Literature Sources Analysis Using Deep Impostor Approach

Presented by: Ofir Uziel, Emanuel Davidov

Supervisor: Prof. Zeev Volkovich, Dr. Renata Avros

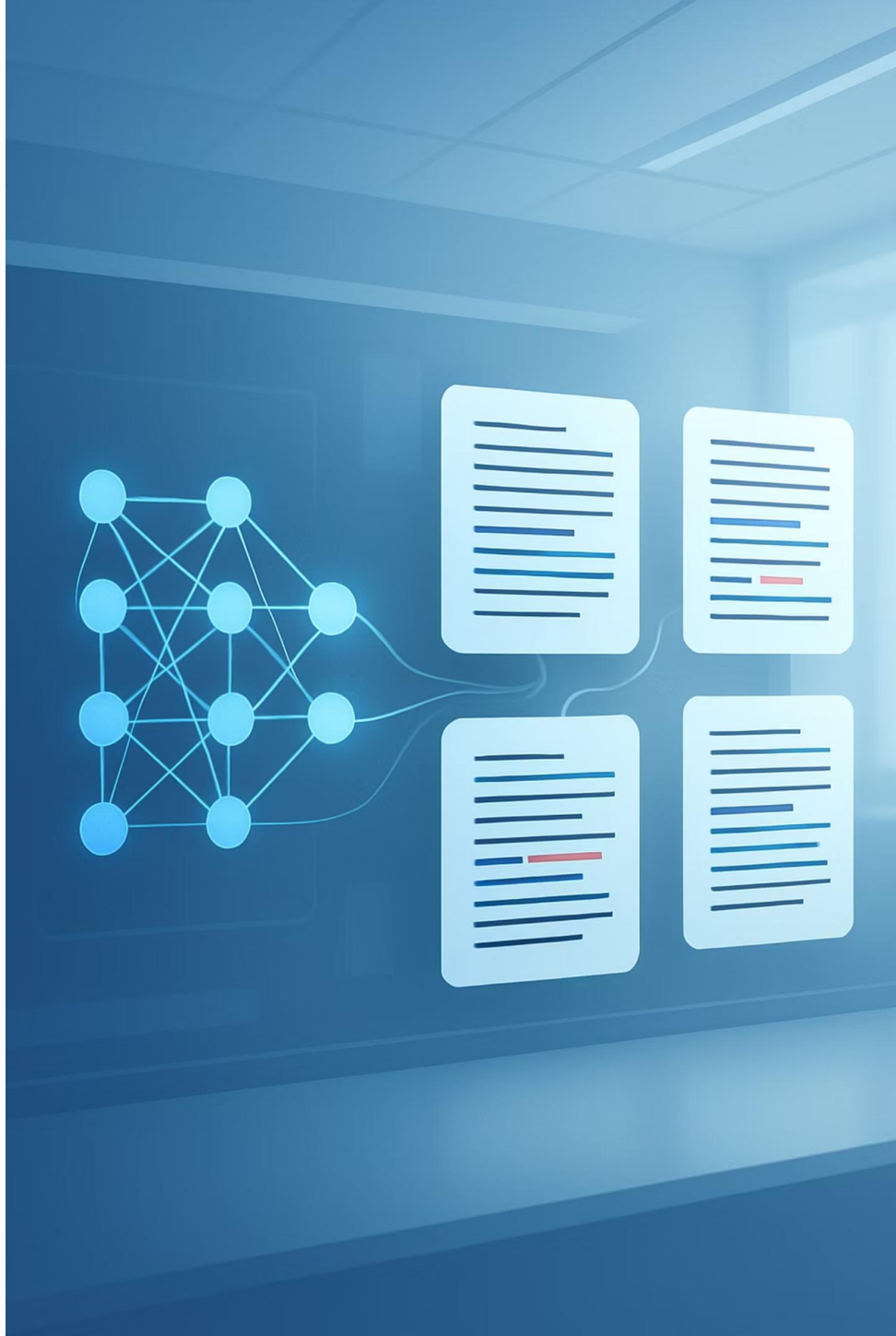


Table of Contents

Problem & Solution

Problem Definition, Project Purpose, Proposed Solution Overview

System Architecture

Detailed System Architecture, Data Preparation, Embedding Generation

Models & Implementation

Classification Models, Signal Construction, Boundary Detection, UML Diagrams, Implementation Workflow

Evaluation & Conclusion

Testing Plan, Innovation, Applications, Evaluation Methodology, Summary and Future Directions

Problem Definition

1 Traditional Assumptions

Current methods like stylometry, authorship attribution models, or n-gram analysis typically assume that each document has a single, consistent author. This makes them blind to collaborative works or sections written by editors, causing misleading conclusions.

2 Detection Challenges

Most algorithms classify entire documents, but stylistic shifts can happen within them. for example, between a letter's introduction and its body, or when content is copied from multiple sources. These internal boundaries go undetected.

3 Technical Limitations

Existing approaches rely on labeled data (known authorship), which is rare. They also produce opaque results that lack interpretability, especially when used on unseen or composite texts.

Project Purpose

Unsupervised Framework

Train classifiers using unrelated “impostor” corpora, without labeled examples from the target document. This makes the method flexible and domain-independent.

Alternation Point Detection

Identify where the writing style changes inside a single document. These shifts may indicate author boundaries, edits, or insertions.

Noise Filtering

Remove small fluctuations or false stylistic deviations using aggregation and signal smoothing, so only major stylistic boundaries are detected.

Interpretable Visualization

Provide visual plots of stylistic signals over the document timeline, clearly marking detected boundaries for human inspection.

Our goal is to build a system that can detect internal stylistic boundaries in documents, without relying on pre-labeled data. By training deep classifiers on unrelated impostor texts and analyzing classifier behavior across the document, we can detect stylistic shifts, filter noise, and visualize transitions to support forensic or editorial analysis.

Proposed Solution Overview

1 Deep Impostor Method

Based on the method introduced by Avros and Volkovich, we train a deep neural model to distinguish between two unrelated “impostor” corpora. This classifier learns a strong stylistic boundary, which we then use to evaluate each chunk of the target document and track how its style shifts over time.

2 Stylistic Classification

Apply the trained impostor classifier to batches from the target document. Each batch receives a probability score indicating its alignment with one of the two impostors.

3 Signal Generation

Aggregate batch predictions into chunk-level scores, forming a 1D signal representing how stylistic alignment changes across the document.

4 Multiview Detection

Repeat the entire process across many impostor pairs. Apply statistical changepoint detection (including wavelet transforms), and confirm stylistic boundaries only where multiple classifiers agree.

Detailed System Architecture

1 Text Segmentation

Split documents into fixed-length batches, then group batches into larger chunks. Each chunk will become a unit of stylistic analysis.

2 Word Embedding

Convert text batches to vector representations using either Word2Vec or BERT, depending on the classifier.

3 Stylistic Classification

Train a binary classifier on each impostor pair. Apply it to every chunk in the target document to produce a stylistic probability score.

4 Signal Creation Analysis

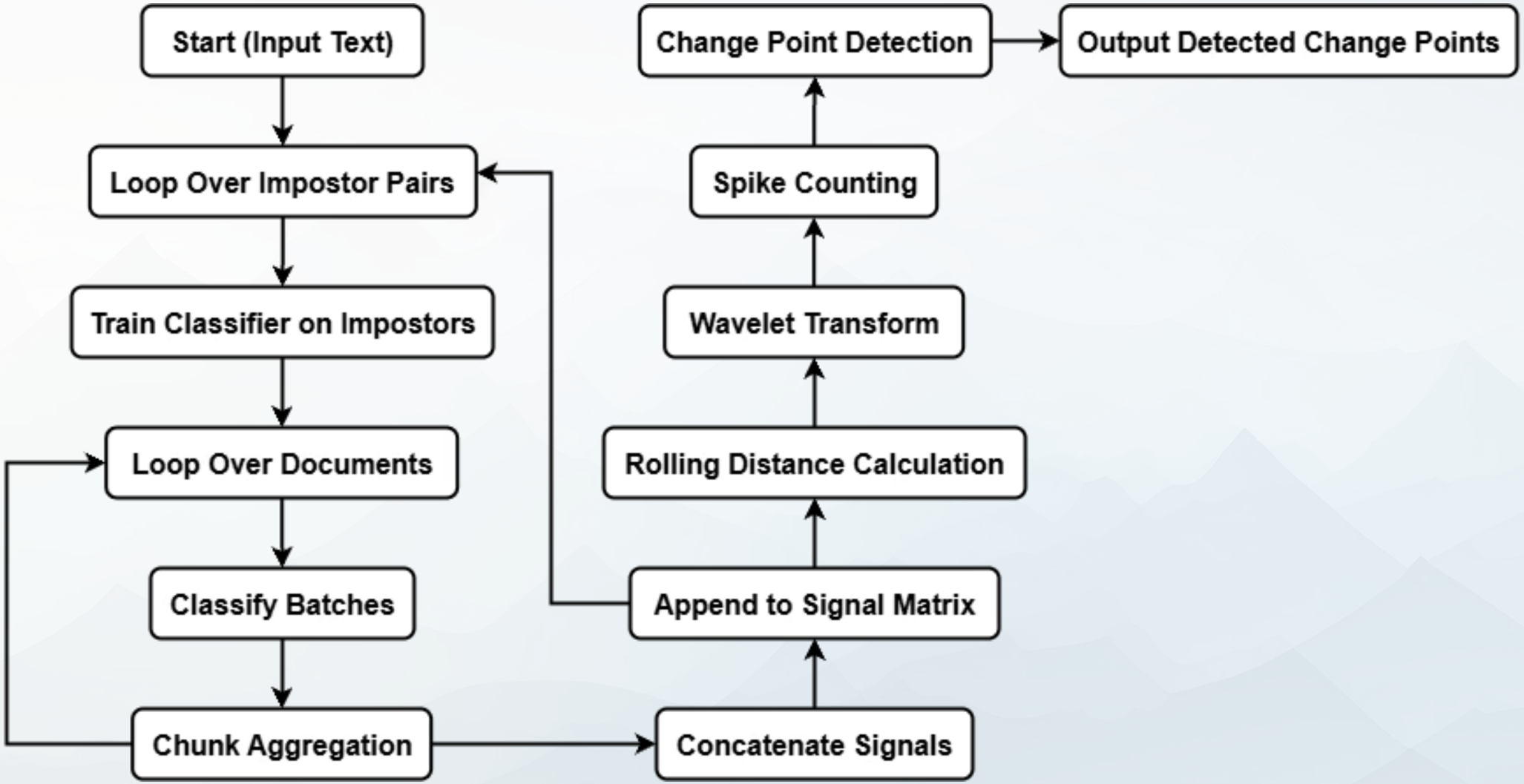
Combine the per-chunk classifier outputs to form a signal representing stylistic alignment over the document. Apply rolling distance and wavelet transform to identify stylistic shifts.

5 Boundary Detection

Run multiple classifiers using different impostor pairs. Use voting to detect consistent peaks across models. Only peaks with broad agreement are marked as final stylistic boundaries.

The system architecture integrates multiple components working in sequence to process text, generate meaningful representations, and identify stylistic boundaries through sophisticated analysis techniques.

General Model FlowChart



Data Preparation

Document Input

Raw text documents

Tokenization

Fixed-length batches (e.g. 50 words)

Chunking

Several batches per chunk (e.g. 400 words)

The data preparation process ensures consistent, structured input for the stylistic modeling system. By breaking documents into standardized units, we create uniform segments that can be effectively analyzed for stylistic patterns and variations.

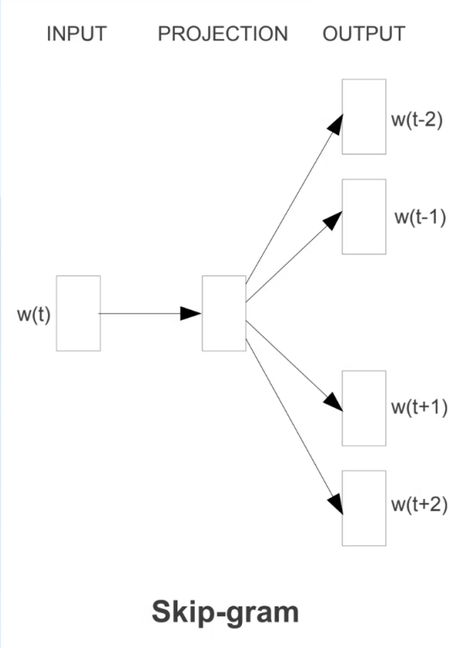
This systematic approach allows for more reliable comparison across different parts of a document, enabling the detection of subtle stylistic shifts that might indicate changes in authorship or significant editorial intervention.

Embedding Generation

Word2Vec (Skip-Gram)

A computationally efficient embedding technique that captures lexical-level features by predicting context words from target words.

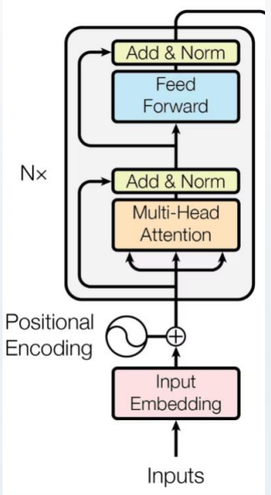
- Captures word relationships
- Efficient processing
- Strong lexical representation



BERT (Transformer)

A context-sensitive embedding approach that captures sentence-level semantics through bidirectional attention mechanisms.

- Context-aware representations
- Captures semantic nuances
- Pre-trained on massive corpora

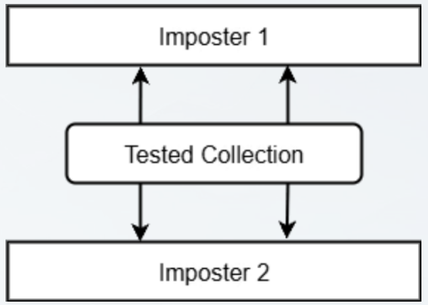


Both embedding techniques transform text into numerical vectors that represent semantic and stylistic properties. These vectors are then used as inputs to our classification models, which learn to detect stylistic differences. Together, Word2Vec and BERT ensure broad coverage of both surface-level and deep semantic stylistic cues.

Stylistic Classification Models

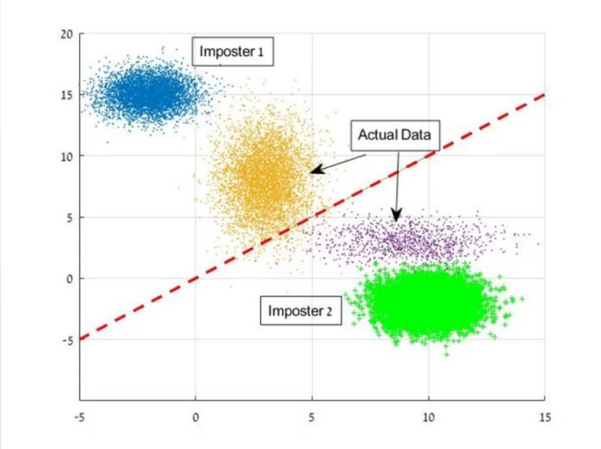
CNN-BiLSTM Model

Combines convolutional neural networks to extract local textual patterns with bidirectional LSTM networks to preserve long-range dependencies. This hybrid architecture effectively captures both n-gram features and sequential context.



BERT Classifier

Leverages transformer architecture for contextual embeddings and classification. Excels at detecting nuanced stylistic differences through pre-trained language understanding and targeted fine-tuning for style classification.



Both models are trained to distinguish between two impostor corpora. Once trained, they are applied to the target document to assess how each text segment aligns stylistically with either impostor, producing a signal of stylistic variation.

Stylistic Signal Construction

1 Batch-Level Classification

Generate probability scores for each batch against impostor corpora

2 Chunk-Level Aggregation

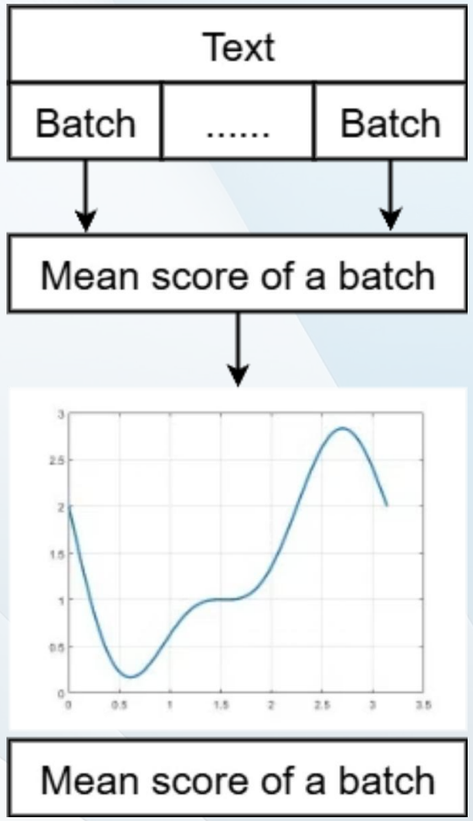
Combine batch probabilities into chunk scores

3 Continuous Signal Generation

Create a continuous signal reflecting stylistic alignment

4 Signal Interpretation

Analyze the signal for stylistic consistency and identify sharp transitions. These shifts indicate candidate change points later validated through statistical analysis.



Changepoint Detection

Rolling Distance Calculation

Compute average squared stylistic differences between consecutive chunks to identify potential transition points in the document's stylistic flow.

Changepoint Identification

Apply a Discrete Wavelet Transform (DWT) to the rolling distance signal to extract multi-scale detail coefficients. Identify local maxima in these coefficients as statistically significant stylistic shifts.

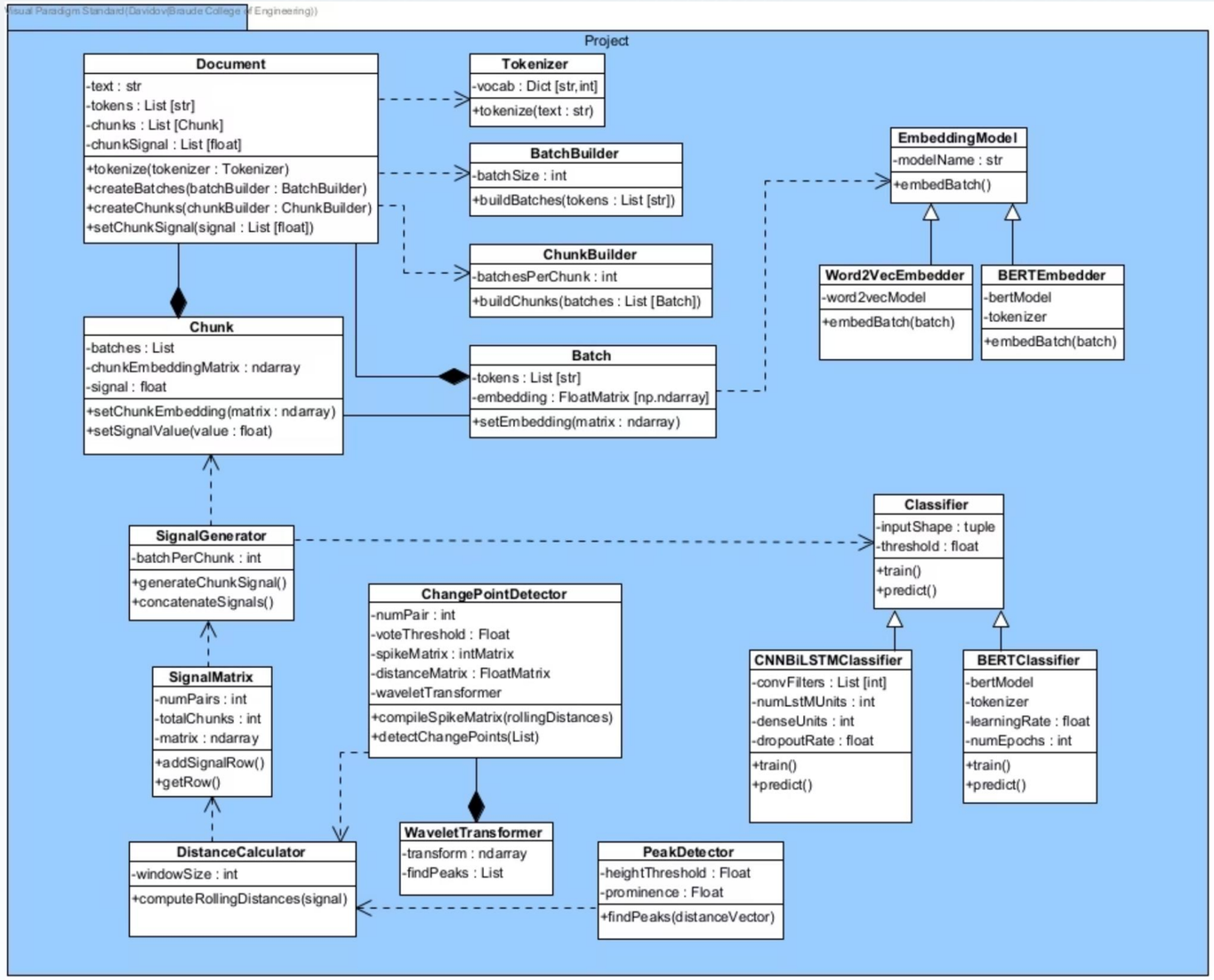
Consensus Voting

Apply a majority voting mechanism across multiple classifier outputs to validate peaks and ensure reliable, robust detection of genuine stylistic changes.

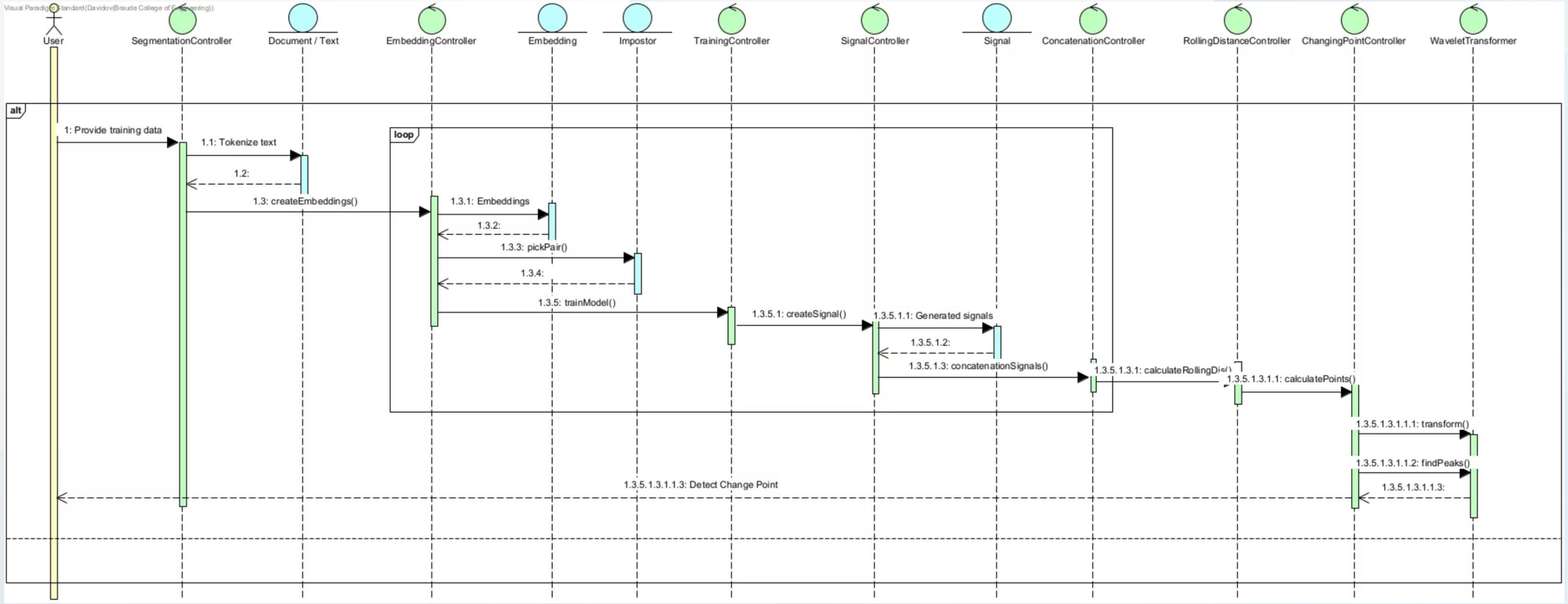
Boundary Finalization

Establish final stylistic boundaries based on consensus results, providing clear demarcation of stylistic shifts within the document.

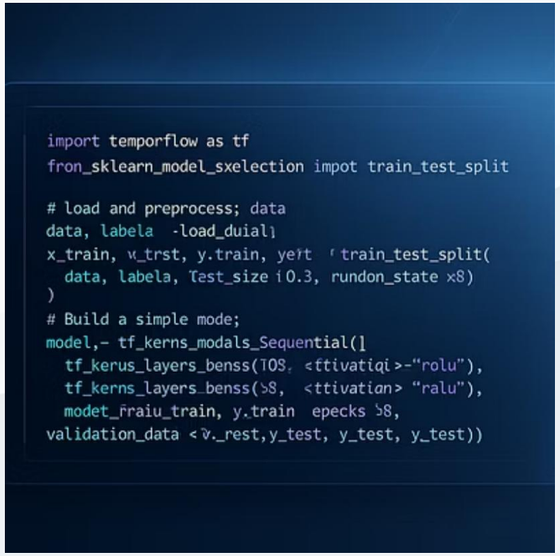
UML - Class Diagram

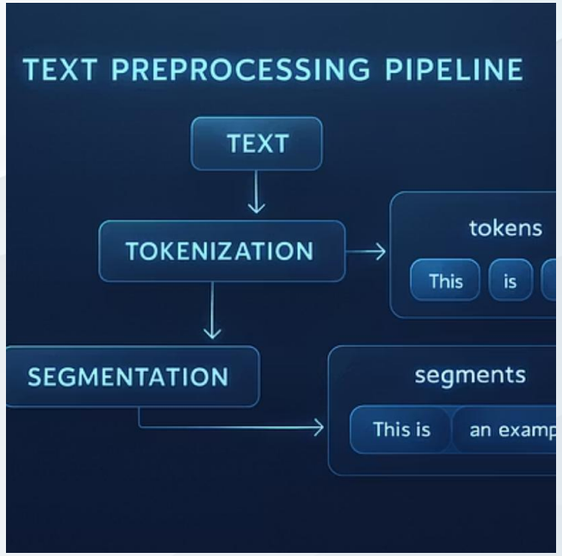


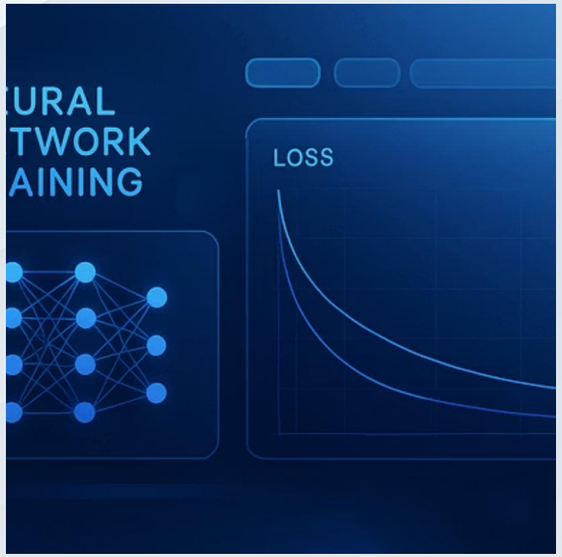
UML - Sequence Diagram

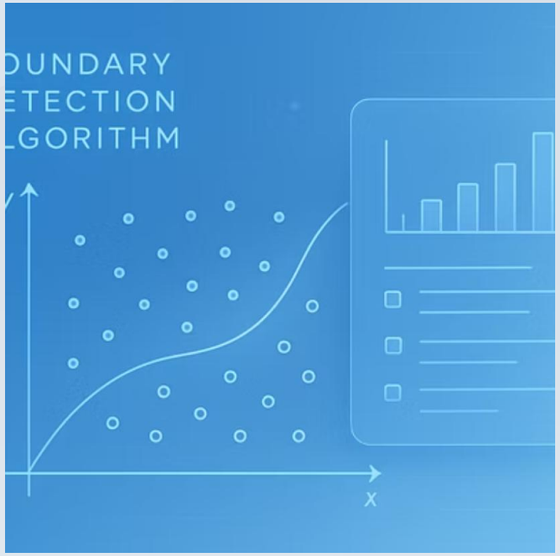


Implementation Workflow









The suggested implementation is going to be based on Python 3.10 in the Google Colab environment with a comprehensive technical stack including TensorFlow, scikit-learn, NumPy, pandas, transformers, gensim, and NLTK. The workflow progresses through distinct stages as mentioned on slide 6.

Testing Plan

Testing Objective

This plan outlines our intended evaluation strategy. No testing has been performed yet. Our testing objective is to verify the system's reliability in identifying stylistic shifts within documents.

Ground Truth Creation

We will create synthetic texts with manually annotated boundaries to serve as ground truth for evaluation.

Testing Procedure

The testing procedure will involve multiple impostor-classifier configurations to ensure robust performance across different scenarios.

Evaluation Metrics

Evaluation will focus on precision (accurate boundary detection), recall (complete boundary coverage), and F1-score (balanced performance measure), with particular attention to minimizing false positives while maximizing true positive detection.

Formal Testing Plan

Component	Test/Unit	Input	Expected Output	Evaluation Method
Tokenizer & Batching	Tokenization accuracy, batch integrity	Raw text documents (diverse styles)	Correctly tokenized sequences, fixed-size batches	Manual inspection, assert batch dimensions, vocabulary verification
Embedding Generation	Semantic coherence, dimensional consistency	Tokenized text batches	Contextually relevant numerical vector representations	Cosine similarity for related texts, dimensionality checks
Classifier (CNN/BERT)	Stylistic discrimination accuracy	Embeddings of stylistic chunks, ground truth labels	Correct classification of stylistic origin	Precision, Recall, F1-score against ground truth (synthetic data)
Chunk Aggregation	Correct combination of chunk outputs	Classifier outputs (labels/probabilities)	Coherent, aggregated stylistic representations	Cross-validation with known changes, consistency checks
Signal Generation	Signal robustness and responsiveness	Aggregated chunk data	Clear stylistic signal indicating shifts	Visual inspection of plots on synthetic data with known boundaries
Rolling Distance Calculation	Accuracy of distance metric computation	Stylistic signal data	Quantified stylistic divergence between windows	Numerical validation against manual calculations, sensitivity analysis
Spike/Peak Detection	Precision and recall of boundary identification	Rolling distance signal	Accurate detection of significant peaks	Comparison against ground truth boundaries
Voting Mechanism	Consensus formation for final boundaries	Multiple impostor-classifier boundary proposals	Consolidated, robust final stylistic boundaries	Agreement rate, F1-score on combined results vs. ground truth
Evaluation Metrics	Correctness of metric calculations	Detected and ground truth boundaries	Accurate Precision, Recall, F1-score	Manual verification, comparison with standard library implementations

Evaluation Methodology

Dataset Preparation

We create test documents by concatenating distinct author segments with clear stylistic differences. These documents are then manually annotated to establish ground truth boundaries that serve as the reference for our evaluation metrics.

Performance Metrics

Our evaluation employs precision, recall, and F1-score metrics with a margin of error (±1 chunk) to account for minor inaccuracies in boundary detection. This approach provides a comprehensive assessment of the system's detection capabilities.

Visualization

We plot stylistic signals with both predicted and actual boundary indicators, enabling visual comparison of system performance against ground truth and facilitating intuitive understanding of detection accuracy.

Innovation & Contributions

The following outlines how we believe our proposed approach may offer methodological innovation and real-world value as the project develops.

Methodological Innovation

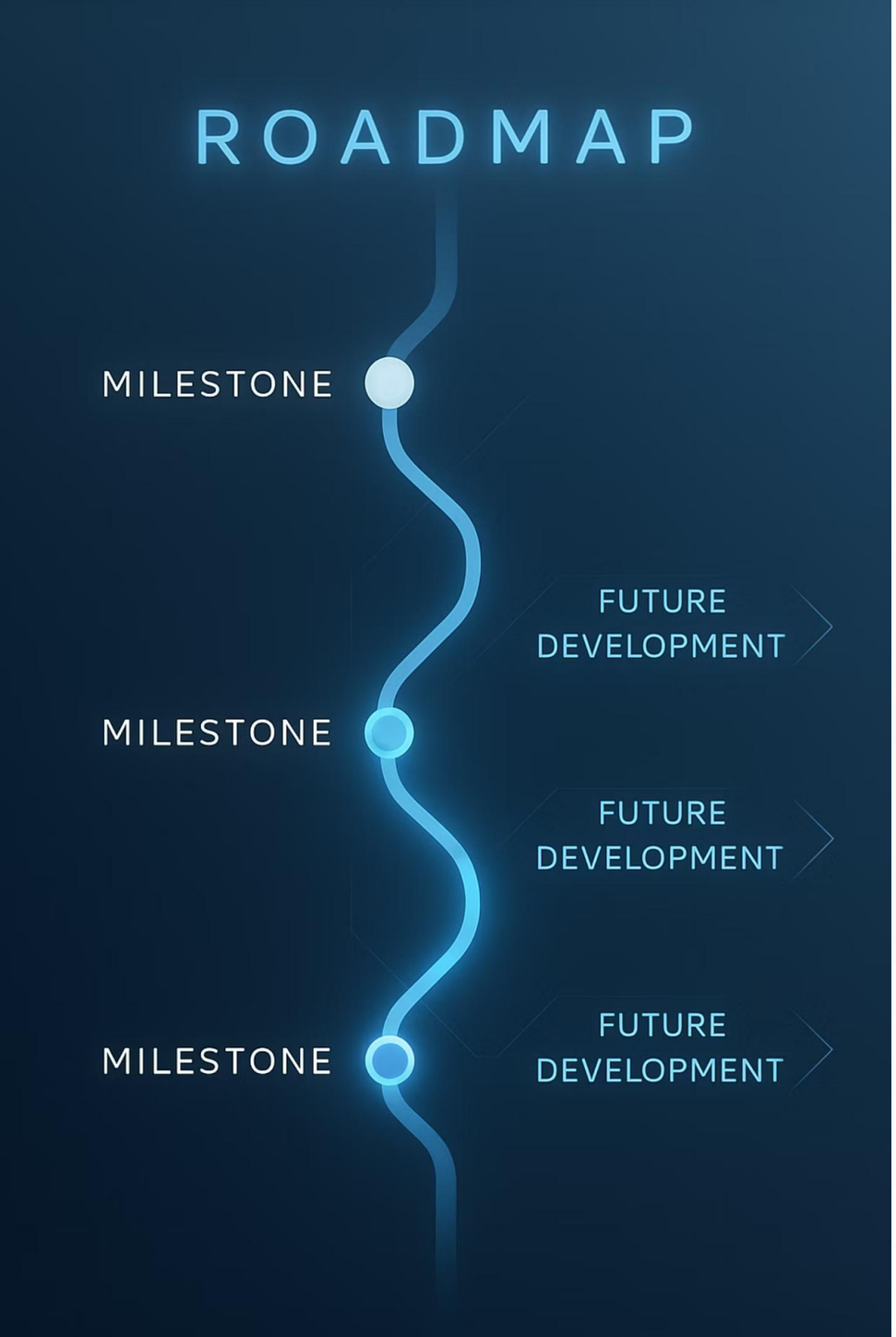
Our approach introduces deep neural networks into the impostor methodology, creating a novel unsupervised and interpretable stylistic segmentation framework. This may represent a meaningful advancement over traditional methods that rely on supervised learning or basic statistical techniques.

- Integration of deep learning with impostor techniques
- Unsupervised boundary detection
- Interpretable stylistic analysis

Scientific & Practical Contribution

If successful, the system could enhance the analysis of complex, multi-authored texts, supporting transparency in literary forensics, digital editing, and plagiarism detection. These capabilities would be valuable in both academic and practical content verification contexts.

- Enhanced multi-author text analysis
- Improved literary forensics tools
- Advanced plagiarism detection



Summary and Future Directions

Phase A Achievements	Phase B Next Steps
Clear definition of stylistic segmentation problem	Comprehensive classifier training and optimization
Developed robust deep impostor-based pipeline	Large-scale real-world testing and benchmarking
Established testing and evaluation framework	Fine-tuning voting mechanisms for accuracy
	Explore multilingual and cross-genre generalization

At the end of Phase A, we have defined the stylistic segmentation problem and designed a comprehensive solution approach using deep impostor methods. Our robust pipeline and evaluation framework provide a solid foundation for further development.

Moving forward, we will focus on optimizing classifiers, conducting extensive real-world testing, refining our boundary detection mechanisms, and exploring applications across different languages and genres to enhance the system's versatility and practical utility.

Thank You

Acknowledgments

Special thanks to Dr. Renata Avros and Prof. Zeev Volkovich for guidance and support throughout this research project. Additional appreciation to Braude College Software Engineering Department for providing resources and academic environment.

Contact Information

For questions, feedback, or collaboration opportunities, please reach out to Ofir Uziel. Your insights and suggestions are valuable for the continued development of this research.

Questions Welcome

Thank you for your attention! We welcome any questions about the methodology, implementation details, or potential applications of our stylistic analysis framework.