Author Identifier

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

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.

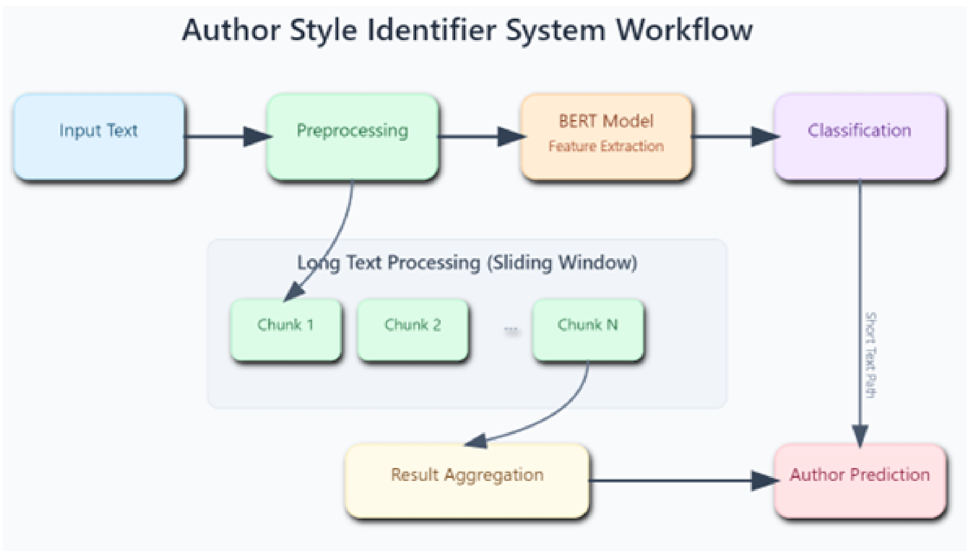
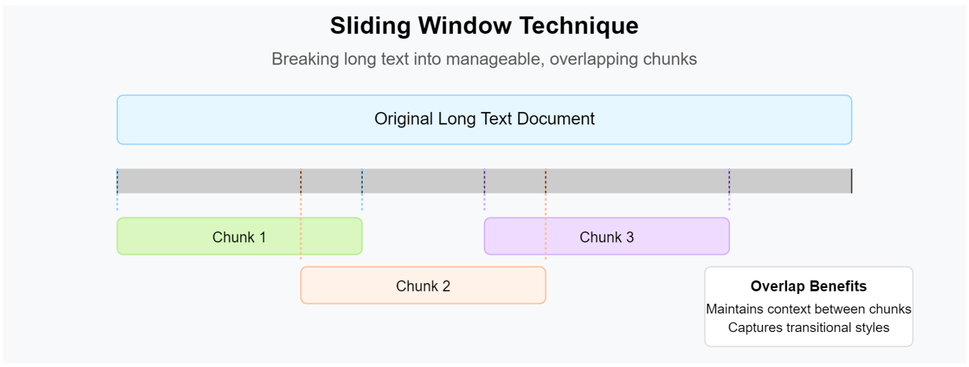


Figure 1: Author Style Identifier System Workflow.

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.



Sliding Window Technique
Breaking long text into manageable, overlapping chunks
Original Long Text Document
Chunk 1
Chunk 3
Chunk 2
Overlap Benefits
Maintains context between chunks
Captures transitional styles

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.

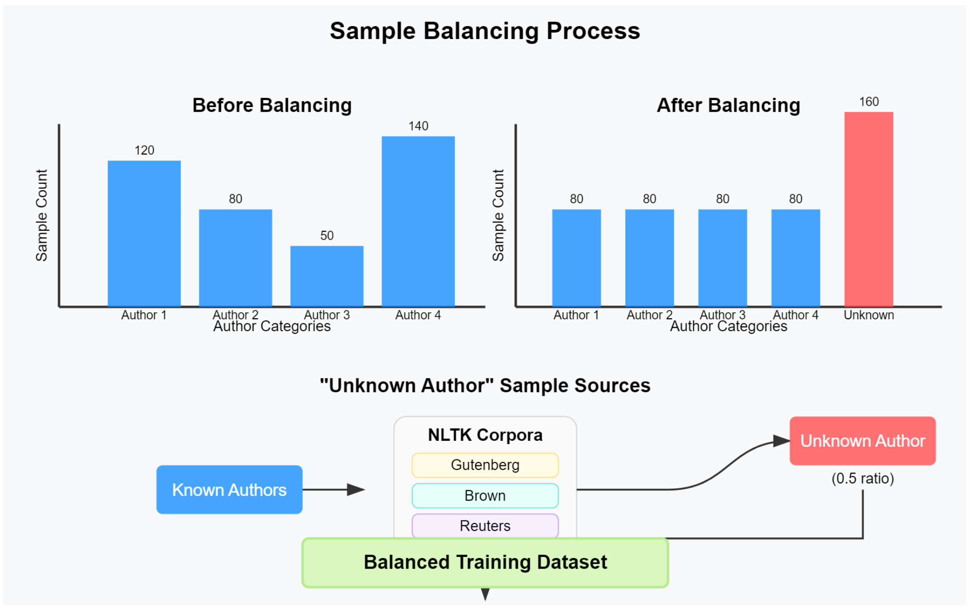
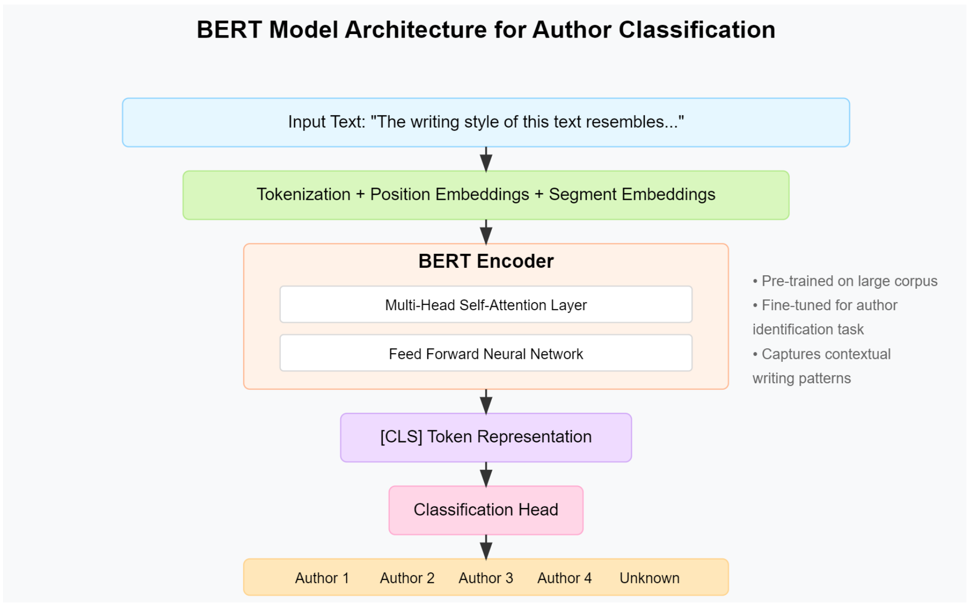


Figure 3: Sample Balancing Process.

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.



BERT Model Architecture for Author Classification
Input Text: "The writing style of this text resembles..."
Tokenization + Position Embeddings + Segment Embeddings
BERT Encoder
Multi-Head Self-Attention Layer
Feed Forward Neural Network
• Pre-trained on large corpus
• Fine-tuned for author identification task
• Captures contextual writing patterns
[CLS] Token Representation
Classification Head
Author 1
Author 2
Author 3
Author 4
Unknown

Figure 4: BERT Model Architecture for Author Classification.

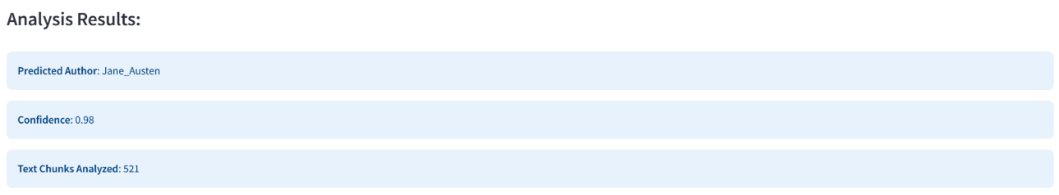
4.1.2 Parameter Tuning Strategy

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".

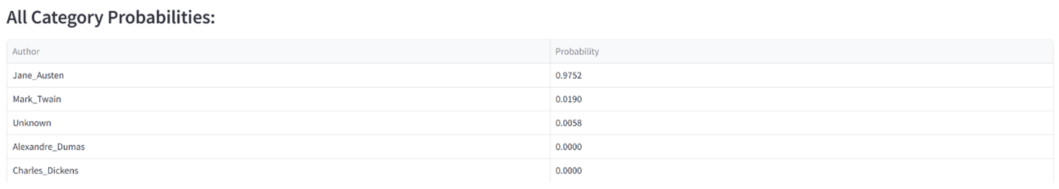

Analysis Results:

Predicted Author: Jane_Austen

Confidence: 0.98

Text Chunks Analyzed: 521

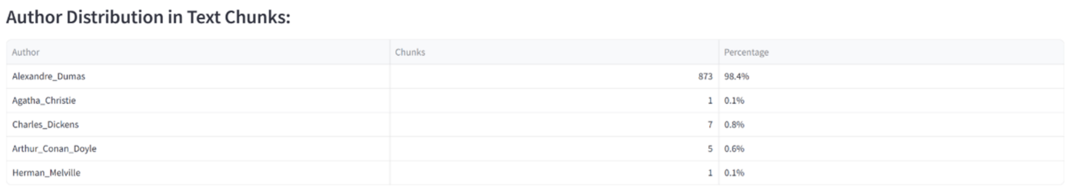
Figure 5: Example Analysis Results Interface (1).


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 *GPT-2* 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.

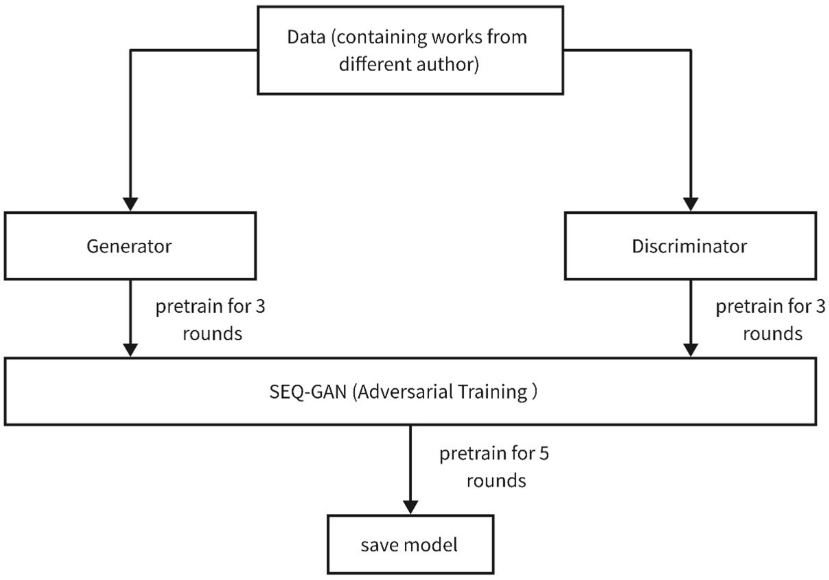


Figure 8: *SeqGAN* Training Process.

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.

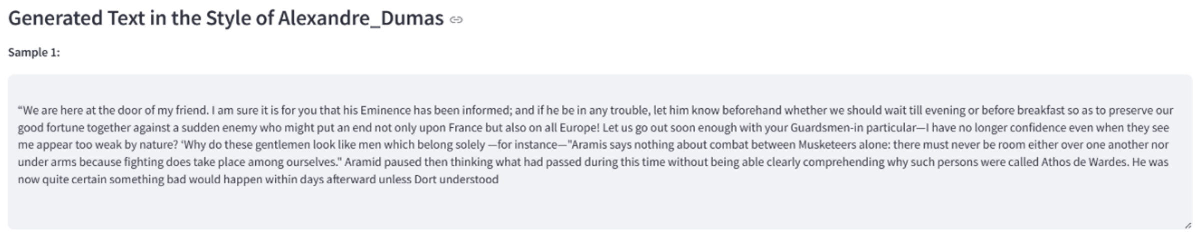
In this architecture, the discriminator's goal is to help the generator better produce the target author's writing style by evaluating generated samples. We once tried using a pre-trained discriminator model from `train.py` (which underwent more epochs of training) directly in our adversarial training, 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.

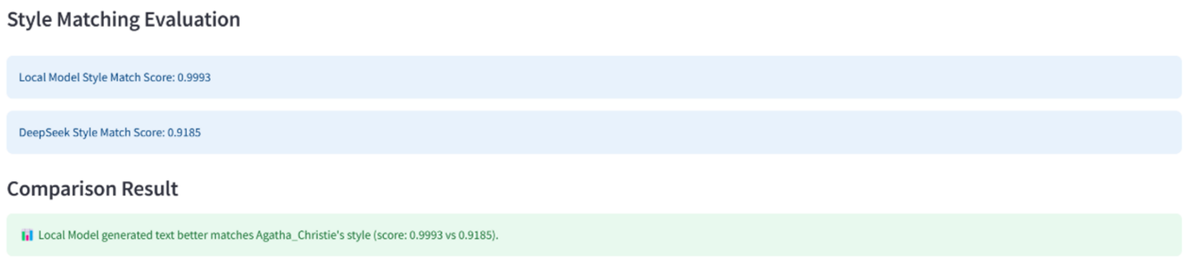


Generated Text in the Style of Alexandre_Dumas

Sample 1:

"We are here at the door of my friend. I am sure it is for you that his Eminence has been informed; and if he be in any trouble, let him know beforehand whether we should wait till evening or before breakfast so as to preserve our good fortune together against a sudden enemy who might put an end not only upon France but also on all Europe! Let us go out soon enough with your Guardsmen-in particular—I have no longer confidence even when they see me appear too weak by nature? 'Why do these gentlemen look like men which belong solely —for instance—"Aramis says nothing about combat between Musketeers alone: there must never be room either over one another nor under arms because fighting does take place among ourselves." Aramid paused then thinking what had passed during this time without being able clearly comprehending why such persons were called Athos de Wardes. He was now quite certain something bad would happen within days afterward unless Dort understood

Figure 9: Example Generated Text.



Style Matching Evaluation

Local Model Style Match Score: 0.9993

DeepSeek Style Match Score: 0.9185

Comparison Result

📊 Local Model generated text better matches Agatha_Christie's style (score: 0.9993 vs 0.9185).

Figure 10: Style Matching Evaluation Interface.

4.3 API Integration

The system leverages external APIs to enhance its text generation capabilities and provide comparative outputs. We integrate with the DeepSeek API and utilize Hugging Face for model deployment and access.

4.3.1 DeepSeek API Integration

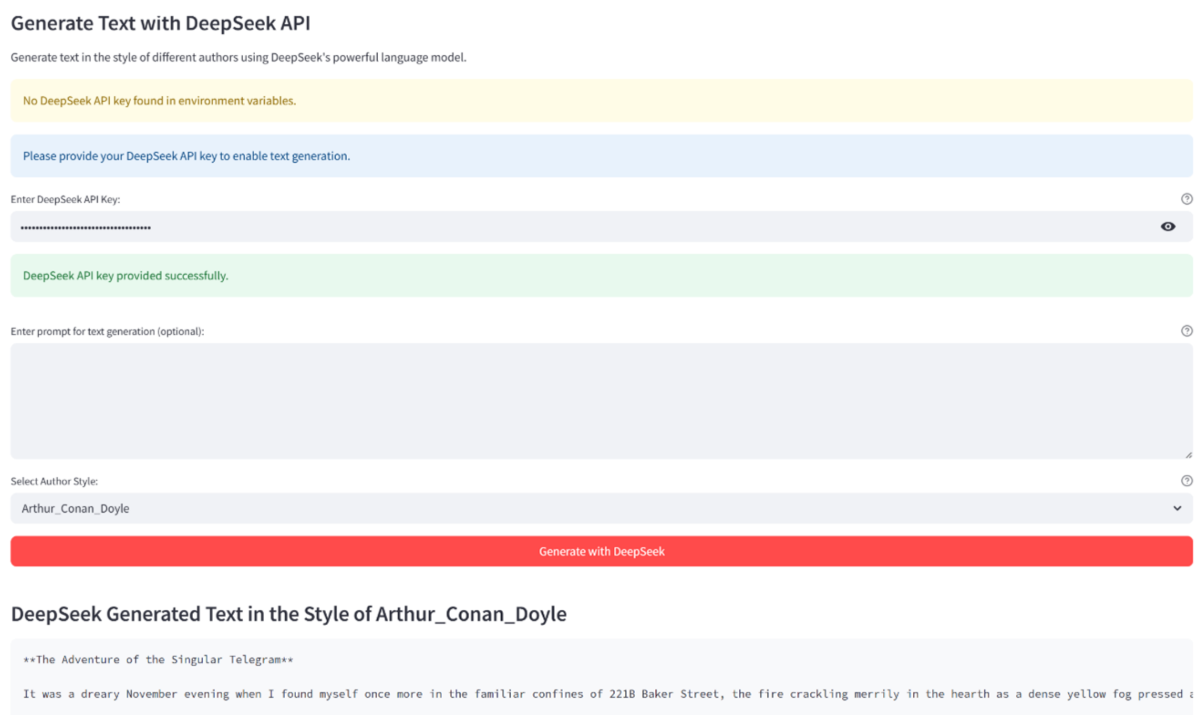
To incorporate advanced generative capabilities, the system interfaces with the *DeepSeek R1* large language model. Access is facilitated through a dedicated API endpoint, requiring authentication via a unique API key.

Authentication: Connection to the DeepSeek service is authenticated using the API key:

`sk-5026edabe797479492b0ed7c9d8ad0ca`.

This key must be securely managed and included in the request headers for all API calls.

Functionality: The API allows sending prompts (potentially including context or author style tags) and receiving generated text responses from the *DeepSeek R1* model. The specific parameters and request/response formats are defined by the DeepSeek API documentation. Figure 11 illustrates the conceptual interface for interacting with this API.


Generate Text with DeepSeek API

Generate text in the style of different authors using DeepSeek's powerful language model.

No DeepSeek API key found in environment variables.

Please provide your DeepSeek API key to enable text generation.

Enter DeepSeek API Key:

••••••••••••••••••••••••••••••••••••

DeepSeek API key provided successfully.

Enter prompt for text generation (optional):

Select Author Style:

Arthur_Conan_Doyle

DeepSeek Generated Text in the Style of Arthur_Conan_Doyle

```
**The Adventure of the Singular Telegram**

It was a dreary November evening when I found myself once more in the familiar confines of 221B Baker Street, the fire crackling merrily in the hearth as a dense yellow fog pressed
```

Figure 11: Conceptual Diagram of the DeepSeek API Interaction.

4.3.2 Hugging Face Model Deployment and API

For accessing our custom-trained models (Identifier and Generator), we utilize the Hugging Face platform. This provides a robust infrastructure for model storage, versioning, and remote access.

Model Repository: The trained Identifier and Generator models are uploaded to a private Hugging Face repository. This ensures controlled access while allowing the application to fetch the models as needed. Access requires appropriate authentication tokens associated with the Hugging Face account.

Access Tokens: Separate tokens are configured within the application logic – one specifically for identifying the desired model (Identifier or Generator) and another for authorizing the download or remote inference call.

Model Loading Strategy: To optimize performance and ensure resilience, the application employs a prioritized model loading sequence:

1. **Local Cache Check:** The system first attempts to load the required model (Identifier or Generator) from a local cache directory within the project folder. This provides the fastest access if the model has been previously downloaded.
2. **Download via CLI:** If the model is not found locally, the application attempts to download it from the Hugging Face repository using the `huggingface-cli` tool. This command-line interface handles the download process and places the model files in the designated local directory. Successful download populates the local cache for future use.
3. **Remote Loading (Fallback):** If both the local cache check and the download attempt fail (e.g., due to network issues or CLI errors), the application falls back to loading the model directly from the Hugging Face Hub remotely using the appropriate library functions (like those provided by the ‘transformers‘ library). This ensures functionality even if local storage or download mechanisms are unavailable, albeit potentially with higher latency.

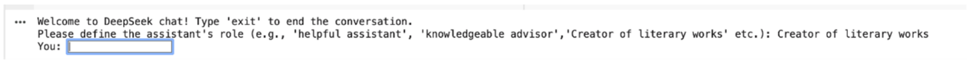
4.4 User Interface Design

The user interface is designed to be intuitive and task-oriented, guiding the user through the process of defining the interaction context and initiating text generation.

4.4.1 Chat Interface: Context Definition

The primary interaction point is the chat interface. To ensure focused and relevant outputs, the user is first prompted to define the context or role of the AI assistant.

Role Specification: Upon initiating a chat session, the user is presented with an input field or selection mechanism to specify the assistant's role. Examples include "Assistant for project analysis," "Business planning consultant," "Creative writing partner," or "Technical support assistant." This initial step helps tailor the subsequent interaction and model responses to the specific task domain. Figure 12 shows this initial role definition step.



```
Welcome to DeepSeek chat! Type 'exit' to end the conversation.
Please define the assistant's role (e.g., 'helpful assistant', 'knowledgeable advisor','Creator of literary works' etc.): Creator of literary works
You:
```

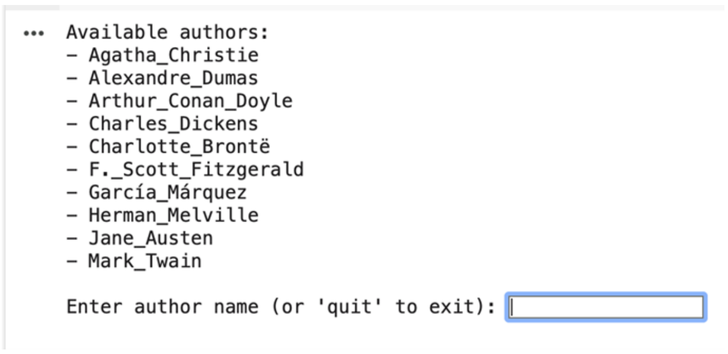
Figure 12: Chat Interface: Initial Role Definition Prompt.

Interaction Flow: Once the role is defined, the main chat area becomes active, allowing the user to input prompts or questions relevant to the chosen role.

4.4.2 Chat Interface: Author Style Selection and Generation

A key feature of the interface is the ability to influence the generated text style by selecting an author tag.

Author Tag Input: The interface provides a mechanism for selecting author style tags. As depicted in Figure 13, this could involve dropdowns, buttons, or a tag input field pre-populated with the available author names (derived from the 10 author name tags used during training). The system might randomly select one or allow the user to choose explicitly.



```
Available authors:
- Agatha_Christie
- Alexandre_Dumas
- Arthur_Conan_Doyle
- Charles_Dickens
- Charlotte_Brontë
- F._Scott_Fitzgerald
- García_Márquez
- Herman_Melville
- Jane_Austen
- Mark_Twain

Enter author name (or 'quit' to exit):
```

Figure 13: Chat Interface: Mechanism for Selecting Author Style Tags.

Comparative Generation: After receiving the user's prompt and the selected (or randomly assigned) author tag, the system initiates text generation using two different models:

1. **Custom Model Output:** The user prompt and author tag are processed by our custom-trained Generator model (loaded via the Hugging Face strategy described earlier).
2. **DeepSeek Model Output:** The user prompt (potentially adapted or augmented with style information based on the tag) is sent to the *DeepSeek R1* model via its API.

The interface then displays both generated text segments side-by-side or sequentially, allowing the user to compare the outputs from the custom model and the DeepSeek model.

4.4.3 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.

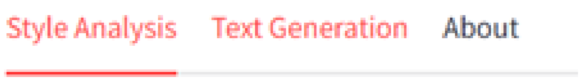


Figure 14: GUI Tabs.

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.

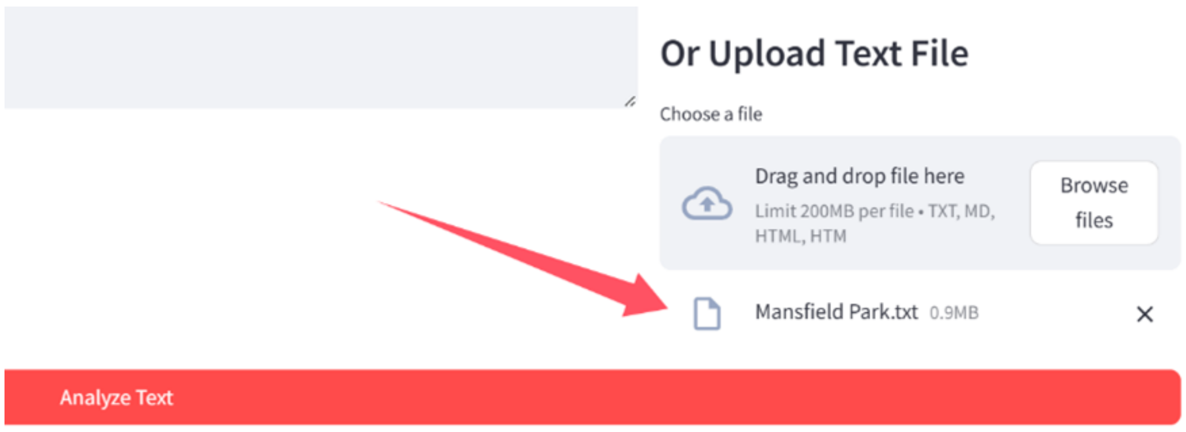


Figure 15: GUI: Style Analysis Tab.

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

This section details the performance assessment of the developed Authorship Identifier and Stylistic Text Generator models. The evaluation focuses on quantitative metrics derived from controlled experiments and provides an analysis of the observed results, contextualized within known challenges and advancements in the fields of stylometry and text generation.

5.1 Identifier Performance

The accuracy of the authorship identifier was evaluated using a dedicated test set comprising new articles distinct from the training data. For each of the ten authors included in the system, 20 text samples were selected. Each sample contained approximately 80 words, resulting in a total test set of 200 samples. The identifier's task was to assign each sample to its correct author.

The frequency and percentage of correct identifications for each author are presented in Table 2. This table serves as the primary quantitative basis for evaluating the identifier's effectiveness, breaking down performance on a per-author basis to reveal specific areas of strength and weakness. This granularity is essential for diagnosing potential issues related to specific authorial styles or model limitations.

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%

The results indicate strong performance for the majority of authors. Seven out of the ten authors were identified with high accuracy (90% or above). However, the identifier's performance was notably lower for Charlotte Brontë (60%), F. Scott Fitzgerald (55%), and Herman Melville (55%). This discrepancy warrants further investigation into potential contributing factors.

Several elements could explain the reduced accuracy for these specific authors. Firstly, these authors might possess inherently more varied writing styles across their different works, or their styles might be characterized by greater subtlety compared to the other authors in the dataset. Stylistic variation within a single author's oeuvre is a known challenge in authorship attribution, potentially making it difficult to establish a consistent stylistic "fingerprint" that generalizes well, especially when attributing previously unseen works. The model might overfit to the style of the specific texts included in the training set, leading to poor performance when tested on different works by the same author.

Secondly, the short length of the text samples used for testing (approximately 80 words) presents a significant challenge. Authorship attribution, particularly forensic attribution, becomes considerably more difficult as text length decreases because shorter samples provide less stylistic evidence. Many stylometric features rely on statistical patterns that only become stable and discriminative over longer stretches of text. Accurately attributing short, informal texts like tweets or forum posts is recognized as a particularly hard problem in the field.

Thirdly, it is possible that the writing styles of Brontë, Fitzgerald, and Melville share certain linguistic features with other authors within the dataset, leading to confusion and misclassification by the model. Furthermore, the quality of the training data cannot be discounted; potential noise, inconsistencies, or unrepresentative samples within the training data for these authors could have hindered the model's ability to learn a robust representation of their styles.

Beyond data and inherent stylistic challenges, the underlying architecture of the identifier, based on *bert-base-uncased* fine-tuned for sequence classification, may contribute to these limitations. Standard transformer models like BERT, while highly effective for many NLP tasks involving semantic understanding, achieve classification typically by aggregating information into a summary representation, often derived from the special `[CLS]` token's output embedding. This aggregation process, while suitable for capturing the overall meaning or topic, might inadvertently obscure or average out the fine-grained, sequential stylistic nuances that unfold across sentences and are potentially crucial for distinguishing authors with complex or variable styles, particularly within short text fragments. BERT's pre-training objectives, Masked Language Modeling (*MLM*) and Next Sentence Prediction (*NSP*), primarily equip it for semantic and local contextual understanding , potentially leaving it less sensitive to the subtle, longer-range stylistic dependencies or variations that characterize certain authors.

The model might excel at identifying local lexical or syntactic markers but struggle when the distinguishing features lie in the flow and structure across multiple phrases or sentences, a limitation exacerbated by the short sample length. This suggests that the identifier's specific failures on Brontë, Fitzgerald, and Melville are unlikely to be random but may point towards a systematic difficulty of the standard BERT classification approach in capturing stylistic variation and subtlety within limited textual data. Consequently, achieving significant improvements for these challenging authors might necessitate moving beyond simple fine-tuning towards approaches that explicitly model sequential stylistic patterns or hierarchical text structures.

5.2 Generator Performance

The capabilities of the custom-developed stylistic text generator were evaluated by generating three text samples for each of the ten target authors, resulting in a total of 30 generated texts. The authorship identifier, previously evaluated, was then used to assess these generated texts, calculating a confidence score indicating the likelihood that the text was written by the target author. These scores were compared against those obtained for texts generated by a large, general-purpose language model, *DeepSeek R1*, attempting to mimic the same authors.

Tables 3 and 4 present the confidence scores for the texts generated by the custom model and *DeepSeek R1*, respectively. These tables provide the quantitative basis for comparing the stylistic fidelity of the two generators, as measured by the identifier. The per-author breakdown reveals specific strengths and weaknesses in style mimicry for each generator.

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

The comparison reveals a significant difference in performance. The custom generator, developed using a *GPT-2* base model and trained with a Sequential Generative Adversarial Network (*SeqGAN*) architecture, achieved consistently high confidence scores across all authors, with average scores generally exceeding 0.96. In contrast, *DeepSeek R1* demonstrated reasonable performance only for Agatha Christie, Alexandre Dumas, and Arthur Conan Doyle. For the remaining seven authors, its performance was notably poor, particularly for F. Scott Fitzgerald, Jane Austen, and Gabriel García Márquez, where the average confidence scores were extremely low (below 0.17).

This quantitative difference is reflected qualitatively. For example, when tasked with generating text in the style of F. Scott Fitzgerald, the custom model might produce: ”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*, consistent with its low confidence score for this author, might generate more generic text like: ”The evening came over the green grass. The air was damp. People talked quietly as the moon watched.” While grammatically sound, this lacks the specific stylistic nuances that the identifier associates with Fitzgerald.

The superior performance of the custom generator can be attributed to its specialized training regimen. The *SeqGAN* architecture employs the authorship identifier as a discriminator, providing direct feedback to the generator. The generator, modeled as a reinforcement learning agent, updates its policy (i.e., its text generation strategy) to maximize the reward signal received from the discriminator. This adversarial process compels the generator to specifically optimize for producing text that the identifier recognizes as stylistically similar to the target author. This contrasts sharply with large, general-purpose models like *DeepSeek R1*, which are pre-trained on vast datasets for broad language understanding and generation capabilities but lack this targeted, author-specific adversarial fine-tuning unless explicitly configured for it. The high confidence scores achieved by the custom generator demonstrate its success in learning to mimic the identifier’s perception of each author’s style, potentially even learning to exploit any biases or specific features the identifier weighs heavily.

However, this strong focus on stylistic fidelity, enforced by the *SeqGAN* training loop, can introduce a trade-off with other desirable qualities of generated text, notably global coherence and semantic naturalness. Achieving deep logical consistency and a natural flow of ideas across sentences and paragraphs requires modeling complex, long-range discourse structures. Standard pre-training objectives, such as the causal language modeling used for *GPT-2* or the *MLM*/*NSP* tasks for BERT-like models, may not inherently equip models with the sophisticated capabilities needed to capture these structures effectively, especially when the generation process is heavily constrained by optimizing a specific stylistic reward signal. The *SeqGAN* framework itself, often providing rewards based on the assessment of complete sequences, might struggle to implicitly enforce paragraph-level coherence or logical progression. This challenge is likely amplified by the use of a relatively small base model (*GPT-2* small), which has limited capacity for modeling long-range dependencies compared to larger architectures. The combination of limited model capacity and the intense optimization pressure towards stylistic mimicry from *SeqGAN* may leave insufficient ”bandwidth” for learning and maintaining complex coherence structures. Therefore, generating text that is simultaneously stylistically accurate and semantically sound with logical flow remains a significant challenge in controlled text generation, particularly under resource constraints. Addressing this may require approaches that explicitly target coherence alongside style during training.

6 Conclusion and Future Work

In summary, the performance evaluation indicates that the Author Identifier system achieves high accuracy for a majority of the authors tested. However, challenges persist, particularly when dealing with authors who exhibit potentially more variable styles across their works or when analyzing very short text samples. These challenges highlight potential limitations in how standard transformer architectures capture sequential stylistic nuances. The custom-trained Generator, despite its smaller scale (*GPT-2* base) and resource-constrained training environment, demonstrates a superior capability in mimicking specific authorial styles compared to the large general-purpose model *DeepSeek R1*, as measured by the high confidence scores awarded by the Identifier. This success is attributed to the targeted optimization provided by the *SeqGAN* training framework. Nevertheless, this intense focus on style can lead to a trade-off with the global coherence and naturalness of the generated text, a challenge exacerbated by the base model's size.

6.1 Identifier

The Author Style Identifier system, based on a fine-tuned *bert-base-uncased* model for sequence classification, successfully identifies authorship for many cases but shows limitations linked to stylistic variability, short text length, and potentially the inherent information processing mechanisms of standard transformer architectures. Based on the current system and the performance analysis, future development can proceed along several strategic directions.

Immediate Steps:

- **Parameter Optimization:** Conduct further systematic hyperparameter tuning (e.g., learning rates, batch sizes, optimizer parameters like *AdamW* settings, scheduler types) to potentially improve performance, especially for the currently challenging authors.
- **Alternative Pre-trained Models:** Experiment with alternative transformer architectures known for potentially capturing contextual information differently or more robustly, such as *RoBERTa*, *DeBERTa*, or *ELECTRA*, contingent on available computational resources.
- **Data Expansion and Refinement:** Expand the dataset to include more authors and potentially more diverse texts (different genres, periods) from existing authors. Careful data augmentation strategies could be employed. Crucially, maintain rigorous procedures for managing class balance and continue optimizing the ratio and composition of the "Unknown Author" category, followed by model retraining.

Architectural/Methodological Enhancements (Resource-Conscious): The analysis suggests that the identifier's struggles with certain authors and short texts might stem from limitations in capturing sequential stylistic patterns using standard BERT classification heads relying on aggregated representations like the `[CLS]` token. To address this specifically, while being mindful of computational resources, the following enhancements are proposed:

- **Hierarchical Modeling:** Explore architectures that process text hierarchically. One approach involves encoding smaller text units (e.g., sentences or fixed-size chunks) individually using the existing fine-tuned BERT encoder (or potentially a frozen base BERT to minimize computational overhead during this stage). The resulting sequence of unit-level embeddings would then be fed into a lightweight sequential model – such as an *LSTM*, *GRU*, or a small Transformer layer – responsible for the final author classification.
  Rationale: This method explicitly models the sequence and evolution of stylistic features across text units, potentially capturing patterns missed by single-pass aggregation. It directly addresses the hypothesized weakness in handling sequential stylistic information and could improve robustness for authors with variable styles or for short texts where the sequence of features is critical. Employing a lightweight second-stage model ensures computational feasibility. This aligns conceptually with research on hierarchical text classification and using sentence embeddings as input to sequence models.
- **Auxiliary Training Objectives:** Investigate the use of auxiliary learning tasks during the fine-tuning process. Instead of solely optimizing for the primary authorship classification loss, the model could be simultaneously trained on a secondary task designed to enhance its sensitivity to relevant stylistic features. For instance, an auxiliary task could involve predicting the correct order of shuffled sentences within a stylistically consistent text block (adapting the concept of *NSP* for stylistic coherence) or predicting specific stylistic markers if they can be reliably identified and labeled.
  Rationale: Auxiliary tasks are known to improve model generalization and representation learning by

providing additional learning signals. A task focused on sequence or discourse properties, tailored to style, could compel the BERT encoder to pay closer attention to the sequential patterns hypothesized as weak points in the current setup. This approach offers potential improvements without necessitating major architectural changes or significantly larger models, making it a resource-efficient option.

These architectural and methodological enhancements represent targeted strategies to address the specific limitations observed in the performance evaluation, focusing on improving the model's ability to capture sequential and variable stylistic information, which appears crucial for the challenging authors and short text scenarios encountered.

Longer-Term Goals:

- **Explainability:** Implement techniques to enhance model interpretability, such as visualizing attention maps or extracting feature importance scores (e.g., using *LIME* or *SHAP*), to better understand why the model makes certain predictions and identify the linguistic features it relies upon.
- **Qualitative Summarization:** Explore adding functionality to provide qualitative summaries of the detected writing style based on the input text, moving beyond simple classification.

6.2 Generator

The custom text generator, built upon *GPT-2* and trained using *SeqGAN*, demonstrated strong performance in mimicking authorial styles as judged by the identifier, significantly outperforming a large baseline model. However, this came at the cost of potential trade-offs in text coherence, exacerbated by the small base model size and resource limitations during training (personal laptop GPU preventing use of larger models like *DeepSeek v3* or extensive hyperparameter sweeps). Future work should aim to maintain stylistic fidelity while improving coherence and overall text quality.

Immediate Steps:

- **Discriminator Integration:** If the identifier (discriminator) is improved through the future work outlined above, integrate the enhanced version back into the *SeqGAN* training loop. A more accurate discriminator could provide better guidance to the generator.
- **Data Quality:** Intensify efforts on data cleaning and preprocessing for the training corpora. Residual issues in the data can lead to the generation of incorrect words or awkward punctuation. While larger models might be more robust to noisy data, high-quality foundational text remains crucial for optimal performance.
- **Resource-Permitting Scaling:** When computational resources (more powerful GPUs, cloud platforms like Google Colab) become available, explore training with larger base models (e.g., larger *GPT-2* variants, other architectures like *DeepSeek v3*). Larger models possess greater capacity to capture complex language patterns, potentially improving both style and coherence.

Architectural/Methodological Enhancements (Resource-Conscious): Given the current resource constraints, significant improvements in coherence might be achievable through more sophisticated training methodologies and potentially minor architectural adjustments, rather than solely relying on scaling the base model size. The style-coherence trade-off observed likely stems from the limited capacity of *GPT-2* small combined with *SeqGAN*'s primary focus on optimizing the stylistic reward signal from the discriminator.

- **Improving Coherence Modeling (Hierarchical Approaches):** Investigate hierarchical generation techniques, adaptable even to the smaller *GPT-2* base model. This could involve strategies like generating a high-level outline or plan first, and then generating sentences conditioned on this plan as well as preceding context. Architectures inspired by hierarchical text classification might also be relevant.
  Rationale: By explicitly modeling the text structure at a higher level, such approaches provide better scaffolding for maintaining topic consistency and logical flow over longer passages. This could compensate for the base model's limitations in handling long-range dependencies and mitigate the tendency of *SeqGAN* to prioritize local stylistic mimicry over global coherence.
- **Coherence-Focused Training Objectives (Auxiliary Learning):** Explore augmenting the *SeqGAN* training objective with auxiliary loss terms specifically designed to reward text coherence. This could involve incorporating metrics related to semantic similarity between adjacent sentences, penalizing abrupt topic shifts, training the model to predict sentence order within its own generated text, or even using a separate pre-trained coherence scoring model to provide an additional reward signal alongside the discriminator's

style score.
Rationale: The standard *SeqGAN* reward primarily reflects stylistic similarity as judged by the discriminator. Adding an explicit objective for coherence provides a direct learning signal for this property, helping the generator balance the competing demands of style and sense. This offers a pragmatic way to improve coherence without immediate reliance on much larger, resource-intensive base models, making it a viable path forward within existing constraints.

Refining Generation Control: Beyond *SeqGAN*, explore other controlled text generation techniques that might offer finer control over balancing stylistic attributes with content fidelity and coherence.

These enhancements focus on augmenting the training process and potentially the generation architecture to explicitly address the observed coherence limitations, offering resource-conscious pathways to improve the generator's output quality beyond simply scaling the model size.

6.3 Broader Societal Implications

The development of sophisticated technologies for authorship identification and style generation carries significant societal implications that extend beyond technical performance. On the positive side, these tools hold potential for enhancing accessibility, aiding literary and historical analysis by uncovering stylistic patterns or resolving authorship disputes, personalizing content discovery platforms, and assisting in forensic linguistics and investigations.

However, the potential for negative impacts necessitates careful consideration and proactive mitigation strategies. The ability to accurately mimic writing styles raises concerns about the generation of convincing disinformation, sophisticated forms of plagiarism, or the creation of fake content attributed to specific individuals or organizations. Authorship identification tools, while useful for attribution, could also be misused for deanonymization, potentially compromising the privacy of individuals who wish to write anonymously or pseudonymously. Furthermore, like many machine learning models, both identifiers and generators are susceptible to inheriting and potentially amplifying biases present in their training data, leading to unfair or skewed outcomes.

Therefore, the continued development and deployment of these technologies must proceed ethically. This includes prioritizing transparency, fairness, and accountability in model design and use; investing in robust methods for detecting generated or manipulated text; and engaging in ongoing dialogue about the societal consequences and appropriate governance frameworks for these powerful tools.