Author Identifier

Huang Chenyu – A0295813W
Huang Tianle – A0296702Y
Zhu Yihang - A0297220H

April 26, 2025

Contents

1 Preface

- Introduction Website: https://yates-z.top/author_analyzer.html
- Github Project: https://github.com/Yates-zyh/Author-Identifier
- Hugging Face Models:
  - Identifier: https://huggingface.co/Yates-zyh/author_identifier
  - Generator: https://huggingface.co/fjxddy/author-stylegan

2 Business Problem

In today's rapidly evolving digital landscape, the way individuals consume information has become increasingly fragmented. While the proliferation of platforms like short-form video, social media, and countless websites offers unprecedented access to diverse content, this very diversification paradoxically intensifies the challenge of engaging deeply with written material and verifying its origins. Compelling excerpts and impactful statements frequently circulate without proper attribution, leaving readers disconnected from the original source and context. Existing search methodologies often prove inadequate in navigating this complex environment. The prevalence of casual paraphrasing, inaccurate reposting across myriad online platforms, and the sheer volume of digital text make the task of tracing original sources exceptionally difficult and time-consuming.

This difficulty extends beyond mere inconvenience, imposing significant costs on individuals and society. It breeds user frustration and results in wasted research time as people struggle to locate original works or explore related content by the same author. More critically, the opacity surrounding content origins creates fertile ground for the spread of misinformation and disinformation, eroding trust in digital information ecosystems. Furthermore, it increases the potential for unintentional plagiarism and facilitates the misappropriation of intellectual property, undermining creators' rights. The challenge is further amplified by the advent of sophisticated Large Language Models (LLMs), which can generate human-quality text at scale, blurring the lines between human and machine authorship and making robust attribution methods indispensable for maintaining content integrity. This lack of verifiable attribution hinders genuine engagement and makes derivative reading or analysis based on unattributed fragments a prohibitively effort-intensive task for users. The impact is systemic, affecting not only individual users but also the credibility of academic research, the integrity of journalistic reporting, and the accountability sought in legal contexts.

The Author Identifier project directly confronts these issues by employing advanced techniques in stylometry, the quantitative analysis of linguistic style. This field operates on the principle that authors possess unique and often subconscious stylistic "fingerprints" manifested through patterns in their lexical choices, syntactic structures, punctuation habits, and overall text organization. By extracting and analyzing these distinctive characteristics, the Author Identifier technology enables accurate authorship attribution (identifying the likely author of a text from a known set) , authorship verification (determining if two texts were written by the same author) , and nuanced text similarity analysis based on writing style rather than just topic. This core capability underpins powerful applications across diverse industries, addressing critical business and societal challenges. Key application areas include Content Recommendation & Personalization, Literary & Historical Authorship Attribution, Plagiarism Detection , Forensic Linguistics , and Brand Voice Analysis. Ultimately, by providing a means to verify the origin and authenticity of written content, the Author Identifier project aims to foster greater trust and accountability in the digital realm.

2.1 Content Recommendation & Personalization

- **Problem:** Digital media platforms, e-commerce sites, and streaming services grapple with the challenge of information overload, striving to present users with relevant content from vast catalogs. Conventional recommendation systems often rely on explicit metadata like genre or keywords, or on collaborative filtering techniques that analyze past user interactions. While useful, these methods frequently fail to capture the more subtle aspects of user preference related to how content is written—its specific style, narrative voice, complexity, and emotional tone. This can lead to recommendations that feel generic or misaligned with a user's tastes, resulting in diminished engagement and satisfaction. Readers often develop affinities for authors based not just on subject matter but on their unique stylistic expression.
- **Solution:** The Author Identifier technology facilitates a more sophisticated form of content-based recommendation. By analyzing deep stylistic features—such as sentence length distributions, vocabulary richness,

syntactic complexity, and characteristic n-gram patterns —potentially combined with sentiment or emotional analysis , the system can create richer profiles of both content and user preferences. It can group authors exhibiting similar writing styles and identify the specific stylistic elements a user favors. Consequently, it can recommend new articles, books, or even video transcripts that align with a user's preferred authorial style and emotional resonance, even across different topics or genres. This moves beyond recommending what a user might like to how they might appreciate it being presented, leading to more personalized, engaging discoveries and fostering greater user loyalty and retention.

2.2 Literary & Historical Authorship Attribution

- **Problem:** Scholars in literature and history frequently confront texts whose authorship is disputed, anonymous, or masked by a pseudonym. Famous examples include the Federalist Papers , debates surrounding Shakespearean works , and numerous historical documents or anonymous publications. Traditional attribution methods, often relying on limited historical records or subjective interpretations of style, can lead to inconclusive or enduring debates. Establishing accurate authorship is fundamental for correct historical interpretation, robust literary analysis, and verifying the provenance and authenticity of important texts.
- **Solution:** Computational stylometry, as implemented in the Author Identifier project, offers an objective, data-driven approach to these longstanding challenges. The system quantifies and compares a wide array of linguistic markers—including function word frequencies, sentence length variability, syntactic constructions, and character or word n-grams —between the text in question and the known writings of potential authors. By calculating the statistical similarity or distance between stylistic profiles , the system provides quantitative evidence to support or refute specific attribution hypotheses. This computational approach can significantly aid researchers in resolving historical authorship disputes and verifying the authenticity of literary and historical texts, potentially offering more accessible and reproducible analyses compared to purely qualitative methods. However, the reliability of such analyses depends heavily on the availability of sufficient text lengths for both the questioned document and comparison corpora.

2.3 Plagiarism Detection

- **Problem:** Maintaining academic, journalistic, and creative integrity requires robust methods for detecting plagiarism. Standard software primarily identifies verbatim copying but struggles with more sophisticated forms of unoriginal work. These include extensive paraphrasing that obscures direct text matching, the use of translation or text-spinning tools to disguise sources , "thesaurus plagiarism" where words are systematically substituted , contract cheating involving ghostwriters , and intrinsic plagiarism where plagiarized sections are interwoven with original writing. The increasing capability of AI to generate text presents novel challenges for ensuring originality. Undetected plagiarism undermines educational standards, devalues original authorship, and infringes upon intellectual property rights.
- **Solution:** The Author Identifier's focus on stylistic analysis provides a powerful complement to traditional plagiarism detection methods. Rather than solely matching text strings, it examines the underlying authorial style. This allows the system to: (1) Identify significant inconsistencies in writing style within a single document, flagging potential instances of intrinsic plagiarism where different sections may originate from different authors. (2) Compare the stylistic profile of a submitted work against the known style of the claimed author to detect potential ghostwriting or unauthorized use of another's identity. (3) Recognize stylistic similarities between a submitted text and potential source materials, even when significant paraphrasing has occurred. This capability is vital for upholding academic honesty and protecting intellectual property, including software code. This approach moves detection towards a more proactive assessment of authorial authenticity rather than purely reactive text matching.

2.4 Forensic Linguistics

- **Problem:** In legal and investigative contexts, determining the authorship of questioned documents—such as anonymous threats, ransom notes, fraudulent communications, online harassment, or disputed confessions—is often crucial. Law enforcement agencies, intelligence services, and legal teams need reliable methods to link communications to suspects, verify the authenticity of statements, or profile unknown authors based on their writing. Traditional investigative techniques may lack the specific linguistic expertise needed, and subjective analysis can be challenged in court. Establishing authorship or generating a linguistic profile can provide critical leads or supporting evidence.

- **Solution:** Stylometry serves as a core technique within forensic linguistics, offering systematic methods for analyzing authorship in legal settings. The Author Identifier technology can analyze subtle, often unconscious, stylistic markers in questioned documents—including idiolectal patterns , grammatical choices , punctuation usage , lexical preferences , characteristic errors , and n-gram frequencies. Comparing these features against known writing samples enables: (1) Authorship Attribution/Verification: Providing statistical evidence to link a document to a specific individual or determine common authorship across multiple texts. (2) Author Profiling: Inferring potential demographic traits (age, gender, region, education) or psychological characteristics of an unknown author from their language use. (3) Authenticity Assessment: Evaluating whether a document (e.g., a confession or statement) is consistent with the typical writing style of the person purported to have produced it. These techniques can also be applied in software forensics to trace the origins of malicious code. However, forensic applications face significant hurdles, including frequently short text lengths , noisy data like transcripts , and the possibility of adversarial stylometry, where authors deliberately attempt to disguise their style or imitate another's. Such obfuscation efforts can significantly degrade classifier performance , requiring cautious interpretation of results, particularly in high-stakes scenarios.

2.5 Brand Voice Analysis

- **Problem:** Organizations invest significant resources in cultivating a unique brand voice—the consistent tone, style, and personality conveyed through all written communications. Maintaining this voice across diverse channels (marketing materials, website copy, social media, customer support, internal communications) is vital for building brand identity, fostering customer trust, and ensuring effective engagement. However, achieving consistency becomes increasingly difficult as content creation involves multiple internal teams, external agencies, freelancers, and AI generation tools. Deviations from the established brand voice can dilute brand recognition, confuse audiences, and ultimately reduce the impact and return on investment of marketing and communication efforts.
- **Solution:** Adapting stylometric analysis for brand voice offers an objective way to monitor, measure, and enforce brand consistency at scale. By defining a target brand voice profile—based on exemplary content or detailed style guides specifying attributes like formality, sentence structure, vocabulary, and tone —the Author Identifier technology can be used to: (1) Automatically assess whether newly created content aligns with the desired brand voice. (2) Provide actionable feedback to human writers or content teams to guide revisions and ensure adherence. (3) Train and guide AI content generation models (such as those from OpenAI or Anthropic) to consistently produce outputs that match the specific brand voice, moving beyond generic text. (4) Streamline internal review and editing processes, reducing bottlenecks and accelerating content publication while maintaining quality standards. This transforms brand voice management from a subjective editorial task into a data-informed process, enabling objective measurement and improvement of consistency across all organizational communications.

3 Dataset

This project utilizes literary works from several renowned authors as its dataset for training and validating author style identification and generation models. The process of data collection, cleaning, and organization is detailed below.

3.1 Data Sources

The dataset primarily originates from the Project Gutenberg Online Library. A custom crawler script was employed to automatically retrieve classic, copyright-free literary works. The selection focuses on 10 famous authors known for their distinctive writing styles, as summarized in Table 1. To enhance the model's ability to differentiate these authors from others, supplementary corpus data from NLTK libraries was incorporated to represent "unknown authors". These supplementary samples were randomly drawn from the Gutenberg, Brown, Reuters, and WebText corpora available within NLTK. While utilizing public domain works, care was taken to ensure fair representation and proper acknowledgment of the original sources as part of ethical data handling practices.

Table 1: Target Authors and Data Collection Summary

Author Name	Representative Work
Jane Austen	Pride and Prejudice
Charles Dickens	A Tale of Two Cities
Mark Twain	The Adventures of Tom Sawyer
Arthur Conan Doyle	The Adventures of Sherlock Holmes
Charlotte Brontë	Jane Eyre
F. Scott Fitzgerald	The Great Gatsby
Herman Melville	Pierre; or The Ambiguities
Alexandre Dumas	The Three Musketeers
Agatha Christie	The murder of Roger Ackroyd
Gabriel García Márquez	One Hundred Years Of Solitude

3.2 Data Collection Process

Data collection was implemented using an automated crawler script. This process involved creating dedicated directory structures for each target author, searching for their works on the Project Gutenberg website, and subsequently locating and downloading the e-books specifically in plain text format. Responsible web crawling practices, such as appropriate request intervals and setting up user agents, were followed to ensure compliant data collection. Certain limitations were applied: a maximum of 10 representative works were gathered per author to maintain balance, and only works written in English were included for language consistency.

3.3 Data Preprocessing

Following collection, the raw text files were cleaned using a dedicated preprocessing script to ensure data quality suitable for model training. This involved removing Project Gutenberg-specific headers and footers containing metadata like copyright notices and license information. Any non-main content occasionally found at the end of texts, such as editor's notes, was also eliminated. Furthermore, excessive consecutive blank lines were deleted. The script utilizes multi-level identification strategies to accurately locate the start and end of the main literary content by searching for standard Gutenberg markers, common chapter patterns, or significantly long paragraphs to determine the text boundaries. Mechanisms were included to handle potential file encoding issues, ensuring UTF-8 consistency, alongside error handling and logging.

3.4 Dataset Organization

The resulting cleaned dataset was organized systematically. It was divided into two main sets stored in separate directories: `data_train/` for model training and `data_val/` for validation using unseen texts from the same authors. Within these directories, further subdirectories were created using each author's name, containing the corresponding plain text files of their works, named consistently based on original titles. Each selected literary work provides substantial text, typically averaging 50,000 to 100,000 words. All texts are in the `.txt` format for uniform processing. A final manual review ensured the completeness and representativeness of the works included for each author. This systematic workflow guarantees a high-quality dataset, providing a solid foundation for the author style identification system.

4 Approach

4.1 Identifier

The Author Style Identifier component is engineered to analyze textual content and ascertain its authorship by identifying unique and distinctive writing patterns characteristic of specific authors. Its development encompassed a standard machine learning workflow, beginning with rigorous model training and parameter optimization, followed by implementation for inference, and concluding with integration into a user-friendly graphical interface.

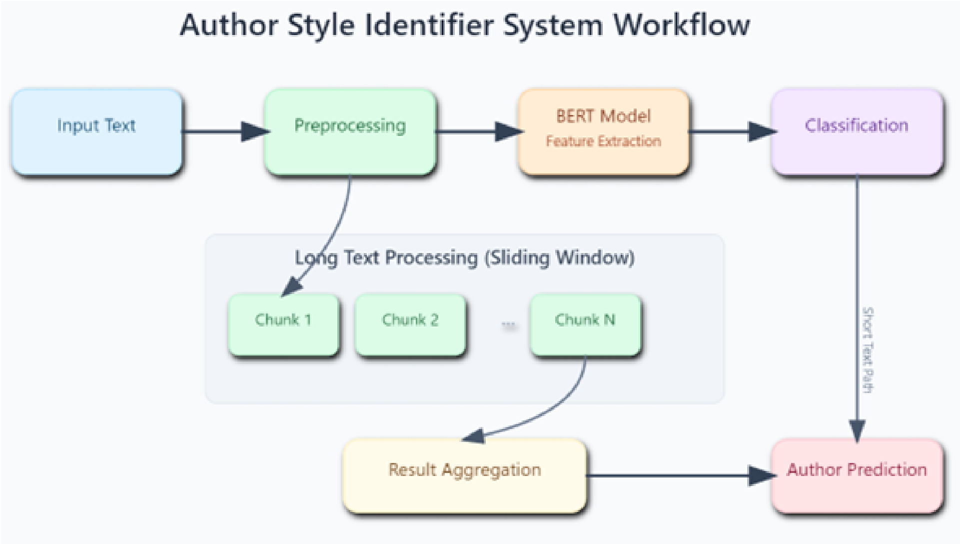


Figure 1: Author Style Identifier System Workflow.

4.1.1 Model Training and Data Preparation

The foundation of the identifier lies in the model training process. This script employs a fine-tuning strategy based on the BERT (Bidirectional Encoder Representations from Transformers) architecture, specifically the `Bert-base-uncased` pre-trained model. This model was chosen for its strong performance on various NLP tasks, including text classification, and its relatively manageable size compared to larger models, making it suitable for fine-tuning. The primary goal is to classify text segments according to their author.

Data preparation involves processing text files sourced from various authors, organized within structured directories. To handle potentially long documents, a **sliding window technique** is utilized. This method segments texts into manageable, overlapping chunks, ensuring contextual continuity while adhering to the model's maximum input length (typically 512 tokens). Tokenization is performed using the Bert Tokenizer corresponding to the `Bert-base-uncased` model. While BERT operates as a "black box," it is hypothesized that the model learns to distinguish authors based on a combination of subtle features, including characteristic vocabulary choices, sentence structure complexity, punctuation patterns, dialogue conventions, and overall narrative tone.

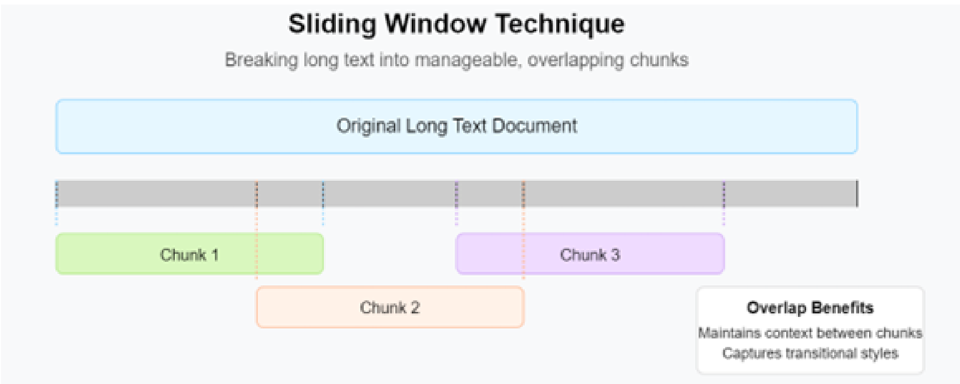


Figure 2: Sliding Window Technique.

A crucial aspect of training is addressing potential data imbalance. The function we use includes an option to equalize the number of training samples across all known authors, preventing the model from developing a bias towards authors with more available text data. Furthermore, to enhance the model's real-world applicability and its ability to discern when a text *doesn't* match any known author, an "Unknown Author" category is introduced. Samples for this category are sourced from diverse NLTK corpora like Gutenberg, Brown, Reuters, and WebText. The quantity of these "Unknown" samples is typically set relative to the average number of samples per known author, providing the model with negative examples during training.

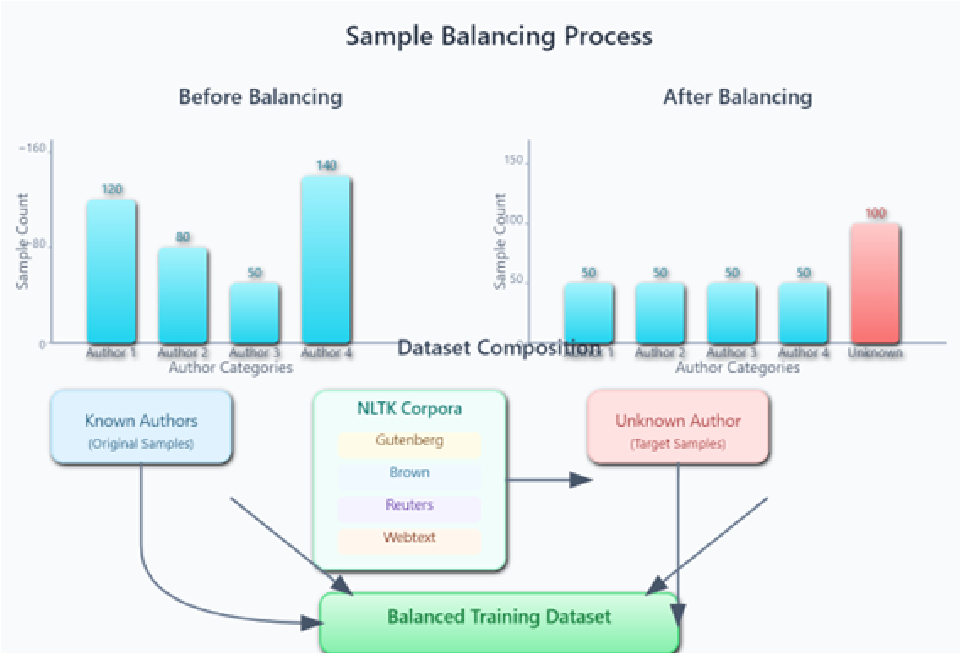


Figure 3: Sample Balancing Process.

The final model architecture is `BertForSequenceClassification`, loaded with weights from `Bert-base-uncased` and modified with an output layer sized to accommodate all known authors plus the "Unknown" category. The training loop incorporates several best practices: AdamW optimizer with weight decay, a linear learning rate scheduler with warmup, **early stopping** based on validation loss, **gradient clipping**, and model checkpointing. The process automatically utilizes GPU (CUDA) if available. Model performance is continuously evaluated using accuracy, loss, F1-score, classification reports, and confusion matrices. The F1-score is particularly important as it balances precision and recall, providing a robust measure for potentially imbalanced classes (especially considering the "Unknown" category). Confusion matrices are crucial for identifying specific misclassification patterns between authors, highlighting areas where the model might struggle.

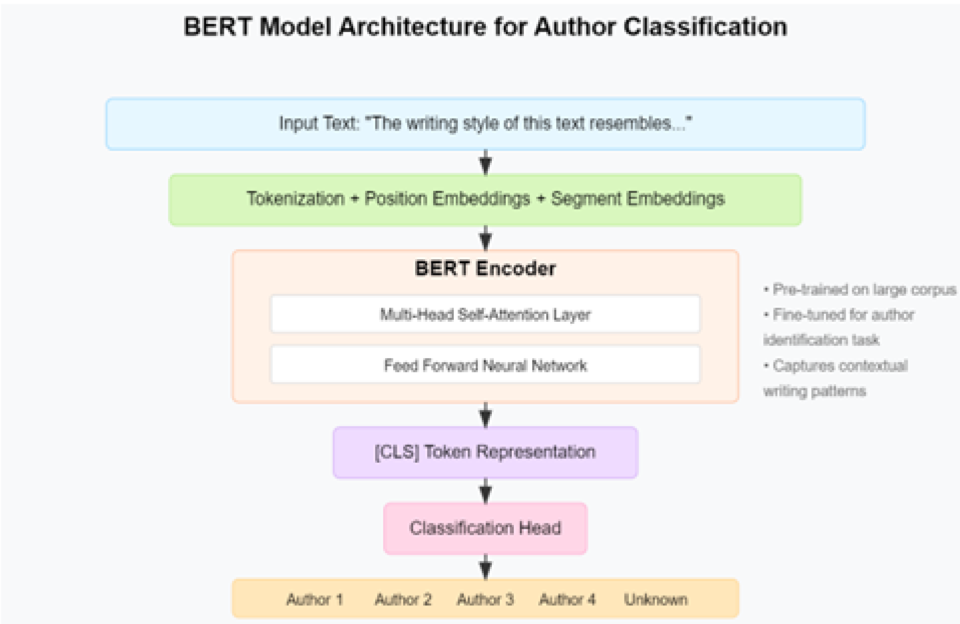


Figure 4: BERT Model Architecture for Author Classification.

4.1.2 Parameter Tuning Strategy

Following initial training, **parameter tuning** was conducted to optimize performance, as outlined in the report. A key parameter explored was the ratio of "Unknown Author" samples to known author samples. Experiments varying this ratio (e.g., 0.1, 0.3, 0.5, 0.7, 1.0) indicated that a ratio of 0.5 yielded the optimal balance. This ratio enhanced the model's ability to generalize and correctly identify unknown authors without significantly impairing its accuracy on known authors. The tuning framework involved pre-extracting samples for consistency across experiments, training multiple models varying only the target parameter, recording comprehensive metrics (accuracy, F1, loss), and using visualizations of training curves to compare results and identify the best configuration.

4.1.3 Inference Implementation (AuthorIdentifier Class)

The trained and optimized model is implemented within the `AuthorIdentifier` class in `identify.py` for practical use. This class encapsulates the logic for loading the model (either from a local path like `author_style_model` or directly from a Hugging Face repository, e.g., `Yates-zyh/author_identifier`), managing Hugging Face API tokens for accessing private or gated models, and automatically detecting the appropriate compute device (CPU/GPU). It handles the loading of the tokenizer, the classification model, and associated metadata, which includes details about the training process and label definitions. The class provides robust error handling and fallback mechanisms, attempting to download the model using a token if provided, otherwise loading from a local cache or directly from the Hugging Face Hub.

For analyzing text, the implementation mirrors the data preparation strategy. If a text exceeds a certain length, it's broken down into overlapping chunks. Each chunk is processed individually, which tokenizes the input, feeds it to the model, and obtains probability distributions over the author labels via SoftMax applied to the model's logits. A confidence threshold (default 0.6) is applied: if the highest probability for a known author falls below this threshold, the prediction defaults to "Unknown Author".

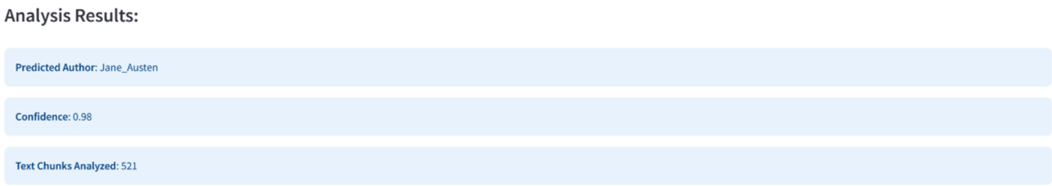
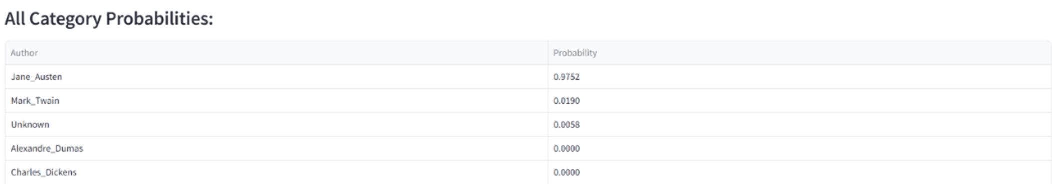


Figure 5: Example Analysis Results Interface (1).

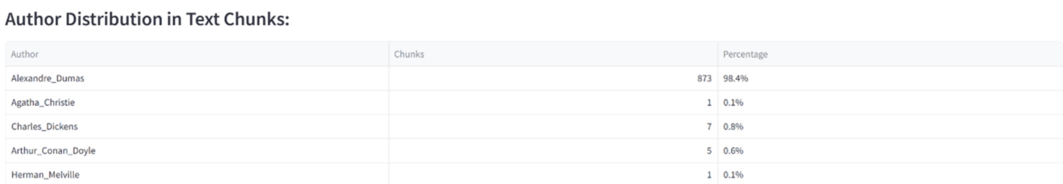


All Category Probabilities:

Author	Probability
Jane_Austen	0.9752
Mark_Twain	0.0190
Unknown	0.0058
Alexandre_Dumas	0.0000
Charles_Dickens	0.0000

Figure 6: Example Analysis Results Interface (2).

For multi-chunk texts, the probabilities from all chunks are aggregated (typically averaged) to produce a final prediction and confidence score. The results are returned in a structured dictionary containing the predicted author, confidence score, the full probability distribution, and optionally, chunk-level predictions and author distribution across chunks. The `analyze_file` method provides convenience for processing text directly from files. The `get_model_info` method returns metadata about the loaded model.



Author Distribution in Text Chunks:

Author	Chunks	Percentage
Alexandre_Dumas	873	98.4%
Agatha_Christie	1	0.1%
Charles_Dickens	7	0.8%
Arthur_Conan_Doyle	5	0.6%
Herman_Melville	1	0.1%

Figure 7: Example Analysis Results Interface (3).

4.2 Generator

In previous experiments, we encountered limitations with standard prompting techniques for large language models (LLMs) like GPT-4o/4.5. Even when provided with an author's name, these models struggled to consistently generate text that our fine-tuned discriminator identified as matching the target author's style. Attempts to guide generation using explicit stylistic summaries derived from statistical analyses (word frequency, sentence patterns, etc.) proved ineffective; the resulting texts often lacked stylistic similarity or were misclassified.

This highlights a limitation of general-purpose LLMs in capturing nuanced, implicit stylistic features without more targeted training objectives. The black-box nature of the BERT discriminator also prevented direct extraction of its judgment criteria for use in prompting. An attempt to use NLTK-based word contribution analysis to guide generation resulted in logically incoherent output.

Therefore, we adopted a Generative Adversarial Network (GAN) approach, specifically SeqGAN, which is well-suited for sequential data generation and allows for adversarial training towards a specific goal—in this case, mimicking authorial style as judged by our discriminator. Our primary goal was to train a generator capable of producing logically coherent text while achieving high similarity scores for the target author. Due to computational constraints (training on a personal machine), we opted for the 137 million parameter GPT-2 model released by OpenAI as the generator base. GPT-2 provides a solid foundation for text generation, and its smaller size allowed for feasible training and fine-tuning within the SeqGAN framework.

4.2.1 Model Structure

The modules of the generator section mainly include three parts: a generator based on the GPT2 model, a discriminator based on the BERT architecture, and an adversarial training architecture based on SeqGAN.

The discriminator used for GAN training adopts the default model of `Bert-base-uncased`. A unique index is created for each author in the dataset, and an additional "Unknown" category is added to identify texts that do not belong to any known author. Text samples are first loaded from each author's directory, then long texts are split into chunks of approximately 256 tokens, ensuring sentence integrity.

A sample balancing strategy is also applied, with the training sample limit for each author set to 1000 samples. If an author has more than 1000 samples, 1000 samples are randomly selected for training. Additional samples from the NLTK corpus are added via the `nltk_samples` parameter as part of the "Unknown" category to enhance performance.

For the generator part, the original GPT-2 model architecture is used, including the Transformer decoder and the language model head. Text is first tokenized using the GPT-2 Byte Pair Encoding (BPE) tokenizer, which breaks down vocabulary into subword units, effectively handling rare words and setting the padding token as the end-of-sequence token. In the input processing section, the raw text is converted into token IDs, truncated or padded to a fixed length (set to 512), and an attention mask is created to identify valid content.

4.2.2 Model Training

The training process is based on the standard language model objective using autoregressive prediction. In the generation part, autoregressive generation is employed: a token is generated at each step based on the preceding sequence. Temperature sampling is applied to control the sharpness of the distribution, Top-k sampling is used to only consider the k highest probability candidates, Top-p (nucleus) sampling dynamically selects candidates whose probability exceeds a threshold, and a repetition penalty is applied to avoid text repetition. Texts of up to 200 tokens can be generated. The model trained for each author is saved after training is completed. The training process is as shown in the figure below.

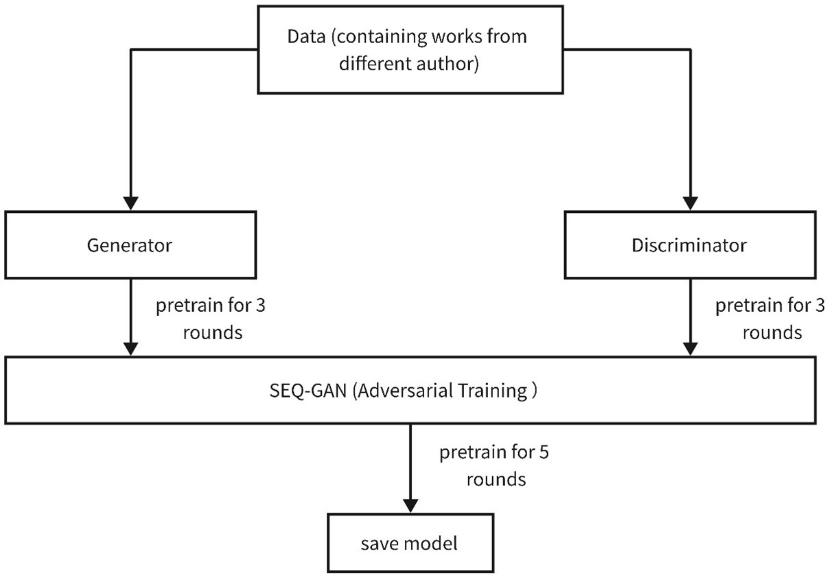


Figure 8: SeqGAN Training Process.

During the pre-training phase, both the generator and discriminator undergo three rounds of training. The generator in the pre-training phase already focuses on learning the specific target author's writing style from their texts. After pre-training is complete, a total of five rounds of adversarial training are carried out. The goal of adversarial training is to refine the generator's ability to produce text that the discriminator classifies as belonging to the target author's writing style.

In each round, the generator produces 20 texts, and during training, the temperature value gradually decreases from 1.0 (by 0.05 each epoch) and the Top-k value decreases from 50 (by 2 each epoch), encouraging the generator to transition from exploration to exploitation. The discriminator calculates the probability that each sample belongs to the target author, which serves as the reward value for that sample. The model with the highest average reward is saved as the best model. Thus, during the adversarial training process, the generator is guided to develop towards producing text with higher reward outcomes that better mimic the target author's style.

In this architecture, the discriminator's goal is to help the generator better produce the target author's writing style by evaluating generated samples. We once tried using a pre-trained discriminator model from `train.py` (which underwent more epochs of training) directly in our adversarial training, but this resulted in the discriminator being too strong. The highly trained discriminator was too critical of generated samples, providing consistently low rewards that failed to effectively guide the generator's improvement.

Therefore, in our final implementation, we use a more moderately trained discriminator from `trainseq.py` (with just 3 epochs of training) that provides more useful gradient signals during the adversarial training process.

4.2.3 Model Output

For each author in the dataset, we retain the best-performing model from the adversarial training in `trainseq.py` as the generator. When a generation request is received, the corresponding author's generator is called to generate a 200-token (approximately 100–150 words) text. To enhance the stability of the model's output, we configure it to automatically generate 10 samples per request and output the one with the highest score.

Additionally, the model itself is uploaded to Hugging Face and can be accessed remotely. It is important to note that evaluating the quality of generated text solely based on the discriminator's similarity score presents challenges; aspects like semantic coherence, creativity, and avoidance of repetitive patterns require additional qualitative assessment, which remains an area for future exploration.

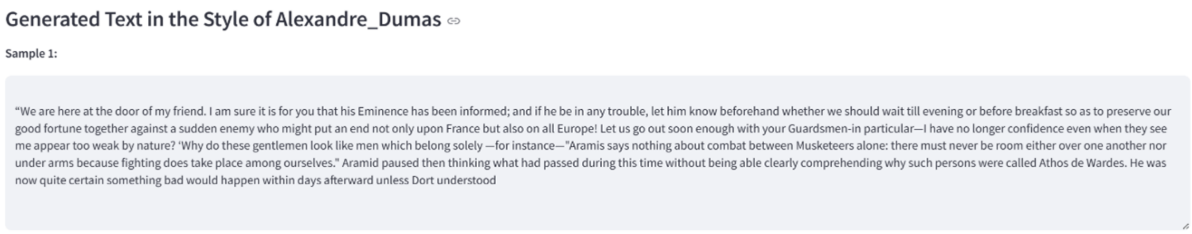


Figure 9: Example Generated Text.

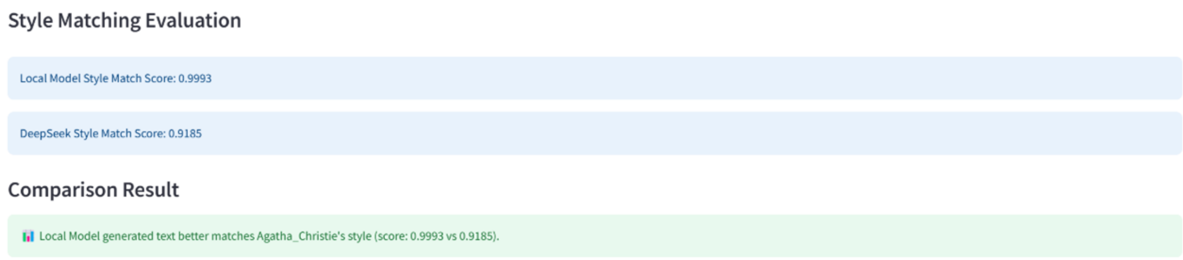


Figure 10: Style Matching Evaluation Interface.

4.3 API

4.3.1 DeepSeek API

Call the DeepSeek R1 model. Its API key is `sk-5026edabe797479492b0ed7c9d8ad0ca`.

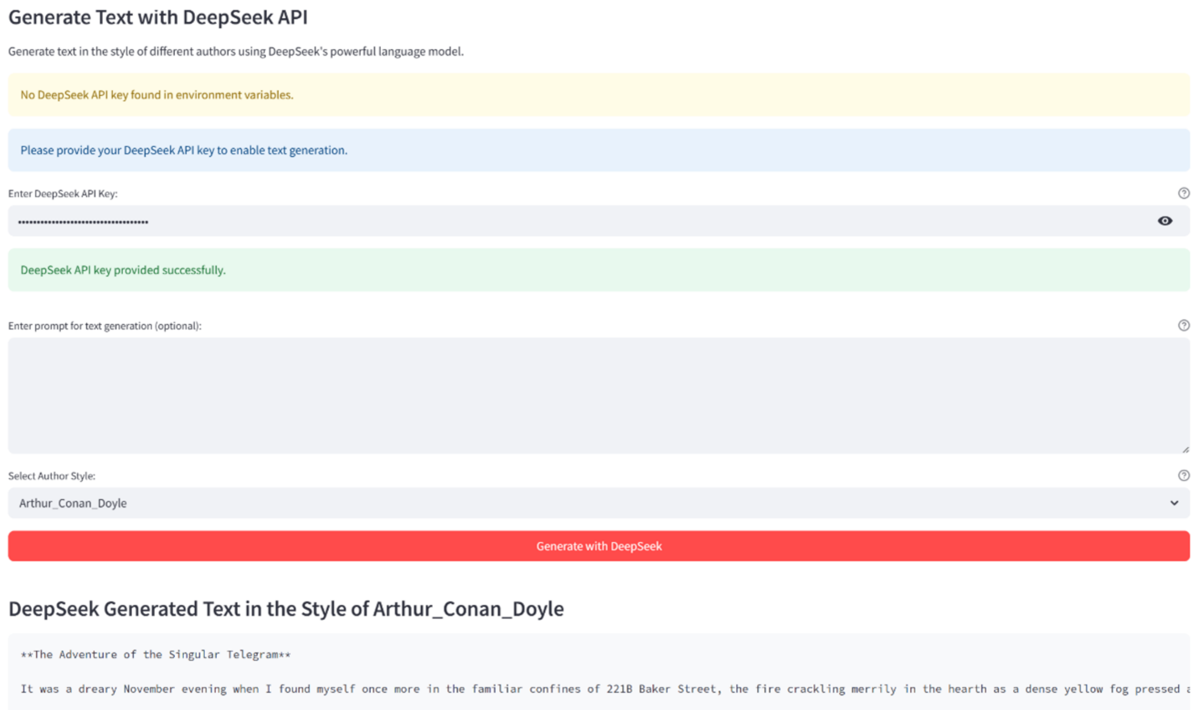


Figure 11: DeepSeek API Interface.

4.3.2 HuggingFace API

In the project, we uploaded the trained Identifier and Generator models to our personal Hugging Face repository for remote access. The code logic separates the tokens for identification and generation. Once the corresponding token is recognized, the Identifier and Generator models can be downloaded into the project folder individually using the `huggingface-cli`. For model loading, the code follows this priority order:

1. Attempt to load the model locally.
2. If the model does not exist locally, attempt to download it using `huggingface-cli`.
3. If the download fails, load the model directly from Hugging Face remotely.

4.4 User Interface Design

4.4.1 Chat Interface

In the chat interface, ask the user to define the role of the assistant, such as "Assistant for project analysis," "Business planning consultant," or "Technical support assistant," to ensure that the conversation stays focused on a specific task or objective. It is shown in the following image.

Figure 12: Chat Interface: Role Definition.

Afterwards, you can randomly select input tags from the 10 author name tags. Ultimately, two segments of text will be output: one generated by our own trained model, and the other generated by DeepSeek. The tags for inputting the authors' names are shown in the following image.

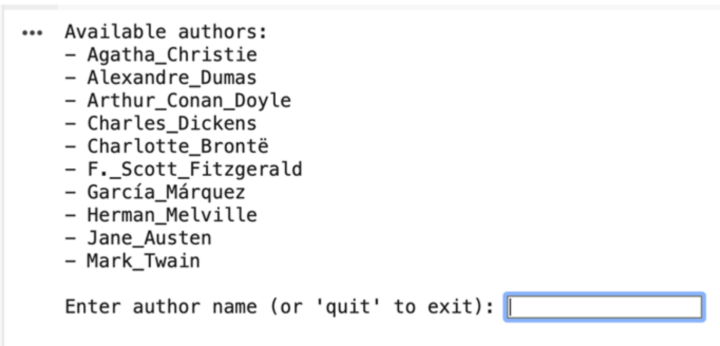


Figure 13: Chat Interface: Author Selection.

4.4.2 Graphical User Interface

The application features a Streamlit web interface organized into three main tabs: **Style Analysis**, **Text Generation**, and **About**, facilitating easy navigation between core functions.

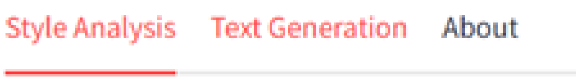


Figure 14: GUI Tabs.

The **Style Analysis** tab is dedicated to identifying the author style of a given text. Users can either paste text directly into the provided area or upload a text file.

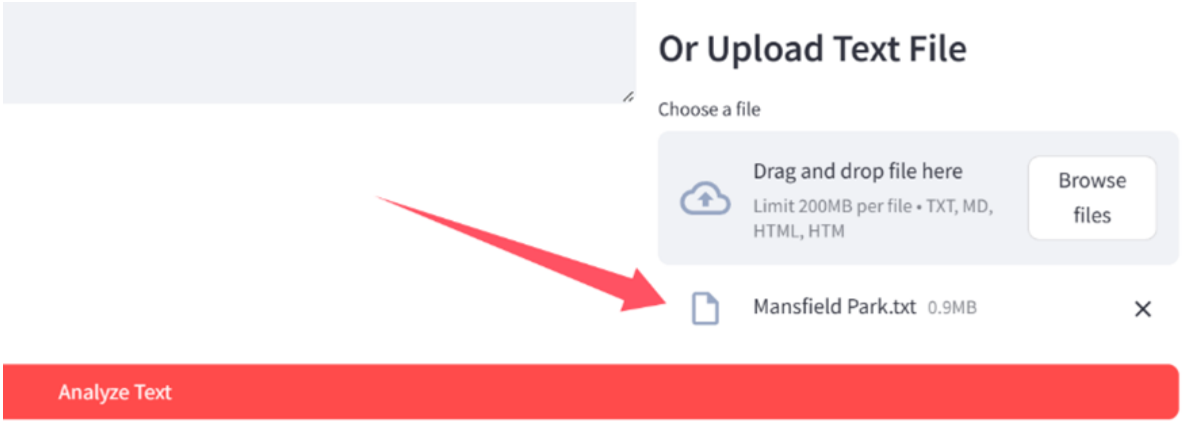


Figure 15: GUI: Style Analysis Tab.

An optional input field allows for the entry of a Hugging Face token, which can grant access to private models or help avoid rate limits; otherwise, public models are utilized by default. A key setting is the adjustable confidence threshold slider, which dictates the minimum probability needed for a specific author classification. Predictions falling below this threshold are labeled as "Unknown Author".

Clicking the "Analyze Text" button initiates the AuthorIdentifier and text processing. The results displayed include the predicted author, the associated confidence score, and, for longer texts, the number of chunks analyzed along with a breakdown of author distribution across these chunks. A comprehensive table lists the probabilities for all potential authors. Finally, details about the specific identification model used are presented.

The **Text Generation** tab centralizes text creation features and is further divided into three sub-tabs.

- The first, *Local Model Generation*, employs fine-tuned models, potentially hosted on Hugging Face, to generate text. It requires selecting an author style and permits an optional starting prompt. Users can configure the number of text samples and the maximum length using sliders. An optional Hugging Face token can also be provided. The generated text samples are then displayed.
- The second sub-tab, *DeepSeek Generation*, utilizes the DeepSeek API. This requires a DeepSeek API key, obtainable either from the `OPENAI_API_KEY` environment variable or direct user input. Users select the author style and can add an optional prompt before generating and displaying the text.
- The third sub-tab, *Style Comparison*, enables a direct comparison of outputs generated by the Local Model and DeepSeek for the same author. It shows the most recently generated texts from both sources side-by-side. If comparable texts exist, a "Compare Writing Styles" button activates a discriminator model to assess which text better aligns with the target author's style, presenting comparative scores.

Lastly, the **About** tab offers detailed information about the application. It covers the project's purpose, the underlying technologies (BERT, GPT-2, DeepSeek API, Streamlit, etc.), specific model identifiers, the list of supported authors, and general usage guidelines. It also provides instructions for configuring necessary API tokens through a `.env` file, displays the status of these configurations, and shows detailed metadata about the identification model if an analysis was previously run.

5 Performance

5.1 Identifier Performance

Test the accuracy of the identifier by selecting new articles, different from the training set, as the test set. For each author, select 20 text samples, each with about 80 words, totaling 200 samples as the test set, and observe the accuracy.

Table 2: Identifier Correct Identification Frequency

Author	Correct Identification Frequency	Correct Identification Percentage
Agatha Christie	20	100%
Alexandre Dumas	19	95%
Arthur Conan Doyle	19	95%
Charles Dickens	18	90%
Charlotte Brontë	12	60%
F. Scott Fitzgerald	11	55%
García Márquez	20	100%
Herman Melville	11	55%
Jane Austen	19	95%
Mark Twain	20	100%

Based on the data in the table above, it can be concluded that the identifier demonstrates high accuracy (90% and above) for seven of the ten authors. However, performance is notably lower for F. Scott Fitzgerald, Charlotte Brontë, and Herman Melville, with accuracy ranging from 55% to 60%. Several factors could contribute to this discrepancy. These authors might possess inherently more subtle or varied writing styles across their works, making a consistent "fingerprint" harder to capture, especially in short text samples. Stylistic variation between individual works by the same author is a known challenge in attribution. Additionally, the short length of the test samples (approx. 80 words) might provide insufficient stylistic evidence for reliable attribution, a common issue in stylometry. It's also possible that their styles share certain features with other authors in the dataset, leading to confusion, or that the training data for these authors contained more noise or inconsistencies compared to others. Forensic authorship attribution, in general, faces difficulties with limited text and potential stylistic overlap.

5.2 Generator Performance

Test the capabilities of the generator produced by ourselves by generating 3 texts for each author, totaling 30 texts. Then, use the identifier to calculate the confidence scores and compare the generated texts with the results from DeepSeek R1, as shown in the table below.

Table 3: Generator Performance (Our Model) - Average Confidence Scores

Author	Score 1	Score 2	Score 3	Average Score
Agatha Christie	0.9993	0.9993	0.9996	0.9994
Alexandre Dumas	0.9992	0.8934	0.9983	0.9636
Arthur Conan Doyle	0.9987	0.9934	0.9993	0.9971
Charles Dickens	0.9990	0.9992	0.9991	0.9991
Charlotte Brontë	0.9710	0.9189	0.9923	0.9607
F. Scott Fitzgerald	0.9989	0.9986	0.9993	0.9989
García Márquez	0.9913	0.9984	0.9982	0.9960
Herman Melville	0.9991	0.9942	0.9834	0.9922
Jane Austen	0.9989	0.9987	0.9989	0.9989
Mark Twain	0.9913	0.9905	0.9984	0.9934

Table 4: Generator Performance (DeepSeek R1) - Average Confidence Scores

Author	Score 1	Score 2	Score 3	Average Score
Agatha Christie	0.9992	0.8862	0.8237	0.9030
Alexandre Dumas	0.7321	0.8742	0.9916	0.8660
Arthur Conan Doyle	0.9703	0.7631	0.8733	0.8689
Charles Dickens	0.9021	0.8762	0.6782	0.8188
Charlotte Brontë	0.3421	0.1342	0.1745	0.2169
F. Scott Fitzgerald	0.0513	0.1032	0.0872	0.0806
García Márquez	0.3498	0.1362	0.2456	0.1657
Herman Melville	0.1452	0.0912	0.7651	0.3338
Jane Austen	0.0783	0.0713	0.1402	0.0966
Mark Twain	0.2963	0.1634	0.5263	0.3287

Overall, the texts generated by our generator perform very well, with confidence scores consistently high (average scores generally above 0.96). In contrast, DeepSeek R1 performs reasonably well only for the writing styles of Agatha Christie, Alexandre Dumas, and Arthur Conan Doyle, while its performance is notably poor for the other seven authors, particularly Fitzgerald, Austen, and García Márquez, where confidence scores are very low.

Qualitatively, this difference is apparent. For instance, generating text in the style of F. Scott Fitzgerald, our model might produce a snippet like: ”The twilight draped itself over the manicured lawns, a melancholic sigh in the humid air, hinting at promises whispered and inevitably broken under the careless gaze of the moon.” This attempts to capture Fitzgerald’s evocative, somewhat somber tone. DeepSeek R1, reflecting its lower confidence score for this author, might generate something more generic: ”The evening came over the green grass. The air was damp. People talked quietly as the moon watched.” While grammatically correct, it lacks the specific stylistic nuances the identifier associates with Fitzgerald.

In general, our self-developed small generator model, trained using a SeqGAN architecture , outperforms the highly capable large model DeepSeek R1 in replicating the specific writing styles of these authors, as measured by our identifier. However, it’s worth noting that strongly enforcing stylistic similarity can sometimes pose a trade-off with the overall coherence or naturalness of the generated text. Achieving the right balance between stylistic fidelity and semantic integrity remains a challenge in controlled text generation.

6 Conclusion and Future Work

In summary, the performance evaluation indicates that the Author Identifier achieves high accuracy for a majority of the authors tested, though challenges remain with authors exhibiting potentially more variable styles or when analyzing very short texts. The custom-trained Generator, despite its smaller scale, demonstrates superior capability in mimicking specific authorial styles compared to a large general-purpose model like DeepSeek R1, achieving high confidence scores from the Identifier.

6.1 Identifier

The Author Style Identifier system is designed to determine the authorship of text by analyzing unique writing patterns. It utilizes a machine learning approach, specifically a model based on the BERT architecture (`bert-base-uncased`), fine-tuned for a sequence classification task. Key aspects of its development and functionality include:

- **Model Training:** Employs a standard machine learning workflow, processing text from various authors. Uses a sliding window technique for long documents. BERT tokenizer performs tokenization.
- **Data Handling:** Addresses data imbalance by equalizing training samples. Introduces an ”Unknown Author” category trained on diverse corpora. Parameter tuning set the optimal ”Unknown” sample ratio to 0.5.
- **Architecture and Optimization:** Core model is `BertForSequenceClassification`. Incorporates best practices: AdamW optimizer, learning rate scheduling, early stopping, gradient clipping, GPU acceleration. Performance evaluated using accuracy, F1 score, confusion matrices.

- **Inference Implementation:** `AuthorIdentifier` class encapsulates the model. Handles model/tokenizer loading (local/Hugging Face Hub), API tokens, device detection. Analyzes text by chunking, predicting per chunk, applying a confidence threshold (default 0.6). Aggregates results for multi-chunk texts.

Based on the current system, future development can proceed in several directions. Immediate next steps could involve further parameter optimization or experimenting with alternative pre-trained models (e.g., RoBERTa, other transformer variants) to potentially improve accuracy, especially for the currently challenging authors. Expanding author coverage by sourcing more data, carefully incorporating it into training while managing balance, and retraining the model is another key priority. Longer-term goals include enhancing model explainability, perhaps by implementing functionality to highlight key phrases or stylistic features driving a prediction (e.g., attention maps) , and potentially adding qualitative style summarization based on input text.

6.2 Generator

In this project, we used GPT-2 as the foundation and adopted the SeqGAN architecture as our training strategy to develop a generator capable of producing text in the style of a specified author. Compared to some models that have not been fine-tuned, our generator achieved higher similarity scores when evaluated using our trained identifier. Of course, there is still room for optimization and improvement in the project. Due to the project's resource limitations, there is still significant room for improvement in the generator component. Since the model was trained on a team member's personal laptop using a GPU (RTX 4060), we were only able to use the smallest version of the GPT-2 model as the foundation for the generator.

In recent years, with the open-sourcing of various models like DeepSeek v3, better options for generators have become available. These cutting-edge generators feature more parameters and more innovative training methods, offering stronger guarantees in terms of the coherence and quality of generated text.

Unfortunately, models like DeepSeek v3 cannot be trained on personal laptops, and constraints in budget and time also prevented us from using platforms like Google Colab to rent higher-performance GPUs for training. This also limited our ability to train the model to mimic the writing styles of more authors. Further enhancements might include integrating improvements made to the discriminator (identifier) back into the fine-tuning process for the generator, potentially enhancing its performance further. Addressing data cleanliness remains an important area for future work; although we employed various cleaning methods, residual issues sometimes resulted in the generation of incorrect words or punctuation. While larger LLMs can mitigate some data quality problems , obtaining high-quality foundational text datasets is crucial. Due to time and resource constraints, progress here was limited.

Looking ahead, exploring larger base models (like more advanced GPT variants or other architectures) when resources permit is a clear path forward. Refining the training data and potentially incorporating more sophisticated controlled generation techniques could also yield improvements.

The development of sophisticated authorship identification and style generation technologies carries broader societal implications. Positively, these tools can enhance accessibility, aid literary and historical analysis, personalize content discovery, and assist in forensic investigations. However, potential negative impacts also warrant careful consideration, including the risk of misuse for generating disinformation or fake content, facilitating sophisticated plagiarism, raising privacy concerns through deanonymization, and perpetuating biases present in training data. Ethical development and deployment, alongside robust detection methods, will be crucial as these technologies continue to evolve.