Text-based Author Identification

Hongming Fu, Ruize Ma, Yi Sun, Yu Gu
(Authors are listed in alphabetical order to reflect equal contributions to the research.)

Brown University
May 9th, 2024

Github: https://github.com/RuizeMa666/CSCI2470_Final_Project_AutherDetective/
Devpost: https://devpost.com/software/author-detective-texts-traits-and-tea-leaves

1. Introduction

This project aims to develop an architecture for predicting the authors of some given texts. Text-based author identification serves as a critical tool in literary analysis and academic research. Many texts, especially historical ones, come with unclear authorship. This project intends to accurately attribute such works to their rightful creators, thereby ensuring that authors are recognized and acknowledged for their contributions. The literary domain, often obscured by the use of ghostwriters, calls for enhanced transparency in authorship attribution. By identifying the true authors behind texts, this research seeks to foster greater clarity and accountability. Furthermore, a detailed examination of distinctive writing styles, patterns, and features inherent to each author not only enriches the understanding of individual works but also augments the collective appreciation of literature.

To address these objectives, a committee architecture comprising six predictive models has been developed. It consists of two CNN-based models, three RNN-based models, and one Transformer model. This ensemble approach is designed to leverage the strengths of each model type, thereby enhancing the accuracy and reliability of author predictions across diverse textual datasets.

2. Data

2.1 Dataset

The dataset employed in this study comprises 53 books authored by 23 distinct writers. These texts are divided into 2,300 excerpts, each consisting of approximately 3,000 characters. To facilitate robust model training and evaluation, the dataset is split into training and testing sets. Seventy percent of the data is allocated for training purposes, ensuring adequate exposure to varied literary styles and nuances. The remaining thirty percent serves as the test set, used to assess the generalization capabilities of the developed models across unseen textual segments.

2.2 Preprocessing

Preprocessing of CNN-Based Model

During the preprocessing phase for the CNN model, the initial step involved reading the dataset and determining its encoding via the chardet library. Subsequently, each text underwent a cleaning process in which specific characters were removed and all letters were converted to lowercase. The words were then lemmatized utilizing the lemmatization capabilities of the Natural Language Toolkit (NLTK). Following this, a comprehensive vocabulary was constructed encompassing all words extracted from the texts. To further process the texts, I developed a function named create_n_grams. This function is designed to accept a list of sentences, from which it generates n-grams, subsequently transforms these n-grams into one-hot encoded vectors, and finally adjusts these vectors by either padding or truncating them to conform to a predetermined length.

Preprocessing of RNN-based Model

The preprocessor for RNN-based model and transformer model both introduces external, pre-established and pre-trained packages for tokenization and vocabulary building. The RNN-based models use GloVe for word embedding with dimensions of 100. GloVe based word embedding is constructed with word-word co occurrence matrix and captures fine-grained semantic and syntactic meaning. Such a pre-trained embedding model can accurately reflect semantic meaning and gives our model a better chance of learning text information.

For each piece of text, we have also padded each text sequence so that all text in a batch has the same length. The input text for each datapoint can naturally have different length since each sentence is of different length.Padding enables this by ensuring that all sequences in a batch are of equal length.The purpose of such padding is to make it suitable for batch processing in neural networks. Fixed size text information is ideal for computational execution in CUDA and can lead to finer performance. Equal length text enables CUDA to enforce parallel procession and allows it to shorten training time significantly.

Preprocessing of Transformer:

For the Transformer based model, prior to training, the textual data undergoes a series of preprocessing steps to ensure compatibility with the BERT-based transformer model. Initially, the BertTokenizer from the 'bert-base-uncased' model is employed to tokenize the texts. This tokenizer converts each text excerpt into a sequence of tokens that are consistent with the format expected by the BERT model. The tokenization process includes truncating or padding the sequences to a uniform length of 128 tokens to accommodate the model's fixed input size requirement, and generating corresponding attention masks that enable the model to differentiate between meaningful tokens and padding. Subsequently, the author labels are encoded into a numerical format to facilitate model training. A unique identifier is assigned to each author, creating a mapping from author names to discrete label indices. This encoding transforms the author names associated with each text excerpt into a tensor of numerical labels, which are used as the target outputs for model training.

3. Methodology

3.1 Basic CNN and multi-channel multi-model voting CNN

To start with, we first tried the basic 1D-convolutional CNN model, which usually has a strong capacity of capturing features of sequential data including context. Our structure setup includes an embedding layer with an embedding size of 16, followed by a dropout layer with an adaptive dropout rate, a 1D-convolutional layer equipped with 500 filters, a kernel size of 3, and normal distribution activation, and finally, a max-pooling layer. Despite achieving a high training accuracy of 99.88%, the validation accuracy stagnated around 54%. We thought there could be an issue that the model formed an overly segmented decision boundary because the model learns features that are complex but not necessarily effective for generalization, which hindered its predictive capabilities.

To enhance our model’s predictive capabilities, we've developed a new multi-channel, multi-model CNN architecture. This innovative setup includes several CNN modules, each featuring distinct channels. Within each channel in each CNN module, we maintained the same structure of the formal 1-D convolutional CNN model, despite that we added a flattened layer for the following concatenation. We also varied the initialization of weights across channels to capture diverse features during training. The outputs from these channels are concatenated and then processed through a dense layer, which outputs a probability for each of the 23 authors. Each CNN module produces its own predictions which are then aggregated through a voting mechanism to arrive at a final decision based on the majority vote. Prior to processing the data through this multi-channel, multi-model architecture, we preprocess our data into several batches. Each batch contains differently, but equally distributed data to ensure balanced learning of each author's features across the modules, thereby aiming to reduce bias and variance.

This revised architecture could potentially enhance the accuracy of the final predictions. We addressed the issue of the model forming an overly segmented decision boundary by limiting the amount of learning data per module, thus preventing overfitting. Meanwhile, we maintained the complexity essential for robust feature extraction by incorporating multiple CNN modules, each with several channels, thereby preventing underfitting. This approach balances the need for detailed feature recognition with the ability to generalize across new, unseen data. The new architecture is able to give us correct predictions even if multiple modules fail.

3.2 RNN-based Model

For RNN-based implementation, the architecture is less complex than what we have learned from the lecture. Traditionally, RNN-based models take an encoder-decoder structure since they are trying to solve a “many to many” problem: the model takes a string as input and another coherent string as output. With each token processed, the internal parameters of the RNN-based unit are updated based on the loss gradient. This step helps the model to learn over time which features of the text are important for predicting the author. Since in this question, the intended output is the probability distribution among authors, the problem becomes a “many to one” problem. Thus, we only need to implement the encoder section and leave the decoder alone.

The Transformer model follows a similar approach. By only implementing the encoder part, our transformer model will be able to produce fixed-size vectors that represent probability distribution among 23 authors, while still taking advantage of mechanisms like self-attention.

For an RNN-based model, including LSTM, Bi-LSTM, and GRU, we only need an RNN layer and a dense layer. The core unit, the RNN layer, takes in each token sequentially and updates its internal parameters. After each token input, the unit will produce a fixed-size vector that represents the probability distribution of authorships. We take the final output of our core unit as the final prediction. The authorship probability distribution can be achieved after passing such a vector through a dense layer with a softmax function.

3.3 “Committee” Model

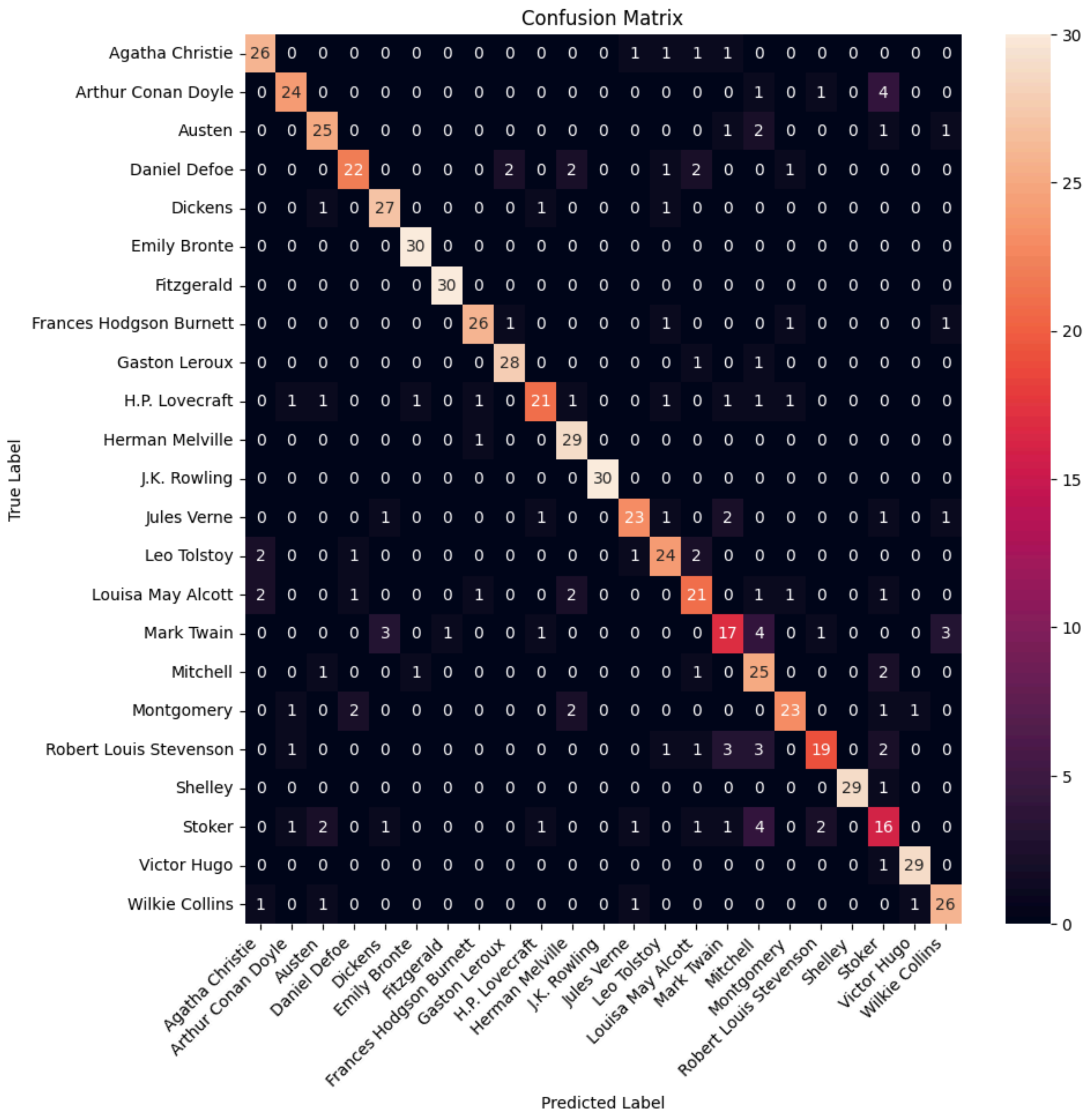
Currently, we possess a suite of five finely tuned models, each distinguished by unique operational mechanisms. Recognizing the monotony of relying solely on the model with the highest accuracy, we sought to explore methodologies that could leverage the collective strengths of multiple models. Consequently, we commenced a detailed analysis to evaluate the advantages and disadvantages inherent in each of the five models.

Each model demonstrates optimal performance within specific contexts. For instance, our analysis revealed that the 1-gram CNN model primarily performs word frequency analysis, enabling it to capture the distinct vocabulary styles of various authors accurately. However, it fails to discern the interrelationships among words. In contrast, the recurrent neural network (RNN) models, including GRU, LSTM, and BiLSTM, excel in recognizing long-term dependencies among data points but may underperform given the constraint of processing around 3000 characters. Meanwhile, the transformer model, theoretically the most precise, is more susceptible to overfitting.

While we are analyzing our models' predictions, we found there are some potential scenarios: Firstly, the simpler models may yield implausible predictions for complex authorial styles that the more sophisticated models can adeptly handle. Secondly, the intricate models could overly complicate patterns, resulting in erroneous predictions, whereas the simpler models might capture the essence more accurately. Thirdly, predictions concerning one author by a model could provide insightful data for forecasting the writing probabilities of other authors.

To address these challenges, we have devised a novel "committee" approach. The committee's idea is to combine a group of classifiers(models) and treat each model as an expert. We then let those experts vote to get our final prediction result. Unlike vanilla committees that treat models as experts and only give one vote to each model, we will treat each model's predicted value for one author as one expert since the prediction on one author may also provide information to other authors. We allow them to individually contribute to the overall prediction concerning the authorship of a given text. Since we have five models and 23 authors, we will have 5*23 = 115 experts in our committee. We initiate the process by concatenating the softmax predictions from all models (a n * 115 matrix). The elements within each row of this concatenated matrix act as the opinions of specialized experts for the corresponding text excerpts. In order to make sure our resulting matrix has the correct shape, we need to use a weight matrix of shape 115*23, each column represents the influence, or power, each expert wields in authorship determination. We employed gradient descent to optimize this weight matrix, calibrating the influence of each expert based on their performance during training, which is quantified using the cross-entropy loss metric. After that, we will take the softmax value with respect to the product of the concatenated matrix and the optimized weight matrix. The final output from this committee represents a consensus prediction, reflecting the combined expertise of all models involved.

4. Results



	Agatha Christie	Arthur Conan Doyle	Austen	Daniel Defoe	Dickens	Emily Bronte	Fitzgerald	Frances Hodgson Burnett	Gaston Leroux	H.P. Lovecraft	Herman Melville	J.K. Rowling	Jules Verne	Leo Tolstoy	Louisa May Alcott	Mark Twain	Mitchell	Montgomery	Robert Louis Stevenson	Shelley	Stoker	Victor Hugo	Wilkie Collins
Agatha Christie	26	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
Arthur Conan Doyle	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	4	0	0
Austen	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	1	0	1
Daniel Defoe	0	0	0	22	0	0	0	0	2	0	2	0	0	1	2	0	0	1	0	0	0	0	0
Dickens	0	0	1	0	27	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
Emily Bronte	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fitzgerald	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Frances Hodgson Burnett	0	0	0	0	0	0	0	26	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1
Gaston Leroux	0	0	0	0	0	0	0	0	28	0	0	0	0	0	1	0	1	0	0	0	0	0	0
H.P. Lovecraft	0	1	1	0	0	1	0	1	0	21	1	0	0	1	0	1	1	1	0	0	0	0	0
Herman Melville	0	0	0	0	0	0	0	1	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0
J.K. Rowling	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
Jules Verne	0	0	0	0	1	0	0	0	0	1	0	0	23	1	0	2	0	0	0	0	1	0	1
Leo Tolstoy	2	0	0	1	0	0	0	0	0	0	0	0	1	24	2	0	0	0	0	0	0	0	0
Louisa May Alcott	2	0	0	1	0	0	0	1	0	0	2	0	0	0	21	0	1	1	0	0	1	0	0
Mark Twain	0	0	0	0	3	0	1	0	0	1	0	0	0	0	0	17	4	0	1	0	0	0	3
Mitchell	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	25	0	0	0	2	0	0
Montgomery	0	1	0	2	0	0	0	0	0	0	2	0	0	0	0	0	0	23	0	0	1	1	0
Robert Louis Stevenson	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	3	3	0	19	0	2	0	0
Shelley	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	1	0	0
Stoker	0	1	2	0	1	0	0	0	0	1	0	0	1	0	1	1	4	0	2	0	16	0	0
Victor Hugo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	29	0
Wilkie Collins	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	26

Fig 1: Confusion Matrix of the Committee Model

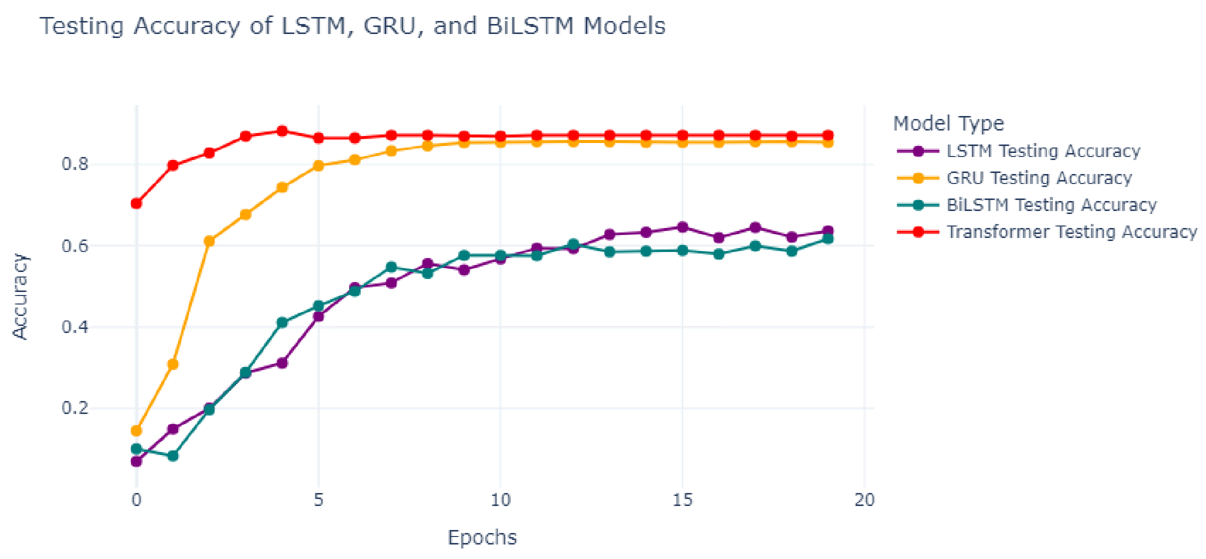
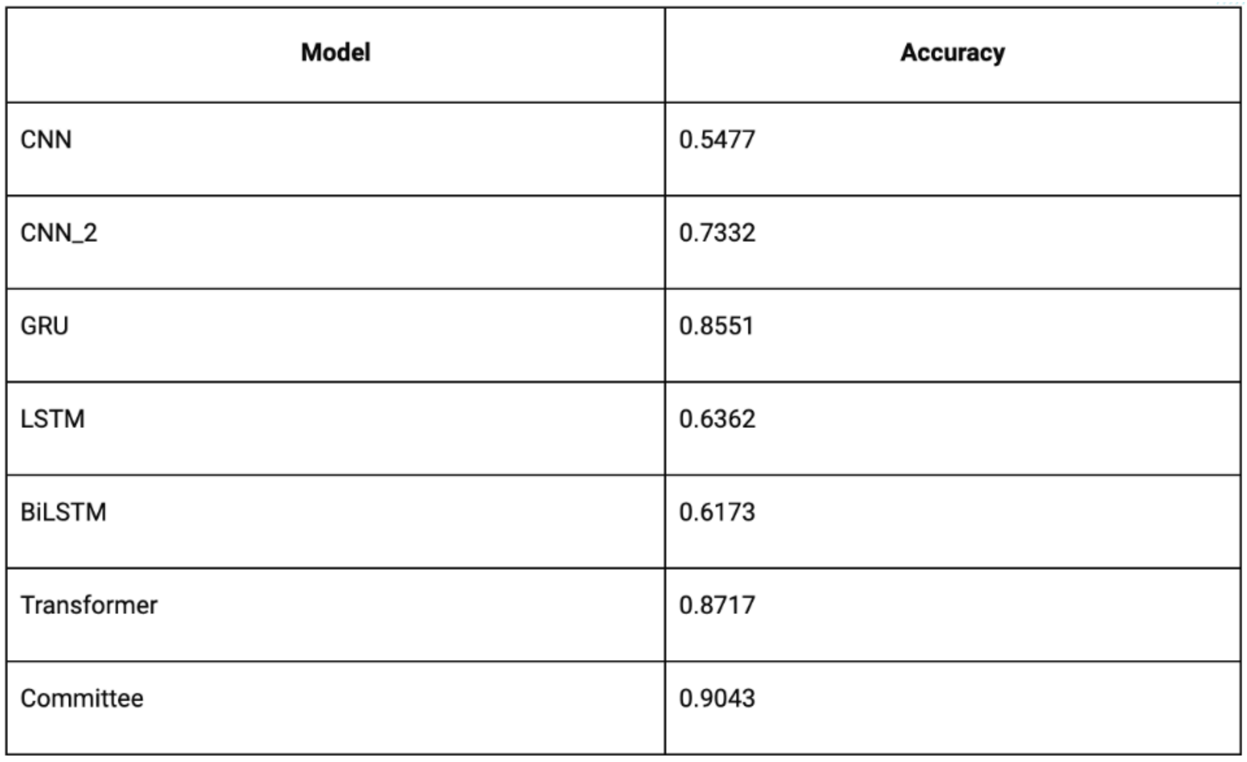


Fig2: Testing result of RNN-based Model and Transformer Model



Model	Accuracy
CNN	0.5477
CNN_2	0.7332
GRU	0.8551
LSTM	0.6362
BiLSTM	0.6173
Transformer	0.8717
Committee	0.9043

Fig3: Testing Accuracy of Individual Models and the Committee Model

The confusion matrix displayed in Fig1 shows the committee prediction result. We can observe that the committee model excels at predicting authors like J.K. Rowling, Bronte, and Fitzgerald. These authors are all known for their distinctive writing style, which explains the good accuracy. On the other hand, the model's prediction of Mark Twain, Stevenson, and Stoker can be less satisfactory. The model can still have uneven predictive power among different authors, even with the help of committee mechanisms.

According to Fig3, we can see that the original CNN model can only reach an accuracy of 54.77%. This suggests that it is unable to effectively capture numerous patterns in the authors' writing styles. However, subsequent enhancements through the implementation of an adaptive dropout rate and a multi-channel multi-model architecture led to a significant improvement, raising the accuracy to 73.32%.

Regarding the RNN models, the GRU model stands out, demonstrating superior performance with an accuracy of 85.51%. This enhanced performance can be attributed to the relative simplicity of the GRU architecture, which features two gates as opposed to the three gates in LSTMs. This simpler configuration may be particularly well-suited to datasets that are neither exceedingly large nor complex.

As anticipated, the Transformer model outperformed all other individual models in terms of accuracy(87.17%). This outcome aligns with expectations given the Transformer's advanced capabilities in handling sequence data.

Upon integrating the outputs from all five models, our composite 'committee' model achieved the highest accuracy(90.43%). This result underscores the benefits of leveraging diverse model architectures to enhance overall performance by capturing a broader array of linguistic features.

5. Challenges

One of the challenges we face in this task is limited computational resources. Training text information is no task for the CPU and impossible to finish like our assignments. Thus, we have to take advantage of GPU and CUDA as well as cuDNN for performance. However, Tensorflow doesn’t support CUDA for Win 11. Therefore, we have rewritten all our codes for its implementation with PyTorch. The issue with PyTorch is compatibility. While ensuring our PyTorch and Torchtext versions, we also need to guarantee some of our functions and attributes implemented in our code are available. Ultimately, setting up these environmental factors consumes almost an entire day of coding time.

Due to computational constraints, we encountered issues with Tensorflow when attempting to process a dense layer with a hidden size exceeding 300,000 in our committee model. This large hidden size is necessary because our architecture involves reshaping this layer into a tensor of dimensions (5, 13154, 23), which represents five probability matrices, each of size (13154, 23), before performing a reduction sum on these matrices to consolidate them into a single (13154, 23) matrix.

6. Reflections

The committee architecture implemented in this study achieves an accuracy of 90 percent, surpassing the performance of any individual model within the ensemble. This result underscores the effectiveness of leveraging multiple models to enhance predictive accuracy in text-based author identification.

This project can be enhanced in several ways in the future. Expanding the dataset to include a broader array of authors and literary works could potentially improve the robustness and adaptability of the model. Additionally, the deployment of more advanced computational resources would significantly reduce processing times, accommodating larger volumes of data more efficiently. Furthermore, implementing a masking strategy for sensitive tokens within the training data—such as protagonists' names and locations—could prevent the model from over-relying on these elements for author prediction, thereby fostering a deeper learning of literary style and syntax.

One significant takeaway of this project is the enhanced capability in data cleaning processes. The data was meticulously collected and transformed into formats that are more amenable to algorithmic analysis, emphasizing the importance of preparatory data management. Furthermore, a deeper comprehension of the gradient descent algorithm was achieved through manual calculations during the development of the committee architecture. This hands-on approach provided invaluable insights into the optimization challenges and solutions in deep learning. Additionally, the project underscored the efficacy of collaborative development. Working as a team facilitated a more efficient debugging process and accelerated the development cycle, demonstrating the substantial benefits of collaborative efforts in complex computational tasks.

7. References

1. Brownlee, J. (2020, September 2). *How to develop a multichannel CNN model for text classification*. MachineLearningMastery.com. https://machinelearningmastery.com/develop-n-gram-multichannel-convolutional-neural-network-sentiment-analysis/
2. Better digit recognition with a committee of Simple Neural Nets | IEEE conference publication | IEEE Xplore. (n.d.). https://ieeexplore.ieee.org/document/6065510/