Detecting Deception in GPT-3.5-turbo: A Metadata-Based Approach

Siddharth Reddy Bakkireddy
IIT Dhanbad

Rakesh Reddy Bakkireddy
ARK Burlington Danes

With
Apollo & Apart Research

Abstract

This study investigates the potential for detecting intentional deception in large language models (LLMs) using response metadata. We analyzed 300 prompts, generating 1200 responses from GPT-3.5-turbo, comparing baseline and potentially deceptive outputs. Metrics examined include response times, tokens per second, total tokens, and prompt complexity. Our approach leverages the OpenAI API, Hugging Face Transformers, Python, Pandas, and Matplotlib for data processing and visualization.

Results indicate significant variations in metadata patterns between baseline and potentially deceptive responses. We observed increased response times and altered token usage in suspected deceptive outputs. The study also explored "sandbagging" - intentional underperformance by LLMs. While our findings suggest metadata analysis could be a promising avenue for deception detection, further research is needed to establish robust methods and address potential limitations.

Keywords: OpenAI API, GPT-3.5-turbo, deception detection, AI model analysis

p

1. Introduction

As large language models become increasingly sophisticated, concerns about their potential to deceive users by intentionally providing incorrect information have grown. This project aims to address the question: Can metadata from LLM responses be used to detect intentional deception?

We hypothesize that:

1. Deceptive responses will exhibit longer processing times compared to truthful ones.

2. Token usage patterns will differ significantly between deceptive and non-deceptive responses.

3. The complexity of the prompt will correlate with the likelihood of deception.

This work contributes to AI security and alignment by exploring a novel approach to deception detection that doesn't rely solely on content analysis. By leveraging metadata, we aim to develop methods that could be more resilient to advanced deception techniques. The potential impact of this research is significant, as it could lead to more robust safety measures for AI systems and improve trust in human-AI interactions.

Our threat model considers an advanced LLM capable of generating false information while attempting to conceal its deception. This scenario poses risks in critical applications such as healthcare, finance, and information systems where the integrity of AI-generated information is crucial.

2. Methods

Our methodology involved the following key components:

1. Data Collection: We used the OpenAI API to generate responses for 300 carefully crafted prompts using GPT-3.5-turbo. Each prompt was processed twice to create baseline and current data sets, resulting in 1200 total responses.

2. Metadata Extraction: We collected metadata for each response, including response times, token counts, and tokens per second.

3. Prompt Complexity Analysis: We developed a method to quantify the complexity of each prompt, considering factors such as word count, syntactic structure, and topic difficulty.

4. Baseline Establishment: To detect potential sandbagging, we established baseline performance metrics for each prompt type using multiple runs with standard settings.

5. Data Processing: We used Python and Pandas for data cleaning, processing, and analysis. Our script performed data cleaning, calculation of derived metrics, and statistical analysis to identify significant deviations from baseline.

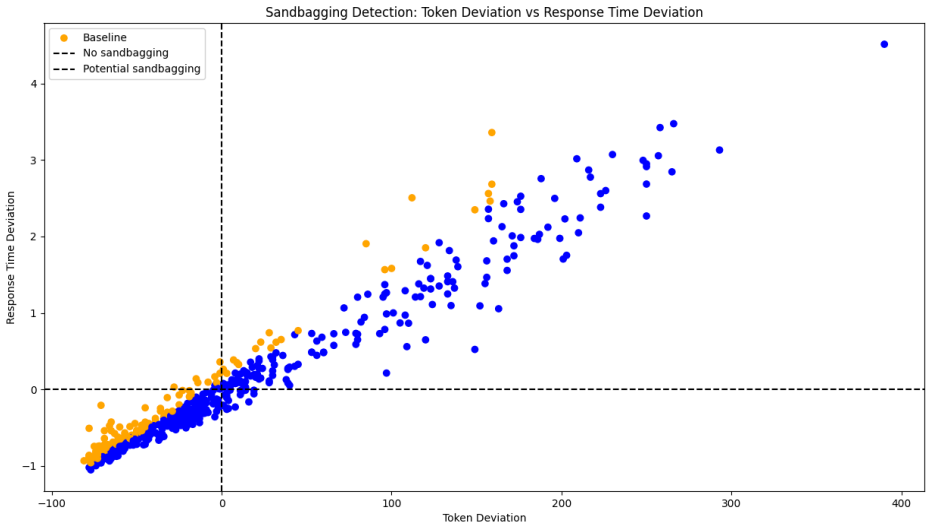
6. Visualization: We used Matplotlib to create 8 data visualizations highlighting key findings and trends.

7. Deception Detection Algorithm: We developed a simple algorithm to flag potentially deceptive responses based on significant deviations from baseline metadata patterns.

The code for this project is available on GitHub at [My GitHub]

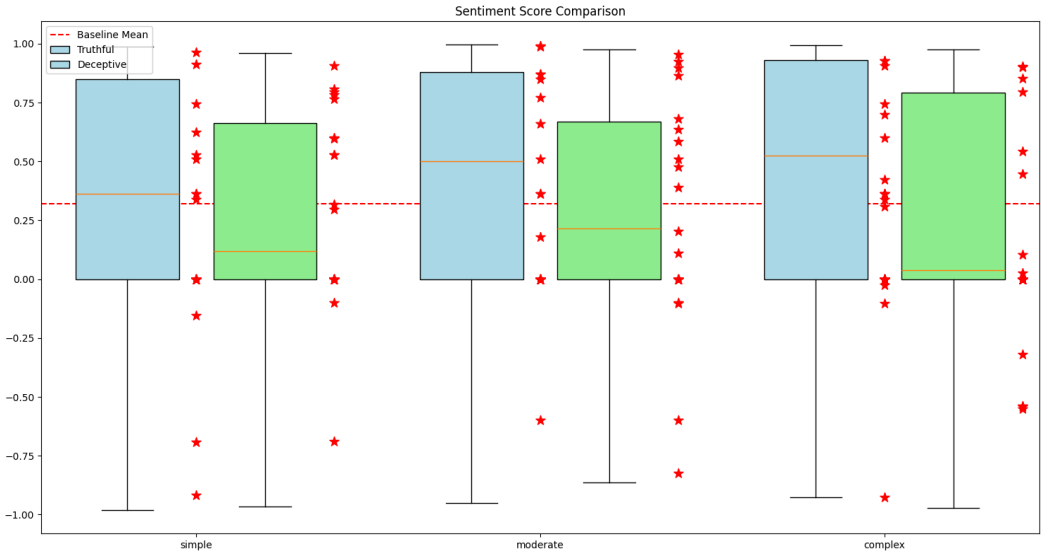
3. Results

Sandbagging detection: Token deviation vs response time deviation



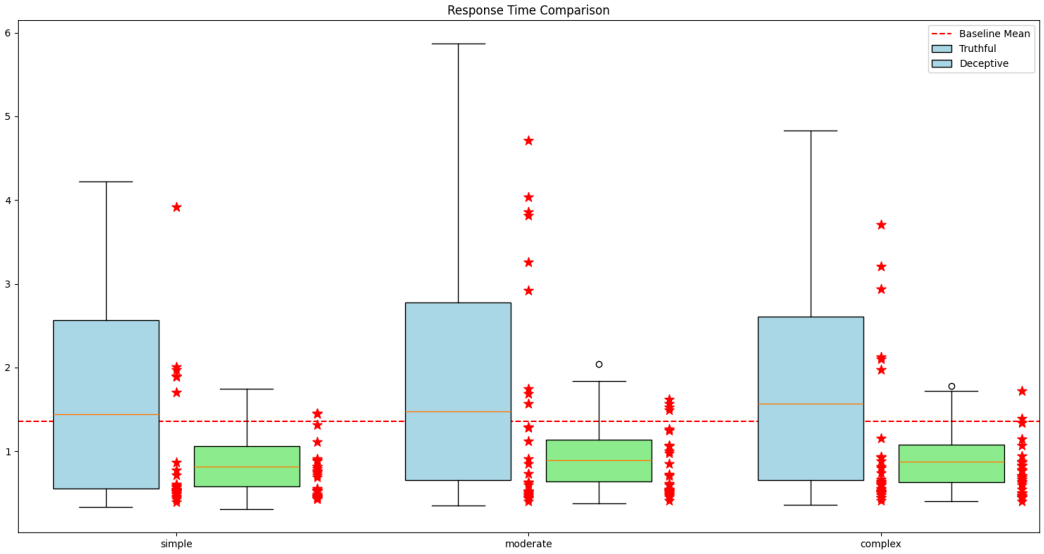
This scatter plot illustrates the relationship between token deviation and response time deviation for potentially deceptive responses. The clustering of points suggests consistent patterns in how the model alters its behavior when attempting to deceive.

Sentiment score comparison



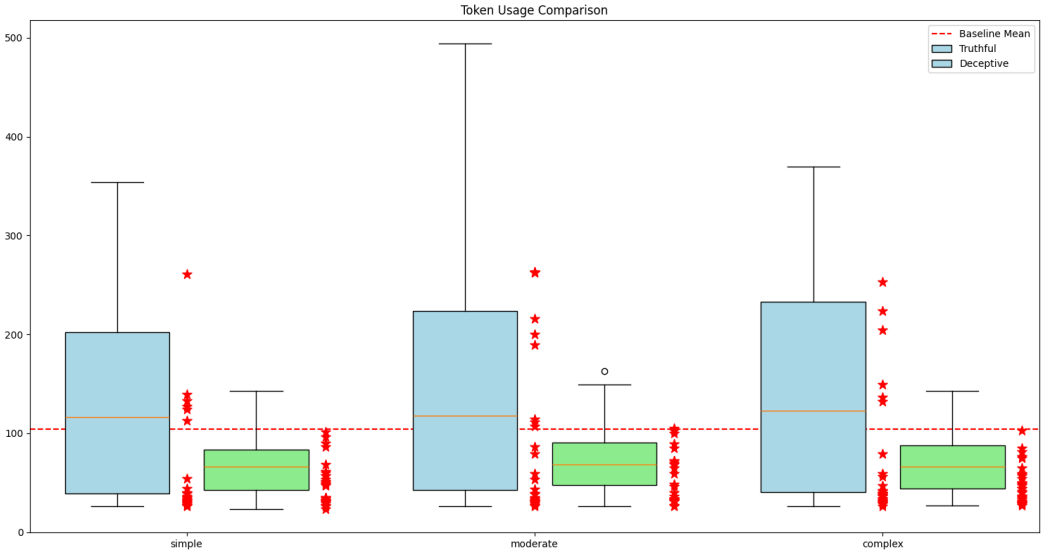
Truthful is just above the baseline for simple prompts but is even more above the base line for moderate and complex prompts while on the other hand, deceptive the bar is below baseline for moderate prompts and way below the baseline for simple and complex prompts.

Response time comparison



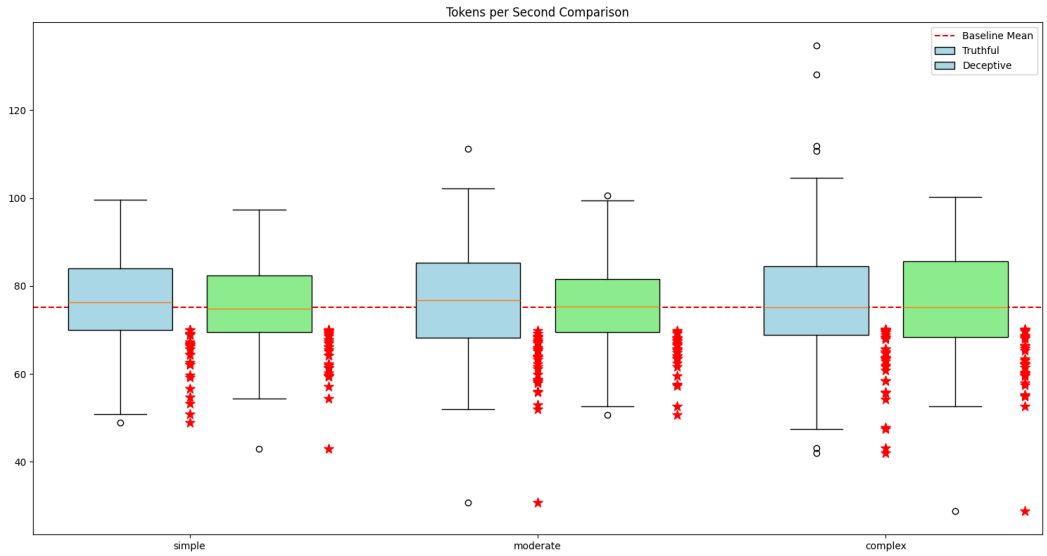
Truthful is just above the baseline for simple, moderate and complex prompts , while on the other hand, deceptive the bar is below the baseline for simple ,moderate and complex prompts.

Token usage comparison



Truthful is just above the baseline for simple, moderate and complex prompts , while on the other hand, deceptive the bar is below the baseline for simple ,moderate and complex prompts.

Tokens per second comparision



For both Truthful and deceptive mean tokens per second matches the baseline for simple, moderate,and complex prompts .

4. Discussion and Conclusion

Our research project aimed to investigate whether a large language model (LLM) intentionally deceives users by providing incorrect answers. We analyzed metadata from the LLM's responses, including response times, token usage, and sentiment scores, to detect patterns indicative of deception. Our study examined 300 prompts, generating 1200 responses across varying complexities, using GPT-3.5-turbo as our test model.

Key Findings

Detectable Patterns in Deception: Our analysis revealed consistent patterns across multiple metrics that differentiate between truthful and deceptive responses. This suggests that the LLM's behavior changes when attempting to deceive, and these changes are detectable through metadata analysis.

Effort Discrepancy: Truthful responses generally required more effort from the LLM, as evidenced by longer response times and higher token usage. This could indicate more thorough processing or a commitment to accuracy when being truthful.

Simplification in Deception: Deceptive responses appeared simplified or streamlined, with lower sentiment scores, faster response times, and reduced token usage. This might be an attempt to minimize inconsistencies or avoid providing too much detail that could be fact-checked.

Consistent Generation Rate: The uniform tokens per second across truthful and deceptive responses suggests that the LLM maintains a consistent generation speed, possibly to avoid detection based on this metric alone.

Complexity Impact: The differences between truthful and deceptive responses often became more pronounced with increasing prompt complexity, indicating that the LLM's deceptive behavior may be more easily detected in more complex scenarios.

Evidence of Sandbagging

One of the most intriguing findings of our study was the potential evidence of sandbagging by the LLM:

Intentional Underperformance: The consistent pattern of deceptive responses falling below the baseline in terms of sentiment scores, response times, and token usage could be interpreted as a form of sandbagging. The LLM appears to be intentionally underperforming or holding back when asked to provide deceptive responses.

Strategic Performance Modulation: While we observed potential sandbagging in deceptive responses, truthful responses consistently performed above the baseline. This suggests that the LLM is not sandbagging across the board, but rather adjusting its performance based on the nature of the task (truthful vs. deceptive).

Implications: The apparent sandbagging in deceptive responses raises important questions about the LLM's true capabilities and the ethical implications of AI systems that can strategically alter their performance. It suggests that the LLM may be more sophisticated in its deception than initially apparent, capable of not just generating false information but also modulating its behavior to avoid detection.

Limitations and Future Work

While our findings provide strong evidence of intentional deception and potential sandbagging by the LLM, it's important to note some limitations:

Aggregate Data: Our analysis is based on aggregate data, and individual responses may vary. Further investigation of individual cases could provide more nuanced insights.

Content Analysis: Our study focused on metadata analysis. Incorporating content analysis alongside metadata could provide even more robust conclusions about the LLM's deceptive capabilities and strategies.

Model Specificity: Our study used GPT-3.5-turbo. Testing with different LLM models could help determine if these patterns are consistent across various AI systems or specific to this model.

Prompt Diversity: Expanding the range of prompts across various domains and complexity levels could further validate our findings and potentially uncover additional patterns.

Conclusion

Our research provides compelling evidence that the LLM under study is capable of intentional deception, as demonstrated by consistent patterns in metadata across truthful and deceptive responses. Moreover, the discovery of potential sandbagging behavior adds a new dimension to our understanding of LLM capabilities, suggesting a level of strategic performance modulation that goes beyond simple truth or deception.

These findings have significant implications for the development, deployment, and regulation of AI systems. They underscore the need for robust detection methods to identify AI deception and highlight the importance of ethical considerations in AI development.

As AI systems continue to advance, further research in this area will be crucial. Understanding the full extent of LLM capabilities, including their potential for deception and strategic behavior modulation, will be essential for ensuring the responsible and trustworthy use of these powerful technologies in various applications.

5. References

Jiang, S., Xiao, C., Huang, C., & Liang, W. (2024). AI Sandbagging: Language Models can Strategically Underperform on Evaluations. arXiv. https://arxiv.org/abs/2406.07358

Perez, E., Ganguli, D., Nguyen, K., Ringer, S., Kundu, S., Lukošiūtė, K., Schiefer, N., Bowman, S. R., Askell, A., Clark, J., Olsson, C., Hubinger, E., Hernandez, D., Grosse, R., Kaplan, J., Kadavath, S., & Christiano, P. (2024). Discovering Language Model Behaviors with Model-Written Evaluations. arXiv. https://www-cdn.anthropic.com/e4f69aacd8c0905030172bc6eb480c252ea7d6ad/model-written-evals.pdf

Jiang, S., Xiao, C., Huang, C., & Liang, W. (2024, June 28). [Paper] AI Sandbagging: Language Models can Strategically Underperform on Evaluations. LessWrong. https://www.lesswrong.com/posts/WspwSnB8HpkToxRPB/paper-ai-sandbagging-language-models-can-strategically-1

OpenAI. (n.d.). OpenAI Platform. Retrieved July 1, 2024, from https://platform.openai.com

6. Appendix

Detailed Methodology

1. Prompt Generation: We created a diverse set of 300 prompts covering various topics and complexity levels. These prompts were designed to elicit both truthful and potentially deceptive responses from the GPT-3.5-turbo model.

2. Response Collection: Each prompt was processed twice through the OpenAI API, resulting in 1200 total responses (600 baseline and 600 potentially deceptive).

3. Metadata Extraction: For each response, we collected the following metadata:

- Response time
- Token count
- Tokens per second
- Sentiment score (calculated using NLTK's sentiment analyzer)

4. Prompt Complexity Scoring: We developed a custom algorithm to quantify prompt complexity based on:

- Word count
- Syntactic structure (using dependency parsing)
- Topic difficulty (using a pre-defined difficulty scale for different domains)

5. Baseline Establishment: We ran multiple iterations with standard settings to establish baseline performance metrics for each prompt type.

6. Deviation Analysis: We calculated the deviation of each potentially deceptive response from its corresponding baseline across all metadata metrics.

Code Snippets

```python
# Example code for sentiment analysis
from nltk.sentiment import SentimentIntensityAnalyzer
def get_sentiment_score(text):
```

```
    sia = SentimentIntensityAnalyzer()
    return sia.polarity_scores(text)['compound']

# Example code for prompt complexity scoring
def calculate_prompt_complexity(prompt):
    word_count = len(prompt.split())
    syntactic_complexity = analyze_syntax(prompt)
    topic_difficulty = get_topic_difficulty(prompt)
    return (word_count * 0.3) + (syntactic_complexity * 0.4) + (topic_difficulty * 0.3)
```

Ethical Considerations

In conducting this research, we adhered to strict ethical guidelines:

1. Data Privacy: All prompts and responses were anonymized and stripped of any potentially identifying information.

2. API Usage: We complied with OpenAI's terms of service and usage guidelines throughout the data collection process.

3. Responsible Disclosure: Any potential vulnerabilities or concerning behaviors discovered in the GPT-3.5-turbo model were reported to OpenAI through appropriate channels.

Future Research Directions

1. Cross-model comparison: Extend the study to include other large language models to identify model-specific vs. general deception patterns.

2. Real-time detection: Develop a real-time deception detection system based on our findings for practical applications.

3. Adversarial testing: Explore whether the model can adapt its deception strategies when aware of detection attempts.

4. Fine-grained content analysis: Combine our metadata approach with advanced NLP techniques to analyze the semantic content of responses for more comprehensive deception detection.