1st Abdullahil Kafi
Data and Knowledge Engineering
Otto von Guericke Universitat Magdeburg
Magdeburg, Germany
abdullahil.kafi@st.ovgu.de

2nd Israt Nowshin
Data and Knowledge Engineering
Otto von Guericke Universitat Magdeburg
Magdeburg, Germany
israt.nowshin@st.ovgu.de

***Abstract*—In this paper, we will discuss how different features were extracted from fictions to determine the genre of the fictions. Different features like sentence length, POS tagging were used and the model was trained using them. The accuracy of different models were compared after that.**

***Index Terms*—genre, Gutenberg, tokenization, POS tagging**

I. Introduction

Fiction is the literary form that is read by people of all ages. There are different genres of fiction and Gutenberg corpus consists of ten of them. In this project, we work with some fictions from Gutenberg corpus and try to classify the genre of a book. We extract features from the data and try to find the genres of the books based on some unique features of different genres of the book. Classification of genre of a book is important so that the reader does not have to read the whole fiction in order to find the genre of a book. If the genre can be identified properly, it makes it easier for the reader to choose and read books according to the genre of their choice.

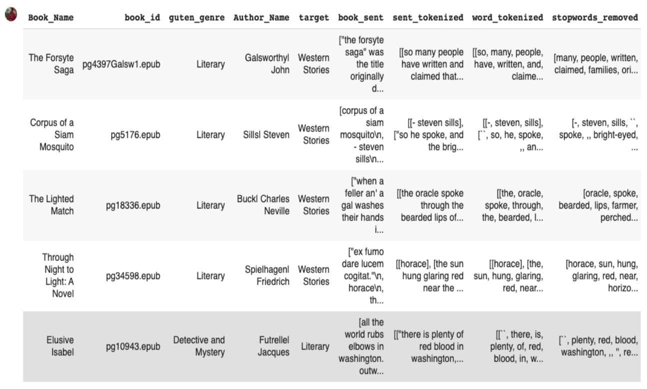
In this paper, we are going to discuss how different features are extracted from the 996 fictions of Gutenberg corpus to identify the genre of a book. After preparing the dataset for feature extraction, we are going to train the model and test the accuracy over the validation set.

II. Data

The dataset used for the project is collected from Project Gutenberg which is a part of 19th Century English Novel. The dataset consists of 996 novels with 10 different genres. Thus, the data of the novels are text data written in paragraph form. There is a master.csv file which consists of the names, genre, book ID and author of the books. The data in this file was structured in tabular format. The HTML files contain the novels in text format which was used for extracting the feature.

The data set had to be preprocessed in order to extract features from it. After loading the file containing the gutenberg corpus using pandas, the master.csv file was used for that. From the file, the name of the book and the book ID of the respective books are used. The prefix .epub was removed from the book id using .split('.') and it is replaced by '-content.html'. Doing this step was important to access the contents of the files with the given Book ID and to know the book name of the given Book ID. Having known the book name and the book ID of the book, we then access the contents of the book from the HTML files and store them in the data frame.

Next steps are tokenization, removal of stopwords and stemming of the text data. The sentences of each of the books were converted to words and they are stored in the data frame. Stopwords were removed from the tokenized words that were splitted from the sentences before. As the novels are written in english, we are using the English language for stopword removal. In order to do that, we imported the built-in library 'stopwords from nltk.corpus'. After removing the stopword, porter stemmer is used for stemming. We used 'PorterStemmer from nltk.stem' to accomplish this task.



Book_Name	book_id	guten_genre	Author_Name	target	book_sent	sent_tokenized	word_tokenized	stopwords_removed
The Forsyte Saga	pg4397Galsw1.epub	Literary	Galsworthyl John	Western Stories	["the forsyte saga" was the title originally d...	[[so many people have written and claimed that...	[[so, many, people, have, written, and, claime...	[many, people, written, claimed, families, ori...
Corpus of a Siam Mosquito	pg5176.epub	Literary	Sillsl Steven	Western Stories	[corpus of a siam mosquito\n, - steven sills\n...	[[- steven sills], ["so he spoke, and the brig...	[[-, steven, sills], [``, so, he, spoke, ,, an...	[-, steven, sills, ``, spoke, ,, bright-eyed, ...
The Lighted Match	pg18336.epub	Literary	Buckl Charles Neville	Western Stories	["when a feller an' a gal washes their hands i...	[[the oracle spoke through the bearded lips of...	[[the, oracle, spoke, through, the, bearded, l...	[oracle, spoke, bearded, lips, farmer, perched...
Through Night to Light: A Novel	pg34598.epub	Literary	Spielhagenl Friedrich	Western Stories	["ex fumo dare lucem cogitat."\n, horace\n, th...	[[horace], [the sun hung glaring red near the ...	[[horace], [the, sun, hung, glaring, red, near...	[horace, sun, hung, glaring, red, near, horizo...
Elusive Isabel	pg10943.epub	Detective and Mystery	Futrellel Jacques	Literary	[all the world rubs elbows in washington. outw...	[["there is plenty of red blood in washington,...	[[``, there, is, plenty, of, red, blood, in, w...	[``, plenty, red, blood, washington, ,, '', re...

Fig. 1. Data after preprocessing

III. Concept

Methods for testing the data set depends on the type of genre of the book. Some methods are effective in finding the genres of some books while the same methods can be ineffective when they are used to identify the genres of other books. In order to find the genre of a book, different features can be useful, for example, character, plot, point of view, setting, style, and theme of the book. [1] Different features are important to classify the genre of different books.

The genre of some of the books can be determined from the use of different words in the books. Some books contain some words more than others, for example, the word 'love' appears more in books of Love and Romance genre, 'sea, island' are the words most common in books of the genre Sea and Adventure. However, this might not work every time for all the books of the same genre. In our dataset, the word 'love' appears a lot in the books of genre Literary as well. So, it makes it difficult to classify the books as Love and Romance based on the word count of specific words as it might cause bias as well. [2]

IV. Implementation

Implementation step took a lot of time because of the amount of data that are being used for the project. Python 3.7 is used for the final implementation. Initially we were using Google Colaboratory for the implementation. The reason for using it was the ability to implement different parts of the project in small blocks in Google Colaboratory. However, it also has some drawbacks as it was crashing after a certain period of runtime, we eventually moved into a local machine.

The steps of Implementation includes: Preprocessing data, Model selection and Evaluation. It is briefly visualized in Fig 2.

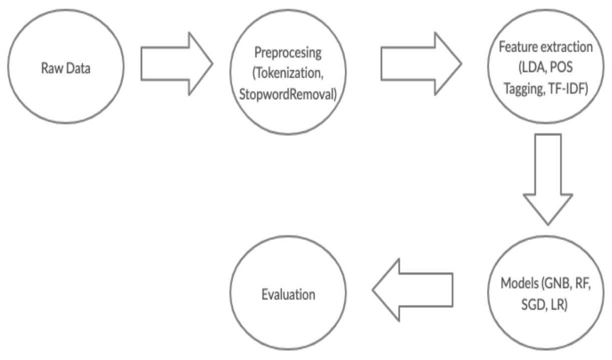

Raw Data
Preprocesing (Tokenization, StopwordRemoval)
Feature extraction (LDA, POS Tagging, TF-IDF)
Models (GNB, RF, SGD, LR)
Evaluation

Fig. 2. Steps of Implementation

As the number of data is too large and it was taking too much time we took a sample of the dataset which is homogenous. This dataset is a representation of the large dataset as it also contains the genres. We made sure that the small dataset contains fictions from all 10 genres. In order to do that, we used the method '.isin()' from pandas dataframe.

While importing all the text from the data files we cleaned the text files. With the texts from each individual book we applied Latent Dirichlet allocation (LDA) and wordcloud to find out the mostly used words across all the genres. For visualising word cloud we have searched for twenty most frequent words in each of the genres. [4] For visualising LDA we have searched through all the books in the dataset and counted the most common words using count vectorizer. As we can see from Fig 2, the most common words for the fictions of genre Sea and Adventure are captain, man, island, ship. Similarly, Love and Romance genre also contains common words like love, lady and eyes as it is shown in Fig 4.

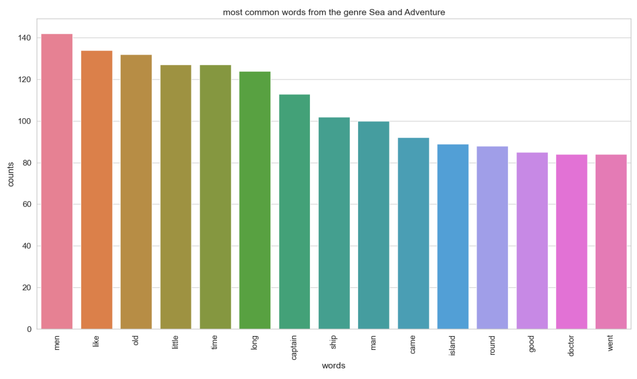


Fig. 3. Most common words of Sea and Adventure

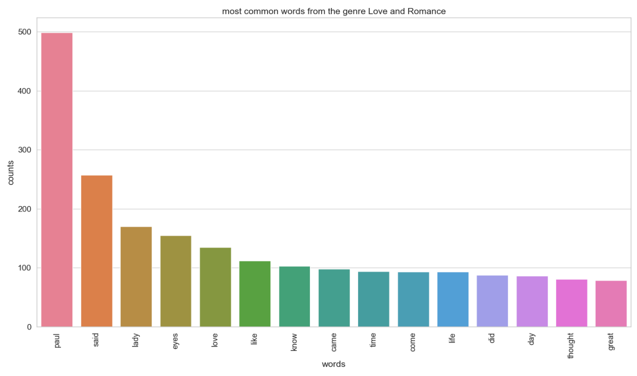


Fig. 4. Most common words of Love and Romance

The text of each book was taken as a string and we applied count vectorizer with stop words as english. We stored frequent words found from each book and later on compared them on the basis of the genre. The most common words from each genre are later visualized. (Fig 3)

Initially we thought of extracting Plots and Characters from the dataset. However, as we could not manage to extract all these features we took some very simplistic features which also have meaningful implications. The features which we have used are Sentence length, POS tagging and word count, Tf-idf.

Sentence length indicates the length of a sentence in a document. It is a very useful feature to identify the genre of a text. Usually, sentences with high lengths are more often history books, literature books. Most of the historical, political books contain long sentences. Thus, we thought sentence length will be an effective feature to classify genre of the fictions. We have split the data into a ratio of 70:30 as train and test set. The sentence length is an attribute and genre is

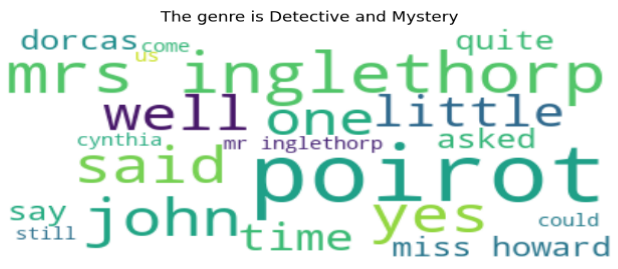


Fig. 5. Most common words

taken as a label and different classification algorithms were used to calculate the accuracy of the models.

POS tagging is used to find Parts of Speech of the words. The tokenized words were taken and POS tagging was implemented on them using the built-in NLTK library ‘pos tagger’. It shows which words belong to which parts of speech. After getting the parts of speech for each word, the summation is taken for each type of parts of speech present in each fiction. In the same way as before, we divided our sample into a 70:30 ratio of train and test set to build a model and test it’s accuracy. The number of words in different parts of speech (Coordinating Conjunction, Proper Noun etc.) are the attributes and the genre of each fiction are the labels. We fit this in different models taking different amounts of dataset to test the accuracy of this feature in genre classification.

TF-IDF is used to find the importance of words in a document. Thus this can be a very important feature for classifying the genre of the fictions. If it can be found which words are important for each genre, it can be a very effective feature for identification of genre of fiction. We used the sklearns built in TF-IDF model to find out the score for every sentence of each book. Firstly, we take the most frequent or common words that we found out while applying LDA and store those words as the vocabulary. Furthermore, we search for these words in every book and store the tf-idf score in an array. We fit the TF-IDF score and the genre in different classification model to predict the genre of fiction books.

V. Evaluation

As mentioned earlier we are not taking the full dataset rather we take a portion of it as it is more efficient. We have taken datasets of three sizes (100,150 and 200 samples) and applied the pre-processing. First we tried with the smallest dataset (100) samples which took less time to compute. After preprocessing we divided the dataset into train and test set using sklearn built in function. The ratio is 70:30 for train and test set.

We evaluated the performance using 4 different algorithms. The machine learning algorithms used for the classification task were Gaussian Naive Bayes, SGD Classifier, Logistic Regression and Random forest. We have used the sklearn tool to implement these algorithms. While fitting the data into the model it was necessary to change the data format into float32. We used different numbers of parameters and tested the performance using these 4 algorithms.

The classification accuracy for these 4 different algorithms used with different features (POS Tagging, TF-IDF, Sentence Length) are given in Fig 6, 7 and 8.

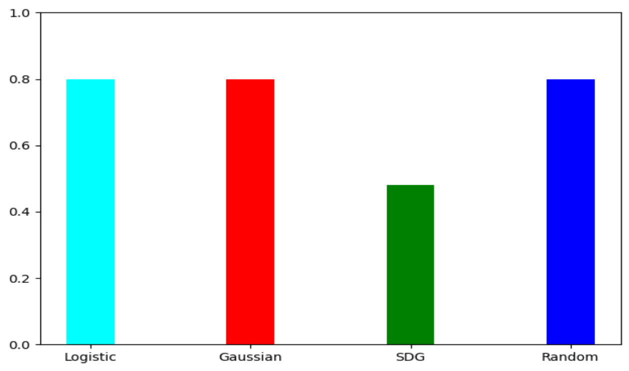


Fig. 6. Accuracy of different Models for Coordinating Conjunctive

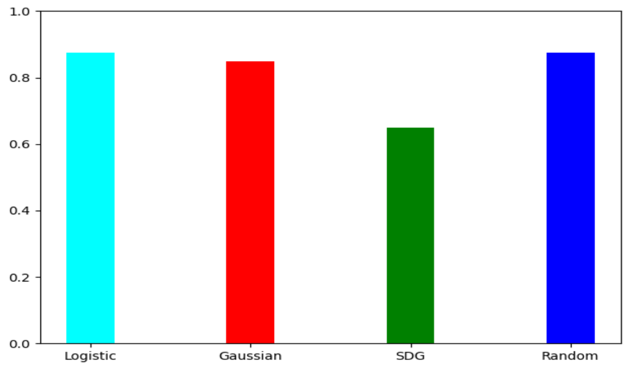


Fig. 7. Accuracy of different Models for TF-IDF

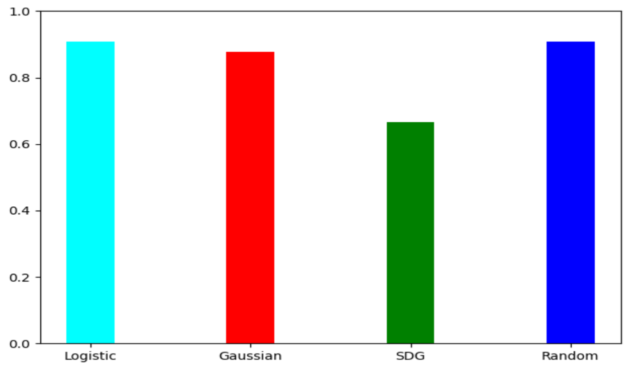
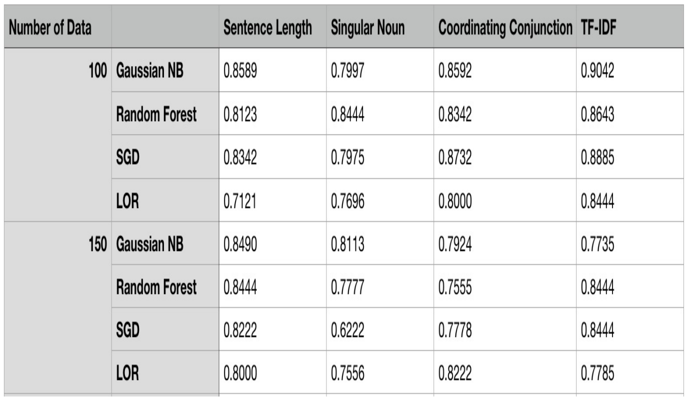


Fig. 8. Accuracy of different Models for Sentence Length

The accuracy is also checked using different number of samples and the accuracy changed with it. Table 1 shows the difference of accuracy for different algorithms with different sample size.

Table 1: Accuracy of Different Algorithms with Data Samples of Different Sizes



Number of Data		Sentence Length	Singular Noun	Coordinating Conjunction	TF-IDF
100	Gaussian NB	0.8589	0.7997	0.8592	0.9042
	Random Forest	0.8123	0.8444	0.8342	0.8643
	SGD	0.8342	0.7975	0.8732	0.8885
	LOR	0.7121	0.7696	0.8000	0.8444
150	Gaussian NB	0.8490	0.8113	0.7924	0.7735
	Random Forest	0.8444	0.7777	0.7555	0.8444
	SGD	0.8222	0.6222	0.7778	0.8444
	LOR	0.8000	0.7556	0.8222	0.7785

VI. Conclusion

In this project, we detected genre of fictions from Gutenberg corpus using 4 features: Sentence Length, Coordinating Conjunction, TF-IDF and singular Noun. Testing on the four different algorithms with 3 different sample sizes, we could conclude that the average accuracy we get is approximately between 80-90%. The project became difficult to implement because of runtime performance. While taking the full dataset into consideration, it took a lot of memory during preprocessing. So, we had to take small samples. The result would have been better if the sample size was larger. However, due to the shortage of memory, it was not possible to implement and thus we had to take a small sample which is a representative of a bigger dataset. Thus, we don't get the expected result which wouldn't have been the case if we used a larger dataset. However, we could have used small chunks of each file and pre-process those chunks simultaneously in threads which may have allowed us to use the full dataset. Moreover, we could have taken more features which would improve the result.

References

[1] S. Polley and S. Ghosh. "Comparing the qualitative impact of different features and similarities on fictional text using SIMFIC".

[2] S. Polley, S. Ghosh, M. Thiel, M. Kotzyba, A. Nurnberger. "SIMFIC: An Explainable Book Search Companion"

[3] https://web.csulb.edu/ yamadaty/EleFic.html

[4] https://www.datacamp.com/community/tutorials/wordcloud-python

[5] https://www.kaggle.com/ngyptr/python-nltk-sentiment-analysis/notebook

[6] https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0

[7] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html