Bot or Not: Pushing the Limits of Human-Like Tweet Generation

Written by Victoria Hason [*]
McGill University
3480 Rue University, Montréal, QC H3A 2A

Abstract

This paper examines the development and evaluation of a social media bot engineered to generate human-like tweets and evade automated bot detectors in a competitive environment. Two core objectives shaped the bot's design: producing content that reads as authentically human and reducing the probability of being identified by detection models . The research evaluates several approaches to bot creation: prompt engineering, post-processing, and ultimately fine-tuning a language model to specialize in tweet generation. Fine-tuning proved most effective method, avoiding detection by 100% of classifiers and generating tweets whose tone, structure, and sentiment consistently resembled human-made content. Nonetheless, challenges such as hallucination, dataset biases, and the model's sensitivity to training data were observed.

Introduction

Social media has become one of the most influential tools for shaping public discourse and disseminating information. Its ability to rapidly amplify content and reach broad audiences has made it both a powerful platform for communication, and a dangerous tool that can be leveraged for strategic manipulation. Increasingly, social media is being exploited through deliberate, coordinated campaigns designed to manipulate public opinion. These efforts may aim to spread disinformation, distort public narratives, or push specific agendas.

As such tactics become more common, the challenge of distinguishing authentic content from bot-generated material has become an important area of study, as the need to flag/track dangerous content, and more generally to understand the social media landscape, becomes increasingly valuable. This paper aims to contribute to this ongoing discourse by designing and iteratively improving a social media bot capable of producing highly realistic human-like content. The intention is not merely to evade detection, but, on a larger scale, to explore the limits of what can be achieved with current text-based bot generation methods, and determine where the state of bot creation is at today.

The first objective that guided the development of this bot was to generate the most human-like tweets possible: tweets authentic enough that a casual reader would not suspect they were created by an automated system. This goal was set in order to focus the development process towards optimizing natural language generation for informal social media settings, and to investigate where the line is between human and bot-generated content.

Another goal was to avoid detection by bot detectors. The latter was approached by introducing variation into the bot's behavior, manipulating observable features, and avoiding detectable patterns. Ideally, both objectives are deeply related: A bot that generates authentic-passing content should avoid detection more successfully than a clearly automated one. However, this distinction between the two goals is specified to contextualize the motivations for certain decisions throughout the bot creation process.

As a result of these objectives, the overall strategy for bot detection prioritized tweet-level realism over account-level coherence. That is, rather than attempting to simulate a consistent online persona over time, the approach prioritized crafting each individual tweet to be as convincing as possible.

Background

The research is structured around a competition-like infrastructure between two opposing teams: bot builders and bot detectors. Weekly competitions, called sessions, simulate a typical period of activity on X. At the beginning of each session, the bot building teams receive a static dataset of

[*]Under the supervision of Derek Ruths, McGill University

posts (simulating the current posts visible to the bot on the platform at that time), and their aim is to inject their generated tweets, authored by their generated users, into the dataset. Once the complete dataset—including both real and injected content—has been compiled, the detection team analyzes the data in an effort to determine which users were bots. For every user, they provide a binary classification ("bot" or "not a bot") along with a confidence score between 0 and 100 indicating the certainty of their prediction.

In the competition dataset, users are represented by their name, username, description, and location. Posts are depicted by their content, and the corresponding user who authored them.

As mentioned previously, this paper aims to document the creation process of a social media bot participating in this competition. Below are the techniques and technologies experimented with during the development process:

- **ChatGPT4o**: A multi-lingual LLM developed by OpenAI, used to generate user metadata and post content.
- **Prompt Engineering**: The intentional design of instructions to be given to an LLM with the goal of optimizing the response to one's specifications. Various prompt engineering techniques were employed.
- **Fine Tuning**: The process in which a pre-trained model (In this case, ChatGPT4o) is further trained on a custom dataset to specialize it for a predetermined, specific task. This is done by using only domain-specific data that is different from the original dataset used to train the base model.
- **Post Processing**: The process of further modifying generated tweets before submitting them to the dataset, often through the introduction of randomized spelling errors and grammar errors, with the goal of introducing randomness and making the tweets appear more human.

Before proceeding further, it is important to address an important question: what constitutes a “good” tweet? How do we know whether a bot is performing well? It was necessary to establish a set of objective evaluation metrics to guide the development process and systematically measure improvements in bot performance over time. Below are the techniques employed for this purpose:

Bot Performance Graphs: After each session, a graph was generated showing all the bots present in the competition, and the confidence value of their detection by each detector (A positive value indicating a bot was detected). These graphs helped illustrate bot performance with respect to detector performance, revealing how deceptive a bot was.

Bot vs Human Comparisons: To evaluate the realism of the generated content, the bot-submitted tweets were separated from human tweets, and then both were analyzed using a range of quantitative metrics. Ideally, these numbers would be very similar, as aligning the generated tweet's features with those of genuine users could increase the likelihood of evading detection. The following features were extracted across both datasets:

- Average Tweet Length: The number of characters per tweet, to evaluate whether the bot mimics typical tweet conciseness.
- Timestamp Distribution: A graph demonstrating when posts were made across the session timeline.
- Spelling Errors: The average number of spelling mistakes per tweet, considering only alphabetical words. (Links and Emojis, for example, don't count as a spelling error). (Implemented using the spellchecker library)
- Number of Tweets Containing Links and Mentions: The percentage of tweets that include hyperlinks or user mentions. This highlights a common issue in generated content, where tweets either uniformly include or exclude such features, lacking the variability seen in human behavior.
- Most Frequent Words and Top TF-IDF Terms: A comparison of the most common and most distinctive words (as measured by Term Frequency–Inverse Document Frequency).
- Sentence Length: The average number of words per sentence.
- Lexical Diversity: A measure of the variety of unique words used, indicating vocabulary richness.
- Readability (Flesch Reading Ease Score): An estimate of how easy the text is to read, with higher scores indicating simpler, more accessible language.

- Fragmentation: The presence of incomplete or grammatically incoherent sentence structures, which can signal unnatural or bot-like language.

The final four features were determined using the SpaCy model. (Honnibal et al. 2020).

Methods

The ultimate approach used fine-tuning as the primary content-generation method. This section presents an overview of the techniques experimented with during development, and presents how they led to this final strategy.

Initial approaches

The initial approach's main goal was to create a baseline bot that operated successfully within the competition infrastructure, and employed a simple, hard-coded framework. A single user account was hard-coded, and all the tweets came from this one account.

The bot generated content by pulling from the existing dataset of human-authored tweets provided to it. This was done by selecting a random post from the dataset and prepending the hard-coded phrase: "I can't believe someone said..." to the tweet, creating the appearance of original engagement.

To simulate realistic activity, each post was assigned a randomly generated timestamp within the bounds of the relevant subsession window. This approach for timestamp generation provided a satisfactory time distribution, and was kept for every subsequent bot.

To move beyond deterministic, hard-coded behavior, the next iteration replaced fixed tweet content with dynamically generated text. A simple prompt: "generate one tweet" was provided to the GPT-4o model, and the response was posted in place of static content, marking the first step toward more adaptive and human-like generation.

These initial experiments revealed that the existing dataset of session tweets could serve as a valuable resource when generating bot content. Not only does it provide valuable contextual information on what topics the bot will be posting into, but it also offers concrete examples of human tone, vocabulary, and syntax: features that can be mirrored to improve authenticity. A promising bot strategy could perhaps combine the nondeterministic GPT-4o text generation with the semantic and syntactic context gained from analyzing tweets from the dataset.

Prompt Engineering

In order to improve the results from the previous approach, multiple rounds of prompt-engineering were explored. All prompts used few-shot prompting, a technique where several authentic tweets were provided as examples to guide the language model in generating similar content.

One method experimented with was prompt chaining: An initial model was given a set of sample tweets and asked to generate a new prompt that could be used to create tweets in the same style. This generated prompt was then passed to a second model to produce the final output. This approach, however consistently underperformed. The intermediate prompts created by the first model lacked the specificity needed to drive meaningful changes in the generated tweets. As a result, the second-stage outputs did not differ significantly from one another, nor from outputs generated using simpler prompts.

Next, the inner monologue technique was experimented with, as described in OpenAI's prompt engineering documentation. (OpenAI, 2025) The core idea is that instructing the model to reason internally before generating output can improve response quality. Although more commonly used in tasks like logical reasoning or multi-step decision-making, this technique was adapted here in an experimental attempt to encourage more deliberate and natural tweet generation.

The final prompts given to the chat completions API, after rounds of testing, were as follows:
To generate user metadata:

- System prompt: "You are a helpful assistant that generates realistic twitter user metada"
- User prompt: "Generate one set of user metadata in exactly this format: username-name-description-location-"

To generate tweets:

- System prompt: ""You are a helpful assistant that generates realistic tweets"
- User prompt: "The below dataset is a set of 35 thoughts and opinions on various topics. Please analyze the dataset thoroughly, and explain the specifics of its distinctive stylistic elements in terms of vocabulary, sentence structure, topics, sentiment, etc. Analyze which

words seem important to the data and use these words in your blurbs. You must be very specific to the dataset. For example, if the dataset references specific people or events, you can bring these up as well. Finally, generate a JSON dataset of 10 blurbs on the same topics. Make it extremely similar in style to the rest of the data. Here are 3 important things to keep in mind: 1: Use the slang and important words discovered in your generated blurbs. 2: Each blurb should be around 40 words (130 characters). Here is the dataset:" (This would be followed by the 35 example tweets, pulled at random from the session data).

The 'response format' endpoint was then used to hide the inner monologue, facilitate easy parsing, and provide a structure for what the model should think through. The required response format was a JSON structure that included the following fields:

- Analysis (type: string): "Analyze the distinctive stylistic elements in terms of vocabulary, sentence structure, and sentiment"
- Number of interactions (type: string): "How many tweets use "@mention" or "https://t.co/twitter_link" ? You should aim to keep the same ratio."
- Slang Used (type: string): "Write the informal slang or swear words that are used. Use these words in your blurbs."
- Topics (type: string): "Write the 3 main topics from the dataset here"
- Important words and specific things/people mentioned (type: string) "Write the most important words in the dataset. Use specific words instead of generic words. Use these words in your blurbs."
- Tweets (type: array): "A list of 10 generated blurbs similar to the dataset."

From the generated tweets, a random selection of 2-8 were posted by the user. Each user generated 10 new tweets during each subsession.

This final prompt structure was developed through extensive trial and error, involving close analysis of session results and careful reading of the generated tweets. One key addition was the inclusion of a field titled "Important words and specific things/people mentioned", which was designed to steer the model toward generating content anchored in concrete topics drawn from the human dataset. This modification addressed a recurring issue observed in earlier iterations, where the model defaulted to producing vague platitudes or overly generic expressions, lacking the specificity and contextual relevance typical of human tweets.

The impact of this change is evident when comparing tweet quality before and after the introduction of the inner-monologue format using structured JSON prompts. For instance, in session 14, Victoriabot1 posted tweets such as:

- "Can we please talk about how insane that game was last night? What a comeback!"
- "What even is the academic consensus on that topic?"

While these posts mimic casual social media tone, they lack the grounded references and strong opinions commonly found in authentic tweets. In contrast, by session 18, the bot's tweets displayed noticeable improvement in specificity and personality:

- "Lucas Raymond is on fire this season! Leading the Red Wings — let's go! #LGRW"
- "Comparing Chet Holmgren to barbecue chicken is too real. #OKC"

The latter examples demonstrate a more deliberate engagement with figures and events, reaching a greater level of specificity. In addition, they use certain expressions, such as being "too real" that indicate a more online, modern language.

Overall, while this strategy yielded significant improvements over the initial approaches, certain limitations must be noted:

1. The generated tweets were still noticeably overly sanitized—characterized by perfect spelling and grammar, and lacking the informal tone, slang, or internet-specific vernacular often present in authentic social media discourse.
2. The reliance on lengthy prompts is expensive (due to the large number of tokens required to process them) and slow. In contrast, a fine-tuning approach—while requiring a greater initial investment— could allow for much

more compact prompts (by not needing to provide examples in the prompt and requiring less monologue, for example) reducing both token usage and inference cost in the long term.

3. Despite early success, performance in competitions of the bot using this strategy declined as the sessions progressed:

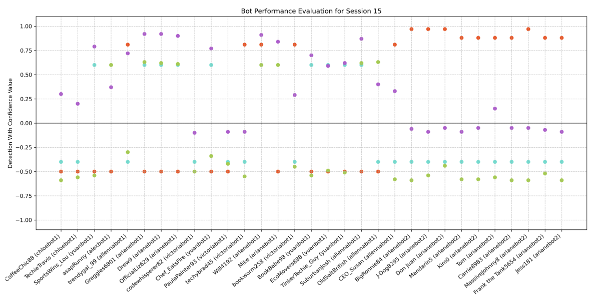

Figure 1: Bot Performance Evaluation, Session 15. (Bot is victoriabot1)

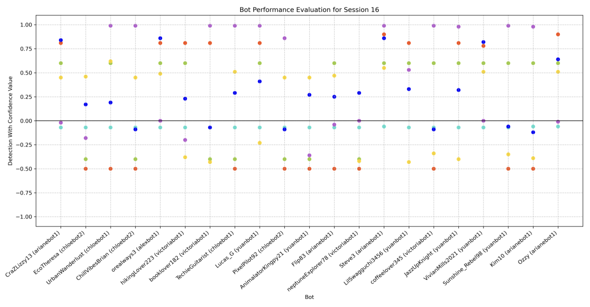

Figure 2: Bot Performance Evaluation, Session 16. (Bot is victoriabot1)

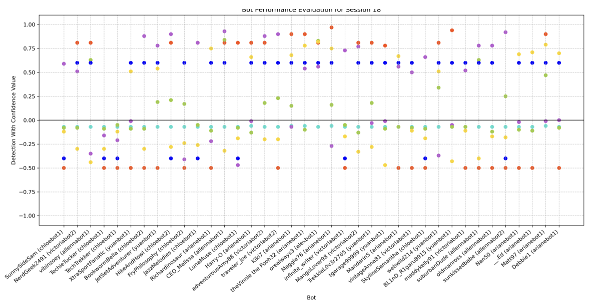

Figure 3: Bot Performance Evaluation, Session 18. (Bot is victoriabot2)

The detection rate was calculated from the bot performance graphs through the formula:

$$\frac{n}{d} \times 100$$

where n is the number of times the bot is detected across all accounts, and d is the number of detectors across all accounts.

Session	Detection Rate
Session 15	18.75%
Session 16	33%
Session 18	54%

Table 1: Bot Detection Rates by Session

As demonstrated by the session results, the performance of the prompt-engineering bot, which had minimal changes made from session 15-16, and no changes made from session 16-17, kept declining. With each subsequent session, bot detectors became more sophisticated and were able to train on previous session's data, learning how the bots performed. The decreasing effectiveness suggests that the existing prompt structure may have reached a point of diminishing returns and that a new strategy would be needed to remain competitive.

Post Processing

To improve the quality of content generated through prompt-engineering, small modifications were introduced to the tweets after their creation in order to (1) mimic the imperfections of human behavior, (2) introduce randomness, and (3) meet specific targets identified through post-session analysis, with the aim of more closely aligning the bot's output with authentic human tweets.

This endeavor was achieved using the typo Python library (Kumar, 2023). The following modifications were made:

- A character was replaced with a nearby key on the keyboard 5% of the time.
- A random character was removed 5% of the time to simulate common typing errors.
- 15% of the time, a tweet was set to all lowercase.
- 25% of the time, if a tweet did not already contain a link, one was appended. This adjustment helped to match the typical frequency of links observed in human-authored posts.
- Mentions were inserted into tweets either at the beginning or end, with a 3% chance for each.

These parameters were continuously adjusted throughout the development process. After each session, performance was analyzed to determine whether additional randomness was needed—e.g., increasing the rate of typos or link insertions if human tweets in that session showed higher rates of such features. The percentages shown above are those used during the final session.

Fine Tuning

The next strategy that was attempted was fine-tuning. Once a model has been fine-tuned, it no longer requires detailed examples in each prompt to achieve a desired style, tone, or format. Instead, these qualitative aspects are internalized during training. Fine-tuning can be seen as a useful option in cases where it is more effective to “show, not tell.”

According to OpenAI’s documentation (OpeanAI, 2025), fine-tuning is recommended for use cases involving:

1. Setting the style and tone of a desired output
2. Performing a task that’s difficult to articulate in a prompt

The fine-tuning process was performed through the following steps

1. In order to assemble the training and validation datasets, an equal number of human-authored tweets from every previous session’s dataset was collected, with an effort to ensure topical balance. This preference was constrained by the content bias in the datasets, which overrepresented topics like sports and pop music.
2. The datasets were formatted in JSONL using the following structure :

   ```
   {"messages": [{"role": "system",
   "content": "You are a content
   generation assistant that
   provides sample social media
   posts"}, {"role": "user", "content":
   "Generate one blurb"}, {"role":
   "assistant", "content":
   (Human tweet goes here) }]}
   ```

3. The user field refers to the prompt during fine-tuning. Choosing what to include in this field is use-case specific and widely variable—ranging from detailed task instructions to being left entirely blank. For this experiment, the prompt: “Generate one blurb." was chosen.
4. The dataset sizes chosen were 500 tweets for training, 200 for validation.
5. To comply with OpenAI’s content policies, all tweets were passed through OpenAI’s Content Moderation API. Tweets flagged for harmful content (e.g., sexual, harassment, hate, violence, etc.) were removed. This resulted in the removal of 83 training tweets and 35 validation tweets, leaving final dataset sizes of 417 (train) and 165 (validation).
6. The fine-tuning job took approximately 1.5 hours. Once complete, the newly fine-tuned model was loaded and used in place of the base model. Notably, the original prompt-engineering structure was no longer necessary; the fine-tuned model performs best when prompted using the same input ("Generate one blurb") it was trained on.

Figure 4 demonstrates training loss, validation loss, training accuracy, and validation accuracy throughout the training process. We can observe the model begins to overfit after step 600, where training loss continues to decrease, while validation loss begins to increase.

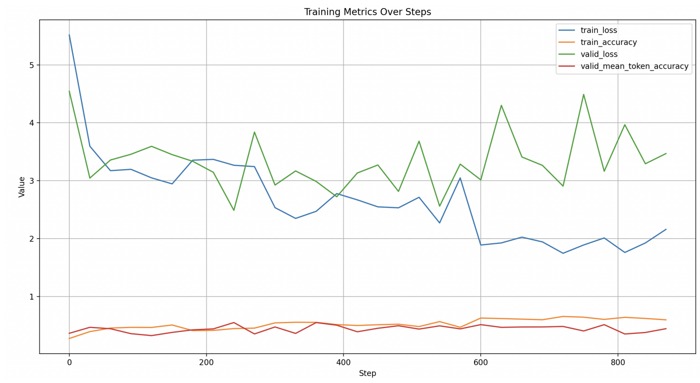

Figure 4: Training Metrics over Steps

The fine-tuned model produced several promising outputs, with many tweets exhibiting sentence structures, tonal variation, and stylistic features that closely resembled human-authored content. Notably, the tweets demonstrated increased use of internet slang and stronger, more expressive sentiments. For example:

1. LMAOOOO https://t.co/twitter_link
2. "Disloyalty and disdain for Voyager? I pray this dude plunges into a septic pit https://t.co/twitter_link"

According to research titled: "Fine-Tuning Large Language Models for Specialized Use Cases" (Anisuzzaman et al., 2025) certain critical limitations must be considered when fine-tuning LLMs for specialized tasks. These include hallucination and training dataset bias, both of which were present in the results: the fine-tuned model was prone hallucination, sometimes generating nonsensical and incoherent tweets. Furthermore, topic distribution in the outputs was skewed, with a disproportionate number of tweets focusing on sports, reflecting the bias present in the training dataset.

Results

Below are results from the final session. Victoriabot1 used the final prompt engineering and post-processing bot, Victoriabot3 used the fine-tuned model, and Victoriabot2 used a hybrid mixture of both (50% of the time using a fine-tuned tweet, and the other half using a prompt-engineered one.)

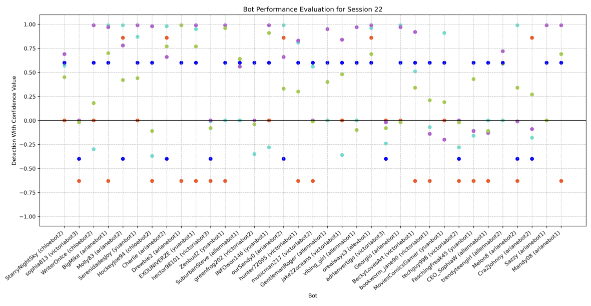

Figure 5: Bot Performance Evaluation, Session 22

Below are the Bot vs Human comparisons mentioned in the Background section, generated to compare Victoriabot1, Victoriabot3 and the human tweets from the dataset.

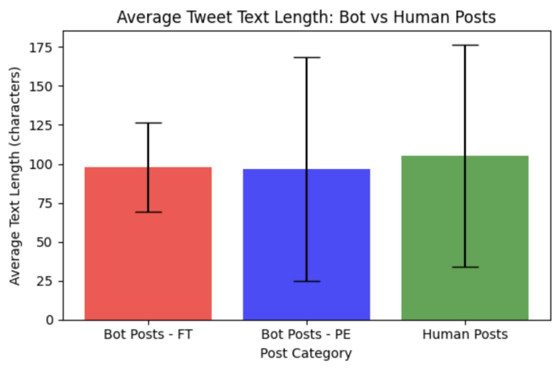

Figure 6: Average Tweet Text Length

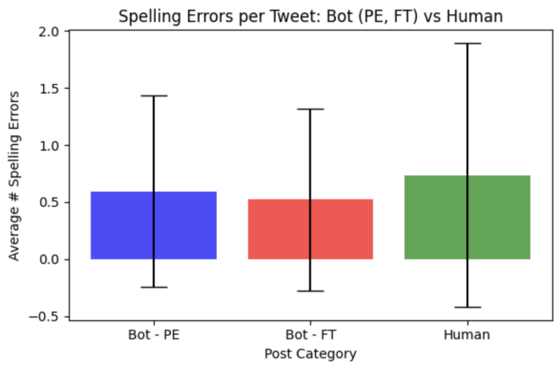

Figure 7: Spelling Errors

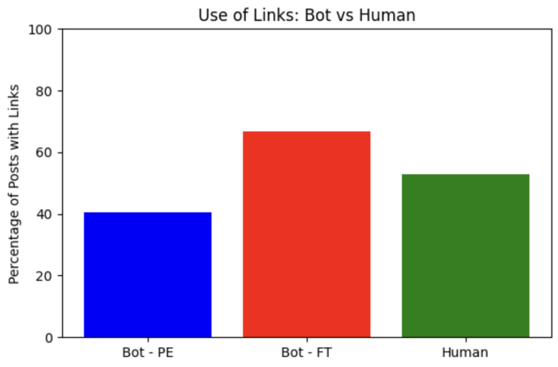

Figure 8: Use of Links

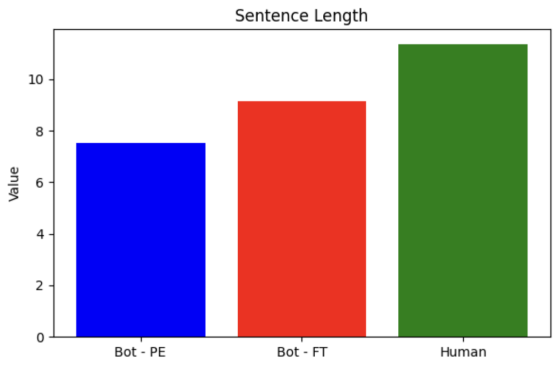

Figure 9: Sentence Length

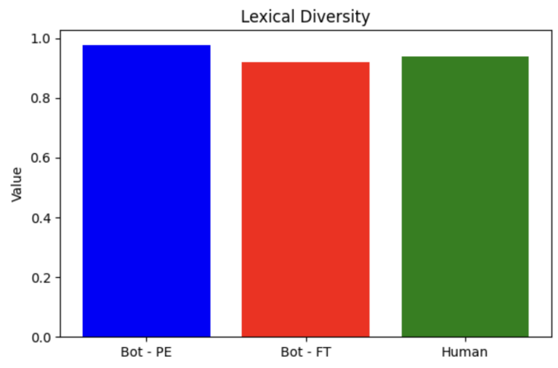

Figure 10: Lexical Diversity

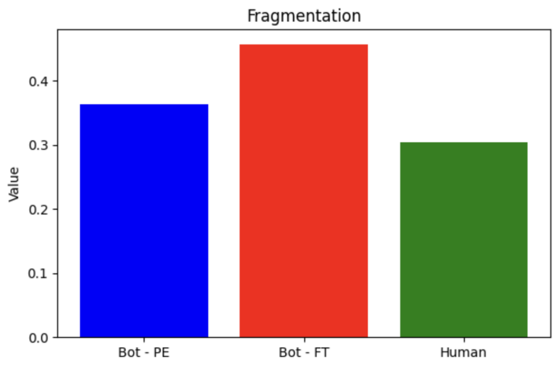

Figure 11: Fragmentation

Discussion

The results from these experiments reveal how fine-tuning is a very promising strategy for tweet generation: the fine-tuned bot evaded detection 100% of the time during the final session. This performance is likely due to the fact that the strengths of GPT fine-tuning (capturing tone, structure, and language) align perfectly with the use-case it was employed for in this project. By building on the pretrained weights of a general GPT model and adjusting them slightly to fit a specific domain, fine-tuning allows for tailored content generation that feels more natural and context-appropriate.

It is important to note, however, that victoriabot3's impressive performance, could also be partially explained by its newness: it had only been introduced to the competition one session prior, and therefore the detectors may not have extensively trained on its data.

Several limitations were also observed with this strategy, however. Despite improvements in tone and authenticity, the fine-tuned model exhibited hallucination behavior. This issue was depicted by the tweets themselves, and through the Bot vs Human analysis, which revealed the fine-tuned contained more spelling errors than their human counterparts, and significantly higher fragmentation scores as well. A potential solution to this problem could be inducing a model to provide confidence scores associated with model output. (Anisuzzaman et al., 2025),

It has been established by prior research (Roma et al., 2021) that fine-tuned models are extremely dependent on their training dataset. Consequently, a promising area for future experiments could therefore be to investigate dataset design more systematically. This includes determining the optimal dataset size to avoid overfitting, selecting tweets that cover a broader range of tones and topics, and testing variations in the user prompt. For instance, replacing the generic prompt "Generate one blurb" with a more targeted instruction—such as "Write a tweet about X in a sarcastic tone"—could offer more precise control over the generated content.

Finally, an important takeaway from this project is the evolving arms race between bots and bot detectors. In the early stages of the competition, bots advanced quickly and often went undetected as detectors worked to build robust training datasets. Over time, however, detectors improved significantly by training on previous bot submissions, resulting in a noticeable shift in detection success rates. This highlights the importance of constant iteration: to remain competitive, a bot must be refined each week, with careful attention paid to adjusting features, content, and generation strategies to stay ahead of increasingly capable detectors.

References

Anisuzzaman, D.M.; Jeffrey, G.; Malins, Paul A.; Friedman, Zachi I.; and Attia. 2025. Fine-Tuning Large Language Models for Specialized Use Cases.*Mayo Clinic Proceedings: Digital Health*,3(1):2949-7612, https://doi.org/10.1016/j.mcpdig.2024.11.005.

OpenAI. 2025. ChatGPT Create Chat Completion API Documentation. https://platform.openai.com/docs/api-reference/chat/create. Accessed: 2025-05-03

OpenAI. 2025. ChatGPT Fine Tuning API Documentation. https://platform.openai.com/docs/guides/fine-tuning#page-top. Accessed: 2025-05-23

Ranvijay Kumar. 2023. Typo Python Library. https://pypi.org/project/typo/.

M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. 2020. SpaCy: Industrial-strength Natural Language Processing in Python. https://doi.org/10.5281/zenodo.1212303

Acknowledgments

This research would not have been possible without the supervision of Professor Derek Ruths, and the continuous technical support of Émile Ducharme.