Project Report

Devansh Kantesaria(2022112003)

1 Initial Approach

1.1 Introduction

The approach employs a dual-perspective analysis through a novel weighted ensemble architecture that combines sentence-level sequential analysis with document-level holistic processing. This design attempts to capture both local linguistic patterns and global structural features that might differentiate between AI and human writing.

1.2 Methodology

1.2.1 Text Preprocessing

The preprocessing pipeline employs several steps to standardize input text:

1. **HTML Cleaning**: BeautifulSoup is used to extract clean text from potential HTML-formatted inputs
2. **URL and Email Removal**: Regular expressions filter out URLs and email addresses
3. **Character Normalization**: Non-alphanumeric characters (except basic punctuation) are removed
4. **Unicode Normalization**: Text is normalized using NFKD standardization
5. **Case Normalization**: All text is converted to lowercase for consistency
6. **Sentence Segmentation**: SpaCy's natural language processing pipeline divides text into sentence units

This thorough preprocessing approach ensures consistent inputs regardless of source formatting, while preserving meaningful linguistic elements like sentence boundaries that are crucial for downstream analysis.

1.2.2 Embedding Generation

The feature extraction process utilizes transformer-based embeddings:

1. **Base Model**: RoBERTa serves as the foundation model for generating contextual embeddings

2. **Dimension Reduction**: A projection layer reduces RoBERTa's 768-dimensional embeddings to 30 dimensions
3. **Dual Representation**: Each text is represented in two ways:
    - **Sentence-level embeddings**: A sequence of embeddings for each sentence
    - **Document-level embedding**: A single embedding for the entire text

The dimension reduction not only makes computation more efficient but also focuses the representation on the most discriminative features for the specific AI detection task through the trainable projection layer.

1.2.3 Memory Efficiency Considerations

To handle large datasets efficiently, the implementation includes several memory optimization techniques:

1. **Batch Processing**: Embeddings are generated in small batches
2. **Partial Storage**: Progress is periodically saved to avoid data loss
3. **GPU Memory Management**: Data transfer between CPU and GPU is carefully managed
4. **Empty Text Handling**: Special handling for edge cases like empty documents

1.3 Model Architecture

The detection system employs a weighted ensemble that combines two complementary neural network architectures:

1.3.1 BiLSTM Component

The Bidirectional LSTM model processes the sequence of sentence embeddings:

1. **Input**: Variable-length sequence of sentence embeddings (batch_size $\times$ num_sentences $\times$ embedding_dim)
2. **Architecture**:
    - Two-layer bidirectional LSTM with 32 hidden units
    - Final hidden states from both directions are concatenated
    - A fully connected layer maps to a single output node with sigmoid activation
3. **Function**: Captures sequential patterns and relationships between sentences

The bidirectional nature allows the model to analyze each sentence in the context of both preceding and following sentences, potentially identifying unnatural transitions or patterns that might indicate AI generation.

1.3.2 MLP Component

The Multi-Layer Perceptron processes the document-level embedding:

1. **Input**: Document embedding (batch_size $\times$ embedding_dim)

2. **Architecture**:
    - Three fully connected layers (input_dim $\rightarrow$ 32 $\rightarrow$ 16 $\rightarrow$ 1)
    - ReLU activations between layers
    - Dropout regularization (0.3 rate) to prevent overfitting
    - Final sigmoid activation
3. **Function**: Identifies global patterns and features in the complete text

This component focuses on holistic document characteristics that might distinguish AI-generated text, such as overall coherence, thematic consistency, or vocabulary distribution patterns.

1.3.3 Weighted Ensemble

The two model outputs are combined through a learnable weighting mechanism:

1. **Weighting Parameter**: A single learnable parameter determines the contribution of each model
2. **Sigmoid Transformation**: The raw weight is passed through a sigmoid function to constrain it between 0 and 1
3. **Weighted Sum**: The final prediction is $w \times$ MLP_output $+ (1 - w) \times$ BiLSTM_output

This adaptive weighting allows the model to automatically determine which component (sentence-level or document-level analysis) is more reliable for different types of texts, potentially adjusting to different AI generators or writing styles.

1.4 Experimental Setup

1.4.1 Dataset

The dataset combines AI-generated and human-written texts, with binary labels (1 for AI, 0 for human). The data is split using stratified sampling to maintain class balance:

- 80% training set
- 20% test set

1.4.2 Training Configuration

The model is trained with the following parameters:

- **Optimizer**: Adam with 0.001 learning rate
- **Loss Function**: Binary Cross-Entropy
- **Batch Size**: 128
- **Epochs**: 50
- **Reproducibility**: Fixed random seed (42) for deterministic results

1.4.3 Evaluation Metrics

Model performance is assessed using multiple complementary metrics:

- **Accuracy**: Overall correct classification rate
- **F1 Score**: Harmonic mean of precision and recall
- **AUC-ROC**: Area under the Receiver Operating Characteristic curve

2 GAN Based Approach

2.1 Approach Overview

The implemented system employs a Generative Adversarial Network (GAN) architecture modified specifically for text classification. Unlike traditional classification approaches that rely solely on supervised learning, this system utilizes the adversarial dynamic between a generator and discriminator to learn more robust representations of linguistic patterns that distinguish AI-generated text from human writing.

2.2 Theoretical Foundation

The approach is founded on the hypothesis that AI-generated text exhibits subtly different linguistic patterns compared to human text, even when the content appears similar at a superficial level. These differences manifest in:

1. Stylometric features (sentence structure, lexical diversity)
2. Content word distributions and patterns
3. Functional word usage and syntactic structures

These patterns can be difficult to detect with traditional classifiers but become more apparent in feature spaces learned through adversarial training.

2.3 Data Preprocessing Pipeline

The text preprocessing pipeline implements several sophisticated techniques:

2.3.1 Text Normalization and Quality Filtering

- Conversion to lowercase
- Standardization of Unicode characters and whitespace
- Removal of non-standard characters while preserving punctuation
- Quality filtering to remove extremely short texts or those with abnormal character distributions

2.3.2 Strategic Masking

A novel approach is used where content words (nouns, adjectives, proper nouns) are selectively masked while preserving function words. This forces the model to learn deeper structural patterns rather than relying on specific content tokens, enhancing its generalization capabilities across different domains and topics.

2.3.3 Stylometric Feature Extraction

Five key stylometric features are extracted:

- Type-token ratio (lexical diversity)
- Average sentence length
- Sentence length standard deviation
- Noun ratio
- Function word ratio

These features capture writing style characteristics independent of content, providing complementary signals to the deep learning features.

2.4 Model Architecture

The system employs a multi-component architecture:

2.4.1 Base Embedding Model

- Utilizes a pre-trained DistilRoBERTa model as a frozen feature extractor
- Extracts contextualized token representations without fine-tuning
- Provides deep semantic understanding of text content

2.4.2 Generator Network

- Multi-layer neural network with residual connections
- Progressive dimension reduction through hidden layers (768$\rightarrow$512$\rightarrow$256$\rightarrow$128)
- Layer normalization and dropout for regularization
- Transforms BERT embeddings into a feature space optimized for classification

2.4.3 Style Feature Processor

- Dedicated neural network to process stylometric features
- Transforms raw stylometric features into a learned representation
- Enables integration with the main feature pathway

2.4.4 Self-Attention Mechanism

- Allows the discriminator to focus on the most relevant aspects of generated features
- Improves discrimination by highlighting distinctive patterns

2.4.5 Discriminator Network

- Combines generator features and processed stylometric features
- Applies multiple layers with batch normalization
- Outputs probability of text being AI-generated

2.5 Adversarial Training Approach

The training process follows a modified GAN training procedure:

1. **Discriminator Training Phase:** The discriminator is trained to correctly classify texts as human or AI-generated, with gradients flowing through the discriminator only.
2. **Generator Training Phase:** The generator is trained to produce features that fool the discriminator, with inverted labels to create the adversarial objective. The discriminator parameters are frozen during this phase.
3. **Balanced Training:** The discriminator is trained more frequently than the generator (5:1 ratio) to maintain balance, as discriminator training tends to be more stable.

This adversarial dynamic creates a continuous improvement cycle where the generator must create increasingly sophisticated representations, and the discriminator must find increasingly subtle patterns to distinguish between classes.

2.6 Technical Implementation Details

The implementation includes several advanced techniques:

2.6.1 Learning Rate Management

- Separate optimizers for generator and discriminator components
- Learning rate reduction on plateau to navigate the complex loss landscape
- Different initial learning rates (2e-4 for generator, 1e-4 for discriminator)

2.6.2 Progressive Batch Processing

- Strategic alternation between discriminator and generator updates
- Balanced mini-batch training to prevent mode collapse

2.6.3 Evaluation Metrics

- Multiple complementary metrics (accuracy, AUC-ROC, F1 score)
- Validation-based model selection

2.6.4 Visualization

- t-SNE dimensionality reduction for embedding visualization
- Training progress tracking with loss curves
- Feature importance analysis

2.7 Limitations and Future Work

Despite its strengths, the approach has several limitations:

1. **Computational Complexity:** GAN training requires more computational resources than traditional classifiers.
2. **Training Stability:** As with many GAN architectures, training can sometimes be unstable and require careful hyperparameter tuning.
3. **Cross-Domain Generalization:** While strategic masking helps, domain adaptation remains challenging.

3 LightWeight Approach

3.1 Introduction

The system employs a multi-faceted approach combining stylometric analysis, text embeddings, and neural networks while prioritizing computational efficiency. This lightweight solution achieves strong performance by leveraging key linguistic patterns that differentiate human and machine-generated writing without requiring extensive computational resources.

3.2 Approach Overview

The detector implements a hybrid architecture that analyzes text from multiple perspectives simultaneously. The system extracts three distinct types of features:

1. **Stylometric features** capturing the writing style at a global level
2. **Full-text embeddings** representing the semantic content of the entire text
3. **Sentence-level embeddings** processed sequentially to identify patterns in sentence structure and coherence

These complementary features are then combined using a neural network to make the final classification. This approach is based on the understanding that AI-generated text differs from human writing in subtle ways across multiple dimensions, including stylistic consistency, semantic coherence, and linguistic diversity.

3.3 Intuition Behind the Approach

The fundamental intuition driving this detector is that AI language models and human writers exhibit different linguistic fingerprints. While modern AI systems produce increasingly coherent and grammatically correct text, they still display characteristic patterns that differentiate them from human writers. These differences manifest in several key areas:

3.3.1 Linguistic Diversity and Richness

Human writers typically demonstrate greater linguistic diversity and idiosyncrasy. The detector quantifies this through metrics like type-token ratio (lexical diversity), Yule's K (vocabulary richness), and analysis of POS (Part-of-Speech) distributions. AI-generated text often shows more homogenized patterns and potentially more repetitive vocabulary usage compared to the natural variation in human writing.

3.3.2 Structural Patterns

Sentence length variation and structural complexity often differ between AI and human writing. AI systems may produce text with more consistent sentence structures and lengths, while humans naturally vary their sentence patterns. The system captures these differences through sentence length statistics and sequential analysis of sentence embeddings using a bidirectional LSTM.

3.3.3 Function Word Usage

Function words (articles, prepositions, conjunctions) are used differently by AI systems and humans. The detector analyzes function word ratio and distributions as these can reveal subtle differences in how language is structured, even when the content appears similar on the surface.

3.3.4 Punctuation Patterns

The frequency and distribution of punctuation marks often differ between AI and human writing. The detector explicitly measures punctuation ratio as this can be a telling indicator of authorship style.

3.4 Technical Architecture

The detector is built upon three primary components that work together to extract, process, and classify text features:

3.4.1 LightStyleFeatureExtractor

This component performs linguistic analysis to extract stylometric features from the input text. Key features include:

- **Type-token ratio**: Measures lexical diversity by dividing unique word types by total tokens
- **Sentence length statistics**: Calculates average sentence length and variance
- **POS tag distributions**: Quantifies the frequency of different parts of speech
- **Function word ratio**: Measures the proportion of function words to content words
- **Punctuation ratio**: Calculates the frequency of punctuation usage
- **Yule's K**: A measure of vocabulary richness that accounts for text length
- **Average word length**: Captures complexity of vocabulary used

This lightweight feature extraction uses SpaCy's small English model to efficiently process texts without requiring heavy computational resources.

3.4.2 LightEmbeddingGenerator

Instead of using resource-intensive transformer models, this component creates text embeddings using a more efficient approach:

- Employs TF-IDF vectorization to create sparse representations of texts
- Uses unigrams and bigrams to capture word patterns
- Applies dimensionality reduction through a neural network layer to create dense embeddings
- Generates embeddings for both the full text and individual sentences
- Implements caching to avoid redundant computation

This approach drastically reduces the computational requirements while still capturing meaningful semantic information from the text.

3.4.3 LightDetector Neural Network

The core classification model combines multiple types of features using a neural network architecture:

- **Sentence processing**: A bidirectional LSTM processes the sequence of sentence embeddings to capture narrative flow and coherence patterns
- **Style encoding**: A feed-forward neural network processes the stylometric features
- **Text encoding**: A separate feed-forward network processes the full-text embedding
- **Feature fusion**: The outputs from all three branches are concatenated
- **Classification**: The combined features pass through final fully-connected layers with dropout regularization to prevent overfitting

The model outputs a single scalar value which, after passing through a sigmoid function, represents the probability of the text being AI-generated.

3.5 Data Processing Pipeline

The system implements an efficient data processing pipeline:

1. **Text preprocessing**: Cleans input text and prepares it for feature extraction
2. **Feature extraction**: Computes stylometric features and generates embeddings
3. **Caching**: Stores computed features to avoid redundant processing
4. **Batching**: Organizes data into batches for efficient training
5. **Gradient accumulation**: Allows for effective larger batch sizes with limited memory

The pipeline includes error handling to gracefully manage exceptions during processing, ensuring robustness when dealing with unusual or problematic text inputs.

3.6 Training Methodology

The detector employs several strategies to optimize the training process:

- **AdamW optimizer**: Combines the benefits of Adam optimization with proper weight decay regularization
- **Learning rate scheduling**: Dynamically adjusts learning rate based on validation performance
- **Early stopping**: Saves the best model based on validation AUC to prevent overfitting
- **Efficient validation**: Performs validation less frequently to speed up training
- **Batch normalization**: Stabilizes and accelerates training
- **Dropout regularization**: Prevents overfitting by randomly deactivating neurons during training

The model is trained using binary cross-entropy loss, appropriate for the binary classification task of distinguishing AI-generated from human-written text.

3.7 Performance Evaluation

The system evaluates performance using multiple metrics to ensure comprehensive assessment:

- **AUC-ROC**: Measures the model's ability to distinguish between classes across all threshold values
- **Accuracy**: The proportion of correct predictions
- **F1 Score**: The harmonic mean of precision and recall
- **Precision**: The ratio of true positive predictions to all positive predictions
- **Recall**: The ratio of true positive predictions to all actual positives

These metrics provide a complete picture of the detector's performance, allowing for fine-tuning to optimize for specific requirements (e.g., prioritizing precision over recall or vice versa).

3.8 Practical Applications

The lightweight nature of this detector makes it suitable for a variety of practical applications:

- **Content moderation**: Helping identify AI-generated content in online platforms
- **Academic integrity**: Detecting AI-generated assignments or papers
- **Publishing workflows**: Screening submissions for AI-generated content
- **Media verification**: Supporting efforts to identify synthetic news or information
- **Social media analysis**: Analyzing trends in AI-generated content online

4 Discussion

4.0.1 Intuition Behind the Approach

The hybrid architecture is motivated by several key insights about AI-generated text:

1. **Local vs. Global Patterns**: AI-generated text may exhibit distinct patterns at both the sentence level (local) and document level (global)
2. **Sequential Dynamics**: The way sentences flow and connect in AI text often differs subtly from human writing, which the BiLSTM component aims to capture
3. **Feature Complementarity**: Different models may excel at detecting different aspects of AI-generated text, so an ensemble approach leverages their complementary strengths
4. **Adaptive Weighting**: The importance of sentence-level versus document-level features may vary across different texts, making learnable weighting valuable
5. **Dimension Reduction**: The projection layer not only improves computational efficiency but also allows the model to focus on the most discriminative features for this specific task

4.0.2 Advantages

The architecture offers several notable advantages:

1. **Flexibility**: Handles variable-length inputs naturally through the BiLSTM component
2. **Interpretability**: The learned weight parameter provides insight into which level of analysis (sentence or document) is more important
3. **Memory Efficiency**: Batch processing and careful memory management enable handling large datasets
4. **Robustness**: Special handling for edge cases like empty documents or sentences

4.0.3 Limitations

Potential limitations of the approach include:

1. **Computational Complexity**: Using transformer-based embeddings and BiLSTM makes the model relatively computationally intensive
2. **Domain Sensitivity**: The model may be sensitive to domain shifts if trained on specific text genres
3. **Length Dependency**: Very short texts may not provide sufficient sequential information for the BiLSTM component

5 Dataset

- A new dataset which combines:
  - Open-GPT-Text
  - Open-Web-Text

- AI vs Human Text (500K essays)

- Final dataset:
  - Balanced samples of AI and human-generated text from both datasets.
  - Total size: **47,000** text samples.
  - Link to Dataset

6 Results and Comparisons

Approach	**Accuracy**	**F1 Score**	**AUC-ROC**
Initial Approach	0.4435	0.4560	0.4784
GAN-based Approach	0.5460	0.5320	0.5782
Lightweight Approach	0.6227	0.6553	0.6114

Table 1: Performance comparison on the test set for different approaches