Text Tug of War: Unmasking the Machines with ML

Ojaswi Dheer (1447227), Rafsan Al Mamun (1407776), William Chen (1400081)
Group: proj1 13

1. Introduction

Recent advancements in artificial intelligence (AI) text generation, exemplified by tools like ChatGPT, have made them widely accessible for everyday use. However, this has also given rise to issues such as academic plagiarism [3]. As discussed by Malinka et al. [3], there is a growing need to detect, prevent, and responsibly integrate these technologies as AI-driven text capabilities become increasingly common in our society.

2. The Problem

The goal of this project is to create a machine-learning model capable of distinguishing between texts generated by humans and AI from a mixed corpus. We train our models using two separate datasets from different domains. These datasets are structured as dictionaries, containing two keys: 'text', representing word sequences mapped to indices from 0 to 4999, and 'label' indicating ‘0’ for AI-generated data and ‘1’ for human-generated. Domain 1 dataset consists of 19500 samples produced by humans and a single AI model, with equal class distribution. Domain 2 dataset, on the other hand, comprises 14500 samples having substantial imbalance, with 2150 human-generated and 12750 machine-generated texts, produced by 7 AI models, each substantially different from the one in domain 1. The test dataset includes texts from both domains, evenly split with unknown labels. Our primary challenge is to construct a model that can efficiently handle samples from distinct domains and adapt to the challenge of imbalanced data distribution.

3. Main Approach

Stratified Undersampling for Balancing Domain 2 Dataset

In preparation for model development, we addressed the class imbalance present in our dataset, specifically in domain 2, which exhibited an unequal distribution between machine-generated and human-generated text. We employed a stratified undersampling method to achieve a balanced representation of both classes while ensuring the proportion of text generated by each of the 7 models remained consistent, preserving the wide diversity. This resulted in 2150 instances each of machine-generated and human-generated text in the final domain 2 dataset. A significant advantage of this approach is its ability to prevent an overemphasis on dominant classes while retaining crucial model-specific characteristics [4]. However, it is worth noting that this reduces the overall dataset size, resulting in potential information loss [4]. Nevertheless, it effectively mitigated our class imbalance issues, enhancing the robustness of our final models.

Light Gradient Boosting Machine and Support Vector Machine

For our initial machine learning models, we utilized a Light Gradient Boosting Machine (LGBM) [5] and a Support Vector Machine (SVM) [1].

LGBM is an ensemble learning technique that combines multiple decision trees to make predictions [5]. It sequentially adds decision trees to correct errors made by previous trees and employs gradient-based optimization to prioritize challenging samples, such as those with domain differences, making it highly effective at capturing valuable information from such features [5]. LGBM also uses a histogram-based learning strategy for gradient boosting. This discretizes continuous feature values into intervals, which is subsequently advantageous when dealing with high-cardinality categorical features or diverse domains [5]. To train LGBM, we employed a bag-of-words model [6], creating a sparse matrix representation where each row captured word frequencies within the corresponding text instance, using them as features.

In contrast, SVM operates by identifying a hyperplane in a high-dimensional space to separate data points into distinct classes [1]. It aims to maximize the margin between these classes by pinpointing the data points closest

to the hyperplane, ensuring robust classification. SVM achieves this by mapping input features into a higher-dimensional space using kernel functions [1]. In our SVM training, we first applied Principal Component Analysis (PCA) [7] to the entire dataset, reducing its dimensionality to 400 components. PCA is a statistical technique for dimensionality reduction that preserves the most informative features and reduces the computational demands of subsequent model training [7].

We followed a 3-step training process for our models:

1. **Training on Domain 1**: Initially, we trained the models on the entire dataset of domain 1 to capture common patterns and characteristics found in human-generated and machine-generated text within it.
2. **Fine-Tuning on Domain 2**: Next, we fine-tuned the models to adapt to the differences between the two domains, accounting for variations introduced by the 7 different models in domain 2.
3. **Fine-Tuning on Mixed Domain**: Lastly, we fine-tuned the models using a dataset that combines both domains, enabling them to generalize better for testing on mixed Kaggle data.

To optimize hyperparameters, we employed a Grid Search [2] algorithm with 10-fold cross-validation. For LGBM, the best settings were 'gdbt' boosting with a learning rate of 0.1 for domain 1 and 'dart' boosting with a learning rate of 0.01 for domain 2 and the mixed data. On the other hand, SVM used an 'rbf' kernel, a regularization penalty of 10, and a kernel coefficient of 0.001 across all three datasets. Both models achieved notable test accuracies, with SVM scoring 0.796 and LGBM reaching 0.800.

Final Approach: Confidence-Weighted Ensemble Model

The two models have unique strengths and weaknesses: SVM has high discriminative power but tends to be more biased, whereas LGBM has high flexibility but suffers from high variance. Thus, to combine the strengths of both models, we utilized confidence weighting to ensemble them, intending to improve the accuracy and robustness of the final model, and enhance its reliability, especially in ambiguous cases.

This final model incorporated a fusion strategy using the predicted probabilities from each base model. For each data instance, we assessed the models' confidence in their classifications. If either model had high confidence (probability > 0.7 for '1' or < 0.3 for '0'), we assigned the corresponding label. When neither model had sufficient confidence, we resorted to averaging their probabilities and assigning appropriate labels. Specifically, if the average was ≥ 0.5, we assigned a label of '1'; otherwise, we assigned '0'.

The mixed model approach yielded significant performance improvements, achieving a test accuracy of 0.814. Furthermore, the ensemble strategy reduced the risk of overfitting, a common concern in single-model approaches. However, this approach results in higher computation demands, due to multiple model evaluations for each data instance. Additionally, the probability thresholds for determining high confidence will require calibration in different contexts. Nevertheless, the substantial performance improvement and enhanced prediction confidence justify the adoption of this mixed model for addressing the text-detection problem at hand.

Model	**Training Time**	**Train Accuracy**	**Test Accuracy**
Light Gradient Boosting Machine	~15 seconds	0.871	0.800
Support Vector Machine	~8 minutes	0.901	0.796
Confidence-Weighted Ensemble	-	-	0.814

Table 1: Performance of the Final Models

4. Discussion

In our initial analysis, we observed that domain 1 consistently had higher classification accuracy compared to domain 2. This outcome was anticipated, given that domain 2 faced a class imbalance issue, leading models to favor categorizing text as machine-generated. To address this, we explored oversampling and undersampling techniques to balance the training data. While conventional wisdom suggested that oversampling would be more effective, recent research by Garcia et al. [8] showed that oversampling often succeeds when generating safer samples in more homogenous class regions. In our specific project, due to the substantial overlap between our classes in domain 2 caused by AI's intention to mimic human text output, oversampling was unlikely to yield significantly safer samples. In contrast, undersampling proved to be a more effective approach, despite the loss of information from the dropped instances.

To begin with, we used a basic SVM classifier [1] as a starting point for our classification task and trained it on both domains' complete data without hyperparameter tuning. However, this initial approach yielded unsatisfactory outcomes, including a lengthy training time of around 28 minutes and a low test accuracy of just 0.6. This poor performance was likely due to our dataset's high dimensionality, featuring 5000 features. In high-dimensional spaces, SVMs can struggle to find effective hyperplanes for linear class separation, leading to reduced accuracy. To address this, we employed PCA [7] for dimensionality reduction and engaged in more rigorous hyperparameter tuning in subsequent iterations to enhance the model's performance.

An alternative to the bag of words that we also tested was term frequency–inverse document frequency (TF-IDF). TF-IDF aims to capture the relative importance of words, considering that some words are more common than others. However, TF-IDF performed poorly on the combined domain dataset, possibly due to differences in word commonness (document frequency) between the two domains. This indicates that calculating word importance using a combined corpus of very distinct domains diminishes information rather than enhancing it. However, this provided the intuition for initially training on domain 1, then fine-tuning on domain 2 and the entire dataset, with the hopes of improving model generalization by mitigating the impact of the text differences between the two domains.

5. Conclusion

In conclusion, our project addressed the intricate challenge of text generation detection with a comprehensive approach. Leveraging a carefully designed ensemble of SVM and LGBM models, coupled with strategic data preprocessing, we achieved notable success in enhancing accuracy and confidence in the classification of human-generated and machine-generated text. This project demonstrates the potential of such intricate techniques in tackling complex real-world text classification problems.

References

1. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(23), 273–297.
2. LaValle, S. M., Branicky, M. S., & Lindemann, S. R. (2004). On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7–8), 673–692.
3. Malinka, K., Peresíni, M., Firc, A., et al. (2023). On the educational impact of ChatGPT: Is artificial intelligence ready to obtain a university degree? *Innovation and Technology in Computer Science Education*, 1, 47-53.
4. Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020). Machine learning with oversampling and undersampling techniques: Overview study and experimental results. *Information and Communication Systems*, 243-248.
5. Ke, G., Meng, Q., Finley, T., et al. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 3146–3154.
6. Zhang, Y., Jin, R., & Zhou, Z. (2010). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, 1, 43-52.
7. Pearson, F.R.S. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11), 559–572.
8. García, V., Sánchez, J., Marqués, A., et al. (2020). Understanding the apparent superiority of over-sampling through an analysis of local information for class-imbalanced data. *Expert Systems with Applications*, 158, 113026.