CIC-NLP at GenAI Detection Task 1: Leveraging DistilBERT for Detecting Machine-Generated Text in English

Abiola T. O.[1], Ojo O. E.[1], Tewodros A. B.[1], Abiola J. O. [2], Oladepo T. O.[2], Adebanji O. O. [1], Sidorov G.[1], Kolesnikova O.[1]
[1]Instituto Politécnico Nacional, Centro de Investigación en Computación, CDMX, Mexico.
[2]Federal University Oye-Ekiti, Nigeria.
kolesnikova@cic.ipn.mx

Abstract

As machine-generated texts (MGT) become increasingly similar to human writing, these distinctions are harder to identify. In this paper, we as the CIC-NLP team present our submission to the Gen-AI Content Detection Workshop at COLING 2025 for Task 1 Subtask A, which involves distinguishing between text generated by LLMs and text authored by humans, with an emphasis on detecting English-only MGT. We applied the DistilBERT model to this binary classification task using the dataset provided by the organizers. Fine-tuning the model effectively differentiated between the classes, resulting in a micro-average F1-score of 0.70 on the evaluation test set. We provide a detailed explanation of the fine-tuning parameters and steps involved in our analysis.

1 Introduction

The rapid advancement of large language models (LLMs) has revolutionized NLP across all fields (Yigezu et al., 2023; Kolesnikova et al., 2023; Adebanji et al., 2024; García-Vázquez et al., 2023; Laureano and Calvo, 2024; Aguilar-Canto et al., 2023; Ojo et al., 2024). Machine-generated text (MGT) refers to text produced using AI algorithms with little or no human intervention. On the other hand, human-written text (HWT) relies on natural language and contextual understanding in its approach to construction. AI models are able to generate new content by being trained on large text datasets. However, it can be challenging for AI models to replicate the attributes of HWT accurately. In this work, we leverage the intricacies between MGT and HWT to develop a model that can successfully distinguish between the two.

AI models can generate grammatically correct and contextually relevant text, but they often lack true cohesion between the sentences they produce (Zheng, 2024). HWT conveys personal experience, emotions, and cultural understanding in its content, while MGT can only mimic surface-level emotional cues without grasping the underlying sentiment or context, leading to misrepresentation. Detecting MGT means identifying and carefully analyzing misleading or inappropriate content (Garib and Coffelt, 2024). This detection has also become crucial for addressing ethical issues arising from using AI (Ansarullah et al., 2024). LLMs often reflect biases and limitations inherent in the data on which they were trained(Gallegos et al., 2024). MGT may lean toward certain linguistic patterns, expressions, or topics that are common in the training dataset, which can differ from the more varied and dynamic expressions found in HWT. These biases can also help distinguish between MGT and HWT.

To address some of these highlighted issues, the organizers of COLING Workshop 2025 introduced a large novel dataset (Wang et al., 2025), (Chowdhury et al., 2025), (Dugan et al., 2025). The uniqueness of this dataset is the fact that it broadly covers different text types from several human and AI sources, thus making it a valuable dataset to test several proposed models in exploring the attributes of MGT and HWT.

2 Related Work

Early efforts in detecting MGT focused on identifying telltale signs of machine output, such as lack of coherence, repetitive patterns, or unnatural phrasing (Maimone and Jolley, 2023). However, as generative models continue to improve significantly in terms of fluency, the challenge of detecting MGT has become increasingly difficult. Traditional rule-based approaches and ML classifiers have been largely insufficient to keep pace with the increasingly sophisticated text classification capabilities of deep learning (DL) models (Kierner et al., 2023). Recent approaches have thus turned to DL, particularly transformer-based models, which offer state-of-the-art performance

Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect), pages 271–277
January 19, 2025.

in various NLP tasks (Hoang; Damián et al., 2024; Ojo et al., 2023b,a; Soto et al., 2024).

The introduction of transformer models marked a significant breakthrough in NLP. Research by various authors has shown that transformer-based models are effective in performing a range of NLP tasks (Balouchzahi et al., 2021; Ojo et al., 2023c). DistilBERT has recently been explored for various text classification tasks, including sentiment analysis (Sidorov et al., 2023; Eyob et al., 2024; Ojo et al., 2023b) and text summarization (Alshanqiti et al., 2021). Its efficiency and robustness make it a promising candidate for distinguishing between MGT and HWT. The key advantage of using DistilBERT for this task lies in its ability to effectively capture semantic and syntactic patterns in text, which are often subtle yet critical for differentiating between the two types of text.

The authors in (Li et al., 2024) explored the challenges and feasibility of detecting AI-generated text across various domains and LLMs. They highlighted the growing ability of LLMs to generate human-like text, emphasizing the need for effective detection methods to mitigate risks such as misinformation and plagiarism. They argued that previous research had been limited to specific domains or language models, while real-world detectors needed to handle diverse inputs without prior knowledge of their source. To address this gap, they created a testbed containing a variety of human-written and machine-generated texts from multiple LLMs. Their results revealed significant challenges in distinguishing between human-authored and machine-generated texts, especially with out-of-distribution data. The authors also investigated linguistic patterns that could aid in differentiation, using tools like Stanza to analyze named entities, part-of-speech tags, and other features. Their analysis showed that including texts from diverse domains and LLMs reduced the linguistic differences between human and machine-generated texts, making detection more difficult.

The approach described in (Fernández-Hernández et al., 2023) focused on the participation of the Turing-Testers team in the AuTexTification shared task at IberLEF 2023, which aimed at automated text identification. The task consisted of two subtasks: the first involved distinguishing between human-written and machine-generated texts, while the second focused on attributing texts to specific large language models (LLMs). The authors addressed subtask 1 in both English and Spanish, testing a combination of traditional machine learning and deep learning methods, along with integrating additional metadata related to readability, complexity, sentiment, emotion, and toxicity of the texts. The experiments demonstrated the effectiveness of fine-tuning a multilingual BERT uncased model for detecting AI-generated texts. However, the inclusion of additional features, such as metadata, led to a decrease in performance, even compared to traditional machine learning models, suggesting that BERT's architecture alone was sufficient for classifying text. The authors concluded that, while their experiments provided valuable insights into the effectiveness of different models for detecting AI-generated text, further research was needed.

3 AI vs Human Text Detection

3.1 Dataset Analysis

Datasets for training, development, and testing were provided by the organizers through Hugging Face and Google Drive. The datasets consist of the ID, Text, Source, and Label columns, the label columns consist of two numerical values 0 and 1, where 1 marks the text generated by humans and 0 for the text generated by AI. The task involves binary classification of the test texts into human or AI-written categories after running and fine-tuning the model through the training and development stages. The dataset comprises 610,767 training samples with 381,845 samples being machine-generated and 228,822 human-generated text, furthermore, the development dataset consists of 261,758 samples with 163,430 samples being machine-generated texts and 98,328 human texts. Figures 1 and 2 give a view of the sources of the training dataset and the LLM models used to generate texts.

3.2 Application of DistilBERT

Using the Transformers Library from Hugging Face, we took some steps into data engineering by preparing and formatting it to be a suitable input for the model, then employed them in training and evaluation. The methodology adopted in these steps is outlined in Section 4, where we explain how we designed the DistilBERT base model architecture and proceeded to encode and load the dataset for further processing.

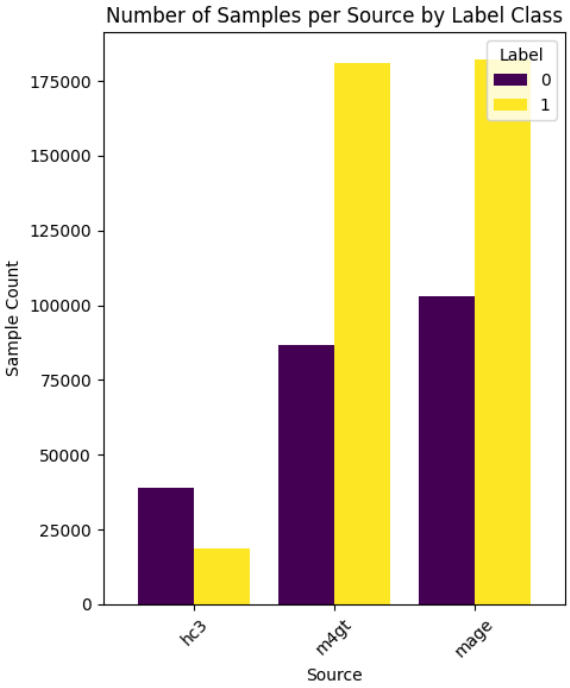

Figure 1: Number of samples by source

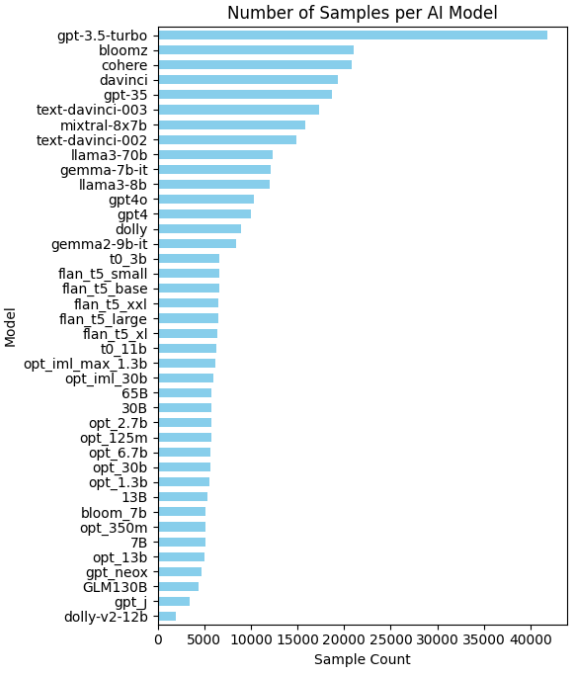

Figure 2: Number of samples generated by different LLM models

4 System Setup and Experiments

4.1 System Setup

We fine-tuned the model by adding a classification layer on the pre-trained DistilBERT architecture to enable it to perform binary classification.

We applied an amount of regularization to the model to prevent overfitting and biases. This was done to address the class imbalance in the dataset and ensure balanced learning across the classes by minimizing potential biases that may arise due to uneven class distributions.

With the adaptation of early stopping and model checkpointing, the best-performing model (based on the F1-score) was saved during training, which helped to prevent overfitting and ensured that the model selected for evaluation had the highest validation performance. This also ensured saving model checkpoints at the end of every epoch.

We used a cosine-annealing learning rate schedule, which gradually reduces the learning rate over time. This scheduling strategy helped stabilize training by lowering the learning rate as the model converged, often resulting in better final model performance.

4.2 Experiments

We adopted the training strategy to focus on fine-tuning the DistilBERT model, specifically the distilbert-base-uncased version; this allowed the model to accommodate the unique characteristics that define the classes in our dataset. This was done by carefully selecting some key hyperparameters and configurations to enhance the models's performance.

The model was trained over 3 epochs, and we leverage available GPU memory, and gradient accumulation steps of 2 to balance memory usage. Mixed precision was utilized to accelerate training by enabling mixed precision training, with early stopping enabled based on F1-score improvements. Additionally, a cosine learning rate scheduler was applied to modulate the learning rate as we optimized batch size for memory and computational efficiency. We used warm-up steps to enable a short warm-up period where the learning rate gradually increases, helping the model to avoid instability at the start of training and smoothing the convergence. Also, we accumulate gradients over two batches before performing an update and using DataCollatorWithPadding to ensure that the input sequences were dynamically padded to the maximum length within each batch, minimizing wasted computation on padding tokens to improve training speed and efficiency. The results of the training and validation stages of the model are displayed in Table 1, and the loss plot and F1-score plot of the training and validation stages are shown in Figures 3 and 4,

Epoch	Training Loss	Validation Loss	Accuracy	F1-score
1	0.1975	0.1759	0.9270	0.9263
2	0.1038	0.1509	0.9439	0.9434
3	0.0541	0.1807	0.9466	0.9461

Table 1: Metrics for each epoch including training and validation losses, accuracy, and F1-score.

Model	Micro-average_F1	Macro-average_F1
DistilBERT	0.7068	0.7242

Table 2: Result obtained from the test dataset

respectively.

We evaluated each epoch, allowing us to monitor the model's progress closely. This helped to ensure that our model's training and validation performance was aligned and aided in identifying overfitting or underfitting early.

Evaluation of the model was done using the micro-average F1-score, which is the default metric for this subtask. The evaluation on the development dataset was performed after each training epoch to track how well the model was doing on the task and closely monitor its progress.

5 Results

In this section, we present the results of the test dataset provided by the organizers of the COLING 2025 conference as predicted by the pre-trained DistilBERT model on the English test supplied through Google Drive. The prediction of the model was uploaded to the Codabench platform featured by the organizers of the conference for submission, and the micro- and macro-average F1 scores were computed accordingly. The results proved the model's ability to distinguish between human-written text and AI-generated text. The details of the results are presented in Table 2.

6 Conclusions

The application of DistilBERT, a compact and efficient transformer model for the detection of AI-generated text illustrates the advancement in the potential of transformer-based models to distinguish nuances in language style and origin. This task addresses specific challenges concerning the size of the dataset and the fact that LLM AI-generated text often mimics human-like patterns, thereby requiring the model to capture subtle stylistic and structural differences. Our approach leverages DistilBERT's efficient architecture and tokenizer capabilities to effectively identify these distinctions with a micro-average F1-score of 0.70. This result highlights DistilBERT's ability to generalize and perform well on the intricate task of text detection and classification, showing that smaller models can yield robust outcomes even in sophisticated NLP tasks.

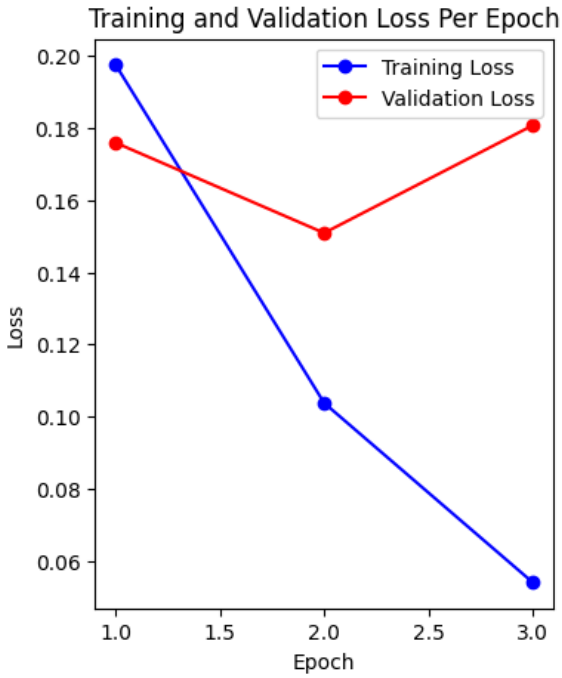

Figure 3: Training and development data loss plot per epoch

The performance obtained in our experiments emphasizes both the viability of DistilBERT in this domain and the promise of continued exploration with other transformer-based models. Future work will focus on expanding the dataset to cover a broader array of AI-generated text from various models and fine-tuning DistilBERT and similar models to achieve even more refined detection capabilities.

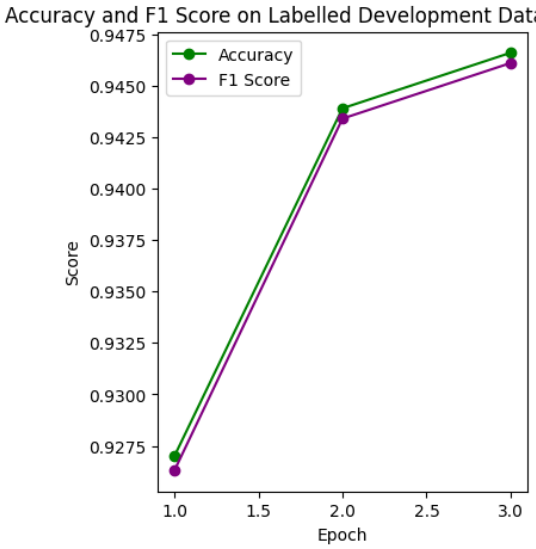

Figure 4: F1-score plot on labeled development dataset

Acknowledgments

The work was done with partial support from the Mexican Government through the grant A1-S-47854 of CONACYT, Mexico, grants 20241816, 20241819, and 20240951 of the Secretaría de Investigación y Posgrado of the Instituto Politécnico Nacional, Mexico. The authors thank the CONACYT for the computing resources brought to them through the Plataforma de Aprendizaje Profundo para Tecnologías del Lenguaje of the Laboratorio de Supercómputo of the INAOE, Mexico and acknowledge the support of Microsoft through the Microsoft Latin America PhD Award.

References

Olaronke Oluwayemisi Adebanji, Olumide Ebenezer Ojo, Hiram Calvo, Irina Gelbukh, and Grigori Sidorov. 2024. Adaptation of transformer-based models for depression detection. *Computación y Sistemas*, 28(1).

Fernando Aguilar-Canto, Marco A Cardoso-Moreno, Diana Jiménez, and Hiram Calvo. 2023. Gpt-2 versus gpt-3 and bloom: Llms for llms generative text detection. In *IberLEF@ SEPLN*.

Abdullah Alshanqiti, Abdallah Namoun, Aeshah Alsughayyir, Aisha Mousa Mashraqi, Abdul Rehman Gilal, and Sami Saad Albouq. 2021. Leveraging distilbert for summarizing arabic text: an extractive dual-stage approach. *IEEE Access*, 9:135594–135607.

Syed Immamul Ansarullah, Mudasir Manzoor Kirmani, Sami Alshmrany, and Arfat Firdous. 2024. Ethical issues around artificial intelligence. In *A Biologist s Guide to Artificial Intelligence*, pages 301–314. Elsevier.

Fazlourrahman Balouchzahi, Hosahalli Lakshmaiah Shashirekha, and Grigori Sidorov. 2021. Mucic at checkthat! 2021: Fado-fake news detection and domain identification using transformers ensembling. In *CLEF (Working Notes)*, pages 455–464.

Shammur Absar Chowdhury, Hind Al-Merekhi, Mucahid Kutlu, Kaan Efe Keleş, Fatema Ahmad, Tasnim Mohiuddin, Georgios Mikros, and Firoj Alam. 2025. GenAI content detection task 2: AI vs. human – academic essay authenticity challenge. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Sergio Damián, Brian Herrera, David Vázquez, Hiram Calvo, Edgardo Felipe-Riverón, and Cornelio Yáñez-Márquez. 2024. Dsvs at pan 2024: Ensemble approach of transformer-based language models for analyzing conspiracy theories against critical thinking narratives.

Liam Dugan, Andrew Zhu, Firoj Alam, Preslav Nakov, Marianna Apidianaki, and Callison-Burch Chris. 2025. Genai content detection task 3: Cross-domain machine generated text detection challenge. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Lemlem Eyob, Tsadkan Yitbarek, Amna Naseeb, Grigori Sidorov, and Ildar Batyrshin. 2024. Enhancing hope speech detection on twitter using machine learning and transformer models.

Alberto Fernández-Hernández, Juan Luis Arboledas-Márquez, Julián Ariza-Merino, and Salud María Jiménez Zafra. 2023. Taming the turing test: Exploring machine learning approaches to discriminate human vs. ai-generated texts. In *IberLEF@ SEPLN*.

Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. 2024. Bias and fairness in large language models: A survey. *Computational Linguistics*, pages 1–79.

Omar García-Vázquez, Tania Alcántara, Hiram Calvo, and Grigori Sidorov. 2023. Llm's for spanish song text analysis and classification using language variants. In *Mexican International Conference on Artificial Intelligence*, pages 118–127. Springer.

Ali Garib and Tina A Coffelt. 2024. Detecting the anomalies: Exploring implications of qualitative research in identifying ai-generated text for ai-assisted composition instruction. *Computers and Composition*, 73:102869.

Thang Ta Hoang. The combination of bert and data oversampling for answer type prediction.

Slawomir Kierner, Jacek Kucharski, and Zofia Kierner. 2023. Taxonomy of hybrid architectures involving rule-based reasoning and machine learning in clinical decision systems: A scoping review. *Journal of Biomedical Informatics*, 144:104428.

Olga Kolesnikova, Mesay Gemeda Yigezu, Atnafu Lambebo Tonja, Michael Meles Woldeyohannis, Grigori Sidorov, and Alexander Gelbukh. 2023. Ginger disease detection using a computer vision pre-trained model. In *Innovations in Machine and Deep Learning: Case Studies and Applications*, pages 419–432. Springer.

Mayte H Laureano and Hiram Calvo. 2024. Computational study of dream interpretations: Psychoanalytic human vs artificial analyses. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–9. IEEE.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2024. MAGE: Machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 36–53, Bangkok, Thailand. Association for Computational Linguistics.

Luciane Maimone and Jason Jolley. 2023. Looks like google to me: Instructor ability to detect machine translation in l2 spanish writing. *Foreign Language Annals*, 56(3):627–644.

Olumide E Ojo, Olaronke O Adebanji, Hiram Calvo, Damian O Dieke, Olumuyiwa E Ojo, Seye E Akinsanya, Tolulope O Abiola, and Anna Feldman. 2023a. Legend at araieval shared task: Persuasion technique detection using a language-agnostic text representation model. In *1st Arabic Natural Language Processing Conference, ArabicNLP 2023*, pages 594–599. Association for Computational Linguistics (ACL).

Olumide E Ojo, Olaronke O Adebanji, Hiram Calvo, Alexander Gelbukh, Anna Feldman, and Ofir Ben Shoham. 2024. Doctor or ai? efficient neural network for response classification in health consultations. *IEEE Access*.

Olumide Ebenezer Ojo, Olaronke Oluwayemisi Adebanji, Hiram Calvo, Alexander Gelbukh, Anna Feldman, and Grigori Sidorov. 2023b. Hate and offensive content identification in indo-aryan languages using transformer-based models.

Olumide Ebenezer Ojo, Hoang Thang Ta, Alexander Gelbukh, Hiram Calvo, Olaronke Oluwayemisi Adebanji, and Grigori Sidorov. 2023c. Transformer-based approaches to sentiment detection. In *Recent Developments and the New Directions of Research, Foundations, and Applications: Selected Papers of the 8th World Conference on Soft Computing, February 03–05, 2022, Baku, Azerbaijan, Vol. II*, pages 101–110. Springer.

Grigori Sidorov, Fazlourrahman Balouchzahi, Sabur Butt, and Alexander Gelbukh. 2023. Regret and hope on transformers: An analysis of transformers on regret and hope speech detection datasets. *Applied Sciences*, 13(6):3983.

Miguel Soto, Cesar Macias, Marco Cardoso-Moreno, Tania Alcántara, Omar García, and Hiram Calvo. 2024. Cognicic at emospeech-iberlef2024: Exploring multimodal emotion recognition in spanish: Deep learning approaches for speech-text analysis.

Yuxia Wang, Artem Shelmanov, Jonibek Mansurov, Akim Tsvigun, Vladislav Mikhailov, Rui Xing, Zhuohan Xie, Jiahui Geng, Giovanni Puccetti, Ekaterina Artemova, Jinyan Su, Minh Ngoc Ta, Mervat Abassy, Kareem Elozeiri, Saad El Dine Ahmed, Maiya Goloburda, Tarek Mahmoud, Raj Vardhan Tomar, Alexander Aziz, Nurkhan Laiyk, Osama Mohammed Afzal, Ryuto Koike, Masahiro Kaneko, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2025. GenAI content detection task 1: English and multilingual machine-generated text detection: AI vs. human. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Mesay Gemeda Yigezu, Tadesse Kebede, Olga Kolesnikova, Grigori Sidorov, and Alexander Gelbukh. 2023. Habesha@ dravidianlangtech: Utilizing deep and transfer learning approaches for sentiment analysis. In *Proceedings of the Third Workshop on Speech and Language Technologies for Dravidian Languages*, pages 239–243.

Wenxin Zheng. 2024. Ai vs. human: A comparative study of cohesion and coherence in academic texts between human-written and chatgpt-generated texts.

A Appendix

Initialize data collator for padding
data_collator = DataCollatorWithPadding(tokenizer=tokenizer) Use the initialized tokenizer

Define the Trainer without early stopping
trainer = Trainer(
model=model,
args=training_args,
train_dataset=train_dataset,
eval_dataset=dev_dataset,
compute_metrics=compute_metrics, Add custom metrics
data_collator=data_collator Use data collator for dynamic padding
)

training_args = TrainingArguments(
output_dir='./results',

eval_strategy='epoch', Evaluate at the end of each epoch
per_device_train_batch_size=256, Larger batch size if memory allows
per_device_eval_batch_size=256,
gradient_accumulation_steps=2, Adjust for GPU memory usage
num_train_epochs=3, Adjust as needed
weight_decay=0.02,
logging_dir='./logs',
logging_steps=10,
fp16=True, Enable mixed precision
bf16=False, Use bfloat16 if supported (set to True for supported hardware)
dataloader_num_workers=8, Increase CPU utilization for data loading
save_strategy="epoch",
load_best_model_at_end=True,
metric_for_best_model="f1", Use F1 score for model selection
save_total_limit=2, Limit saved checkpoints to avoid large storage use
lr_scheduler_type="cosine", Add learning rate scheduling
warmup_steps=500, Warmup to prevent early overfitting
save_steps=1000, Save model periodically
)

Initialize data collator for padding
data_collator = DataCollatorWithPadding(tokenizer=tokenizer) Use the initialized tokenizer

Define the Trainer without early stopping
trainer = Trainer(
model=model,
args=training_args,
train_dataset=train_dataset,
eval_dataset=dev_dataset,
compute_metrics=compute_metrics, Add custom metrics
data_collator=data_collator Use data collator for dynamic padding
)

from transformers import DistilBertTokenizer
from torch.utils.data import Dataset
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
class CustomDatasett(Dataset):
def __init__(self, examples):
self.examples = examples
def __len__(self):
return len(self.examples)
def __getitem__(self, idx):
example = self.examples[idx]
Tokenize the cleaned text
encoding = tokenizer(
example['cleaned_text'],
truncation=True,
padding='max_length',
max_length=512,
return_tensors='pt'
)
return
'input_ids': encoding['input_ids'].flatten(),
'attention_mask': encoding['attention_mask'].flatten(),
'id': example['id']

test_dataset = CustomDatasett(tokenized_testset)