Chatbot Arena Winner & Hardness Score Predictions Using Supervised Machine Learning Models

Ali Shazal, Gary Patterson, Sebastian Mejia
{ali.shazal, rigpin2014, smejia}@berkeley.edu
DATA200
University of California, Berkeley

ABSTRACT
The innovation of Large Language Models (LLMs) is creating new models with varying degrees of capability at a rapid pace. An inherent need to benchmark these models exists to determine which of these models is the best performing. Chatbot Arena is a platform where LLMs compete in head-to-dead battles to rank the best performing models. Winning a battle is determined by a crowdsourced user determining which LLM had the best response to a specific prompt. These chatbots are designed to cater to a wide range of users, whose questions often span fields requiring varying levels of expertise. This study extends the previous work by Chang et al. by analyzing the data from Chatbot Arena. We have developed supervised machine learning models to predict both the winning chatbot in head-to-head battles and the hardness score of the user prompts, using features such as text embeddings, model names, Elo ratings, and prompt clusters. We report a 55.5% test set accuracy in predicting the winning mode using a Logistic Regression model, and a 2.53 test set mean squared error (MSE) for hardness prediction using an ARDRegression model. Our findings contribute to a deeper understanding of LLM capabilities and offer effective methodologies for evaluating and predicting their performance.

1. INTRODUCTION

LLM Agents are becoming increasingly central to AI applications, but concerns remain about their reliability across diverse topics and queries. These chatbots are designed to cater to a wide range of users, whose questions often span fields requiring varying levels of expertise. To better understand LLM performance, Chiang et al. introduced the idea of "hardness" for categorizing user prompts based on their complexity [1]. They applied the DBSCAN clustering algorithm and found significant performance differences in GPT-4-0613's responses. For example, it performed exceptionally well (96.7% accuracy) in computational tasks like Python Game Programming Challenges but struggled (53.3% accuracy) with subjective tasks like Movie Recommendations and Ratings. Their study highlighted that while GPT-4 excels in logic-heavy tasks, it tends to falter in subjective or reasoning-based queries.

Another study by Moros-Dávalos et al. explored using linguistic meta-features to predict the difficulty of prompts [2]. Features like negation and vocabulary complexity were shown to increase the cognitive load for both humans and models, impacting LLM performance. They also examined factors like noise and modality to enhance prompt difficulty predictions. Their work suggested that these meta-features could complement traditional readability metrics, but the broader applicability of this approach remains uncertain. Both studies underscore a common challenge for LLMs: reasoning tasks that require both logic and contextual understanding. For instance, answering a question about the most likely cause of a forest fire, given it wasn't started by humans, demands reasoning beyond simple computations.

In our study, we extend these insights by analyzing a broader range of chatbot models and their head-to-head matchups. While prior studies focused primarily on GPT-4, we investigate multiple chatbot models, using features like embeddings, prompt clusters, and user preferences. Unlike Chiang et al., who used DBSCAN for clustering, we opted for k-means clustering, prioritizing simplicity and interpretability [3]. Our work builds on these foundations by not only evaluating LLM performance but also developing models to predict both the winning chatbot in a conversation

and the hardness of prompts. By exploring a wider set of models and tasks, this research offers a more comprehensive understanding of LLM capabilities. Specifically, this study will be conducted in two tasks, each building its own supervised machine learning model for accomplishing its goal. Task A is focused on predicting the winning model of the Chatbot Arena battles. Task B is focused on predicting the hardness score of the prompt used for each battle.

2. DATASETS

The original dataset is obtained from the HuggingFace Chat Bot Arena Conversations and contained over 33,000 observations in multiple languages [4]. The data available to this study has been preprocessed to remove non-English conversations, conversations with more than one round, and conversations classified as toxic or harmful. All of the data in this study is granular at the level of battles between two models, each row representing data from an individual battle. Each dataset contains 25,282 observations of chatbot battles. The data was analyzed using commonly available scientific programming libraries in the Python language.

2.1 Conversation Data

The conversation data contains qualitative nominal features in JSON line format. The data was read into Python using a Pandas dataframe. There are no missing or null values in the data. The features in this dataset include: the question id, two models, the winning model, and the conversation between each model and the judge. This conversation data required minimal cleaning to extract the prompts given to the models and the responses from each model. The prompts and responses were extracted from the conversation and made into individual features to be evaluated. To evaluate the prompt, the distribution of prompt length was plotted as shown in figure 1. The distribution is slight-skewed and showed that 84% of prompt lengths are less than 250 characters.

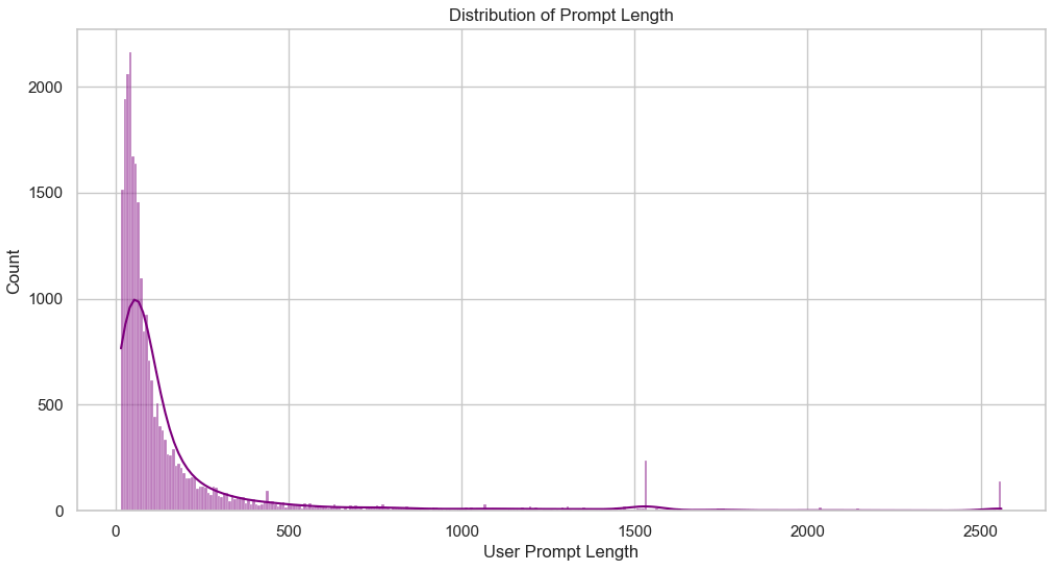

Figure 1: Distribution of user prompt length.

To determine which of the models was the best performing the average win rate for each model was determined. The models were then plotted against each other with their respective win rate. The best performing model was ‘gpt-4’, which was expected based on the resources and funding behind that model. It was surprising that the ‘llama-13b’ model was the worst performing model for the same reason ‘gpt-4’ was the best performing model. The results are summarized in figure 2. Once the top performing models were determined, additional analysis was performed based

on the response length of the top five models. This analysis showed that most of the top performing models responded most frequently in less than 1000 characters. With the exception of the ‘claude-v1’ model which responded most frequently with approximately 1800 characters. The distribution of response lengths for the top performing models was right-skewed, indicating shorter responses are more frequent. The best performing model ‘gpt-4’ responded most frequently with approximately 400 words.

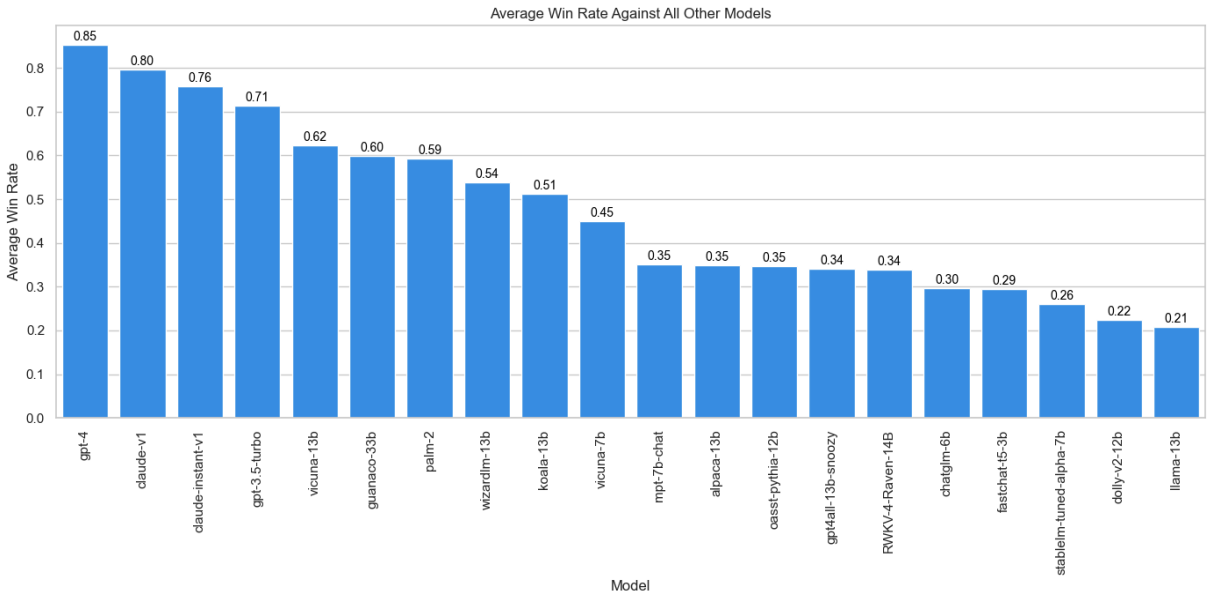

Figure 2: Win rate of each model against all other models.

2.2 Text Embeddings

The text embeddings are 256-dimensional text embeddings for the prompt and responses from the two models in the battle, contained within three auxiliary datasets in Numpy binary file format. There is not a unique key in this data to link it with the conversation data. It is assumed that each observation directly corresponds with an observation in the conversation data. There are no missing or null values in the data. Analysis of the embeddings during EDA involved taking a sample of 1000 embeddings and computing the dot product similarity and returning the top five most similar prompts and responses. Then t-SNE and k-means clustering was performed on a sample of 1000 prompt embeddings to create ten clusters reduced to two dimensions. The winning model for each battle was visualized in the respective cluster as shown in figure 3. We observe that there are definite clusters corresponding to similar prompts. Cluster nine, however, is dispersed throughout the scatter plot. This likely represents prompts from the sample which do not have a majority similarity to a specific cluster.

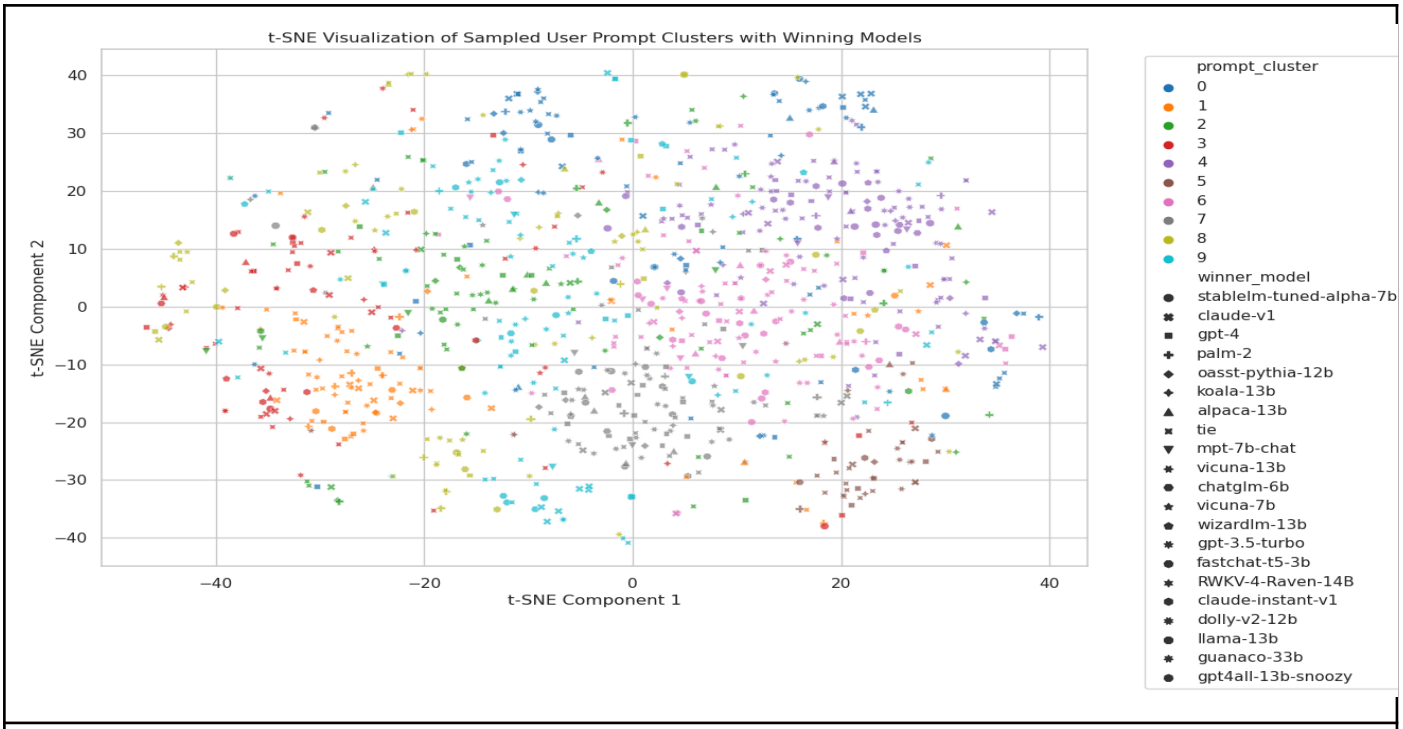

Figure 3: Prompt clustering of winning models from a sample of 1000 battles

The same analysis was performed only for the top five performing models to determine which response cluster contained the most number of wins. For this analysis the number of clusters was reduced to five, the results are summarized in figure 4. It can be seen that cluster two has the most winning models on average while cluster three has the most counts for all models with the exception of 'gpt-3.5-turbo' and 'gpt-4', which have their most counts in cluster two.

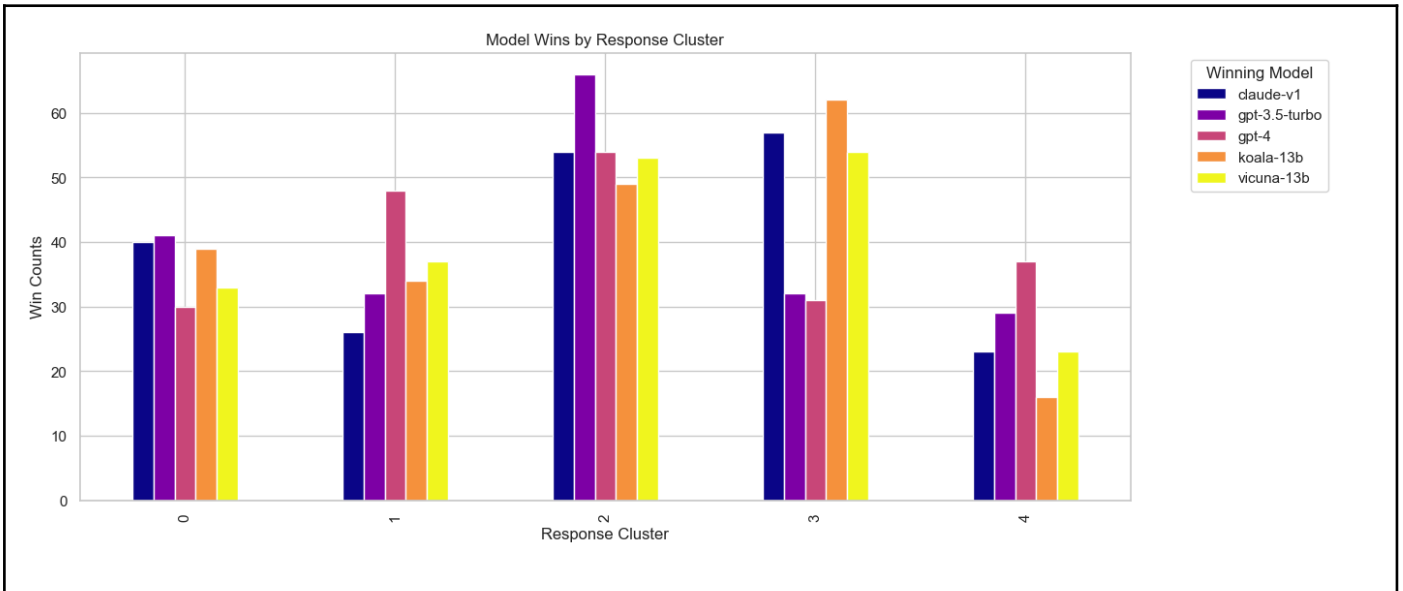

Figure 4: Prompt clustering of top-five performing models from a sample of 1000 battles

2.3 Topic Modeling and Hardness Score Data

The topic modeling and hardness score data contains qualitative nominal features and quantitative nominal hardness

score in JSON line format. The qualitative features are the question ID, the string of text in the prompt, some probabilistic responses from GPT3.5 providing three responses each for ‘topic modeling’, ‘score reason’, and ‘score value’. The score value is the target variable for Task B. This data contained 26 null values and required moderate cleaning. Specifically, some of the probabilistic responses from GPT3.5 were contained within multiple nested lists, and the data needed to be extracted from the lists. The introduction of null values in this dataset required careful handling to ensure the integrity of the data. An additional feature in this dataset was a nested dictionary which contained all of the data in this set. Evaluation was given to the null values to determine if a parsing error resulted in the missing values. After parsing the data from the nested dictionary only a few null values were correct. Observations with null values were dropped from the analysis on the basis of proportion. Dropping this small number of observations relative to the total number of observations is assumed to have a negligible effect on skewing the data or introducing bias.

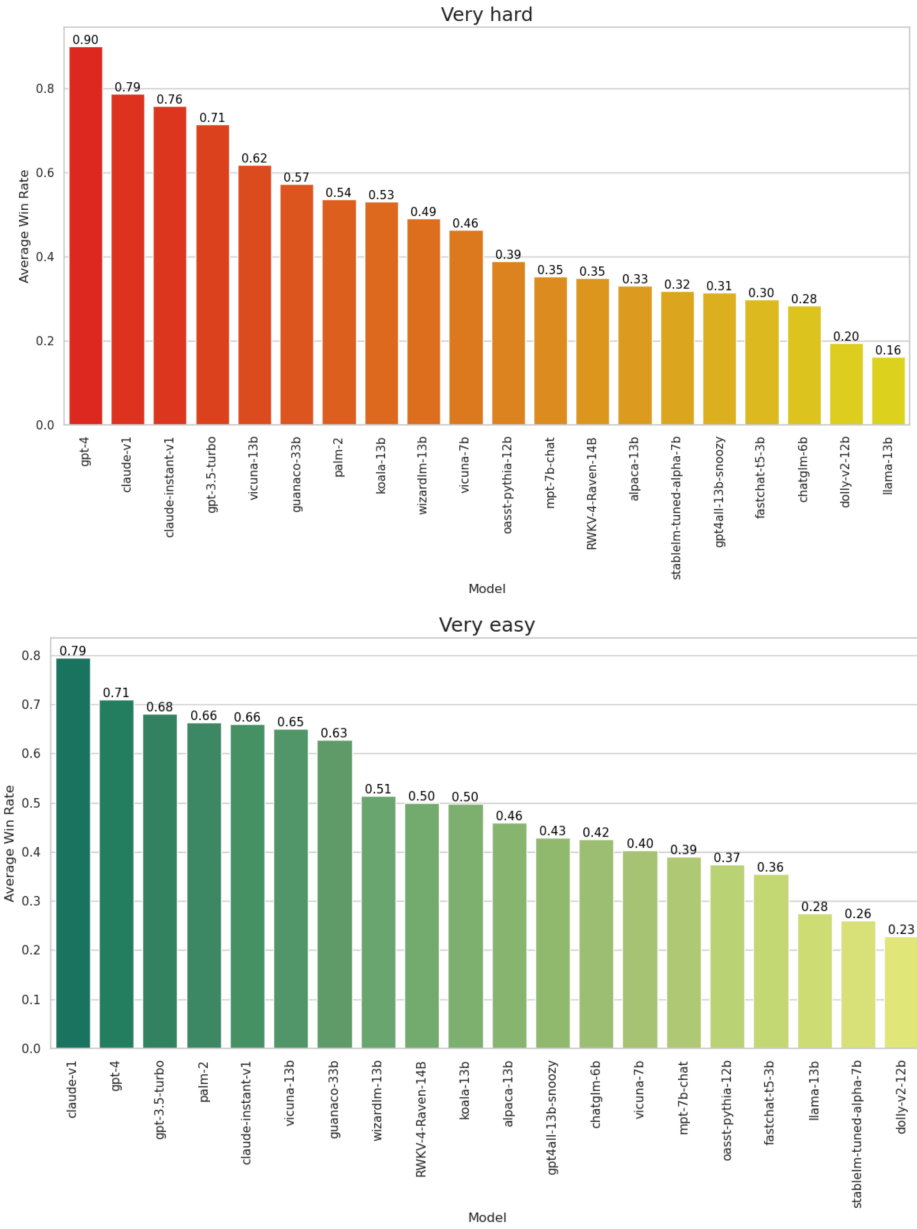

Figure 5: Win ratio versus model for ‘Very Hard’ and ‘Very Easy’ hardness categories

To evaluate this data the hardness scores were grouped into bins representing the difficulty of the question: 'Very Easy', 'Easy', 'Medium', 'Hard', and 'Very Hard'. Each bin contained two values of hardness score. For example, 'Very Easy' contained observations of hardness scores 0 and 1, and so on. Several evaluations were performed based on these bins including plotting the win ratio for the top five models versus hardness categories, plotting the win ratio versus all models for a specific hardness category shown in figure 5, and plotting the distribution of the euclidean distance of the prompt embedding to the response embedding for each hardness category.

By observing figure 5 the results vary as compared to the results of figure 2. The 'gpt-4' model is not the best performing model for all hardness categories. Similarly, the 'llama-13b' model is not the worst performing.

3. TASK A - PREDICTING WINNER CHATBOT MODEL

3.1 Description

Our first task is to predict which chatbot model will be picked as the winner by the user given the above data i.e. a conversation prompt, two chat model names with their corresponding responses to the prompt, 256-dimensional embeddings of the prompt and responses, and other features described in Section 2. Our goal is to create a model that would give 54+% accuracy on the train, validation and test set.

3.2 Feature Engineering

Based on the EDA from section 2, we have used the following features for the modeling task.

3.2.1 Text Embeddings

We are using all three sets of embeddings: prompt embeddings, response A embeddings, and response B embeddings. The reason for this is that the actual prompt and responses directly contribute to which chatbot model gets picked as the winner by the user, which is why we are using their numerical representation in the form of text embeddings. Figure 6 shows us that some models perform better on some clusters of prompt embeddings, and users tend to also favor responses under certain clusters of response embeddings.

We conducted the following experiments when deciding the usage of text embeddings as features:

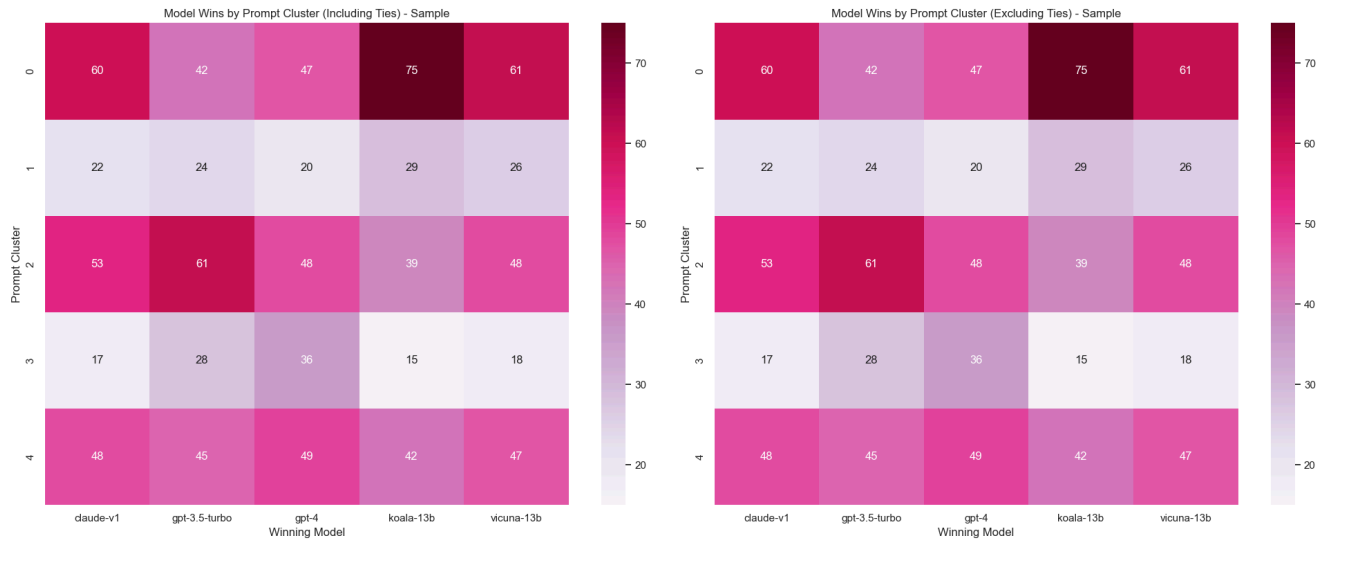

Figure 6: Certain models stand out in specific prompt clusters. For example, GPT-4 shows strong performance in Clusters 0, 2 and 4, showing that it is better suited for certain types of prompts compared to other models. Similarly, GPT-3.5-Turbo did well in Cluster 2, and Koala-13B performed strongly in Cluster 0. Claude-V1 also performed well in Cluster 0. All the models were weak in Clusters 1 and 3.

1. Using each set of embeddings as one feature by taking the mean of all embeddings - we treated this as the baseline.
2. Using each embedding as a separate feature i.e. 256 x 3 = 768 features. This gave a 12% accuracy increase from the baseline.
3. Using dimensionality reduction with Principal Component Analysis (PCA), through scikit-learn library [2], to reduce the number of embedding features from 768 to 400. This gave a 10% accuracy increase from the baseline.

Since the second option provided the best results on the train and validation sets, we used that for the final model.

3.2.2 Model Name

We are utilizing one-hot encoding to convert the model names as binary features, using scikit-learn [5]. We can see in Figure 2 that the number of total wins can vary significantly depending on the model, so having the model itself as a binary feature will provide meaningful information to the model.

3.2.3 Elo Rating

Moreover, as the modeling task is based on predicting a winner chatbot model against another model, the elo rating is relevant. Elo ratings provide a way to compare competitors directly, with higher ratings signifying greater skill [6]. We have added the elo ratings of both the competing models with each prompt [7][8]. Figure 2 shows the win rate of each chatbot model against all other models.

3.2.4 Prompt Cluster

We also included the prompt cluster of each prompt as a separate feature. This was done by first using PCA for dimensionality reduction of prompt embeddings to 150 components and then using K-Means clustering with 4 clusters [3]. These hyperparameters were identified in a series of experiments to achieve the best accuracy on the validation set. The reason we opted for including the prompt cluster on top of using the text embeddings separately was to provide complementary information. The 256-dimensional text embeddings represent granular, high-dimensional semantic information from the user's prompt whereas the prompt cluster provides a lower-dimensional, categorical representation that summarizes the embeddings into broader groups based on similarity. By including both, we aim to preserve detailed information while also providing the model with an abstract, high-level feature that can capture broader clusters in the data. Figure 3 shows that there are concentrations of certain models in specific clusters, which demonstrate areas where these models perform better. Some clusters appear more densely packed with points (e.g., clusters 1 and 3), while others are more spread out.

3.2.5 User Prompt Outliers

In order to make sure that our model does not get skewed by outliers in the user prompt, we dropped the prompts with character length greater than 250. This was done based on the fact that around 84% of the prompts are 250 or less characters long. Figure 6 shows the distribution of user prompt length.

3.2.6 Feature Exclusions

In creating the feature set, certain features were excluded due to their lack of predictive value for the task. This included user prompt lengths and model response lengths as there was no correlation between them and the winning model. We also excluded the chatbot model's predicted topic as a feature due to its high cardinality, with over 10,000 unique values in the dataset of ~25,000 rows. Including such a feature would have significantly increased the dimensionality of the model, potentially leading to sparsity and reduced generalizability. Additionally, the computational cost of encoding this feature outweighed its potential predictive benefits.

3.3 Methodology

3.3.1 Logistic Regression Model

Our task is to predict the winner model given some features. Essentially this is a classification task with the following

4 classes, as given in the data: ‘model_a’, ‘model_b’, ‘tie’, ‘tie (bothbad)’. We opted for a Multi-class Logistic Regression model due to its performance on classification tasks, quick iterability, and ability to capture linear relations well [5][9]. The loss metric used is categorical cross-entropy loss.

For hyperparameter tuning, we used grid search to determine the best regularization technique, regularization parameter, solvers, the number of iterations, and the cross validation folds. To avoid overfitting, our experiments provided the best results with L2 regularization and 5-fold cross validation. This also helped us identify the performance of the model from a bias-variance tradeoff. However, it is important to note here that cross validation resulted in a 1% decrease in accuracy on the validation set and increased the training time from 10 seconds to more than an hour. That is why we ended up dropping cross-validation from our best performing model to save time and resources; it still fulfilled our requirement of 54+% accuracy on the unseen test set.

Our best performing system uses the Limited-memory BFGS solver, 1000 iterations, L2 regularization, and no cross validation [10]. In all our experiments, we use an 80-20 train-validation split as it is a commonly used ratio in machine learning experiments. This split provides a large enough training set for the model to learn meaningful patterns while keeping an adequate validation set to reliably assess model performance. Alternative splits of 90-10 and 70-30 were considered, but the 80-20 split gave the best results in our initial experiments.

3.3.2 Other Models

We also experimented with two other systems.

1. Random forest with the same features as above and default scikit-learn settings [2]. This gave a much lower accuracy on the validation set compared to the logistic regression model. The reason for that was overfitting due to the high dimensionality of input embeddings (e.g., 256-dimensional vectors for user prompts and responses). This overfitting resulted in a lack of robustness when generalizing to the test data.
2. A simple neural network with hyperparameter tuning [5]. The best performing neural network on the validation set ran with a batch size of 32, dropout rate of 0.2, L2 regularization strength of 0.01, learning rate of 0.0001, one hidden layer, and 256 neurons per layer. While this system performed better than random forest, it still did not beat the logistic regression model. There were several reasons for that. One, we saw the need for significantly more data to effectively train the large number of parameters in order to make it more generalizable; even though training accuracies went up to 90%, validation set accuracies still remained below 50%. The high dimensional input combined with limited training examples led to vanishing gradients and convergence issues. Despite hyperparameter tuning, the added complexity of neural networks failed to justify their marginally improved performance. Moreover, in terms of time and resources, the neural network training took 10x more time on average compared to the logistic regression model. Thus, in the interest of time, training resources, simplicity, and generalizability, we opted to move forward with the logistic regression model instead of further exploring neural approaches.

The results of our modeling approaches are described in section 5.

4. TASK B - HARDNESS PREDICTION

4.1 Description

The focus of Task B is to predict the hardness score of the prompt. To predict the hardness score of the prompt we make the following assumption:

- Hardness score is independent of the model evaluating the prompt because any prompt could be evaluated by any model. Therefore, the score is also independent of any feature related to the model, such as the winning model, or the model response.

Given the available data this problem is quite difficult. Additionally, the target hardness score is the predicted hardness

score from GPT3.5. This would make the only relevant features for predicting the hardness of the prompt be: the prompt embedding, the string of text from the prompt, and the features provided by GPT3.5 (topic modeling, score reason, and score value). Since the score reason is not provided in the test data, it cannot be used as a feature in training the model. Additionally, using the topic modeling data as a feature becomes difficult if the topics in the test data are not in the training data. A short analysis on the test data reveals that out of ~2000 unique topics in the test data, only ~800 are in the training data. Therefore any feature engineering of the topic modeling data does not assist in making predictions for hardness scores for the majority of topics in the test data. This leaves the only features available for making predictions to be the string of text in the prompt, and the prompt embedding.

4.2 Feature Engineering

Based on the description above, the only features for making predictions are the string of text in the prompt, and the prompt embedding. These features alone do not provide sufficient information for making predictions. In order to make accurate predictions of the hardness score several features were derived as described below.

4.2.1 Prompt Embedding Statistics

Several statistics were computed for each prompt embedding including: mean, variance, skewness, kurtosis, L2 norm. The mean of the embedding represents the average value of all the dimensions in the vector. A mean close to zero would indicate that the embedding is well centered, otherwise the embedding may contain some asymmetry. The variance of the embedding measures how spread the values are across dimensions. A high variance would suggest that certain dimensions contain more varied information and vice versa. The skewness measures the asymmetry of the distribution of values in the embedding and quantifies the degree to which the distribution is skewed. Kurtosis quantifies the number of outliers in the distribution and evaluates if certain dimensions have outliers that dominate the embedding. The L2 norm is the euclidean norm and provides a measure of the overall length of the embedding vector, which is useful for comparing the relative scale of the embeddings.

4.2.2 Prompt Text Features

A few very basic features were derived from the string of text of the prompt including the number of words, number of characters, and number of unique words. While evaluating the topic modeling data it was determined that there was a relationship between the length of the prompt and the average hardness score for a topic, therefore the length of the prompt was included as a feature. Based on this relationship, the number of words and number of unique words should also be relevant predictors of the hardness score.

4.2.3 Prompt Embedding Clustering

Using a similar approach that was applied to the prompt embeddings for Task A, k-means clustering was used to develop features for predicting hardness score. To derive these features PCA was used for dimensionality reduction to 150 dimensions. This was shown in Task A to capture 86% of the variance of the embedding. Then k-means clustering was used to cluster all prompt embeddings into four clusters. The distance for each prompt embedding to each cluster centroid was calculated and used as a feature for prediction.

4.2.4 Cosine Similarity with the Average Embedding For Each Hardness Score

Since GPT3.5 provides probabilistic responses, the average embedding was determined for each score, for each score value. The average embedding was determined by filtering the prompt embeddings by their hardness score, then determining the average of them. By computing the cosine similarity of each prompt embedding to the average embedding for each score, for each score value, thirty features were created to make predictions. The cosine similarity was chosen over the dot product similarity because it is independent of the magnitude of the embedding vector, meaning that the semantic similarity is emphasized. This feature represents how semantically similar each prompt is to the average embedding for each score.

4.3 Methodology

The purpose of Task B is to predict the hardness score of each prompt using a linear regression model. Additional

evaluation was given into predicting the hardness scores as a classification problem and is discussed further in section 5. To predict the hardness score using linear regression, several linear models from the SKLearn library were evaluated. SKLearn has many linear regression models available for supervised learning tasks [5]. In order to determine which model performs the best with our data, a training loop was created which iteratively looped through several models to determine which model had the best performance. The models chosen for evaluation were:

- Linear Regression
- Ridge
- Lasso
- ElasticNet
- SGDRegression
- BayesianRidge
- ARDRegression

Before training the models we verified that each model accepts normalized feature values between zero and one. The data was then scaled using StandardScaler. Inside of the training loop each model is trained using 5-fold cross validation. The model with the lowest MSE from cross validation was used to make predictions about the validation set. Ideally, the final model would be chosen based on lowest MSE versus the validation set. This would represent the model which performs the best against unseen data. Our best model was determined by chance based on previous feature engineering schemes. This is discussed in more detail in section 5.

5. RESULTS AND DISCUSSION

5.1 Task A

Model	Approach	Experiment	Val Accuracy %	Test Accuracy %
Logistic Regression	1	Avg of vector embeddings as one feature	36.0	-
Logistic Regression	2	Each vector embedding as a separate feature	48.7	-
Logistic Regression	3	Approach 2 with user prompts <= 250 chars	51.0	-
Logistic Regression	4	Approach 3 with dimensionality reduction on embeddings	46.0	-
Logistic Regression	5	Approach 3 with L2 regularization and 5-fold cross validation	49.1	-
Logistic Regression	6	Approach 3 with elo rating	54.7	-
Logistic Regression	**7**	**Approach 6 with prompt embeddings cluster**	**55.7**	**55.5**
Random Forest	8	Features from approach 3	45	-
Neural Network	9	Features from approach 3	49.5	-

Table 1: Task A val and test set results

When trying to predict LLMs' difficulties with user prompts, we find that granular text embeddings and complementary prompt clusters based on their embeddings serve as accurate predictors by providing rich, nuanced

linguistic features to the model. This, combined with filtering length outliers, dimensionality reduction, and other approaches, provide a higher accuracy than the baseline experiment of using the average of vector embeddings as one feature. Some of these models have a low ceiling, such as L2 regularization in which we only achieved minimal improvements through tuning the hyperparameters.

Dimensionality reduction, while useful for reducing computational complexity, had mixed effects on the performance of our models in Task A. Techniques like PCA reduced the embedding dimensions significantly, but this came at the cost of losing critical information. Since the principal components prioritize variance across all features, the model might have discarded subtler but highly predictive dimensions specific to the decision boundary between chatbot responses. For instance, nuanced linguistic features in the embeddings tied to conversational tone or context-specific relevance may have been diluted during dimensionality reduction. This reduced the ability of models like random forests or logistic regression to separate classes effectively, leading to lower classification accuracy.

Elo ratings worked effectively for the task because of their ability to capture pairwise comparisons, which directly aligns with the nature of the winner modeling task. These ratings dynamically adjust based on match outcomes, providing a nuanced representation of relative strengths while smoothing out noise from isolated poor performances. As a compact and interpretable feature, Elo ratings complemented embedding-based features and topic clusters by introducing a performance-driven signal, effectively summarizing historical interactions in a way that enhanced the predictive model's accuracy. Thus they were a key contributor to the model's success.

5.2 Task B

Model	Approach	Feature Engineering Scheme	Train/Val/Test MSE
Ridge	1	Blatant data leakage	1.20 / 0.80 / -
ARDRegression	2	Assuming average hardness scores based on topic	2.11 / 1.80 / 4.8
BayesianRidge	3	Fold based	2.55 / 2.78 / -
ARDRegression	**4**	**Statistics based**	**2.50 / 3.01 / 2.53**

Table 2: Task B train, val and test set results

The initial approach contained results which were misleading due to significant data leakage. The data leakage was a result of directly including information about the target variable in the training data.

A second approach involved assuming the average hardness score for each prompt based on the topic modeling data. Since one of the topic modeling predictions is directly associated with the target variable, the other two topic modeling predictions were used to assume the scores. The assumed score for a topic was the average score for that topic. For the test data, if a topic was not in the training data, the assumed score was chosen based on the most similar prompt in the training set. The topic was assumed to be that of the most similar prompt in the training set, and the applicable scores were assumed based on the topics of that prompt. Although this method showed promising results versus the validation set, the MSE versus the test data was 4.8. This result was surprising, but makes sense. The prediction heuristic introduced by assuming the topic in the test data was the same as the most similar topic in the training data is an inherently bad heuristic. Earlier discussion about topics in the test data revealed that the majority of topics were not in the training data. This means that the most similar prompt could potentially not be a good indicator for the hardness score of the test prompt.

The third approach was an interesting approach for including information about the target variable as a feature while

preventing data leakage. This approach explored the question “when is it okay to include information about the target variable as a feature for modeling”. The solution to that question is fold based feature engineering. Information about the target variable can be included as a feature as long as it is indirectly obtained from the same dataset. To do this the data was split into five folds. For the validation set of each fold, the cosine similarity of each prompt was determined against each prompt in that fold's training set. The five most similar prompts and their associated scores were used as features for predicting hardness scores. This process was repeated for each training-validation set in the five folds to obtain an enriched dataset. This process was not applied to the test data because the MSE of the validation data did not yield sufficient results.

The fourth approach used a statistics based approach to predict the hardness scores based solely on the string of text of the prompt and the prompt embedding. The feature engineering for this method is discussed in more detail in section 4.2. After looping through the SKLearn linear regression models listed in section 4.3, the results were slightly worse on the validation set than the third approach. However, this approach had the best results so far versus the training set with an MSE of ~2.5 for Ridge, BayesianRidge, and ARDRegression. The MSE for the training data is slightly biased because it is computed before rounding the predictions to integers. By normal model selection criteria, the Ridge model would have been chosen as the final model based on an MSE versus the validation set of 2.99. However, the MSE of the Ridge model versus the test model was 2.78. By chance, we decided to test the ARDRegression model against the test data, since this was the best performing model for the second approach. The ARDRegression model evaluated against the test data with an MSE of 2.53. This result is surprising considering the bias in the MSE of the training data, and the MSE of the validation data of 3.01.

6. SOCIETAL IMPACTS AND ETHICAL CONCERNS

As the availability of open-source LLMs becomes increasingly broad, a need to rank these models will always be present. There will be a significant effect on society as the capability of LLMs to perform different tasks with high accuracy increases. The ethical concern behind this issue is “how is society going to manage the transition into the era of AI?”.

One major societal concern with increased use of LLMs to perform tasks is the loss of jobs. As companies decide to automate their processes with AI, the workers who previously performed these tasks are now forced to re-skill in order to continue to provide for themselves and their families. But whose responsibility is it to ensure that these workers are able to continue to provide for themselves? Is it the company that fired them, is it the government in which the individual resides, or is it solely up to the individual? This is where the ethical concern of AI comes into focus. How can we ensure that AI is a tool that people of all demographics can benefit from? As companies automate their processes with AI we can see that the tool is mainly benefiting the companies, and the people who lose their jobs are suffering as a result of its use. It should be clear that all stakeholders in this situation need to take action to ensure that the use of AI is ethical and benefits all demographics as much as possible.

Additional issues have arisen regarding the ethics of innovating LLMs regarding inherent bias interpreting information from different demographics. It is crucial to include innovators from diverse ethnic backgrounds to prevent inherent bias in the training of LLMs. As we are modeling and predicting user preferences, there is the risk of amplifying or perpetuating these biases, especially if they show discriminatory behaviors. The outcomes of this paper can be applied to rank chatbot models that compromise fairness. Moreover, metrics like the elo rating are lossy and indexing heavily on them can take away from qualitative characteristics including, but not limited to, transparency, fairness, and factual correctness.

7. CONCLUSION

In this paper, we tackle the problem of identifying the best-performing open-source LLM and predicting the hardness of user prompts. In Task A, we predict which model performs best in a series of battles. Our Logistic Regression

model achieved a test accuracy of 55.5% using prompt embeddings, response embeddings, Elo ratings, and prompt clusters, demonstrating the effectiveness of these features in capturing user preferences. Logistic regression outperformed other models by effectively generalizing linear patterns while minimizing overfitting. In Task B, we predict the difficulty of prompts presented to models during battles. Our ARDRegression model achieved the best results with a test MSE of 2.53. Using a statistics-based feature engineering approach, we predict hardness scores based on the textual content of the prompt and its embedding. Both models were chosen for their simplicity, interpretability, computational efficiency, and validation set performance.

We believe this effort will support future research aimed at understanding why certain LLMs are favored by chatbot users. This, in turn, will enable the development of better LLM agents, user interfaces, and overall user experience. Future research should also address ethical concerns related to fairness, model biases, and the societal risks posed by AI models.

8. VIDEO PRESENTATION

The video presentation of this report can be found at https://youtu.be/CjE3KPQiOFA.

9. ACKNOWLEDGMENTS

We would like to thank the project TAs Rose Niousha and Jessica Lin for their help throughout the project.

10. REFERENCES

[1] Chiang et al., Chatbot Arena: an open platform for evaluating LLMs by human preference. arXiv.org. https://arxiv.org/abs/2403.04132

[2] F. Moros-Dávalos, F. Martínez-Plumed, and J. Hernández-Orallo, “Using Linguistic Meta-Features to Predict Prompt Difficulty for GPT-4,” *arXiv preprint arXiv:2303.09112*, 2023. [Online]. Available: https://arxiv.org/abs/2303.09112

[3] D. Arthur and S. Vassilvitskii, “k-means++: The Advantages of Careful Seeding,” in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 1027–1035. [Online]. Available: https://theory.stanford.edu/~sergei/papers/kMeansPlusPlus.pdf

[4] Liu, E., Li, S., Zheng, L., Du, N., Zhao, W., Liang, P., Zou, J., & Zhang, C. (2023, May 3). Introducing Chatbot Arena: Benchmarking LLMs in the wild. *LMSYS Org*. https://lmsys.org/blog/2023-05-03-arena/

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Oct. 2011.

[6] A. Elo, *The Rating of Chessplayers, Past and Present*. New York, NY, USA: Arco Publishing, 1978.

[7] R. A. Bradley and M. E. Terry, "Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, Dec. 1952, doi: 10.2307/2334029.

[8] "Bradley-Terry Model Python Implementation," accessed Nov. 25, 2024. [Online]. Available: https://colab.research.google.com/drive/1KdwokPjirkTmpO_P1WByFNFiqxWQquwH#scrollTo=PbTdhkLQp113

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York, NY, USA: Springer, 2009.

[10] D. C. Liu and J. Nocedal, "On the Limited Memory BFGS Method for Large Scale Optimization," *Mathematical Programming*, vol. 45, no. 1–3, pp. 503–528, Aug. 1989, doi: 10.1007/BF01589116.