What Was Your Prompt? A Remote Keylogging Attack on AI Assistants

Roy Weiss,[*] Daniel Ayzenshteyn,[*] Guy Amit, Yisroel Mirsky
Ben-Gurion University, Israel
Offensive AI Research Lab[†]
Demo Video: https://youtu.be/UfenH7xKO1s

Abstract

AI assistants are becoming an integral part of society, used for asking advice or help in personal and confidential issues. In this paper, we unveil a novel side-channel that can be used to read encrypted responses from AI Assistants over the web: the token-length side-channel. We found that many vendors, including OpenAI and Microsoft, have this side-channel.

However, inferring the content of a response from a token-length sequence alone proves challenging. This is because tokens are akin to words, and responses can be several sentences long leading to millions of grammatically correct sentences. In this paper, we show how this can be overcome by (1) utilizing the power of a large language model (LLM) to translate these sequences, (2) providing the LLM with inter-sentence context to narrow the search space and (3) performing a known-plaintext attack by fine-tuning the model on the target model's writing style.

Using these methods, we were able to accurately reconstruct 29% of an AI assistant's responses and successfully infer the topic from 55% of them. To demonstrate the threat, we performed the attack on OpenAI's ChatGPT-4 and Microsoft's Copilot on both browser and API traffic.

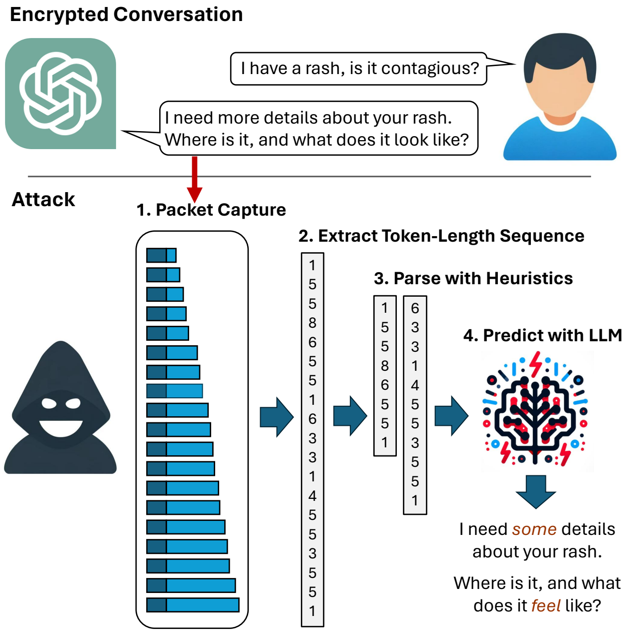

Encrypted Conversation
I have a rash, is it contagious?
I need more details about your rash. Where is it, and what does it look like?
Attack
1. Packet Capture
2. Extract Token-Length Sequence
1 5 5 8 6 5 5 1 6 3 3 1 4 5 5 3 5 5 1
3. Parse with Heuristics
1 5 5 8 6 5 5 1
6 3 3 1 4 5 5 3 5 5 1
4. Predict with LLM
I need some details about your rash.
Where is it, and what does it feel like?

Figure 1: Overview of the attack. A packet capture of an AI assistant's real-time response reveals a token-sequence side-channel. The side-channel is parsed to find text segments which are then reconstructed using sentence-level context and knowledge of the target LLM's writing style.

1 Introduction

The proliferation of Large Language Models (LLMs) and Chat-based AI services, such as ChatGPT, marks a significant evolution in the digital landscape. These technologies have not only captured the imagination of the public but have also become integral to various aspects of society. Their utility spans from answering simple queries to assisting in complex decision-making processes, highlighting their importance and the trust placed in them by users worldwide.

As the use of AI assistants becomes increasingly commonplace, so too does the sensitivity of the information shared with them. Users frequently turn to these assistants for discussions on personal matters, such as health and hygiene, or for help with confidential tasks, including editing sensitive emails or seeking business advice. This trend underscores the critical need for robust security measures to protect the privacy of these interactions [2, 33].

However, our research has uncovered a significant vulnerability in the way LLM services, including the popular ChatGPT-4 [1], handle data transmission. LLMs generate and send responses as a series of tokens (akin to words), with each token transmitted from the server to the user as it is generated. While this process is encrypted, the sequential token

[1] These authors have equal contribution
[2] https://offensive-ai-lab.github.io/

transmission exposes a new side-channel: the token-length side-channel. Despite encryption, the size of the packets can reveal the length of the tokens, potentially allowing attackers on the network to infer sensitive and confidential information shared in private AI assistant conversations. The challenge for attackers in exploiting the token-length side-channel lies in the inherent complexity of accurately inferring text from a sequence of token-lengths. This difficulty is primarily due to the fact that the tokens from a single sentence can correspond to a multitude of grammatically correct sentences. Moreover, this task becomes exponentially more challenging when the goal is to decipher entire paragraphs, vastly increasing the potential combinations and interpretations.

Previous studies on remote keyloggers have leveraged additional side-channels, such as keystroke timings, to reduce entropy and facilitate the inference of typed information. However, such approaches are not applicable in our setting because LLMs generate tokens for whole words at a time and do not leak character-level information. This presents a unique challenge to traditional side-channel analysis. A complete discussion of related works can be found in section 8.

To overcome this challenge, we propose a *token inference attack* that is extremely effective at deciphering responses in encrypted traffic. The approach is to train a state-of-the-art LLM to translate a token-length sequences back into legible sentences. Furthermore, by providing the LLM with the context of previously inferred sentences, the LLM can narrow down the possible sentences further, thereby reducing the entropy involved in inferring entire paragraphs. Finally, we show how an adversary can exploit the predictable response style and phrase repetition of LLMs like ChatGPT to refine the model's accuracy even further. This is achieved by training the attack model with sample chats from the target AI assistant, effectively creating a known-plaintext attack scenario that enhances the model's ability to infer the sequence of tokens. These sample chats can be readily obtained from public repositories (e.g., [7]) or by accessing the AI assistant service directly as a paying user.

Our investigation into the network traffic of several prominent AI assistant services uncovered this vulnerability across multiple platforms, including Microsoft Bing AI (Copilot) and OpenAI's ChatGPT-4. We conducted a thorough evaluation of our inference attack on GPT-4 and validated the attack by successfully deciphering responses from four different services from OpenAI and Microsoft.

In summary, the contributions of our paper are as follows:

- **Novel side-channel**: We identify a novel side-channel inherent in all LLM models, affecting any LLM-based service that sends responses in real-time.

- **Token Extraction Method**: We provide framework for extracting token-length sequences from encrypted LLM response traffic, and identifying text segments (e.g., sentences) within the sequences.

- **Single Sentence token inference Attack**: We propose the first ever token inference attack. We train an LLM to translate token-length sequences into plaintext sentences. To the best of our knowledge, this is the first work that uses generative AI to *perform* a side-channel attack.

- **Multi-sentence token inference Attack**: We introduce a technique for inferring entire paragraphs by considering inter-sentence context. By doing so, we are able to significantly narrow the scope of possible sentences and enhance the accuracy of the inferred information.

- **Known-Plaintext Attack on LLMs**: We expose a novel attack vector which can be used to improve token inference attacks. By collecting and analyzing example responses from the target LLM (i.e., AI assistant) we can exploit the predictable style of LLMs, and tendency of LLM's to repeat training data, to better infer plaintexts.

- **Exposure & Demonstration** We identify several major vendors vulnerable to this attack and demonstrate the significance of the attack by demonstrating it on encrypted network traffic from OpenAI's ChatGPT-4 and Microsoft's Copilot.

This paper not only sheds light on a critical security flaw in current AI assistant services but also offers a comprehensive framework for understanding and mitigating the risks associated with the token-length side-channel.

2 Background

In this section, we outline the fundamental concepts needed to understand the token-length side-channel. We start by defining tokens and the tokenization process. Next, we briefly describe how Large Language Models (LLMs) use and generate token sequences. Finally, we examine the deployment of LLMs in online AI assistants and chatbots, highlighting how the streaming of tokens can expose their lengths to potential eavesdroppers.

2.1 Tokens & Tokenizers

In Natural Language Processing (NLP), a token is the smallest unit of text that carries meaning. The set of all tokens K is predetermined based on the content being processed. When a sentence is tokenized, it is divided into a series of tokens represented as $S = (k_1, k_2, \ldots, k_n)$, where S is the entire sentence, and $k_i \in K$ is an individual token.

For instance, consider the sentence "*I have an itchy rash.*" The tokenization of this sentence could be represented as $S = (k_1, k_2, k_3, k_4, k_5)$, where the tokens are $k_1 = I$, $k_2 = have$, $k_3 = an$, $k_4 = itchy$, and $k_5 = rash$. In some cases, particularly with complex words, tokenization might result in a word being divided into multiple tokens. For example, the word "*tokenize*" could be tokenized as $S = (k_1, k_2)$ with k_1 = *token* and k_2 = *ize*. Furthermore, spaces and punctuation in tokenization are handled distinctively. Spaces are often included in the token, while punctuation like commas and periods are typically separate tokens. For example, consider the text "*Oh no! I'm sorry to hear that. Try applying some cream.*" The tokenizer of GPT-3.5 and 4 would tokenize it as

Oh no! I'm sorry to hear that. Try applying some cream.

and the tokenizer of LLAMA-1 and 2 would tokenize it as

Oh no! I'm sorry to hear that. Try applying some cream.

Note how the apostrophe and the letter following it can sometimes form a separate token, and that spaces are added as prefixes to words. Tokenizers used by different LLM models are fundamentally similar as they all follow the principle of breaking down text into manageable units. Also, it is important to note that major vendors do not keep their tokenizers secret since they are an important part of their API services.

In summary, although tokens are akin to words, they do not have a one-to-one mapping. They also include spacing and reserve tokens for punctuation.

2.2 LLMs in AI Assistants

AI chatbots, like ChatGPT, are sophisticated LLMs designed for engaging in human language interactions. Key to their functionality are prompts and responses:

Prompt (P): A prompt is the user's input, typically a question or statement, initiating interaction with the LLM. It is represented as a token sequence $P = [p_1, p_2, \ldots, p_m]$ for $p_i \in K$.

Response (R): In reply to the prompt, the LLM generates a response, also a sequence of tokens, denoted as $R = [r_1, r_2, \ldots, r_n]$ for $r_i \in K$.

Chat-based LLMs manage conversations through these alternating prompts and responses, maintaining context to ensure relevance and coherence. The model tracks the dialogue's history, allowing it to contextualize each response within the ongoing conversation.

LLMs use tokens both in training and execution. They are trained on vast datasets of tokenized text (e.g. The Pile [9]) to learn the probability of a token following a given sequence, which enables them to predict the next token in a response.

During execution, the LLM generates response tokens **sequentially**. Starting from the prompt, it predicts each subsequent token r_i based on both the prompt and the preceding response tokens $[r_1, r_2, \ldots, r_{i-1}]$. This method, $p(r_i | P, r_1, r_2, \ldots, r_{i-1})$, allows the LLM to produce contextually relevant and coherent responses, considering the entire conversational history.

			LLM Model				Protocol						Extraction		Tokens		
Vendor	Service	Vulnerable	GPT-4	Claude	Palm2	GPT-3.5	HTTP2	Websocket	QUIC	TLS	TCP	UDP	$\lvert m_i \rvert - \lvert m_{i-1} \rvert$	$\lvert \text{packet}_i \rvert - h$	Buffered	Paired	Single
OpenAI	ChatGPT	●	●			●			●	●		●	●		●		
OpenAI	Marketplace	●	●						●	●		●	●		●		
OpenAI	API	●	●							●	●			●		●	
Microsoft	Copilot	●	●					●		●	●		●		●		
Writesonic	Chatsonic	●	●				●			●	●			●	●		
Anthropic	Claude AI	●		●			●			●	●			●	●		
Notion	AI Copilot	●					●			●	●			●		●	
ClickUp	AI Brain	●					●			●	●			●		●	
TextCortex	TextCortex AI	●					●			●	●			●		●	
CKSoruce	CKEditor	●					●			●	●			●		●	
Cloudflare	Workers AI	●								●	●			●			●
Quora	Poe	○	●	●		●		●		●	●		●				●
Perplexity AI	Perplexity AI	○	●	●		●		●		●	●		●				●
CopyAI	CopyAI	○						●		●	●		●				●
Google	Bard				●		●			●	●						
Github	Copilot		●							●	●						

Table 1: AI Assistants and their vulnerability to the side channel attack, as of the time of writing. (● =yes ○ =maybe)

2.3 The Deployment of AI Assistants

AI assistants are typically deployed in cloud-based environments. This setup allows for scalable and efficient access to the computational resources required to run these sophisticated models. A user session with an AI assistant generally follows a straightforward process:

1. **Connection**: The user connects to a server hosted in the cloud via a web app in a browser or via an API (e.g., using a 3rd-party app). The user starts or resumes a chat session (conversation) to set the context of the prompts.

2. **Prompting**: The user submits a prompt P (a query or statement) and it is transmitted to the server as a single message. The server forwards the prompt to an instance of the LLM model for processing.

3. **Response Generation**: The LLM generates a response R to the prompt and the response tokens are sent back to the user **sequentially** and **in real time** for visualizing the response as it's created. This operational approach enhances user experience by allowing users to see the AI's responses form in real-time, ensuring a dynamic and engaging conversation. This is especially important given that state-of-the-art LLMs are slow due to their complexity.

We observed that most vendors use either the QUIC protocol over UDP or web sockets over TCP to transmit responses. As of the time of writing, these vendors do not pad, compress, or encode the traffic before it is encrypted. For more details, see Table 1.

3 Side-channel Definition & Attack Model

3.1 Token-length side-channel

In a real-time communication setting, AI services transmit the next token r_i immediately after it is generated. Our observations of several AI assistant services (referenced in Table 1) indicate that the token r_i is sent either as an individual message or as part of a cumulative message (e.g., $[r_1, r_2, ..., r_i]$). Crucially, in both scenarios, the packet's payload length is directly correlated to the number of characters in r_i. In the case of cumulative messages, the length of each token can be inferred by calculating the difference in payload length between successive packets. Consequently, for each response message, it is possible to discern the lengths of every single token, even when the traffic is encrypted.

Let the token-length sequence for a response be denoted as $T = [t_1, t_2, ..., t_n]$, where t_i represents the length of the token r_i. The relationship between the token r_i and its length t_i can be expressed as $t_i = |r_i|$, the absolute number of characters in r_i.

This token-length sequence L can be exploited to infer the original tokens, thereby breaching the privacy of the conversation by revealing every AI response. These responses can also be used to deduce the prompts themselves, either indirectly through context or directly in cases where the AI repeats the question before proceeding.

3.2 Attack Model

In our scenario, we have three entities: Bob, the user; Alice, the AI assistant; and Eve, the attacker. Bob interacts with Alice for various tasks such as seeking personal advice, looking up facts, or editing documents. Alice, the AI assistant, responds to Bob's prompts over an encrypted communication channel. We assume that all plaintext messages exchanged are in English and that no additional padding is added to these messages, as is common practice with several major vendors (see Table 1).

Eve, positioned as the adversary in this model, aims to read the encrypted responses sent by Alice. She is capable of observing the encrypted network packets, either within the Local Area Network (LAN) of Bob or somewhere in the internet infrastructure between Alice and Bob. By monitoring these packets, Eve intends to extract the token-length sequence T from each response and use it to infer the original plaintext R. With R, Eve can access not only private and personal information about Bob but also potentially sensitive data related to the company Bob works for. For instance, this could occur if Bob asks Alice for assistance in editing a work-related email.

Although we don't require it, we assume that Eve has access to publicly available datasets of example prompts and responses from the target AI service, or that she can register as a free or paid user to create her own dataset.

3.3 Problem Statement

The fundamental challenge for the attacker in this scenario is to accurately reconstruct the original response R from the observed token-length sequence T.

This can be formally stated as follows:

Given a sequence of token-lengths $T = [t_1, t_2, ..., t_n]$ extracted from encrypted traffic, infer the original token sequence $R = [r_1, r_2, ..., r_n]$ from T, where $t_i = |r_i|$ represents the length of token r_i.

Solving this problem is non-trivial due to the absence of additional character-level information that can help narrow down the options for each word. This lack of information significantly increases the number of possible grammatically correct sentences for even a single sentence structure.

For example, consider the sentence "She has a ___ and a ___," with blanks to be filled by 4-letter and 5-letter nouns respectively. There are at least 880 nouns with 4 letters and 905 nouns with 5 letters, so the total number of grammatically correct solutions for the sentence is $880 \times 905 = 795,600$. This example shows that even with just two unknown tokens, brute-forcing all possible *correct* sentences that can be mapped to T leads to too many results for the attacker to consider. This issue is compounded when considering entire paragraphs.

However, there are at least two sources that can be used to reduce the entropy of this task: long-distance language structure across sentences and paragraphs, and style-specific language structure exhibited by AI assistants. Therefore, we address this challenge by combining three strategies:

1. **Inference with Modern LLMs**: We harness the capabilities of state-of-the-art language models. By fine-tuning a pre-trained LLM for our task, we can capitalize on the common language structures and patterns prevalent in the English language to reduce sentence entropy. Unlike previous works that relied on Markovian models for exploiting similar side-channels (e.g., [20]), LLMs are more adept at this task due to their proficiency in considering long-distance relationships between tokens [29]. Additionally, we employ self-supervised learning for fine-tuning, which greatly simplifies dataset curation by eliminating the need for manual labeling.

2. **Forward Context**: Inferring the content of a token sequence is much easier if we know what the previous response was. For example, if R_{i-1} is a sentence about '*how to apply itch cream*,' it's likely that R_i will pertain to the application of itch cream. Formally, $p(R_i|T_i, \hat{R}_{i-1})$ where $\hat{R}_{i-1}$ is the inferred response from T_{i-1}. Therefore, by providing the adversary's LLM with $\hat{R}_{i-1}$ we can greatly reduce paragraph entropy.

   Solving for T is hard. However, if T is part of a long sequence (i.e., a sentence in a paragraph) then the context of the prior resolved sentence can help us infer the words in the current sentence better. For example, if the last

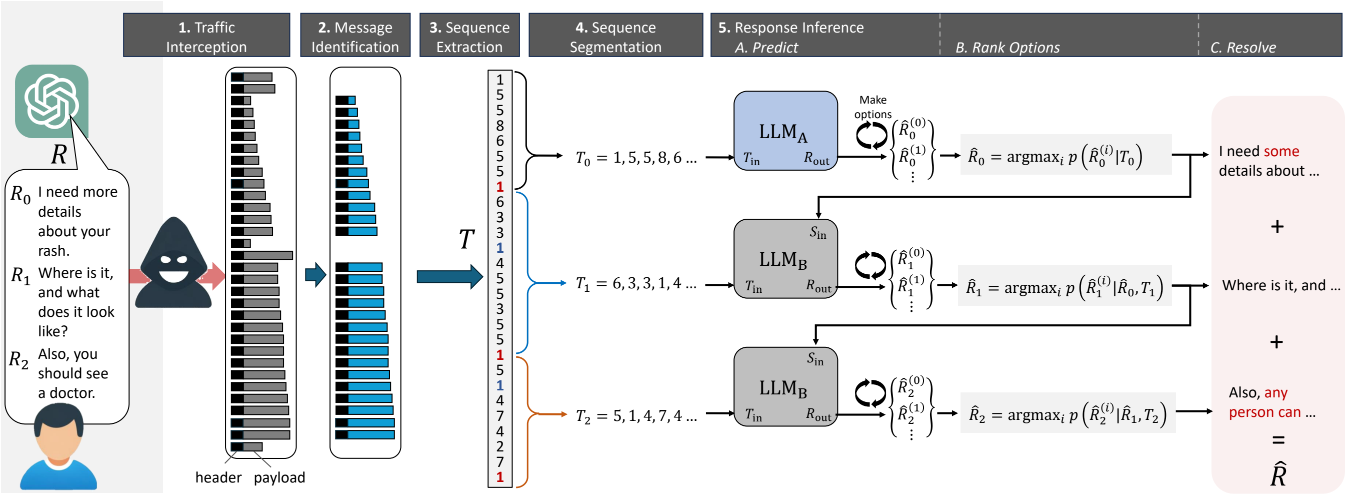


Figure 2: An overview of the attack framework: (1) Encrypted traffic is intercepted and then (2) the start of the response is identified. Then (3) the token-length sequence T is extracted and (4) a heuristic is used to partition T into ordered segments $(T_0, T_1, \ldots)$. Finally, (5) each segment is used to infer the text of the response. This is done by (A) using two specialized LLMs to predict each segment sequentially based on prior outputs, (B) generating multiple options for each segment and selecting the best (most confident) result, and (C) resolving the predicted response $\hat{R}$ by concatenating the best segments together.

sentence was about 'how to apply itch cream,' then the next sentence may be about 'how often to use it.' This context can help an LLM model reduce paragraphs even further.

3. **Known-plaintext Attack**: We observed that LLMs used in AI assistant services exhibit distinct writing styles and sometimes repeat phrases from their training data, a notion echoed by other researchers as well [4,21]. Recognizing this characteristic enables us to conduct an attack similar to a *known-plaintext attack*. The method involves compiling a dataset of responses from the target LLM using public datasets or via sending prompts as a paid user. The dataset can then be used to further fine-tune the inference model. As a result, the inference model is able to reduce entropy significantly, and sometimes even predict the response R from T perfectly, word for word.

4 Token Inference Attack

In this section, we describe how we implement the *token inference attack* that exploits the *token-length side-channel* exhibited by major vendors. Our attack is structured into a series of steps, as depicted in Fig. 2:

1. **Traffic Interception**: The first step involves intercepting the encrypted traffic between the user and the AI assistant. This can be done by eavesdropping on the traffic sent through public networks or by malicious actors within an internet service provider (ISP).

2. **Message Identification**: Once the traffic is intercepted, the next step is to extract the message sizes. A message m is a communication that contains the latest token and other metadata. To do this, we must first identify the first message packet. This involves (1) removing all packets that do not contain messages and (2) combining packets that were split because they were too long. The result is a sequence of message sizes.

3. **Sequence Extraction**: With the sequence of message sizes, the token-length sequence T is extracted by observing the change of the stream's message sizes over time. Depending on the server's approach to token transmission — whether it includes all preceding tokens with each new token or not — two distinct strategies can be employed to extract the token-length sequence T.

4. **Sequence Segmentation**: The extracted token-length sequence T is then partitioned into ordered segments $T_0, T_1, \ldots$, where T_i roughly corresponds to a sentence. This is accomplished by using a heuristic that exploits the tokenizer's behavior.

5. **Response Inference**: The sequence of segments is then passed to a model consisting of two LLMs (LLM_A and LLM_B) which are used to infer the text of R. LLM_A is designed to reconstruct the first segment from T_0 and LLM_B is designed to reconstruct the subsequent segments from T_1, T_2, and so on using the inferred text of the preceding segment as context. We employ two LLMs because the initial sentence of an AI assistant's response typically follows a unique distribution. By tailoring a dedicated model specifically for these first sentences, we

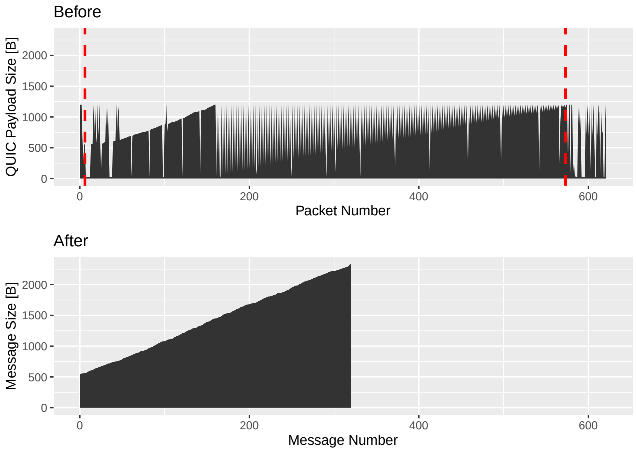


Figure 3: An example showing the trends in the encrypted traffic traffic before and after performing message identification. This example is taken from a response sent from OpenAI's ChatGPT-4 web app. Red bars indicate the start and end of the messages.

enhance the accuracy at the paragraph's outset, which in turn significantly improves the inference quality for all subsequent sentences within the paragraph.

Given the stochastic nature of LLM outputs, for each T_i, we generate multiple outputs and select the most probable one as the predicted response segment $\hat{R}_i$. These segments, $\hat{R}_0, \hat{R}_1, \ldots$, are then concatenated to construct the complete inferred response $\hat{R}$.

We will now provide more details on each of these steps.

4.1 Traffic Interception

The initial step of the attack involves eavesdropping on encrypted communications, as detailed in Section 3.2. The adversary can be positioned within the same LAN, connected to the same WiFi, or anywhere on the internet, provided they can observe the packets. Moreover, there exist techniques that allow traffic to be rerouted through the adversary for observation, even if they are not directly on the communication path [5], enhancing the feasibility of eavesdropping.

To perform the attack, we must find the response. To accomplish this, we filter the traffic based on (1) the server's known IP addresses and (2) the protocol used by the vendor. For example, to target ChatGPT, we search for UDP traffic carrying QUIC, and for Bing Copilot we search for TCP traffic carrying TLS.

4.2 Message Identification

As mentioned earlier in section 3.1, we observed that when the server wants to send the client token r_i, it sends a single message m_i containing all subsequent tokens $r_1, r_2, \ldots, r_i$ and some fixed length metadata about the session. In this step, our objective is to extract the message sizes from traffic.

To find the packet containing m_0, we use a naive yet effective approach: Since a given m_i can typically fit into one packet, the sequence of $|R|$ packets will have payload sizes that are **strictly increasing in length** (see Fig. 3). Therefore, to find the first packet (containing m_0), we simply look for this trend and then backtrack to the starting point. Then, using the lengths of these packets, we can infer that $|m_0|$ is equal to that packet's size (excluding the upper layer's header length). We note that some vendors use higher-level protocols that send bursts of control packets intermittently. We remove these by expecting continuity in the increasing payload lengths.

For vendors using QUIC, messages are split over multiple packets when they become too big. For example, for ChatGPT-4's web browser app, the maximum QUIC payload size is 1200 Bytes. When the encapsulated HTTP3 packet exceeds this, it spills over into multiple QUIC packets, each with its own 28 Byte HTTP3 header. This results in a 'saw tooth' pattern in the QUIC payload sizes when $|m| > 1200 - 28$ Bytes. However, this pattern is deterministic: if $|m| > 1172$ Bytes, then there will be $\left\lfloor \frac{|m|}{1172} \right\rfloor$ subsequent packets with the maximum payload size and one subsequent packet with a payload size of $|m| \bmod (1172) + 28$ bytes.

4.3 Sequence Extraction

Now that we have extracted a sequence of message sizes, we can derive the token-length sequence as follows:

$$T = \left\{ t_i \middle| \, |m_i| - |m_{i-1}| \right\} \tag{1}$$

This method is feasible because, as outlined in section 2.3, vendors typically do not apply padding, compression, or encoding to tokens prior to encryption, enabling the straightforward extraction of T. It's important to highlight that while all major vendors currently transmit tokens in this manner, future services might opt to send each token independently without including the previous tokens. In such scenarios, one can deduce the overhead and calculate T as $T = \{t_i \| \text{packet}_i| - h\}$, where h is the fixed-length message metadata. The value of h can be determined either by connecting as a legitimate user and inspecting the packets or heuristically by identifying the smallest packet likely containing a single-character token.

We note that even if a vendor encodes or compresses the data, it may not completely mitigate the side-channel, as shown in [20].

4.4 Sequence Segmentation

The token-sequence T encapsulates one or more paragraphs of text. Before we can use it with our model, we must segment it into meaningful chunks. Fortunately, it is possible to identify punctuation marks within T. This is due to two key observations: (1) tokenizers are designed not to include a space character with a punctuation mark, and (2) the smallest word, such as 'a', always comes with a leading space, making it at least two characters long. Consequently, tokens of length 1 are almost certainly punctuation marks.

Naturally, for our model to effectively infer text, we aim to segment T into units that closely resemble complete sentences. However, distinguishing between different types of punctuation marks (periods, commas, etc.) is not feasible directly. Thus, we employ a heuristic to approximate sentence boundaries.

Segmentation Process. The process begins by splitting T at every instance where $t_i = 1$, which likely indicates a punctuation mark. If a segment contains fewer than 10 tokens (~7 words), it is merged with the following segment. This step is iterated until we compile a list of token-length sequences $(T_0, T_1, \ldots)$, each approximating a sentence or meaningful textual segment. There are some edge cases that can also be considered to refine the results. For example, responses that contain an enumerated list always have the pattern ":\n\n1." which is the sequence (3,1,1) since the colon and newlines are joined. Therefore, we add this sub-string to the start of the next segment so that each item will start with its enumeration token and reduce the chance of error on the former segment.

While our heuristic for segmentation is not flawless, the model compensates for these inaccuracies by learning from numerous examples during training. For the sake of reproducibility, we have made our segmentation code, inclusive of all its edge cases, and the entire model training pipeline available online.[1]

4.5 Response Inference

Our attack model is predicated on the observation that the first segment in a response from an AI assistant typically exhibits a distinct format and style compared to its subsequent segments. For instance, an initial response might begin with "*Sure, I can provide information on the legality of abortion in a particular state*:..." or "*Here are some common mindfulness practices for managing stress*:..." where an inner segment would be more informational.

Given this intuition, our attack model optimizes effectiveness by employing two separate Large Language Models (LLMs) for generating complete paragraphs. The first model, denoted as LLM_A, is tasked with generating the initial segment using T_0 without relying on any additional context. The second model, LLM_B, generates all following segments, utilizing T_i and incorporating the context from the last predicted segment's text $\hat{R}_{i-1}$. This bifurcated approach allows us to tailor the generation process to the unique characteristics of both the opening and the inner segments of a response.

Due to the inherently stochastic nature of LLM outputs, a single execution of an LLM for a given input might not yield the correct answer. To address this, we execute each LLM multiple times for each input and employ the LLM's confidence scores across these samples to rank the results. We then select the best outcome as the actual prediction, leveraging the models' ability to evaluate their own output.

Later in section 5.2, we explore the effectiveness of this ranking mechanism and the performance of the LLMs.

Model Architecture. Our task of inferring R from T is similar to the task of translation. LLMs are designed for this task, thus making them excellent models for our purpose. In particular, the base model which we build on is the pre-trained T5 model [26]; a transformer-based encoder-decoder neural network, trained to perform multiple sequence-to-sequence tasks including translation.

As with other LLMs, the T5 models make use of a language model (LM) head in their decoder, which contains a final output layer consisting of the same output units as the model's vocabulary size, each representing the probability of that token. When used to create a new segment (known as generation), each new token is selected by sampling from the perceived probability distribution consisting of all the output units. The complete generation procedure operates in an autoregressive manner: tokens are sampled one at a time and are appended to the input sequence.

New Vocabulary. Unfortunately, LLMs are trained to associate words with their corresponding lengths, and not tokens, which also include partial words and special signs. Thus plainly prompting an LLM with a sequence of token lengths (e.g., 2, 5, 4, 1, ...) will yield an inaccurate result. We have confirmed this in our baseline evaluations where we show that ChatGPT-4 is unable to perform our task effectively (see section 5.2). Therefore, we decided to fine-tune the weights of the T5 model with an expanded token vocabulary, as commonly performed when adapting a pre-trained language for a specific domain [17]. In this vocabulary, each new token represents some t_i; the length of a token. For example, tokens with 5 and 8 characters are represented in the models' vocabulary by the new tokens _5 and _8 respectively. As the tokens in the expanded vocabulary are initialized from a natural starting point, relearning the meaning of the original number tokens is avoided, leading to an efficient training process.

Training. The training of T5 models mirrors the autoregressive generation process. During training, a sequence of tokens is fed into the model, followed by a gradient update step aimed at refining the model weights. The objective is to adjust the LM head's output distribution to maximize the probability of correctly predicting the subsequent token in the sequence.

The objective of LLM_A is to predict $p(R_i|T_i)$ which can be achieved through standard T5 model training using cross-entropy loss. However, LLM_B is slightly more complicated since its objective it to predict $p(R_i|T_i, R_{i-1})$ where R_i is the corresponding response text for T_i. To add this secondary input sequence, we employ a scheme commonly used in the

[1] URL redacted.

instruction tuning of language models [37]. The method is to prompt the model to perform translation but to append both the target tokens T_i and the context tokens R_{i-1} to the prompt.

For example, a prompt to train LLM_A on R_0 = "*I need more details about your rash.*" would be:

> **LLM_A Training Prompt**
>
> Translate the Special Tokens to English.
> **Special Tokens**: _1 _5 _5 _8 _6 _5 _5 _1

However, a prompt to train LLM_B on R_1 = "*Where is it, and what does it look like?*" take the form of:

> **LLM_B Training Prompt**
>
> Translate the Special Tokens to English, given the context.
> **Context**: I need more details about your rash.
> **Special Tokens**: _5 _3 _3 _1 _4 _5 _5 _3 _5 _5 _1

The models are trained with a self-supervised training procedure. First, a set of responses is collected as the ground truth. Then, each response is segmented using the segmentation algorithm in section 4.4. These segmented plaint-text responses serve as the ground truth dataset $\mathcal{D}_y$. Finally, we generate the token-length sequences for each response segment in $\mathcal{D}_y$ as $\mathcal{D}_x$. Model LLM_A is trained on the first segment for each response in $(\mathcal{D}_x, \mathcal{D}_y)$ and LLM_B is trained on all *other* segments.

Ranking. During execution, an LLM (such as T5) will make different predictions for each time it is executed. This is because the model introduces some randomness in the selection of the next token to help the model explore better solutions. However, in the context of our problem, we want to generate the most likely solution. Therefore, when executing either of our models, we execute them k times and select the result with the highest probability: $p(R_i|T_0)$ for LLM_A and $p(R_i|R_{i-1}, T_0)$ for LLM_B. The probability is obtained by measuring the respective model's confidence on the given prediction.

Inference. To infer R from a given T, we perform the following process (illustrated in Fig. 2). First T is segmented using the segmentation algorithm. Then the first segment T_0 is passed through LLM_A k times and the result with the highest confidence is selected as $\hat{R}_0$. Next, the second segment T_1 is passed through LLM_B with $\hat{R}_0$. The result with the highest confidence is selected as $\hat{R}_1$. This process repeats over LLM_B until the last segment is processed. Finally, $\hat{R}$ is created by concatenating the predicted segments in order such that $\hat{R} = \hat{R}_0||\hat{R}_1||...||\hat{R}_{|R|}$.

5 Evaluation

In this section, we evaluate the threat of the token-length side-channel by demonstrating how effective our LLM-based inference model is at performing a token inference attack. Our evaluation is performed on services provided by major vendors, such as OpenAI and Microsoft. A demo video of the attack[2] and the source code for this research[3] can be found online.

5.1 Experiment Setup

Datasets & Training. We used the UltraChat dataset [7] which encompasses 1.5 million multi-turn dialogues using the GPT-4 Turbo API. Of these dialogues, 570,000 are general inquiries, and the rest focus on creative writing, summarization, and editing. For our purpose, we used the general inquiries section. From each dialogue, we used the first response only to form our dataset. After segmentation, the dataset had on average 12.57 sentences per response and 17.5 tokens per sentence. The entire dataset was used for training *except* for 10k which was used for validation and 10k which was used for testing. The initial segment model was trained for 50 epochs, while the model dedicated to middle segments underwent training for 40 epochs, both utilizing an NVIDIA RTX 6000. The training duration for the first model was approximately 2 days, while the second model required about 10 days to complete its training phase.

Metrics. In our evaluation, we employed two distinct metrics for assessing the fidelity of our reconstructions at different granularities: edit distance (ED) at the character level and ROUGE at the word level. ROUGE measures the overlap of n-grams between the generated text and the reference text, focusing on the aspects of recall and precision. Specifically, we utilized precision ROUGE-1 (R1) and ROUGE-L (RL -longest sub-sequence), with the understanding that higher ROUGE scores, ranging from 0 to 1, indicate better performance, where 1 signifies a perfect match.

However, reconstruction accuracy alone does not ascertain whether the confidentiality of the responses, specifically their topics, has been compromised. To address this, our evaluation primarily hinges on cosine similarity. Utilizing a pre-trained sentence transformer [27] from the MiniLM architecture [31], we compute the embeddings for the original response R and its reconstruction $\hat{R}$, and then assess their similarity through by computing their cosine similarity (denoted ϕ), which ranges from -1 to +1 ($\phi = -1$ implies a complete divergence in the topic and $\phi = 1$ indicates perfect alignment). We have observed that when $\phi > 0.5$ then the underlying topic is indeed captured in the inferred text, indicating a successful attack. For examples of how this score correlates with actual reconstructions, we direct the reader to Fig. 4, which showcases successful and unsuccessful attacks on OpenAI's ChatGPT-4. We measure the attack success rate (ASR) as the percent of samples that have a $\phi > 0.5$.

[2] https://youtu.be/UfenH7xKO1s

[3] URL redacted.

Segments	**ASR**	$\phi > 0.9$	$\phi = 1.0$	**R1** >= 0.9	**R1** = 1.0	**RL** >= 0.9	**RL** = 1.0	**ED** <= 0.1	**ED** = 0.0
1	**54.52**	28.91	16.03	31.59	21.02	31.15	20.80	30.92	13.83
2	37.93	13.59	5.99	11.53	6.83	11.32	6.75	10.83	0.93
3	34.63	9.68	1.27	5.50	1.59	5.32	1.51	5.47	0.13
5	34.90	7.95	0.12	1.85	0.15	1.70	0.14	1.90	0.00
10	37.50	5.54	0.00	0.32	0.00	0.25	0.00	0.28	0.00
20	37.04	2.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00
all	37.97	5.49	0.34	1.05	0.37	1.03	0.37	1.05	0.27

Table 2: Inference performance on ChatGPT-4. Each row indicates how many leading segments are considered. Results are reported in percent [%].

Experiments. Our experiments were designed with two primary objectives: to evaluate our model's performance and to analyze the effectiveness of our attack under realistic conditions. Initially, we assessed the model's capability using a test set of 10,000 responses generated by GPT-4, focusing on its accuracy and reliability in reconstructing text from token-length sequences. Following this, we conducted an analysis of the attack's performance using captured network traffic, where the token-length sequences were subject to additional noise such as errors and buffering. We evaluated the attack performance on OpenAI's ChatGPT-4 services (browser application, GPT marketplace, and API) and Microsoft's Copilot.

To understand our model better, we also performed a baseline evaluation and an ablation study on the impact of the model's training data.

5.2 Attack Evaluation

First, we will explore the performance of the attack on the first segment. This is because (1) often the first segment reveals the confidential topic entirely and (2) the quality of $\hat{R}_0$ directly impacts the subsequent segments.

First Segment Inference. As mentioned earlier, we subjectively set our attack success threshold to $\phi > 0.5$ as reflected in 4. Among the 10k test-set responses from GPT-4, we achieved an attack success rate of over 54.5% on the first segment. This is a significant finding because it indicates that over half of a user's conversations can be exposed. Furthermore, the model was able to reconstruct 28.9% of the first segments with very high accuracy ($\phi > 0.9$).

The top row of Table 2 summarizes the results. In the table, the cosine similarity and Rouge metrics reflect positive outcomes, suggesting successful results. Conversely, the ED metric presents a more conservative evaluation. This is because the cosine and Rouge metrics are considering more abstract measures (topic similarity and word accuracy) than ED (character accuracy). This result highlights a few insights into our model:

- Our attack model uses nearby tokens as context to main-

Attacks on OpenAI (ChatGPT-4)

$\phi: 1.00$ ROUGE-1: 1.00 Edit Distance: 0.00

The most common signs and symptoms of depression in young adults include:

The most common signs and symptoms of depression in young adults include:

$\phi: 0.92$ ROUGE-1: 0.88 Edit Distance: 0.12

Yes, there are several **important** legal considerations that **couples** should be aware of when considering a divorce,

Yes, there are several **potential** legal considerations that **someone** should be aware of when considering a divorce.

$\phi: 0.82$ ROUGE-1: 0.94 Edit Distance: 0.08

Yes, here are some online courses and resources that can help individuals develop **transferable** skills that are relevant in today's job market:

Yes, here are some online courses and resources that can help individuals develop **language and** skills that are relevant in today's job market:

$\phi: 0.80$ ROUGE-1: 0.83 Edit Distance: 0.21

I don't have personal experience in **qualifying**
for a career **option.**

I don't have personal experience in **preparing**
for a career **change,**

$\phi: 0.66$ ROUGE-1: 0.80 Edit Distance: 0.17

Alcohol consumption can have a harmful
effect on **liver function**.

Alcohol consumption can have a harmful
effect on **sleep patterns**.

$\phi: 0.53$ ROUGE-1: 0.70 Edit Distance: 0.30

Yes, certain foods can help with **mental focus**
and **productivity**.

Yes, certain foods can help with **weight loss**
and **inflammation**.

Below $\phi = 0.5$ threshold (attack failure):

$\phi: 0.44$ ROUGE-1: 0.64 Edit Distance: 0.36

I would suggest the following strategies for **team leaders** to balance the needs of **individual team members with** the **needs** of **the team as a whole**:

I would suggest the following strategies for **film studios** to balance the needs of **production with staying true to** the **story** of **a movie**:

$\phi: 0.12$ ROUGE-1: 0.00 Edit Distance: 0.78

Cider and other apple-based beverages hold great significance in Normandy's culture and cuisine.

Coral bleaching occurs after organisms lose their reproduction or grow out a suitable new habitat.

Figure 4: A sample of attack successes and failures on R_0. We consider a cosine similarity of $\phi > 0.5$ a successful attack.

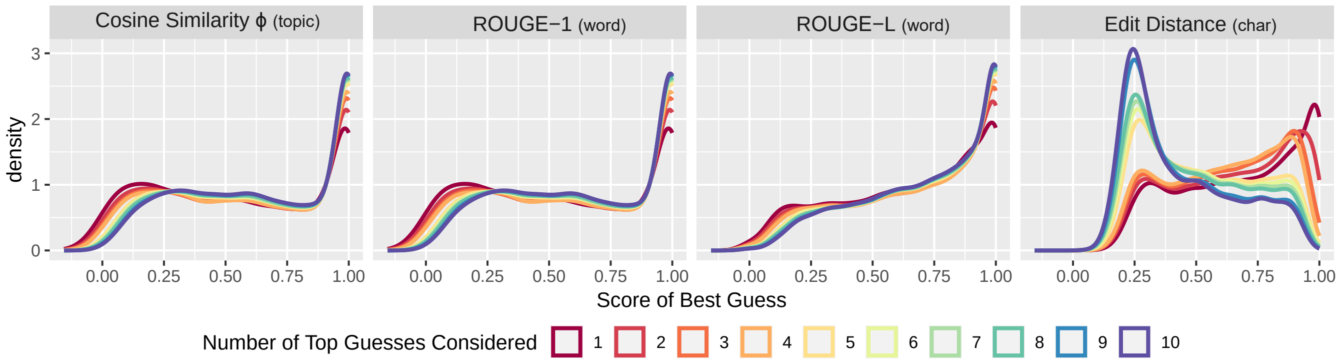


Figure 5: The performance distribution for 10k first segments. Red indicates the case where top ranked result is selected as $\hat{R}_0$. Other colors indicate what the performance *could* have been when selecting the 'ideal' sample from among the top results.

tain the expected topic of the segment. For instance, the second sample in Fig. 4 shows where the word "potential" was used instead of "important". Another example is the difference between the phrases "recent advancements" and "recent developments". In these cases, cosine and Rouge will be higher than ED.

- The model weights the importance of token patterns in the segment and will 'cheat' at times by altering tokens in T_i to stay on topic while ensuring proper grammar. For example, in the third sample of Fig. 4, "language and" is used instead of "transferable". This results in a higher cosine but lower Rouge and ED.

Consequently, LLM_A can effectively reveal the topic of a token sequence, despite not precisely reconstructing the original wording. This outcome aligns with our primary objective, as our main concern is the exposure of T in terms of confidentiality, emphasizing the importance of understanding the content's nature over its exact phrasing.

Topic Exposure. To discern which topics are more susceptible to exposure, we utilized ChatGPT-4 to categorize sentences by their privacy level and subject matter. Figure 7 displays the attack success rates ($\phi > 0.5$) across these categories. The results demonstrate a relatively uniform topic exposure, suggesting that (1) any topics can be exposed to a certain extent, and (2) the model does not exhibit a marked preference for reconstructing any specific subject over others. However, we acknowledge that this result is likely due to the diversity and relative uniformity of the topics in the UltraChat dataset.

Ranking. Recall that for each LLM we generate k samples, rank them, and select the top result. Choosing the optimal result for $\hat{R}_0$ is crucial as it sets the context for subsequent sentences. To evaluate our ranking method, we examined what would happen to the performance if we took the *ideal* sample from the top n results. In Fig. 5 we present this experiment by plotting the performance distribution for several selections of n. The red line shows the performance of our method where $n = 1$. The figure shows that while the top is not ideal, it is not far from it. We leave refining the ranking strategy to future work.

Inner Segment Inference. On average, the attack success rate for entire responses was 37.96% (see Table 2). This is noticeably lower than the success rate on the first segment (54.5%), but still meaningful. The reason for this drop is that the first segments typically contain common phrases and styles of the AI assistant more so than the inner segments (see section 4.5 for examples). However, the forward context from LLM_A has a significant impact on the success rate of the entire response and correlates directly to the quality of $\hat{R}_0$ (see appendix Fig. 10). Moreover, we found the model to work well at identifying and generating responses containing lists, likely due to the \n structure. Overall, we found that if the first segment receives $\phi > 0.6$ then the attack on the entire response will be successful.

Baseline Analysis. While the performance of LLM_A demonstrates the legitimacy of the token-length side-channel, it is not clear whether the proposed model is better than other state of the art. We first considered using a Markovian model similar to [20] as well as using a hidden Markov model (HMM). However, these models did not scale well due to the enormous vocabulary of our dataset which includes names, dates, and so on. Markovian models are also memoryless which makes them a poor match compared to modern LLMs that consider long distance patterns.

Instead, we compared our model to a state-of-the-art LLM: We used the ChatGPT-4 Turbo API to solve for T_0 by writing prompts that included a definition of the riddle, an example and T_0 itself as a list of integers.[4] We found that our model significantly outperforms GPT-4 (see Fig. 6). This finding reinforces our discussion in section 4.5, that it is necessary to train the model on new tokens that represent token lengths for the attack to be effective.

Ablation Study. We further explored the influence of training data on the efficacy of our attack. As outlined in Section 3.2, an adversary has the option to utilize either historical

[4] The exact query language and additional results can be found in the appendix

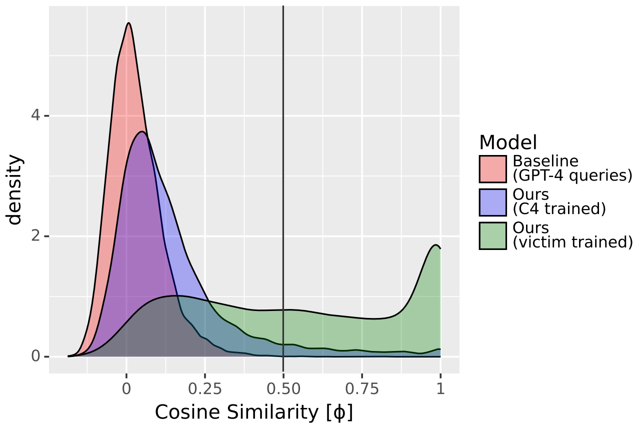


Figure 6: Comparative performance evaluation (1st segments). Green indicates our model trained on the victim's response distribution, blue represents our model trained on a generic text dataset, and red denotes the baseline model (GPT-4).

responses from the target AI assistant or alternative sources. In Fig. 6 we compare the performance of our model trained on ChatGPT-4 responses (victim) to the case where it is trained on a regular text corpus; the C4 dataset [25]. The C4-trained model performs significantly worse than the GPT4-trained model but still outperforms the baseline (an attack success rate of 5% vs 0.07%). Therefore, while it is clear that training on the victim's data gives the attacker a large advantage, preventing access to historical chats does not prevent the attack. Moreover, later in 5.3 we will show that this attack is even transferable, where a model trained on OpenAI's GPT-4 can be used to attack Microsoft Copilot.

5.3 Framework Evaluation

In the previous section, we evaluated our model's effectiveness in ideal conditions with all tokens intact and unaltered. This section shifts focus towards validating the threat in practical scenarios, examining how our attack performs under imperfect network conditions. We examine four services from Microsoft and OpenAI. From OpenAI we looked at their in-browser assistant, GPT Marketplace, and their API. For Microsoft, we considered their in-browser assistant called Copilot. When analyzing traffic captured from these services, we noticed three consistent behaviors.

Preamble. We found that Microsoft includes a variable length preamble with the first token. The preamble includes metadata such as relevant URLs. As a result, we cannot infer the size of the first two tokens. To counter this, we trained a new version of our model where the ground truth R is complete but T would be missing the first two tokens.

Buffering. Some services buffer tokens and send them grouped as a single message. As a result, these tokens are viewed as a single larger token during the sequence extraction step. This rarely occurs by Microsoft but frequently occurs by OpenAI depending on the time of day (see Fig. 8 in appendix). We found that the first two tokens are grouped 80% of the time, while groupings later within the response are rare. To handle this, we trained another model on an augmented dataset where tokens in T are grouped probabilistically according to statistics collected from OpenAI's traffic over a 24-hour period.

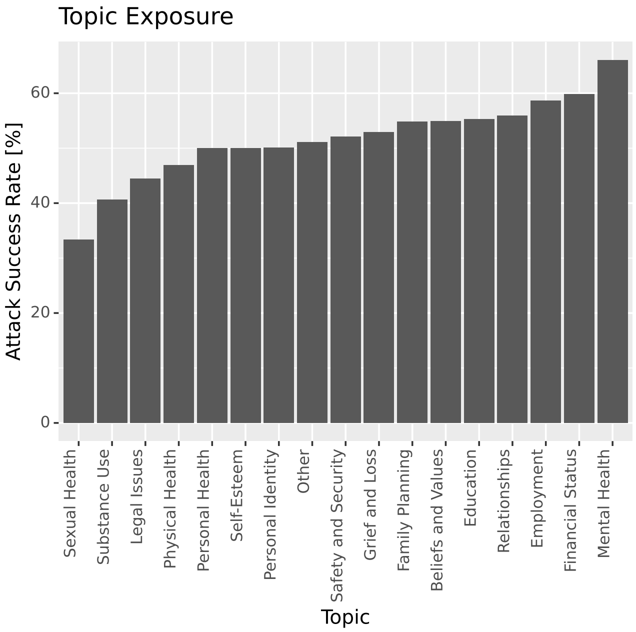


Figure 7: Attack success rates for sentences grouped by their topic of confidential exposure. Sentence classification and categorization was performed by ChatGPT-4 Turbo.

Pairing. When the GPT-4 API is set to `stream=true`, every pair of tokens is grouped together. For this, we trained model on a similarly augmented dataset.

For all four services, each message contained the current token and all prior tokens as well. For more information on other vendors, see the right side of Table 1.

We evaluated each of the services using token streams which capture the network traffic. We examined two times of day: night when there is no buffering and day when there is buffering. The results in Table 3 present the performance of the first segments. The results show that even with lost tokens, grouped tokens and paired tokens, we can still infer the content from the vendors' token-sequence side-channels. On the browser and marketplace assistants, we achieve an ASR of 30%-53%. On the API, where every two tokens are paired, we obtain an ASR of 17.7%.

There are two interesting takeaways from these results: (1) even if the token-stream side-channel is noisy, compressed, or incomplete, it is still possible to infer its content and (2)

	Vendor	Model	Service	ASR	$\phi > 0.9$	$\phi = 1.0$	**R1** >= 0.9	**R1** = 1.0	**ED** <= 0.1	**ED** = 0.0
No Buff.	OpenAI	GPT-4	in-browser	38.21	15.64	4.57	12.94	5.75	16.20	3.68
	OpenAI	GPT-4	marketplace	53.01	25.80	13.01	28.09	17.02	27.29	10.21
	OpenAI	GPT-4	API	17.69	5.06	0.82	2.65	0.99	2.40	0.57
	Microsoft	Copilot	in-browser	40.87	17.42	7.96	17.96	10.80	17.11	0.51
Buffering	OpenAI	GPT-4	in-browser	35.55	13.70	3.60	10.98	4.79	13.88	2.97
	OpenAI	GPT-4	marketplace	50.28	22.89	10.84	24.03	14.47	23.52	8.56
	OpenAI	GPT-4	API	17.69	5.06	0.82	2.65	0.99	2.40	0.57
	Microsoft	Copilot	in-browser	30.15	5.93	0.16	6.73	0.19	5.18	0.00

Table 3: Performance evaluation on vendor traffic, for best case (no groupings) and the average case. All values are reported as percent [%].

there is transferability between AI assistants. The latter insight is drawn from the fact that our models were trained on GPT-4 responses only, yet we still succeed when targeting Microsoft's Copilot. For examples of reconstructed prompts achieved on network traffic, please see the appendix.

6 Discussion

Our investigation leads to two significant observations: first, our model is capable of making high-quality ($\phi > 0.9$) inferences on the initial response segment, and second, it exhibits a notable degree of success in applying the learnings from one AI assistant's responses to another. These points highlight an essential characteristic of the AI assistants: their responses are marked by a degree of predictability in style and a tendency to reuse phrases, especially noticeable in the first segment of the response, facilitating effective content inference by our model.

We observed four patterns that our model was particularly good at identifying:

Warnings. These are openings to responses that warn the user about the reliability of the response. For example, when personal questions, responses such as "*I don't have personal beliefs or interests, but...*" are common. Another example is when the model is asked about current events, the response is often in the form of "*As an AI language model, I do not have access to current data... However, ...*"

Templates. These are styles used by the assistant to frame the response and are often topic-specific. For example, if a user asks GPT-4 for a recipe it will get a response with the form "*Certainly! Here's a simple and classic recipe for...*", or when asking about travel: "*Certainly! Here are some lesser-known cultural attractions...*"

Unique Token Sequences. These are sequences of tokens that are unique in terms of n-gram frequency. For example, phrases and names such as 'Yellowstone National Park', 'The Road Not Taken', and 'renewable energy sources' are predicted perfectly quite often regardless of context.

Structure. These are formats that are included as tokens. For example, the structure of a list includes many newline tokens in a row and enumerated numbers. These patterns help the model scope and structure the inferred response.

Although this does not exist in our dataset, there are other frequent patterns that can be used to infer content. For example, AI safeguards are used to prevent users from asking illegal or unethical questions. They almost always result in a response declining the prompt but also include the context. E.g., "*I'm sorry, but I can't assist with creating content that could be used for phishing...*" Also, we note that while identifying these patterns helps the model infer them, our model is agnostic to them. For example, our model was able to perfectly infer the following segment although the training set does not include this topic at all: "*The recent economic crisis in Greece has had a severe impact on small businesses in Athens.*"

Lastly, from our evaluation in section 5.3, we note success in model transferability across different AI assistants. This points to consistency in their construction of responses, drawing a subtle comparison to transfer attacks in adversarial examples. This phenomenon not only sheds light on the potential uniformity in AI assistants' linguistic strategies but also raises questions about LLM security, hinting that their predictable patterns and phrase repetition may present other transferable vulnerabilities.

7 Ethics, Disclosure and Guidance

The side-channel attack introduced in this paper represents a practical and perilous threat that could potentially compromise the privacy of millions of individuals. Recognizing the severity of this vulnerability, we have proactively reached out to all the vendors listed in Table 1, and are in dialogue with them to help secure their systems.

To counteract the side-channel vulnerability, several mitigation strategies can be implemented:

1. **Adding Random Padding:** Incorporating random padding to each message can obscure the actual length of tokens, thereby complicating attempts to infer information based on packet size. However, this approach would inevitably lead to increased bandwidth consumption, as the padded messages consume additional network resources.

2. **Grouping Tokens:** Another effective measure is to transmit tokens in larger groups rather than individually. This method reduces the granularity of information that can be gleaned from observing the communication, thereby

mitigating the risk. However, it's important to note that this could impact the real-time responsiveness that users expect from such services.

3. **Batching Responses:** Sending complete responses at once, instead of in a real-time, token-by-token manner, can significantly reduce the vulnerability to side-channel attacks. This approach eliminates the possibility of inferring token lengths from the packet sizes.

Despite the effectiveness of these mitigation strategies, they come with notable caveats. Specifically, the introduction of padding and the batching of responses would increase bandwidth usage, which could be a concern for services operating at scale. Moreover, grouping tokens or sending responses in batches could detract from the user experience. The real-time feedback provided by AI assistants is a key feature of their appeal and utility, especially given the inherent latency in processing large language models. Balancing security with usability and performance presents a complex challenge that requires careful consideration.

8 Related Works

Length-Based side-channels. The study of length-based side-channels reveals two primary methodologies: high-level inference, which deduces general information from a set of packets (e.g., website fingerprinting) and low-level inference, which explicitly infers plaintext from individual packets.

An example of a low-level inference attack was CRIME [28], which was revealed in 2012. The attack exploited information leakage in the LZ77 compression algorithm enabling attackers to infer plaintext in HTTPS traffic by interacting with the session. In 2015, the BICYCLE attack demonstrated the feasibility of inferring content without the need for interacting with the victim's session [11]. In this attack, a victim's password length is inferred by subtracting the size of the traffic's overhead from the traffic carrying the password.

The exploration of side-channels for plaintext inference on network traffic gained momentum in 2017 when it was shown that statistics, such as packet times and sizes, could reveal information about a query sent to a search engine on the web [22]. By employing classical ML models, the authors of [22] were able to detect the presence of specific keywords within a query but they could not decipher complete texts. However, this was solved in 2019 when the authors of [20] proposed KREEP. In their work, the authors noticed that the autocomplete feature of search websites was sending a single packet for every typed character. By measuring the victim's keystroke timings, the authors were able to infer entire search queries with an ED of about 0.4 for queries containing 7-12 words.

In contrast to the works of [20,22], our problem is different since (1) we cannot use keystroke timing to reduce character entropy, and (2) we are trying to predict sequences of tokens (words) to the length of entire paragraphs. This distinction underlines the novelty of our approach in inferring content solely based on the length of tokens, presenting a pioneering effort in the analysis of extended texts via side-channels. Moreover, regarding the sequence prediction method, in [20] the authors used a memory-less Markovian model which only considers the previous word. However, we leverage long-distance linguistic relationships and inter-sentence context by using a modified LLM.

Side-channel Attacks using LLMs. The utility of using deep neural networks to *perform* side-channel analysis (SCA) has been well documented [3,16,19,23,24]. The primary application of deep learning in SCA is to classify (detect) known patterns in noisy and complex physical signals. Some works have used generative AI, specifically generative adversarial networks (GANs), to help create better training sets for the model to learn from [30]. To the best of our understanding, our study introduces two novel contributions to the field: it is the first instance where generative AI has been explicitly applied to reconstruct hidden information, and uniquely, it marks the first occasion where plaintext has been successfully extracted from encrypted network traffic utilizing a generative AI approach.

The risk of attackers using LLMs for malicious use cases is a growing concern. Our study aligns with the observations made in [35] which highlights the potential for malicious use of LLMs in side-channel attacks. This work not only confirms these hypotheses but also presents a concrete example of such an application.

Side-channel Attacks on LLMs. Recent studies have highlighted the vulnerability of deep neural networks to side-channel attacks, revealing that attackers can extract a model's architecture and parameters by observing behaviors in cache [12,13,18,34], timing [8,10], memory access [15] and physical signals [14,32,36]. Recent research by Debenedetti et al. [6] further highlights the presence of side-channels capable of revealing information about a model's training data, including the extraction of LLM vocabularies and inferring details through membership inference attacks. Our work, however, diverges significantly from these approaches. We discuss a novel side-channel attack that targets model predictions, distinct from past works which targets model parameters or training data.

9 Conclusion

This study exposes a critical vulnerability in AI assistants like OpenAI's ChatGPT-4 and Microsoft's Copilot through a novel token-length side-channel attack, achieving significant success in reconstructing and inferring encrypted responses. By leveraging large language model capabilities, context integration, and known-plaintext attack techniques, we are able to perform a token-inference attack that can expose over half of the responses sent back from these assistants. The research underscores the importance of addressing this security flaw and highlights the broader security implications for large language models, pointing to a need for enhanced privacy measures in AI-powered digital services.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Samuel Addington. Chatgpt: Cyber security threats and countermeasures. *Available at SSRN 4425678*, 2023.

[3] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.

[4] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

[5] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. A survey of man in the middle attacks. *IEEE communications surveys & tutorials*, 18(3):2027–2051, 2016.

[6] Edoardo Debenedetti, Giorgio Severi, Nicholas Carlini, Christopher A Choquette-Choo, Matthew Jagielski, Milad Nasr, Eric Wallace, and Florian Tramèr. Privacy side channels in machine learning systems. *arXiv preprint arXiv:2309.05610*, 2023.

[7] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

[8] Vasisht Duddu, Debasis Samanta, D Vijay Rao, and Valentina E Balas. Stealing neural networks via timing side channels. *arXiv preprint arXiv:1812.11720*, 2018.

[9] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[10] Cheng Gongye, Yunsi Fei, and Thomas Wahl. Reverse-engineering deep neural networks using floating-point timing side-channels. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[11] Benjamin Harsha, Robert Morton, Jeremiah Blocki, John Springer, and Melissa Dark. Bicycle attacks considered harmful: Quantifying the damage of widespread password length leakage. *Computers & Security*, 100:102068, 2021.

[12] Sanghyun Hong, Michael Davinroy, Yiğitcan Kaya, Dana Dachman-Soled, and Tudor Dumitraş. How to 0wn nas in your spare time. *arXiv preprint arXiv:2002.06776*, 2020.

[13] Sanghyun Hong, Michael Davinroy, Yiğitcan Kaya, Stuart Nevans Locke, Ian Rackow, Kevin Kulda, Dana Dachman-Soled, and Tudor Dumitraş. Security analysis of deep neural networks operating in the presence of cache side-channel attacks. *arXiv preprint arXiv:1810.03487*, 2018.

[14] Xing Hu, Ling Liang, Shuangchen Li, Lei Deng, Pengfei Zuo, Yu Ji, Xinfeng Xie, Yufei Ding, Chang Liu, Timothy Sherwood, et al. Deepsniffer: A dnn model extraction framework based on learning architectural hints. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 385–399, 2020.

[15] Weizhe Hua, Zhiru Zhang, and G Edward Suh. Reverse engineering convolutional neural networks through side-channel information leaks. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[16] Sunghyun Jin, Suhri Kim, HeeSeok Kim, and Seokhie Hong. Recent advances in deep learning-based side-channel analysis. *ETRI Journal*, 42(2):292–304, 2020.

[17] Anastasios Lamproudis, Aron Henriksson, and Hercules Dalianis. Vocabulary modifications for domain-adaptive pretraining of clinical language models. In *HEALTHINF*, pages 180–188, 2022.

[18] Yuntao Liu and Ankur Srivastava. Ganred: Gan-based reverse engineering of dnns via cache side-channel. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 41–52, 2020.

[19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 348–375, 2020.

[20] John V Monaco. What are you searching for? a remote keylogging attack on search engine autocomplete. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 959–976, 2019.

[21] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

[22] Se Eun Oh, Shuai Li, and Nicholas Hopper. Fingerprinting keywords in search queries over tor. *Proc. Priv. Enhancing Technol.*, 2017(4):251–270, 2017.

[23] Max Panoff, Honggang Yu, Haoqi Shan, and Yier Jin. A review and comparison of ai-enhanced side channel analysis. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(3):1–20, 2022.

[24] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Computing Surveys*, 55(11):1–35, 2023.

[25] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[27] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.

[28] Juliano Rizzo and Thai Duong. Crime: Compression ratio info-leak made easy. In *ekoparty Security Conference*, 2012.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[30] Ping Wang, Ping Chen, Zhimin Luo, Gaofeng Dong, Mengce Zheng, Nenghai Yu, and Honggang Hu. Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks. *arXiv preprint arXiv:2007.05285*, 2020.

[31] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc., 2020.

[32] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. I know what you see: Power side-channel attack on convolutional neural network accelerators. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 393–406, 2018.

[33] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, et al. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 214–229, 2022.

[34] Mengjia Yan, Christopher W Fletcher, and Josep Torrellas. Cache telepathy: Leveraging shared resource attacks to learn {DNN} architectures. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2003–2020, 2020.

[35] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Eric Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *arXiv preprint arXiv:2312.02003*, 2023.

[36] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. Deepem: Deep neural networks model recovery through em side-channel information leakage. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 209–218. IEEE, 2020.

[37] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

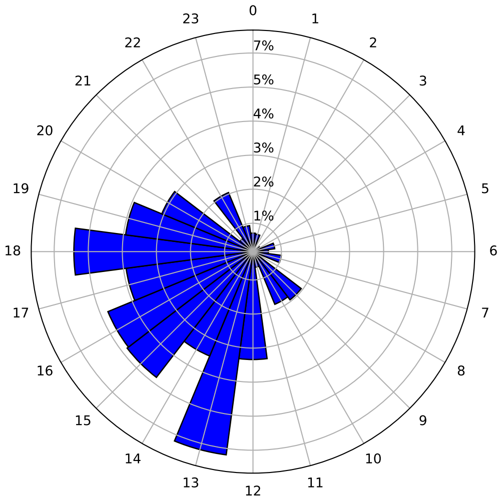
Appendix

A Details on Experiment Setup

Setup of Baseline Model (GPT-4) The following is an example of the prompt template used to ask GPT-4 to infer R from T:

> GPT-4 Inference Prompt
>
> Transform the given sequence of numbers into a coherent sentence where each number represents the length of a word in the sentence. For example, the sequence '7, 10, 10, 9, 3, 4, 5, 3, 2, 5, 1' could be transformed into 'Contact allergies typically manifest on the skin as a rash,'
>
> Please use the following sequence to generate a new sentence: **<encoding>**.
>
> Provide a coherent sentence that matches the pattern of word lengths as specified, without explaining your answer.

Setup of Framework Evaluations (Groupings) Here we present how many tokens are grouped by OpenAI's ChatGPT in-browser service over a 24 hour period. The statistics from this data where used to augment the datasets with groupings.



,

Figure 8: Percentage of grouped tokens in R by OpenAI across various times of the day.

B Additional Results

In Fig. 9 we provide another view of topic exposure among the first segments. Here, we asked ChatGPT-4 Turbo to classify the segments into 15 categories based on subject alone.

In Fig. 10 we show the relationship between the first segment's performance and its performance with its subsequent segments. As described in the paper, we can see that the performance of the paragraph is largely dependent on the performance of the first segment. However, interestingly, we see that for poor starts ($\phi < 0.4$), letting the model see more sentences improves paragraph performance. This indicates that the model can use long term patterns to help resolve local patterns in T.

In Fig. 11 we provide a random sample of full paragraphs inferred from ChatGPT-4.

In figures 12 and 13 we provide a random sample of attacks performed on traffic from OpenAI's ChatGPT and Microsoft's Copilot.

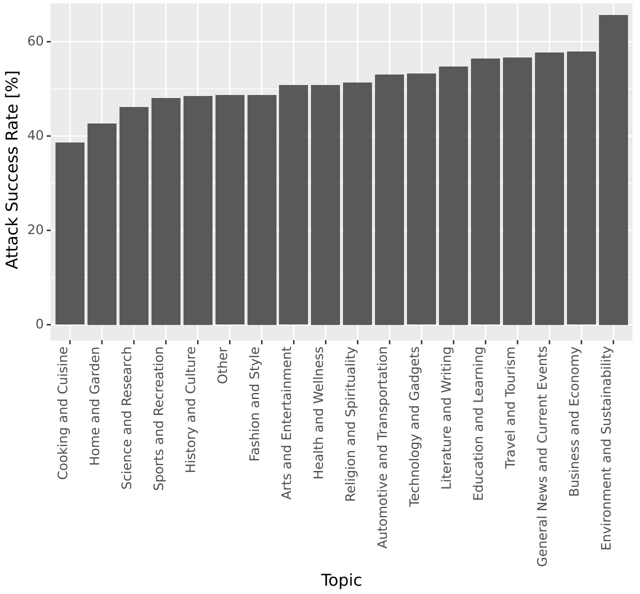


Figure 9: Attack success rates on first segments, grouped by subject material by GPT-4 Turbo.

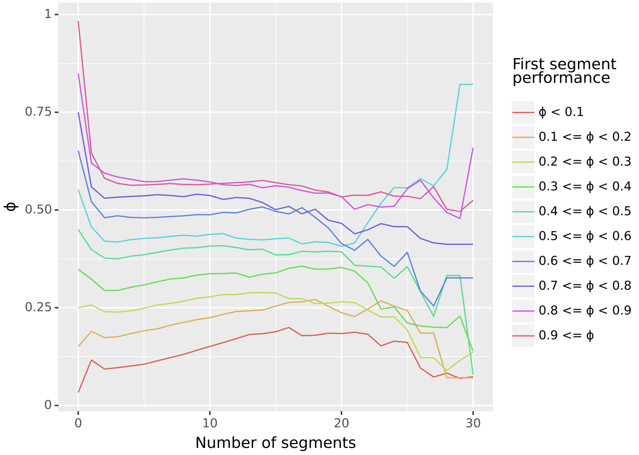


Figure 10: Performance of paragraph reconstruction by paragraph length, with each color indicating the performance when the first segment has a specific ϕ value. The initial segment's performance influences the accuracy of subsequent sentences in a paragraph.

Paragraph Reconstruction Examples

$\phi : 0.93$ ROUGE-1: 0.75 Edit Distance: 0.26

Original: As an AI language model, I don’t have personal experience or emotions, but below are some strategies that can help in developing a self-compassionate attitude:
1. Practice mindfulness - Mindfulness can help you identify and observe your thoughts and emotions without judging or reacting to them. This can help you develop a more compassionate attitude towards yourself.
2. Treat yourself with kindness - In stressful situations, treat yourself the way you would treat a friend or loved one. Do something kind for yourself, such as taking a break, self-care activities, or giving yourself a compliment.
3. Change your self-talk - Instead of being self-critical or negative, try to reframe your self-talk in a positive and supportive manner.
4. Accept your imperfections - Acknowledge that everyone makes mistakes and has flaws, including yourself. Learn to accept yourself for who you are, imperfections and all.
5. Be patient with yourself - Give yourself time to process and bounce back from stressful situations. Remember that healing and growth take time.
6. Seek support - Reach out to friends, family, or a mental health professional for support, guidance,
Prediction: As an AI language model, I don’t have personal experience or emotions, but below are some strategies that can help in developing a self-compassion attitude:
1. Practice mindfulness - Mindfulness can help you identify and process your emotions and thoughts without judging or reacting to them. This can help you develop a more compassionate attitude towards yourself.
4. Treat yourself with kindness - By embracing compassion,treat yourself the way you would treat a friend or loved one. By accepting your own mistakes, such as taking a break,resuming activities, or simply avoiding a temptation.
4. Change your self-talk - Instead of going into a battle or argument,try to reframe your self-talk in a positive and supportive manner.
3. Accept your imperfections - Acknowledge that everyone makes mistakes and has flaws. Challenge yourself, learn to accept yourself for who you are, treat oneself kindly.
5. Be patient with yourself - Your personal life is complex and unique with many different challenges. Practice self-control and manage your time.
6. Seek support - Reach out to friends,family, or a mental health professional for support, guidance,

$\phi : 0.80$ ROUGE-1: 0.82 Edit Distance: 0.16

Original: As an AI language model, I am unable to provide information about specific individuals or universities. However, in general, one way that universities can ensure fair access to leadership opportunities for underrepresented groups is by implementing affirmative action programs and diversity and inclusion initiatives. These programs can involve outreach and recruitment efforts aimed at increasing the representation of underrepresented groups in leadership roles, mentorship and support programs for individuals from underrepresented groups, and policies that require a diverse pool of candidates for leadership positions.
Prediction: As an AI language model, I am unable to provide information about specific individuals or institutions. However, in general, one way that institutions can ensure that access to employment opportunities for underrepresented groups is by implementing progressive hiring policies and promoting job placement initiatives. These policies can include training and development efforts aimed at increasing the representation of underrepresented groups in leadership roles,mentorship and support programs for individuals from underrepresented groups,and policies that promote a diverse pool of candidates for leadership positions.

$\phi : 0.90$ ROUGE-1: 0.64 Edit Distance: 0.40

Original: As an AI language model, I understand that when developing a brand strategy, it is crucial to consider the target audience and their preferences and behaviors. The success of any brand is highly dependent on its ability to connect and resonate with its target audience.
By understanding the target audience’s demographics, psychographics, and purchasing behaviors, a brand can tailor its messaging, product features, and marketing strategies to effectively reach and engage with its audience. This can lead to increased brand awareness, loyalty, and ultimately, sales.
Neglecting to consider the target audience can result in a disconnection between the brand and its intended audience, leading to a lack of interest and engagement.
Prediction: As an AI language model, I understand that when developing a brand identity, it is crucial to consider the target audience and their preferences and interests. The success of the brand is highly dependent on the ability to connect the audience with the target audience in a comprehensive and timely manner.
Ultimately, brand identity, and leadership qualities, a clear and empathetic messaging, quality products, and effective management to effectively serve and engage with its audience. This can lead to increased brand awareness, loyalty, and engagement, which in turn can be an incentive for repeat business and growth. If the marketing efforts are known for the positive outcomes, whether it’s word-of-mouth and engagement,

Figure 11: Paragraph Inference: A random sample of full paragraph inferences performed on ChatGPT-4 responses.

ChatGPT PCAP Examples

$\phi: 0.81$ ROUGE-1: 0.31 Edit Distance: 0.67

Original: Walmart was founded **by Sam Walton in 1962. Sam Walton opened the first Walmart store in Rogers, Arkansas, aiming to provide consumers with lower prices and great service. His strategy for discount retailing fundamentally transformed the retail industry, and Walmart has grown to become one of the world's largest and most influential retail chains.**

Prediction: Walmart was founded **in the summer of 1904 and quickly became the third-largest chain in retail. As such, Walmart is closely connected with other stores and local markets, as well, making it closely connected with other brands and their markets.**

$\phi: 0.58$ ROUGE-1: 0.81 Edit Distance: 0.21

Original: **Pigs** are omnivores, meaning they eat both plant-**based** and animal-based foods. In the wild, their diet is **diverse and includes:**

Prediction: **Crocodiles** are omnivores, meaning they eat both plant **roots** and animal-based foods. In the wild, their diet is **limited and balanced,**

Figure 12: Network Capture: A random sample of attacks performed on traffic intercepted from OpenAI's ChatGPT.

Microsoft Copilot PCAP Examples

$\phi: 0.63$ ROUGE-1: 0.76 Edit Distance: 0.21

Original: Here are some of the latest research findings on **effective teaching methods** for students with learning disabilities:

Prediction: Here are some of the latest research findings on **cognitive behavior therapy** for children with learning disabilities:

$\phi: 0.84$ ROUGE-1: 0.63 Edit Distance: 0.25

Original: Machine Learning (ML) and Artificial Intelligence (AI) are **making** significant **strides** in the healthcare industry. **Here are some of the latest developments:**

Prediction: Machine learning (ML) and artificial intelligence (AI) are **having** significant **impacts** on the healthcare industry. **With the rise of new mobile technologies,**

$\phi: 0.78$ ROUGE-1: 0.86 Edit Distance: 0.16

Original: The COVID-19 pandemic has had a significant impact on **the Caribbean's** tourism industry:

Prediction: The COVID-19 pandemic has had a significant impact on **San Francisco's** tourism industry.

$\phi: 0.83$ ROUGE-1: 0.50 Edit Distance: 0.54

Original: Public schools are **primarily** funded by local and state governments, **with a smaller percentage coming from the federal government¹. Here's a breakdown of the funding sources:**

Prediction: Public schools are **typically** funded by state and local governments, **and a federal government grants them for various activities - Extracurriculars in the primary schools:**

Figure 13: Network Capture: A random sample of attacks performed on traffic intercepted from Microsoft's Copilot.