LLM Classifier Model

Léonard Verschuere; D24126122

Adrien Le Corvec; D24126131

Business Technology, Faculty of Business

Technological Univerity Dublin, Ireland

Programme: TU5535

Module: Predictive Data Analytics

Lecturer: Dr Wael Rashwan

Email: D24126122@mytudublin.ie; D24126131@mytudublin.ie

The decision to work with this dataset is grounded in its unique combination of features, rich contextual diversity, and relevance to ongoing developments in natural language processing (NLP) and AI model evaluation. This dataset offers a structured opportunity to explore the nuanced comparisons between different AI-generated responses, leveraging prompts, responses, and model performance indicators to gain insights into model behaviors, strengths, and potential limitations.

Firstly, the dataset provides examples of prompts paired with responses from various AI models, which makes it ideal for understanding response quality across multiple models. In today's NLP field, where numerous language models with different training architectures (e.g., open-source vs. proprietary) are continually emerging, a dataset that allows direct comparison of model outputs is highly valuable. It provides an empirical basis for evaluating model differences, helping to identify whether specific models consistently perform better, worse, or equally well under different circumstances.

Additionally, the dataset incorporates valuable annotations on winning models and ties, which gives us a concrete metric to predict. With such data, we can examine not only the win-loss patterns but also contextual clues (like prompt types or response length) that might drive these outcomes. A model trained on this dataset can help predict the likelihood of one model winning over another, which in turn aids model selection and optimization efforts. It can also guide research teams in benchmarking models and identifying areas where performance enhancements are needed.

In summary, this dataset was chosen because it provides a fertile ground for exploring inter-model dynamics, comparative analysis, and feature extraction in AI response generation, which are valuable for advancing the understanding and development of competitive, reliable NLP models. It serves as an effective bridge between theoretical machine learning and practical applications, where understanding response quality and consistency is crucial. This dataset holds the potential to shape informed, data-driven strategies for AI deployment and improvement in real-world scenarios, making it an excellent foundation for this project.

Method

For this project we tried using different kind of models such as :

- Linear Regression, to try and label some potential existing patterns such as text length,
- Decision Tree, for its ability to learn via simple decision rules inferred by the data structure, like text sentiment or text length,
- Random Forest Classifier for its known strength in classification, regression, improving accuracy and overfitting control

We then found that we had the best results with the Random Forest Classifier by a significant margin (10+ %), which wasn't really a surprise since it was our favorite choice during the benchmark we did, but we still wanted to try some other options to be sure that would pick the best suited choice.

After choosing the Random Forest Classifier, for its effectiveness in handling complex patterns, we aimed to enhance our accuracy and class balance through systematic hyperparameter tuning.

Key parameters we tuned included:

Number of Estimators: We tested values from 50 to 500 to assess the impact of adding trees on stability and accuracy.

Maximum Depth: Setting maximum depths from 10 to 60 allowed us to manage model complexity and generalization.

Class Weights: Given class imbalances, we used balanced weights to improve recall for underrepresented classes.

During our experiment, we also changed the different variable from the dataset the model will learn from, to determine which had the biggest impact on predictions and accuracy.

Preliminary Experiments

Our tuning produced the following key results:

The first model we have tried treated all text data from the dataset as a single feature, combining them. This proved to be inefficient for the model, as it could not learn from response_a or from response_b. We had an average accuracy of 0.40.

In the second model, we treated each text data as a different feature, and mainly used TFIDVectors to change how our text features were treated by the model. This proved to be better, as the model had an average accuracy of 0.45, but with disparity in the precision, recall and f1-score: winner_model_a and winner_model_b were doing the same scores, while tie was behind.

The third model used the second one as a base, and we used loops and GridSearchCV from sklearn to hypertune our parameters, (such as the number of estimators in our tree, the max depth at which the tree would grow, the minimum number of leaf, the minimum number of splits), increasing its accuracy and result balance: The final model had an average accurary of 0.55, which is far from good, but better than anything we've had with the RandomForest. The winner_model_a and winner_model_b classes showed consistently higher precision and recall, while the tie class had lower recall despite improvements from class balancing. This remains an area for further exploration to improve the model's sensitivity to ties.

Next Steps

Now, based on our preliminary results, here are the steps we are considering :

Feature Optimization and Engineering

Refining and expanding features is key to improving model performance by increasing predictive power and reducing noise. Testing additional features, like embeddings from advanced NLP models (e.g., BERT, GPT), can capture richer semantic information, enabling the model to better understand

nuances between prompts and responses. Additional metrics, such as sentiment scores or readability indices, may further enhance model accuracy.

Dimensionality reduction and feature selection, using methods like PCA or feature importance scores, can reduce the feature set to only the most relevant attributes, boosting efficiency and interpretability. By carefully engineering features, we better capture key patterns that drive model outcomes, leading to a more streamlined, effective model.

Robustness Testing and Validation

Robustness testing ensures a model performs reliably across varied scenarios and remains resilient to data changes. After promising results in training, we validate it with new, unseen, or slightly modified data to test generalizability. For example, introducing variations in prompt phrasing or response lengths simulates different real-world inputs, uncovering vulnerabilities and strengthening model resilience.

Domain-specific testing is equally important. Simulating scenarios in relevant fields (e.g., customer service) in a controlled environment highlights any performance gaps specific to those contexts. Together, these tests confirm the model's reliability and robustness under diverse conditions, supporting consistent performance in practical applications.

Iterative Model Retraining and Updating

In many real-world scenarios, data patterns evolve, so it's crucial for models to adapt. Periodic retraining ensures models stay accurate by incorporating new data, such as shifting language or emerging topics. Automating the data pipeline—covering data collection, preprocessing, and retraining—minimizes manual effort and allows for continuous updates. This iterative process ensures models remain relevant, reliable, and effective in dynamic environments.