Statistical and Deep Learning Approaches for Literary Genre Classification

Chapter · February 2022

DOI: 10.1007/978-981-16-5689-7_26

2 authors:

Anshaj Goyal
Dayalbagh Educational Institute
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Prem Prakash
Dayalbagh Educational Institute
16 PUBLICATIONS **70** CITATIONS
SEE PROFILE

Statistical and Deep Learning Approaches for Literary Genre Classification

Anshaj Goyal and V. Prem Prakash

Abstract The problem of automatically classifying literary texts into their corresponding genres/subgenres occurs in Natural Language Processing (NLP) and has applications in library classification systems and semantic metadata extraction. Existing literature on this problem attempts to reduce the dimensionality of the datasets such that the models are still able to retain the structure and order of the data, considering the high computational complexity of training neural networks on long text sizes. This paper compares several machine learning and deep learning approaches for literary genre classification using the entire texts for classification. Experimental work is performed in a high peformance computing environment that uses general-purpose computing on graphics processing units (GPGPUs) for optimizing the performance of deep learning-based applications. Results are reported on a large collection of freely available texts sourced from Project Gutenberg and show the effectiveness of the proposed approach.

Keywords NLP · Literary genre classification · BERT · Deep learning · RNN · LSTM

1 Introduction

A genre, in the literary context, is a type, or class, that a given work of literature may be categorized as. Such classifications are typically made on the basis of commonalities found in the style of writing, subject or theme, format, use of certain figures of speech, cliches and motifs, similarities in cultural, associated geographical and temporal/historical settings and/or narrative techniques, and so on. A literary work viewed in terms of so many factors may fit into multiple genres, the distinctions between which are themselves fuzzy and overlapping.

A. Goyal (✉) · V. Prem Prakash
Faculty of Engineering, Dayalbagh Educational Institute (Deemed University), Agra, India

V. Prem Prakash
e-mail: vpremprakash@dei.ac.in

S. Tiwari et al. (eds.), *Advances in Data and Information Sciences*, Lecture Notes in Networks and Systems 318, https://doi.org/10.1007/978-981-16-5689-7_26

A key application of literary genre classification occurs in library classification systems, where books have to be categorized and organized. With the advent of large digital libraries and collections, such as Project Gutenberg [1], Google Books [2], and ERIC [3] in the past few years, there has been growing interest in automating the classification of books by genre. Automated genre classification is also a fundamental step in the process of automatically extracting semantic metadata from digital objects [4] and is very useful for fast query and retrieval in large archives containing millions of digital documents. It also results in a deeper understanding of the themes that developed in a complete narrative [5].

Natural Language Processing (NLP) is known to be very effective for word/phrase classification in documents and for text classification and retrieval, operations that are essential in applications such as plagiarism checking, paraphrasing, automated summarization, concise essence extraction, and sentiment analysis.

Machine learning (ML) algorithms and more particularly artificial neural networks (ANN), have been widely used with NLP for applications such as sentiment analysis and text classification, including include convolutional neural networks (CNNs) [6], recurrent neural networks (RNN), and hierarchical attention network (HAN) [7] as well as variants of long short-term memory (LSTM) [8] and autoencoders that represent unsupervised document representation as hierarchies of LSTMs [9].

One challenge with ANNs has been the high computational complexity incurred when processing very large or long text segments. A recent work [10] presents the application of NLP and machine learning for automated genre identification of novels that focuses on ways to reduce dataset size in order to obtain a reasonable trade-off between computation effort and accuracy of classification. However, by exploiting the computational power of massively parallel multi-processor and GPGPU-based computing architectures, it is now possible to develop neural language models that can process a complete novel end-to-end.

This paper applies several ML and deep learning approaches for the classification of English novels and studies their performance on a large dataset of ebooks selected from Project Gutenberg, when considering the reduced dataset strategies proposed in [10] along with a simple full-text strategy proposed in this work. The remainder of the article is organized in the following manner: The datasets used and the major data preprocessing operations performed are described in Sect. 2. The various machine learning and deep learning models used are briefly described in Sect. 3, experimental work is presented in Sect. 4, and conclusions are drawn in Sect. 5.

2 Data and Preprocessing

A broad classification of literary genres distinguishes between compositions (short stories, novels, etc.) rooted in themes that have been created by their authors, and those that are based on actual events, resulting in two genres—fiction and non-fiction—respectively, each genre typically has several subgenres. An author may contribute in more than one genre, and the works of different authors are distinguished by widely

Table 1 Summary of Gutenberg dataset contents

Genre	Count	Length
Science fiction	1186	1,65,874
Adventure stories	595	4,59,556
Love stories	508	4,80,265
Detective and mystery stories	485	4,75,883
Historical fiction	410	5,58,343
Western stories	393	4,63,569

variant writing styles and linguistic preferences. All this makes the content of the data corpora more compilated. Another problem that arises here is the difficulty in finding a suitably structured dataset.

For this work, a subset of the Project Gutenberg collection of openly available eBooks has been used. The Gutenberg website contains over 60,000 ebooks and includes thousands of freely available fiction and non-fiction texts. Metadata related to the books such as author, title, publication date, etc., are also provided [11] A brief genre-wise overview of the Gutenberg dataset is provided in Table 1.

These eBooks along with index file are publicly available on the Project Gutenberg website. In this work, a subset of the Gutenberg [1] dataset, comprising of two subgenres of English fiction novels is used for comparing the various genre classification models. The two selected genres—Horror and Fiction—are almost equal in terms of data count so there is no class imbalance problem. The dataset comprised of these two genres is used for training and testing the machine learning and deep learning models.

The Gutenberg dataset was collected from official Gutenberg data servers, along with a collection of RDF (Resource Description Framework) files containing metadata such as Author name, Novel Id, Category, Keywords, Genre, and Book Length. Of these, the genre was of interest for this work and was linked with the associated novel in a clean data frame.

Headers, footers, and other extraneous data were deemed unnecessary and were, hence, removed before further processing. For statistical modeling approaches, stop words were removed, whereas for deep learning-based models that automatically learn patterns of data, stop words were retained. Common operations like stemming, lemmatization, and sentence tokenizing were performed on both datasets. The datasets and other code files are also available on publicly on github repository.[1]

The dataset was transformed into a vector representation using term frequency-inverse document frequency (tf-idf) vectorization [12] and fed to the machine learning algorithm for training the model.

[1] https://www.github.com/leetanshaj/genre_classification.

3 Machine Learning and Deep Learning Models Considered

Three statistical machine learning algorithms—random forest (RF), Naïve Bayes, and support vector machine (SVM) —and four deep learning approaches—LSTM, CNN, RNN with LSTM and BERT were considered in this work. A brief description of each of these models is provided in this section.

3.1 Random Forest Classifier

The random forest (RF) algorithm [13] is a well-known ensemble model that builds a set of decision trees from a random subset of training data and aggregates the outputs of these decision trees using simple averaging or voting.

3.2 Support Vector Machine Classifier

Support vector machine (SVM) classifiers [14] aim to find a hyperplane in N-dimensional feature space that assigns data points that fall on either side of the hyperplane to different classes such that the distance between the points and hyperplane is maximized. This is achieved with the help of a loss function called hinge loss, computed as $(1 - y * f(x))$. A regularization parameter is also added to the cost function for balancing the margin maximization and loss, and hence, the cost function can be given as:

$$\min_w \lambda ||w||^2 + \sum_{i=1}^{n} (1 - y_i x_i, w)$$

3.3 Naïve Bayes Classifier

The Naïve Bayes classifier [15] is a group of classification algorithms whose decision rule(s) are based on Bayes' theorem of conditional probability [16].

The classifier first converts the set of data items into frequency tables and then generates a likelihood table using the probability of each feature. Thereafter, it calculates the posterior probability, for a given set of features $F = \{f_1, f_2, \ldots, f_n\}$ and classes $C = \{c_1, c_2, \ldots, c\}$ using Bayes' theorem, as follows:

$$P(c|f_1, f_2, \ldots, f_n) = \frac{P(f_1|c)P(f_2|c)\ldots\ldots P(f_n|c)}{P(f_1)P(f_2)\ldots\ldots P(f_n)}$$

According to the assumptions of Bayes' Theorem, the features are independent to each other and, hence, can also be given as:

$$P(y|f_1, f_2 \ldots .f_n) \propto P(c) \prod_{i=1}^{N} P(f_i|c)$$

3.4 Convolutional Neural Network (CNN)

A variant of the CNN [6] is used, wherein a word embedding layer learns a dense k-dimension vector representation for each word in the training phase. The value of k was experimentally set to 100 dimensions here. The word embeddings, thus, created are then input to the convolutional layer using certain filtering methods [17]. During training, a dropout probability of 0.5 was applied following the convolutional layer as regularizer.

3.5 Long Short-Term Memory (LSTM) Classifier

The architecture of the LSTM classifier is designed to "forget" information that is deemed unnecessary. LSTM consists of input and output gates, and a forget gate. The input gate discovers which value from the input should be used to modify the memory with the help of sigmoid and tanh function. A detailed description of the computational processes underlying LSTM may be found in [17]

3.6 Bidirectional LSTM Classifier

Bidirectional LSTM (Bi-LSTM) is also known as recurrent neural network (RNN) with LSTM [18]. RNNs process the previous layer's input to give output concisely for the next layer. The outputs are passed to the next hidden layer. The complexity of increasing variables and finetuned parameters is reduced in this approach. The expression for the current state is given as: $L_r = f(L_{r-1}, M_t)$.

3.7 BERT Classifier

Bidirectional encoder representations from transformers (BERT) [19] use an attention mechanism called transformer in order to learn the behavioral patterns between vector representation of text or general literature text. It is a conceptually simple and empirically powerful tool in which the language model is trained bidirectionally. Transformers are basically used to weigh the inputs given and accordingly assign importance to each input using encoders and decoders. Encoders are used to read the given text and decoders to assign the importance in order to produce a prediction. Though transformers used both, BERT only needs the encoder process for the problem at hand.

4 Experimental Work

The statistical ML and deep learning models presented in Sect. 3 were tested on an NVIDIA DGX-1 System having 8 GPUs with Pascal/Volta daughter cards and HBM 2 memory. The performance of each model when provided with subsets comprised of the first 5000 (FIRST5000), last 5000 (LAST5000), and random 5000 (RANDOM5000) words of the text, as done in [10] was compared with the case where the text was input in its entirety (FULLTEXT). As already explained, stop words were not removed from the data for training the deep learning models, and words were not converted to vectors. They were left intentionally so that neural nets could learn the pattern while including them. The activation function used was sigmoid, and loss function was binary cross entropy for the neural models. The number of epochs trained (for ML) was 5 for a batch size of 64. Each of the DL models was allowed 5 epochs for TOP5000, LAST5000, and RANDOM 5000 instances and 10 epochs for the FULLTEXT instances. A train-test split of 80:20 was used for training the models. The results obtained are given in Table 2. From the tabulated results, it may be seen that, of the three machine learning algorithms considered, the best results were obtained by the tree-based SVM classification algorithm with the vectorised full text as input (FULLTEXT). While results on TOP5000, LAST5000 and RANDOM5000 were comparatively worse, and SVM was still the best performer in all cases except for one instance (TOP5000) where Naïve Bayes classifier reported a better $F1$-score. Several deep learning approaches such as CNN, RNN, LSTM, and BERT were tested on the same datasets. While LSTM did report better accuracy and $F1$-score on the RANDOM5000 instances, the relatively superior results obtained for the other test cases only highlight the fact that RANDOM5000 does not seem to be a good choice for this problem. The RNN algorithm outperformed all the other models in terms of both accuracy score and $F1$-score.

Table 2 Accuracy and $F1$-score of selected machine learning and deep learning models on the selected project gutenberg dataset

Models		TOP5000		LAST5000		RANDOM5000		FULLTEXT	
		Acc	$F1$	Acc	$F1$	Acc	$F1$	Acc	$F1$
Machine learning models	Random forest	0.55	0.53	0.41	0.38	0.25	0.21	0.62	0.61
	SVM	0.58	0.52	0.44	0.41	0.27	0.28	0.85	0.84
	Naïve Bayes	0.57	0.6	0.37	0.39	0.24	0.19	0.81	0.8
Deep learning models	CNN	0.74	0.69	0.52	0.48	0.43	0.37	0.77	0.79
	RNN	0.8	0.7	0.53	0.55	0.41	0.42	0.82	0.75
	LSTM	0.78	0.74	0.52	0.51	0.54	0.43	0.79	0.73
	BERT	0.79	0.71	0.51	0.43	0.32	0.31	0.78	0.81

5 Conclusions

With the advent of digital libraries in recent years, there has been a lot of interest in automating the classification of literary texts according to genre. The problem also has applications in automated extraction of semantic metadata. This paper presents a comparison of several well-known machine learning and deep learning approaches to literary text genre classification. Unlike earlier approaches, the algorithms are implemented on a massively parallel GPGPU computing system and trained on the entire text. The results of the experimental evaluation show that such an approach can yield good performance, with the SVM classifier reporting 85% accuracy of genre prediction among the machine learning models considered. Unlike the ML models, the deep learning algorithms did not perform any additional preprocessing such as stop word removal, yet they report relatively high accuracy and $F1$-scores, which reflects well on the effectiveness of these approaches. Of the four deep learning models compared in the paper, the best results are reported by the RNN approach, which performs better than all the other models considered in this work. The results obtained also show the effectiveness of using fulltext for literary genre classification as compared with other heuristic approaches.

References

1. Hart M (1971) Project Gutenberg. Available www.gutenberg.org. Accessed 17 Apr 2021
2. Google Inc. (2004). Google books. http://books.google.com. Accessed 17 Apr 2021
3. Education Resources Information Center (1966) Available www.eric.gov. Accessed 17 Apr 2021.
4. Abbott D, Kim Y (2008) Genre classification. In: DCC briefing Papers: introduction to curation. Available https://www.dcc.ac.uk/guidance/briefing-papers/introduction-curation/genre-classification. Accessed 18 Apr 2021
5. Kim Y. Automating semantic metadata extraction. Available https://www.digitalpreservationeurope.info/publications/briefs/semantic%20metatada.pdf. Accessed 19 Apr 2021
6. Kim Y (2014) Convolutional neural networks for sentence classification. arXiv
7. Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E (2016) Hierarchical attention networks for document classification. In: Proceedings of the 2016 conference of the north {A}merican chapter of the association for computational linguistics: human language technologies. Association for Computational Linguistics, pp 1480–1489
8. Tang D, Qin B Liu T (2015) Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 1422–1432
9. Li J, Luong M-T, Jurafsky D (2015) A hierarchical neural autoencoder for paragraphs and documents. arXiv
10. Worsham J, Kalita J (2018) Genre identification and the compositional effect of genre in literature. In: Proceedings of the 27th international conference on computational linguistics, pp 1963–1973
11. Gerlach M, Font-Clos F (2020) A standardized project gutenberg corpus for statistical analysis of natural language and quantitative linguistics. In: MDPI
12. Rajaraman A, Ullman J (2011) Data mining. In: Mining of massive datasets. Cambridge University Press, pp 1–17