Proposed Method

Step-by-Step Breakdown

Input Representation

Review sample:
$R = \{w_1, w_2, ..., w_n\}$
Each w_i is a word in the review.

Step 1: Data Augmentation

Action: Randomly insert a token st at a position k in the review → $R^* = \{w_1, ..., w_k, st, w_{k+1}, ..., w_n\}$

Motivation:

- Mimics natural noise and typos in real-world data.
- Prevents the model from overfitting to artificial, overly clean training data.
- Encourages the model to focus on core semantic content rather than surface form.

Step 2: Task A – Binary Classification (Human vs AI)

Model: Fine-tuned BART (Bidirectional and Auto-Regressive Transformer)
Input: Augmented review R*
Output: Binary label (0 = Human, 1 = AI-generated)

Training:

- Use cross-entropy loss to train BART on R* to classify reviews as human or AI-generated.

Motivation:

- BART is a powerful encoder-decoder model capable of capturing nuanced differences in fluency, coherence, and structure — traits often distinguishing human and AI-written text.
- Fine-tuning BART ensures task-specific adaptation while leveraging pre-trained language understanding.

Step 3: Task B – Model Attribution (LLM Identification)

Trigger: Activated only if Task A predicts "AI-generated".

3.1 Feature Extraction using BART

- Extract the semantic vector $f(R^*) \in \mathbb{R}^d$ using the encoder part of the fine-tuned BART.

Motivation:

- Encoded embeddings capture the distinctive writing style and linguistic signature of different LLMs.
- These embeddings serve as a meaningful representation for downstream attribution.

3.2 Perceptual Hashing

- Project $f(R^*)$ into a compact binary hash code space (e.g., via SimHash or learned binary projection).

Motivation:

- Reduces memory and computational cost during inference.
- Transforms high-dimensional semantic data into binary codes that preserve similarity.
- Facilitates efficient search and comparison, ideal for scalable model attribution.

3.3 k-Nearest Neighbors (k-NN) with Cosine Similarity

- Store hash codes from known LLM-generated reviews.
- For a new review, compute its hash and perform k-NN search using cosine similarity to find the most similar embeddings.

Motivation:

- k-NN is non-parametric, allowing easy updates without retraining when new LLMs are added.
- Cosine similarity works well in embedding spaces where direction (rather than magnitude) carries meaning.
- Simple yet effective technique for LLM identification with minimal overhead.

Step 4: Pseudo-Labeling to Improve Learning

Action:

- For unlabeled reviews where model confidence $P(\hat{y}|R) > \tau$, assign pseudo-label $\hat{y}$ and add to training set:

$D_{pseudo} = \{ (R, \hat{y}) \mid P(\hat{y}|R) > \tau \}$

Motivation:

- Expands the training dataset without needing manual labels.
- Helps the model adapt to changing distributions (e.g., new LLMs or writing styles).
- Smooths decision boundaries using confident examples from real-world data.

Training and Testing Protocol

- **Dataset**: Balanced mix of human-written and LLM-generated reviews.
- **Split**: 80% training, 20% testing.

During Training:

- Task A: Train BART using cross-entropy loss for binary classification.
- Task B:
    - Extract semantic features from BART.
    - Apply perceptual hashing.
    - Store hash codes with labels (LLM name).
- Incorporate pseudo-labeled examples iteratively to enrich training.

Inference Pipeline (During Testing)

1. **Input**: A review.
2. **Task A**: Use BART to classify as Human or AI-generated.
3. **If AI-generated**:
    a. Extract $f(R^*)$, apply perceptual hashing.
    b. Use k-NN + cosine similarity on stored hash codes.
    c. Identify the closest matching LLM.

4. **Output**:
    a. If Human: label = Human.
    b. If AI: label = AI + [Identified LLM].