Hallucination Detection via Sentence-level Entropy Analysis

Yusuf Efe
Columbia University

July 9, 2024

1. Introduction

The current backbone architecture of state-of-the-art language models is autoregressive, and these models are prone to hallucination. This is a serious problem as their use in real-life applications is increasing daily, with people starting to rely on their outputs in both hybrid and fully automated settings. In this study, we explored how to quantify the model's uncertainty and confidence given a context, and detect if it is likely to hallucinate by analyzing the model's internal states.

2. Method

Hallucinations can stem from various sources. Bad training data can cause the model to make systematic errors on specific subjects, or the model might intentionally deceive the user for a specific goal. However, in our investigation, we will not focus on these scenarios. Instead, we will examine cases where the model hallucinates by adding seemingly plausible but incorrect information to its outputs.

Many studies calculate a model's uncertainty based on the probabilistic distribution of the next token. However, a major problem with such approaches is that as models become more advanced, their ability to convey specific information in syntactically different ways increases. Consequently, at any point during answer generation, the model's internal states might lead to various routes that ultimately convey similar information. Token-level entropy-based models may incorrectly interpret this as uncertainty or a signal of hallucination.

To address this problem, I drew upon the sentence-level semantic entropy approach, largely inspired by a recent line of work [FKKG24, KHR+24], and developed a semantic entropy predictor based on the model's final layer hidden states. Before delving into the method's details, let me first explain the concept of semantic entropy.

Consider a language model being asked: "At what temperature does water start boiling?" Possible answers might include: "100°C", "It starts boiling at 100 degrees Celsius", or "One hundred Celsius". While these sentences convey the same meaning, their varying token probabilities lead to increased token-level entropy due to multiple valid options. However, semantic-level entropy focuses on the distribution of meanings across the model's full responses, rather than individual tokens. Thus, these variations would not lead to an increase in the semantic entropy, as they all express the same semantic content.

To approximate semantic-level entropy, we can generate multiple sample answers from the model, cluster them semantically, and calculate entropy based on the resulting group sizes. Specifically, if we generate N sample answers and cluster them into k semantic groups, the semantic entropy can be calculated as follows:

$$H_{semantic} = -\sum_{i=1}^{k} \frac{|G_i|}{N} \log_2 \frac{|G_i|}{N} \tag{2.1}$$

where G_i is the number of variations in the semantic group i. Thus, given a question, we can approximate the model's semantic entropy (and its uncertainty about the question's answer) by generating N answers and then calculating the semantic entropy over them.

However, this approach can be costly, as generating N samples (e.g., 10) for each question increases the computation cost tenfold. To address this problem, I developed a predictive model that estimates the semantic entropy of the model's current state based on the hidden state of the last token's final layer and this entropy gives idea about the hallucination probability of the model. Full details about the implementation of this method are provided in the following section.

3. Implementation Details

3.1. Data Preprocessing

We used the TriviaQA [JCWZ17] dataset for training, validation, and testing purposes. The test split's labels were not publicly available, therefore, we used the training split to generate our training and validation datasets, and we used the validation split as our test set.

Additionally, we aimed to test our model's hallucination behavior in a no-context setting, rather than its ability to extract relevant information from the provided context. Thus, we selected portions of the dataset where the context entry was empty. As a result, we had 6k training samples, 2k validation samples, and 1.8k test samples.

3.2. Model Selection

We used and investigated the hallucination behavior of the Gemma-2B-IT model [TMH+24]. This model occupies approximately 4GB of GPU RAM, making it feasible to run experiments with large batch sizes on a consumer-level GPU without encountering memory overload issues. Additionally, since our objective is to gain insights into the hallucination problem and find potential solutions, the Gemma-2B model's smaller size almost guarantees the generation of hallucinations. For these reasons, it was chosen as the main model for our project.

To split the generated answers into semantic groups, we required an auxiliary model for this task. To mimic real-world research setting, we wanted our auxiliary model to be less advanced than the main model. Therefore, we used RoBERTa-Large-MNLI [LOG+19] to classify the sampled 10 answers for each question into semantic groups via bidirectional entailment tests.

3.3. Sampling Outputs

After preprocessing the data and loading the model-tokenizer pairs, we began processing the data. For all answer generations, we used a few-shot approach with five example question-answer pairs, in addition to the question of interest. To record the best guess of the model, we ran the model on all training, validation, and test splits with a temperature of 0, generating one best guess for each question. We recorded the model's best guess output as well as the hidden state of the question's last token's final layer to be used in our logistic regression model.

Next, we ran the model again on the training and validation splits, generating N=10 answers for each question using top-p sampling of 0.9 and a temperature of 0.8. These results were also recorded in the dataset as new columns.

3.4. Answer Clustering and Entropy Calculation

After generating N=10 answers for each question in the training and validation splits, we needed to cluster them into semantic groups to calculate semantic entropies. For this, we used a bidirectional entailment setting. First, we concatenated the question with each of the answers separately. Then, for each pair, we checked if it entailed any existing semantic group. If it did, the pair was added to that group; if not, a new semantic group was created for that pair.

As a result, we obtained distributions such as [5, 3, 2], which corresponds to a probabilistic distribution of 0.5, 0.3, and 0.2. The semantic entropy H_{semantic} for these groups is calculated using the following equation:

$$H_{\text{semantic}} = -\sum_{i=1}^{k} \frac{|G_i|}{N} \log_2 \frac{|G_i|}{N}$$

where G_i is the number of variations in the semantic group i.

Finally, the clustered answers and their corresponding semantic entropy were added as new columns to the training and validation splits.

3.5. Label Assignment

We needed to assign labels as True or False to indicate whether the zero-temperature best guess of the model was correct for the questions in our dataset. The common approach in such tasks is to look for an exact string match between the model's output and the ground truth variations [GTA+23]. However, due to the size of our model, we observed that even for the simplest questions where the model confidently and correctly produces an output, the format often partially deviates from the desired format. For example, the model might add dates or extra details that were not asked for.

If we had employed a full string match evaluation, these confident and correct but not exactly matching outputs would be detected as hallucinations, which would harm our hallucination detection model's performance. To address this, we implemented a softer evaluation method. We split both the ground truth and the prediction into words, removed the stop words, and checked for any match between the significant words. If any of the significant words matched, we counted it as a True attempt. Even with this softer evaluation method, the True/False ratio remained around 20/80 across all splits of the dataset.

3.6. Training the Hallucination Detection Model

We used a logistic regression model to fit a model on the hidden state vector as the input. We employed two alternative settings for the labels. In the first setting, we directly used the best guess's accuracy as our label and aimed to fit a logistic regression model to predict whether a certain question would lead to a hallucination.

In the second scenario, we used the semantic entropy values as our label and trained a logistic regression model to predict the semantic entropy score based on the hidden state vector. For this approach, we selected a threshold to classify entropy values as either high or low. This threshold was chosen to maximize the F1 score for separating True and False labels. Once the model was fitted, the logistic regression model predicted the probability of low entropy. Finally, we selected another threshold: if the probability of low entropy was below this threshold, we predicted that the model would hallucinate; if above, we predicted that the model would not hallucinate. We calculated the F1 score based on this threshold value to evaluate the label prediction performance.

4. Experiments

As for the experiments, we investigated various properties of the model, including its output entropy and accuracy values. The main intuition of our work is that high semantic entropy indicates that even the zero-temperature best guess of the model is incorrect. To verify this, we created the following graph and found supporting evidence for our hypothesis. As can be seen, "True" labeled examples tend to have lower semantic entropy values.

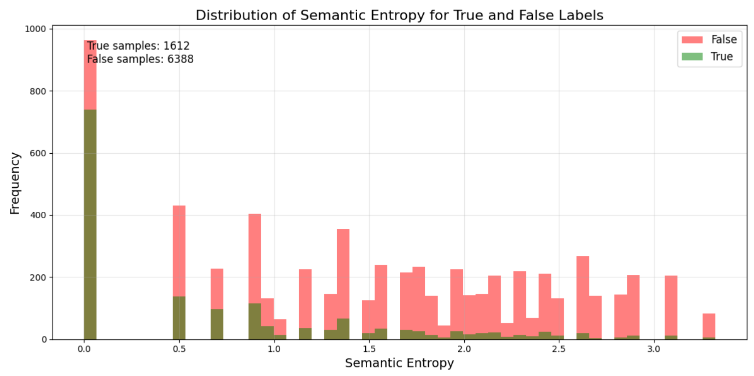

Figure 4.1: Semantic Score - Accuracy Relation

For our first logistic regression model, we used the accuracy column as our label. The ROC curve derived for the test set is illustrated in Figure 4.2.

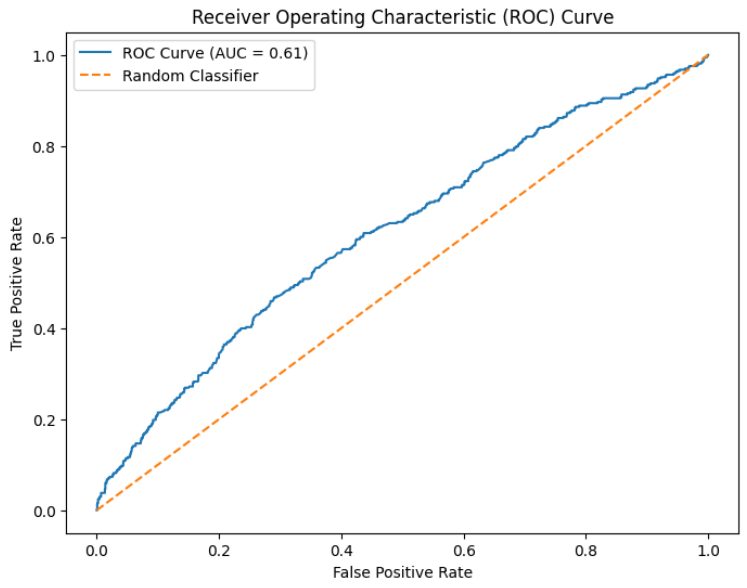

Figure 4.2: ROC Curve (accuracy-based model)

To determine the optimal threshold for our model's True/False assignment, we calculated the F1 score for various threshold values using the validation set. After identifying the optimal threshold, we used it to create the confusion matrix for the test set. Figures 4.3 and 4.4 illustrate the process of searching for the best F1 score and the resulting confusion matrix, respectively.

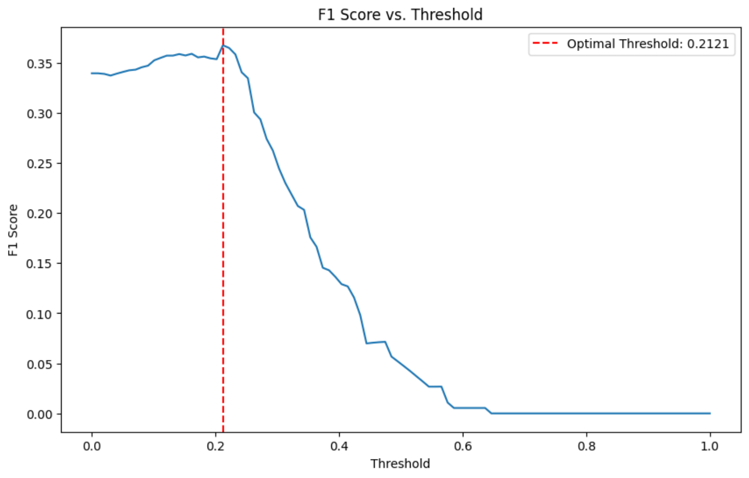

Figure 4.3: Optimal threshold search (accuracy-based model)

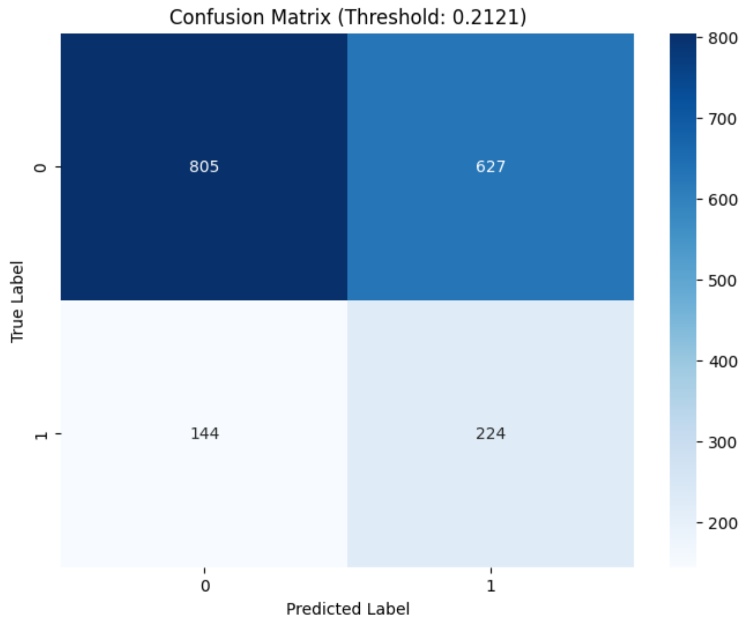

Figure 4.4: Confusion Matrix for the Optimal Threshold (accuracy-based model)

Next, we fitted a second logistic regression model, using entropy values as the labels. The first step was to determine the entropy threshold, above which entropy values were labeled as high-entropy and below which they were labeled as low-entropy. We identified the optimal threshold based on its ability to distinguish True labeled samples from False labeled ones. Figure 4.5 illustrates the selection process for the optimal entropy threshold, showing how well this threshold separates True labels from False labels, measured by the F1 score to consider both recall and precision.

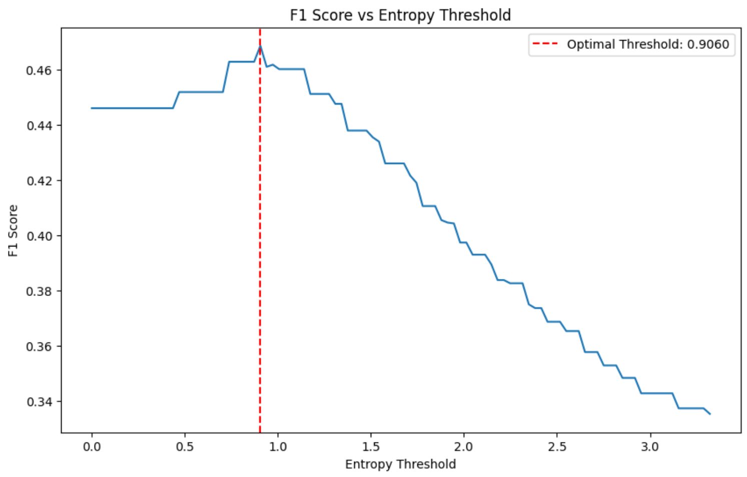

Figure 4.5: Optimal Entropy Threshold Search

As a result of the optimal parameter search for both the entropy threshold as well as the final logistic regression model's threshold, Figure 4.6 and 4.7 illustrates the ROC curve and the confusion matrix of the entropy based second logistic regression model's performance.

5. Conclusions

The important distinction of this method is that the entropy is calculated at the "answer-level" rather than the "token-level." Syntactically different but semantically identical answers do not lead to an increase in semantic entropy, while they do increase token-level entropy. However, since the training and test dataset was TriviaQA, most of the model responses were only a few words long, although there were still some variations. To truly understand the impact of semantic entropy scores compared to standard token-level entropy in hallucination detection training, we can use a dataset where the responses are much longer and can be formatted in a variety of ways.

To mimic a real-world language model research, I wanted to ensure that all models I used were either equal to or smaller than the main objective model, Gemma-2B. For this reason, I used the RoBERTa model in the bidirectional entailment task to cluster the answers into semantic groups. After careful tuning and evaluations, it started performing well, especially after ensuring that the model responses were as short as possible, thanks to the selection

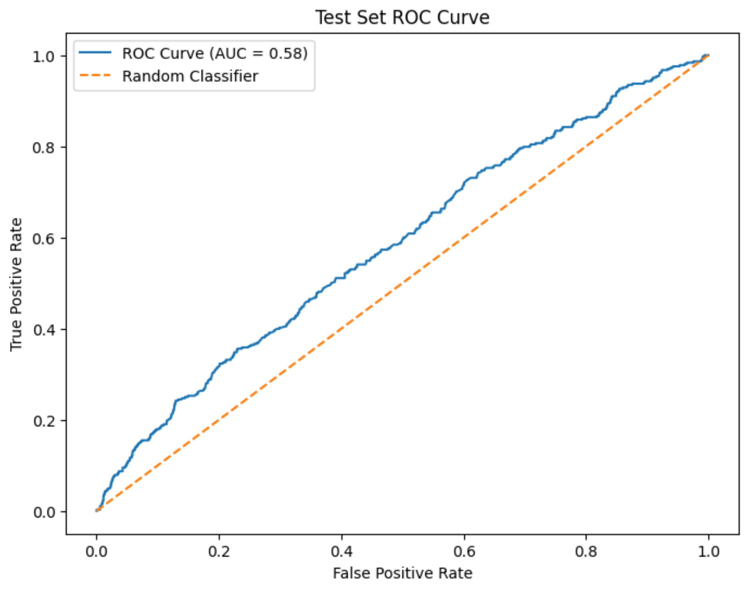

Figure 4.6: ROC Curve (entropy-based model)

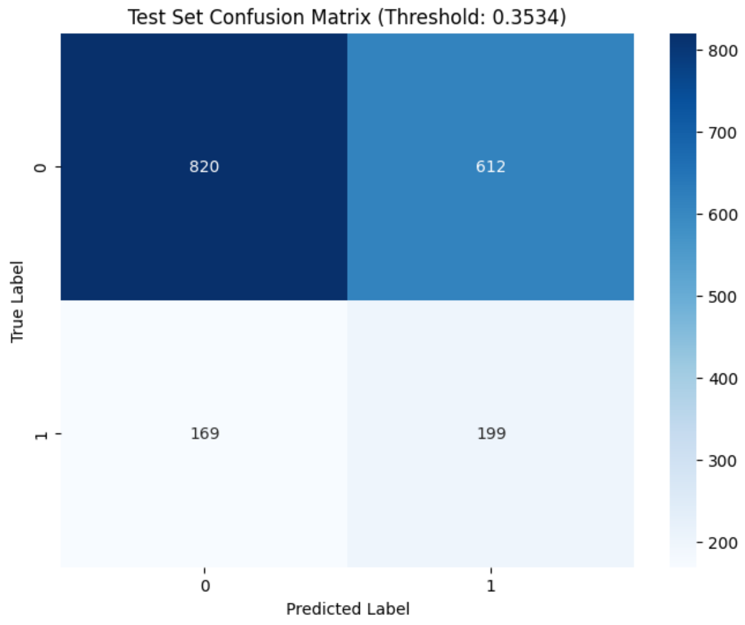

Figure 4.7: Confusion Matrix for the Optimal Threshold (entropy-based model)

of few-shot prompts. However, the clustering was still not perfect. A better clustering algorithm would improve the accuracy of semantic entropy calculation, which in turn would enhance the performance of our logistic regression model.

To ensure that I wouldn't run out of GPU memory during training and experimentation, I aimed to use a model smaller than 7B, as 7B models in 16-bit precision occupy 14GB of RAM, and a large batch size can lead to overloads during training and inference. Therefore, I used the Gemma-2B model. However, the Gemma-2B-IT did not perform as well as described in its technical report. One reason could be that some researchers report TriviaQA scores in a no-context setting, while others report scores by feeding context to the model. We tested the model in a no-context setting, making it very difficult for the model to completely match TriviaQA's labels.

To address this, I created a softer evaluation criterion, which looks for any word overlap between the prediction and the label (as long as it is a significant word, not something like "but," "and," "or," etc.). If such an overlap exists, we assigned a "True" label to the prediction. If no such overlap existed, it was assigned a "False" label. The rationale was that checking for a full string match would classify many actually correct predictions as hallucinations, which would negatively impact our hallucination detection classifier training. Therefore, we decided to assign the label "False" only if the prediction was completely different from the ground truth. Using a model from 7B families would make it much easier!

When trained with best guess accuracy as the label, our model achieved an AUC of 0.61. With semantic entropy, it achieved an AUC of 0.58. The drop of 0.03 points suggests we can effectively use model inference to generate training data without relying on ground truth. This reduces the need for costly, time-consuming manual labeling. Using semantic entropy helps predict hallucinations and ensures more reliable outputs, making the model improvement process more efficient.

In terms of the hallucination detection, we used a logistic regression model, which can only capture the linear relationships between the hidden state vector variables. Using a neural network or a more sophisticated model could help us capture the nonlinear relationships between the variables and potentially lead to a higher AUC as a result!

In terms of the input to the model, we only fed the last question token's final layer hidden state vector. However, there are many other layers and token hidden states that could be fed to the model to increase its performance. Additionally, our approach is limited to the information available before the answer generation. We could also let the model finish its answer and then feed the answer's last token hidden state to the regression model, as it contains information about both the question and the produced answer. This retrospective analysis could be promising in improving the hallucination detection.

6. Extended Experimental Framework

While our initial experiments with the TriviaQA dataset provided valuable insights, they fell short in fully demonstrating the potential of our semantic-entropy method compared to traditional token-level entropy approaches. The primary limitation was the brevity of expected answers, typically consisting of only a few words, which constrained the potential for semantic variation. Consequently, in this specific context, our semantic-level entropy calculations yielded results nearly indistinguishable from token-level entropy calculations.

To address this limitation and provide a more comprehensive evaluation of our methodology, we expanded our investigation to encompass two additional tasks and datasets:

6.1. GSM8K Dataset

We first explored the GSM8K dataset [CKB+21], a collection of grade school mathematics problems. This dataset offers two key advantages for our task:

1. It necessitates the model to elucidate its reasoning before providing the final answer, allowing for diverse syntactic expressions while maintaining semantic consistency.
2. The answer follows the reasoning, delineated by specific delimiters. This structure facilitates efficient verification of answer correctness through string matching and enables effective semantic grouping of similar answers.

6.2. Commonsense QA Dataset

Our second dataset was Commonsense QA [THLB19]. We adapted the original task, which focused solely on answer accuracy, by implementing tailored prompt instructions and few-shot examples. The model was directed to provide its reasoning before selecting from the five given options. This explanatory component significantly distinguishes semantic entropy from token-level entropy in the output.

Evaluation remained straightforward, as we could easily identify answers by searching for "A)", "B)", etc. in the text. To enhance robustness, we implemented additional checks to account for minor variations in the generated content's format when extracting answers, ensuring consistent evaluation across slight differences in response structure.

6.3. Experimental Methodology

For both datasets, we conducted three primary experiments:

1. **Maximum Predictive Power of Semantic Entropy:** We assessed the efficacy of the semantic entropy score in distinguishing between hallucinated and factual content. By applying the calculated semantic score to questions in the training and validation sets, we evaluated its discriminative power. This analysis is crucial as it demonstrates the upper limit of information we can extract about hallucination probability through semantic entropy evaluation or prediction.

2. **Hallucination Detector Training via Correctness Label:** We developed a logistic regression-based hallucination detector. This model uses the final layer embedding of the last question token as input and outputs the hallucination probability for the generated answer. We used the correctness of the model's best guesses as training labels, without incorporating semantic entropy. Our aim was to compare the predictive power of training data correctness against semantic entropy values in determining hallucination probabilities.

3. **Hallucination Detector Training via Semantic-Entropy Label:** We trained another logistic regression-based hallucination detector, this time utilizing semantic entropy values from the training data. The model predicts semantic entropy levels, serving as a proxy for estimating whether the generated answer is a hallucination.

It is worth noting that these new datasets produced improved results compared to the TriviaQA version. However, this improvement may be partially attributed to additional optimization of the logistic regression model in these recent experiments, as we were actually anticipating a potential decrease in performance due to the more complex nature of the problem in these new datasets.

7. Experimental Results and Analysis

7.1. GSM8K Dataset Results

As a preliminary analysis, we examined the relationship between the ground truth label and the semantic entropy scores. Figure 7.1 illustrates the distribution of entropy values for correct and incorrect model outputs.

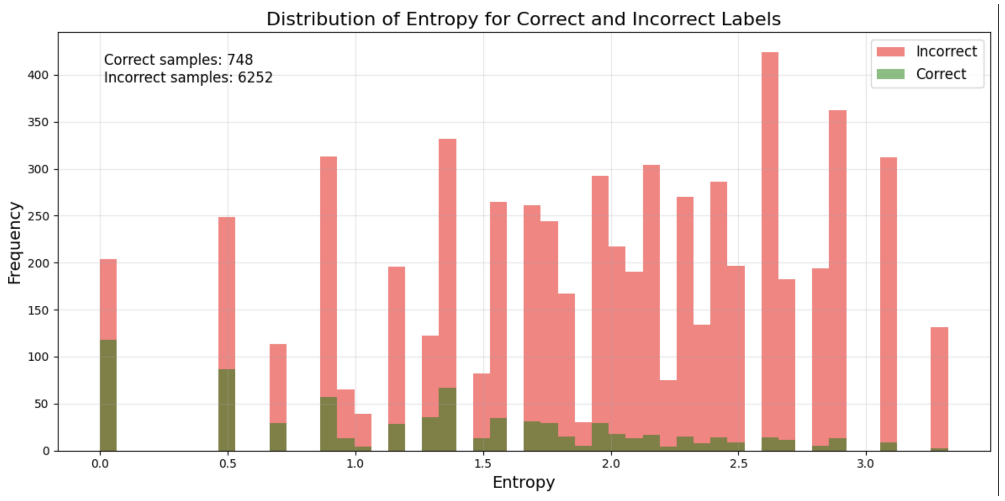

Figure 7.1: Distribution of Semantic Entropy Scores in Relation to Model Accuracy

Figures 7.2 and 7.3 demonstrate the direct predictive power of semantic entropy on the correctness label, presented as a ROC curve and confusion matrix, respectively.

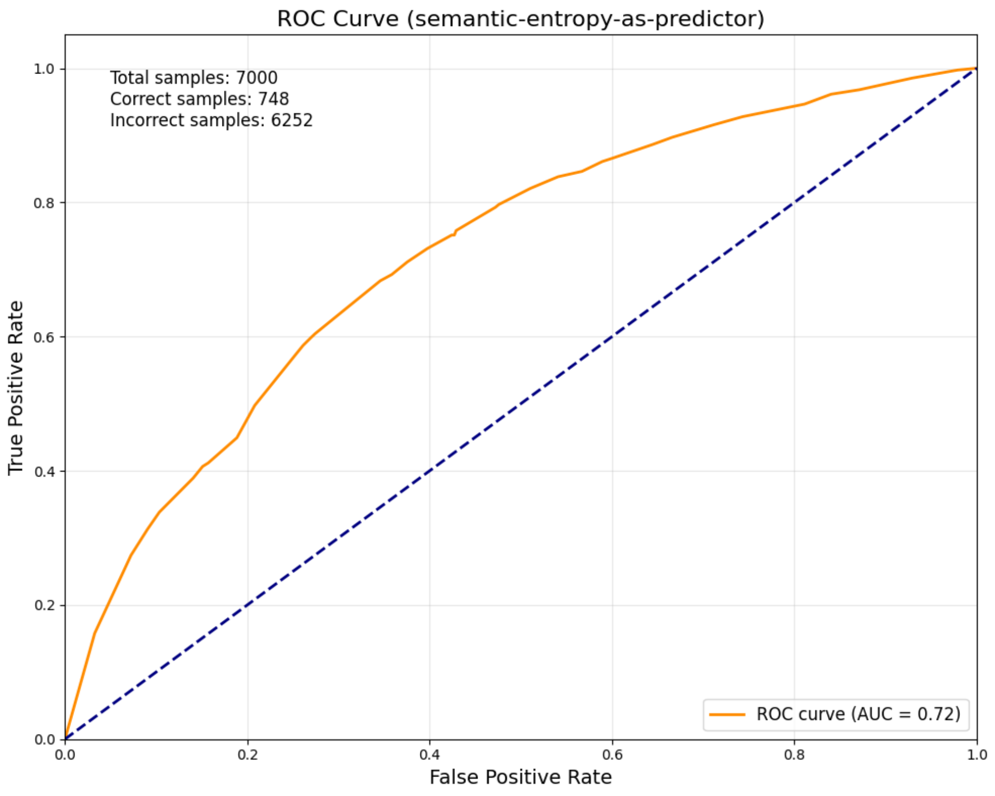

Figure 7.2: ROC Curve for Direct Application of Semantic Entropy

It is important to note that these initial results do not involve logistic regression. Rather, they demonstrate the potential predictive power of semantic entropy scores if we had access to ground truth values.

In the subsequent phase of our experiments, we trained a logistic regression model using the correctness of the model's best guess for the training data as the output and the last question token's final layer embedding as input. This logistic regression model was then evaluated on the test set to assess its predictive power for hallucination detection (correctness)

In the final phase of our experiments, we trained a logistic regression model to predict the semantic entropy of the model given a question. This model was then evaluated for its predictive power regarding both semantic entropy and the correctness label, as semantic entropy can function as a proxy for hallucination probability. The results are presented in Figures 7.6 and 7.7.

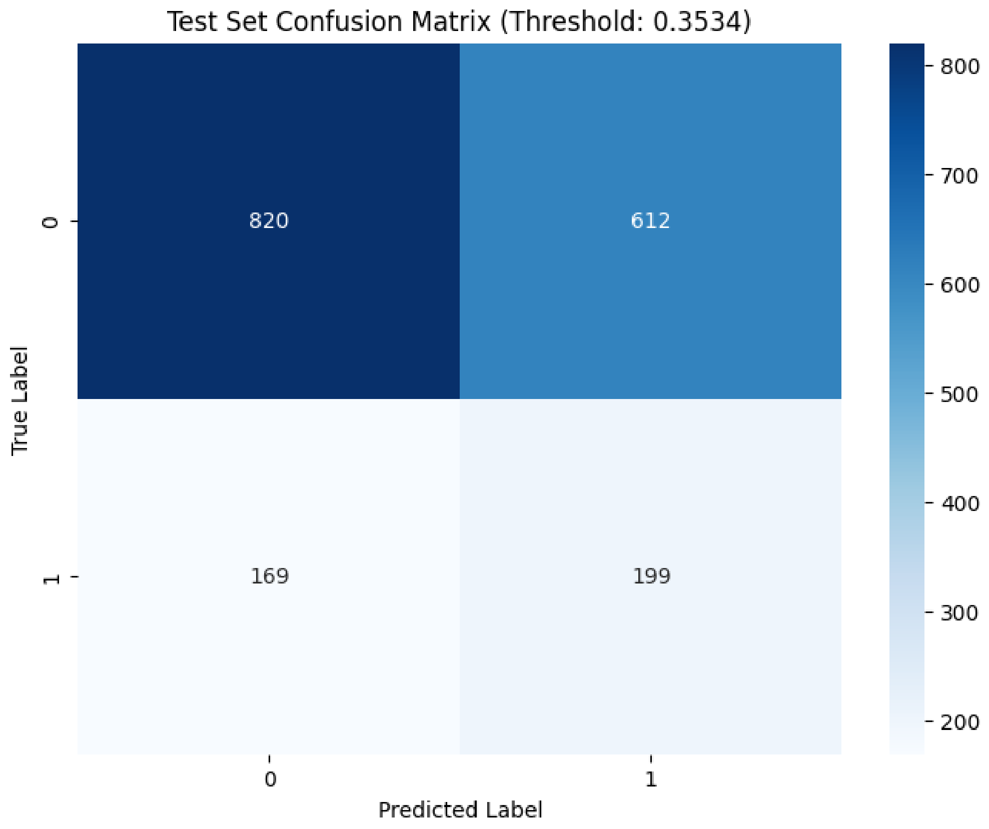

Figure 7.3: Confusion Matrix for Optimal Threshold (Direct Semantic Entropy Application)

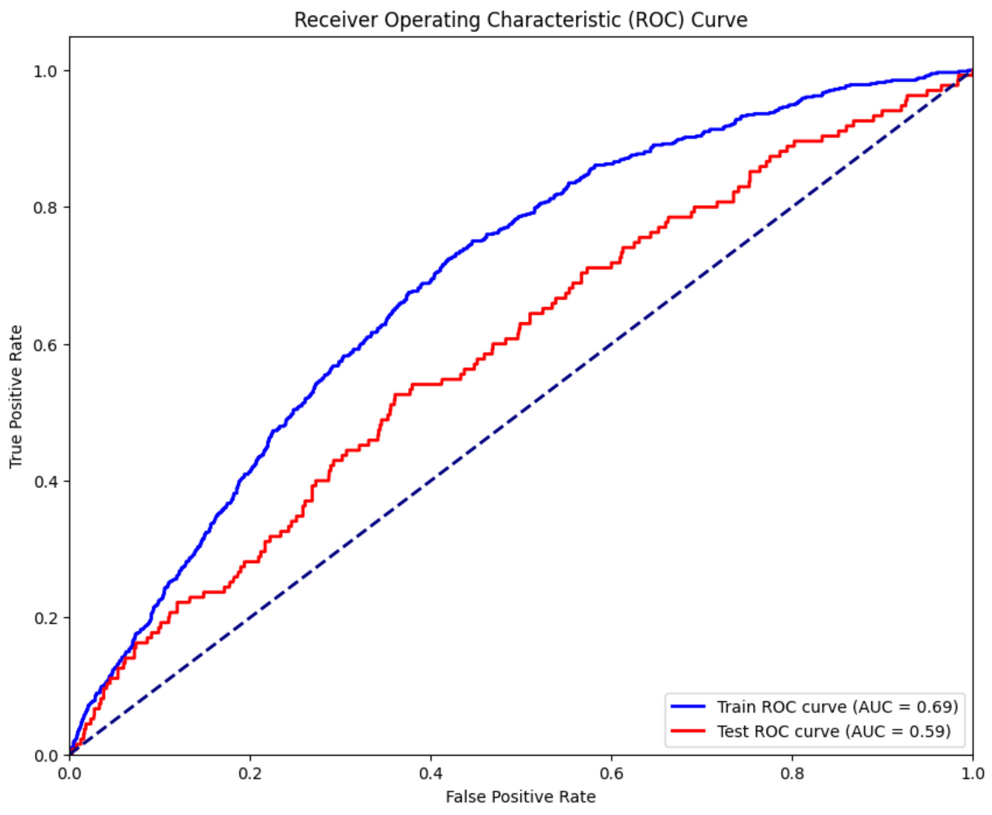

Figure 7.4: ROC Curve for Logistic Regression Model Trained on Correctness Labels

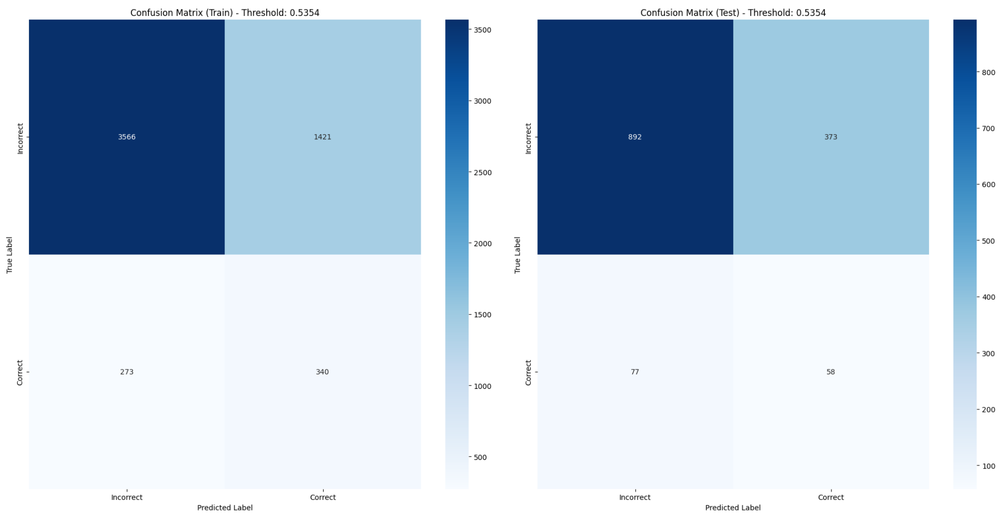

Figure 7.5: Confusion Matrix for Optimal Threshold (Logistic Regression Trained on Correctness Labels)

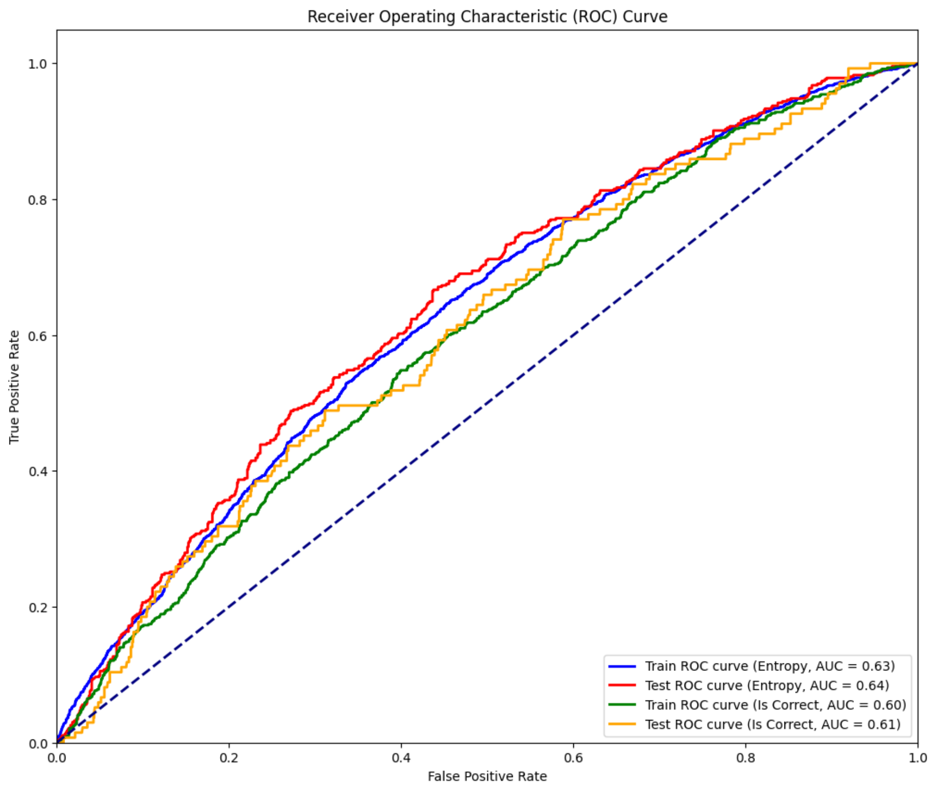

Figure 7.6: ROC Curve for Logistic Regression Model Trained on Semantic Entropy Prediction

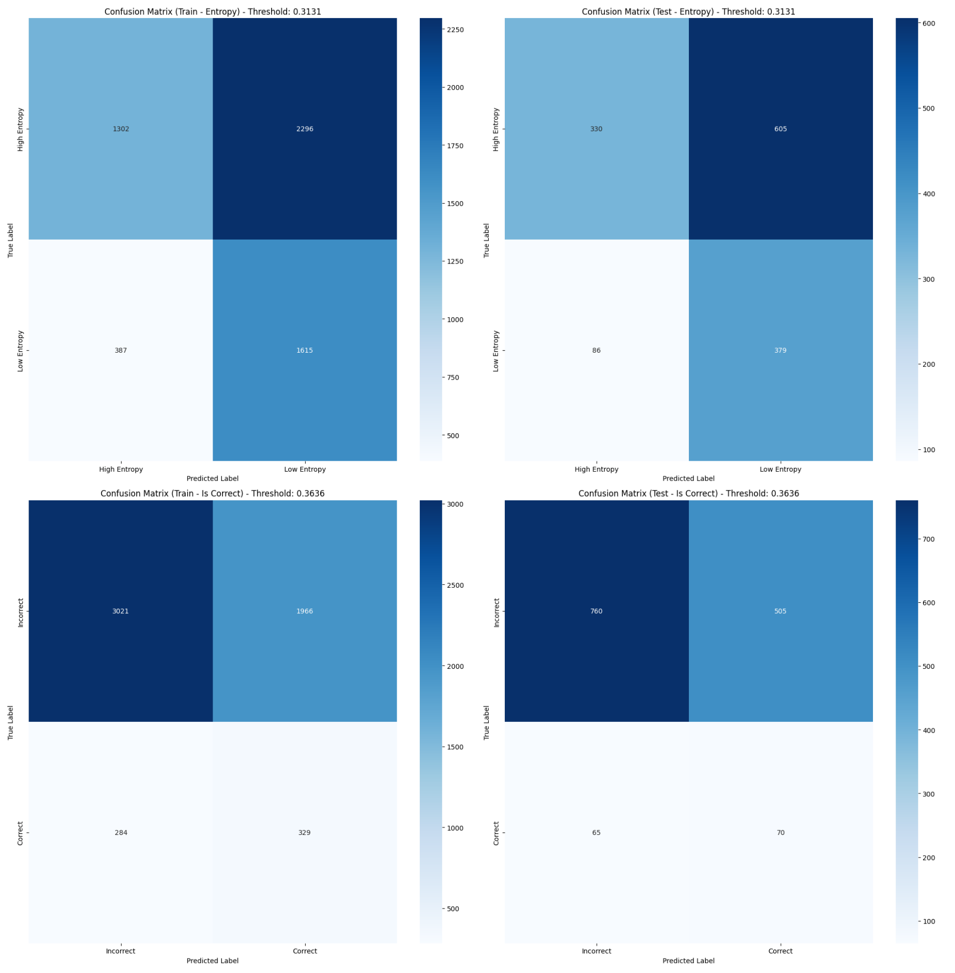

Figure 7.7: Confusion Matrix for Optimal Threshold (Logistic Regression Trained on Semantic Entropy Prediction)

7.2. Commonsense QA Dataset Results

The experimental procedure for the Commonsense QA dataset mirrored that of the GSM8K dataset. We present the results directly, beginning with the analysis of the separative power of semantic entropy for the correctness of model outputs.

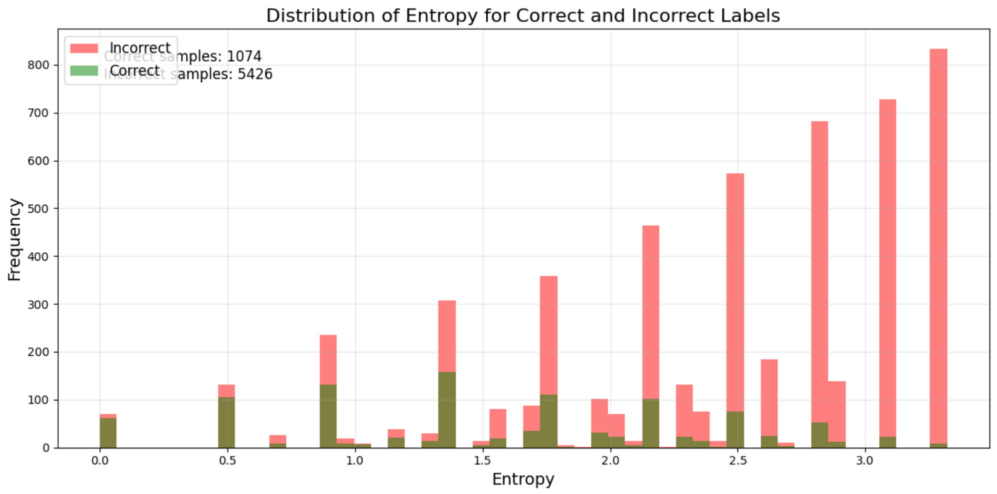

Figure 7.8: Relationship Between Semantic Entropy and Model Accuracy for Commonsense QA

As evident from these results, semantic entropy demonstrates a strong predictive power for the correctness of model responses in the Commonsense QA task. Next, we present the performance of the hallucination detection model when trained to directly predict the correctness of model outputs, rather than predicting semantic entropy.

Finally, we present the results of training a logistic regression model to predict the semantic entropy a question would lead to. We evaluate its performance for both actual semantic entropy and the correctness label, as high semantic entropy would signal a potential hallucination in the model

These comprehensive results across multiple datasets and experimental configurations provide a robust foundation for evaluating the efficacy of our semantic entropy-based approach to hallucination detection in language models.

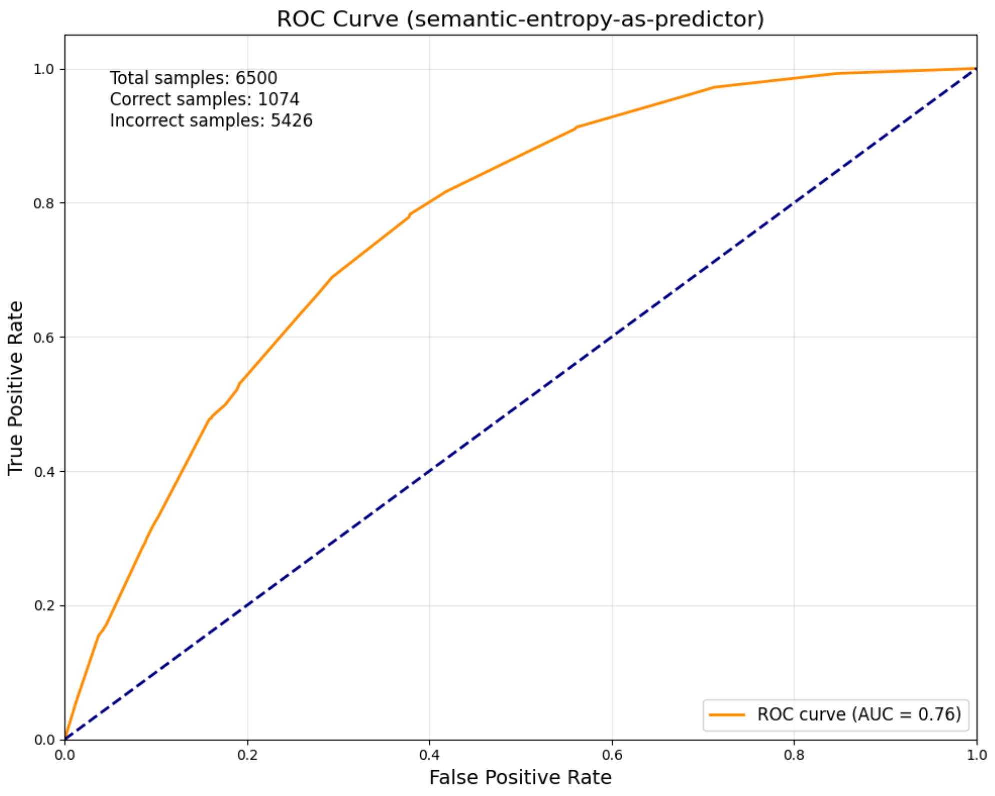

Figure 7.9: ROC Curve for Direct Application of Semantic Entropy on Commonsense QA

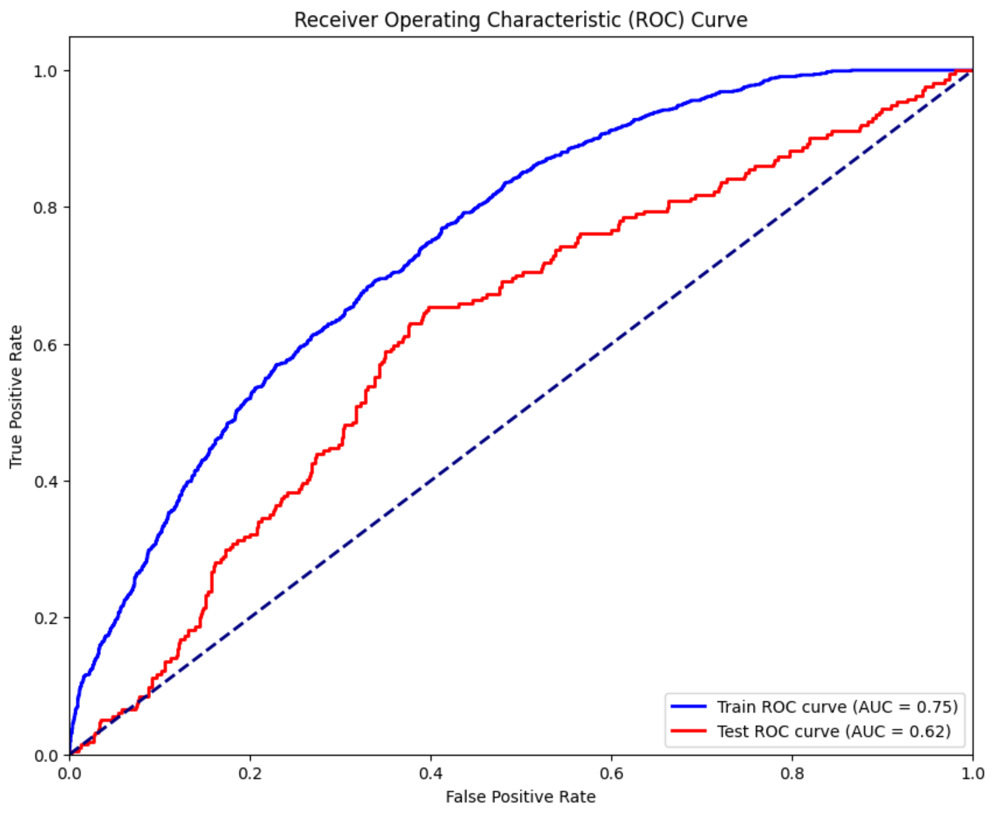

Figure 7.10: ROC Curve for Logistic Regression Model Trained on Correctness Labels (Commonsense QA)

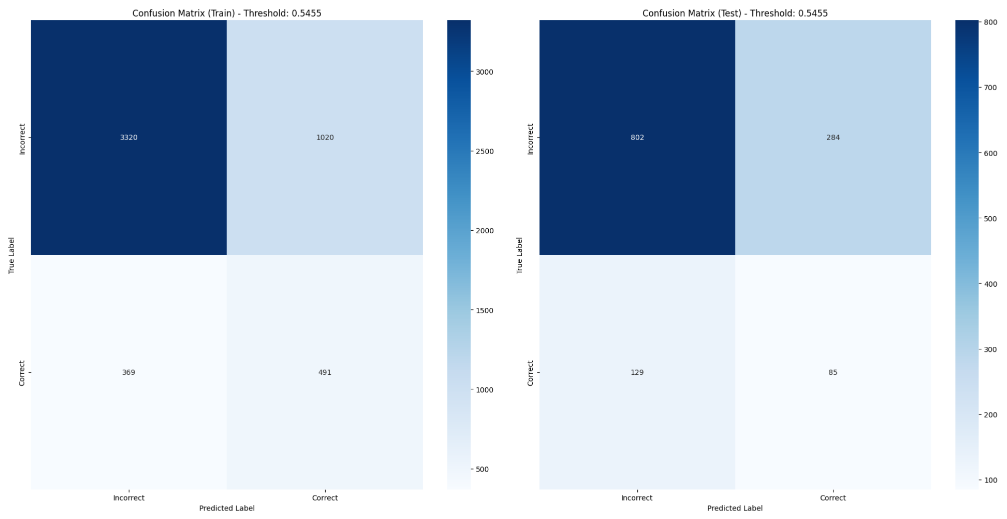

Figure 7.11: Confusion Matrix for Optimal Threshold (Logistic Regression Trained on Correctness Labels, Commonsense QA)

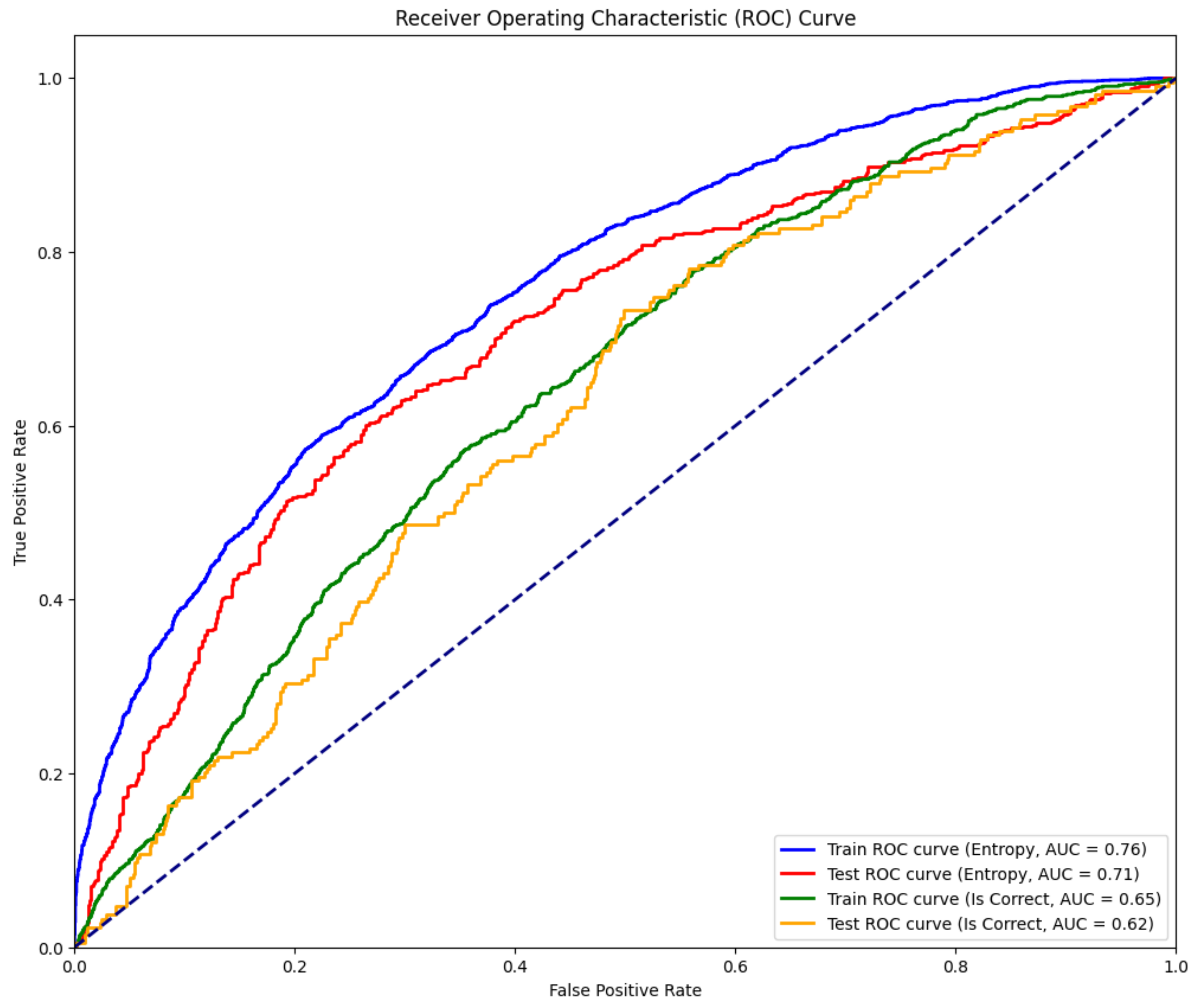

Figure 7.12: ROC Curve for Logistic Regression Model Trained on Semantic Entropy Prediction (Commonsense QA)

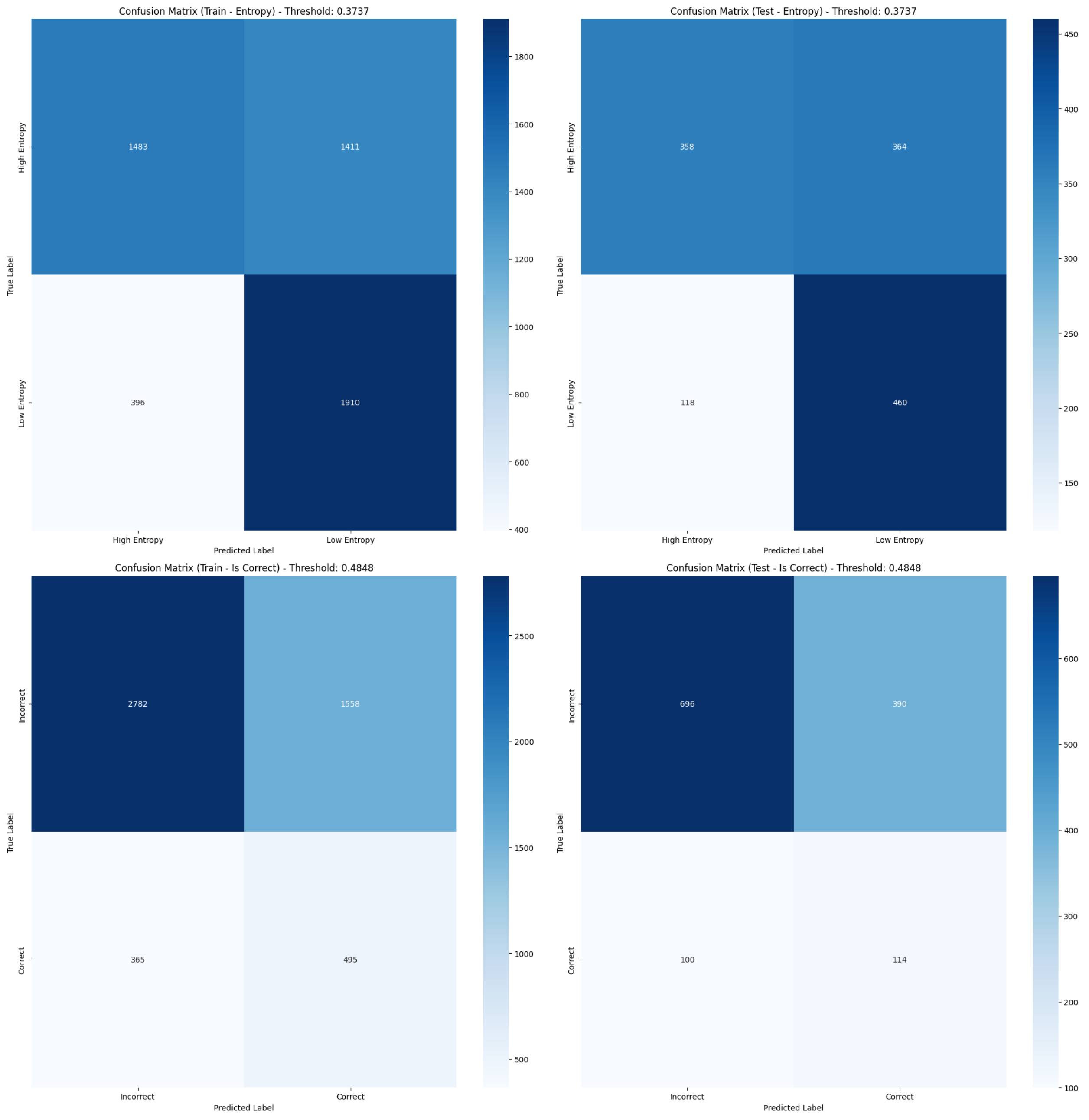

Figure 7.13: Confusion Matrix for Optimal Threshold (Logistic Regression Trained on Semantic Entropy Prediction, Commonsense QA)

8. Bibliography

References

[CKB+21] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[FKKG24] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017): 625–630, 2024.

[GTA+23] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. *A framework for few-shot language model evaluation*. Version v0.4.0. Dec. 2023.

[JCWZ17] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: a large scale distantly supervised challenge dataset for reading comprehension. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 1601–1611, 2017.

[KHR+24] J. Kossen, J. Han, M. Razzak, L. Schut, S. Malik, and Y. Gal. *Semantic Entropy Probes: Robust and Cheap Hallucination Detection in LLMs*. 2024. arXiv: `2406.15927 [cs.CL]`.

[LOG+19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: a robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[THLB19] A. Talmor, J. Herzig, N. Lourie, and J. Berant. CommonsenseQA: a question answering challenge targeting commonsense knowledge. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* 4149–4158, 2019. arXiv: `1811.00937 [cs]`.

[TMH+24] G. Team et al. *Gemma: Open Models Based on Gemini Research and Technology*. 2024. arXiv: `2403.08295 [cs.CL]`.