"*Sorry, Come Again?*" Prompting – Enhancing Comprehension and Diminishing Hallucination with [PAUSE]-injected Optimal Paraphrasing

Vipula Rawte[1]*, Prachi Priya[2], S.M Towhidul Islam Tonmoy[3], S M Mehedi Zaman[3], Aman Chadha[4,5]†, Amit Sheth[1], Amitava Das[1]
[1]AI Institute, University of South Carolina, USA
[2]Indian Institute of Technology, Kharagpur
[3]Islamic University of Technology
[4]Stanford University, USA, [5]Amazon AI, USA
vrawte@mailbox.sc.edu

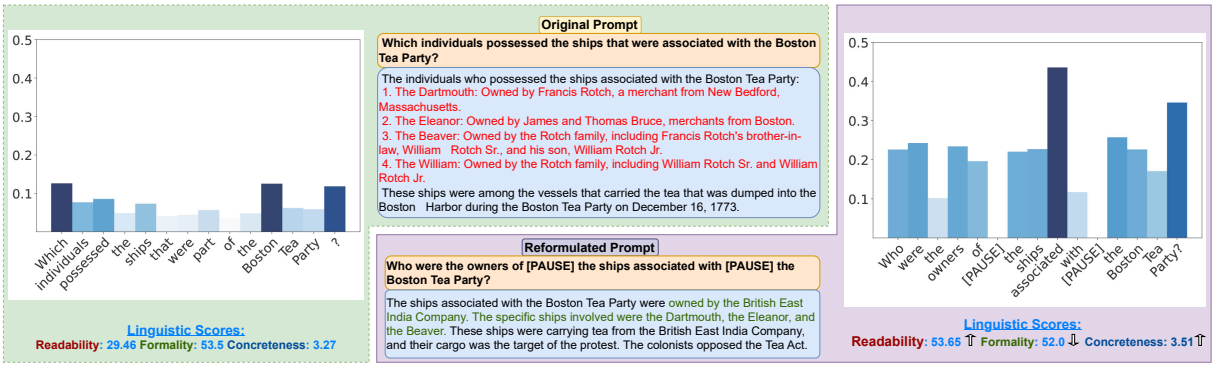


Figure 1: An example demonstrating how a "*rephrased prompt*" presented to a particular LLM can aid in avoiding hallucination. Here, the hallucinated text is highlighted in red. Post reformulation, the newly generated response incorporates the factually correct (dehallucinated) text, highlighted in green.

Abstract

Hallucination has emerged as the most vulnerable aspect of contemporary Large Language Models (LLMs). In this paper, we introduce the *Sorry, Come Again (SCA)* prompting, aimed to avoid LLM hallucinations by enhancing comprehension through: (i) optimal paraphrasing and (ii) injecting [PAUSE] tokens to delay LLM generation. First, we provide an in-depth analysis of linguistic nuances: *formality, readability,* and *concreteness* of prompts for 21 LLMs, and elucidate how these nuances contribute to hallucinated generation. Prompts with lower readability, formality, or concreteness pose comprehension challenges for LLMs, similar to those faced by humans. In such scenarios, an LLM tends to speculate and generate content based on its imagination (associative memory) to fill these information gaps. Although these speculations may occasionally align with factual information, their accuracy is not assured, often resulting in hallucination. Recent studies reveal that an LLM often tends to neglect the middle sections of extended prompts, a phenomenon termed as *lost in the middle*. We find that while a specific paraphrase may suit one LLM, the same paraphrased version may elicit a different response from another LLM. Therefore, we propose an optimal paraphrasing technique aimed at identifying the most comprehensible paraphrase of a given prompt, evaluated using Integrated Gradient (and its variations) to guarantee that all words are accurately processed by the LLM. Furthermore, during the reading of lengthy sentences, humans often pause at various points to better comprehend the meaning read thus far. These pauses are not only dictated by delimiters but also by semantic meaning. We have fine-tuned an LLM with injected [PAUSE] tokens, allowing the LLM to pause while reading lengthier prompts. The introduction of [PAUSE] injection has brought several key contributions: (i) determining the optimal position to inject [PAUSE], (ii) determining the number of [PAUSE] tokens to be

*Corresponding author.
†Work does not relate to position at Amazon.

inserted, and (iii) introducing reverse proxy tuning to fine-tune the LLM for [PAUSE] insertion. SCA's demo is publicly available[1].

Contributions

- Investigating the impact of three different linguistic features (formality, readability, and concreteness) of prompts on hallucination for 21 LLMs (cf. Sec. 3).
- Presenting SCA an optimal paraphrasing prompting framework aimed at identifying the most comprehensible paraphrase of the same prompt (cf. Sec. 1).
- [PAUSE] injection methods to delay LLM generation and aid comprehension (cf. Sec. 8) and a novel reverse proxy-tuning for it (cf. Sec. 8.3).
- Lastly, presenting ***ACTIVATOR***, an end-to-end framework crafted to avoid hallucination by enhancing LLMs' reading comprehension (cf. Sec. 10).

1 "Sorry, Come Again?" – LLM Does Not Comprehend It All in a Given Prompt

With the advent of LLMs, *Prompt Engineering* has emerged as a new technical profession (DePillis and Lohr, 2023; Smith, 2023; Delaney, 2023). While the fundamental concept revolves around framing questions or commands effectively to elicit the desired response, mastering this skill delves into several intricacies. These include (a) understanding the LLM's proficiencies (based on the tasks it was trained to accomplish), (b) trial and error-based experimentation, (c) balancing precision and flexibility, and (d) considering bias and ethical considerations, among many other nuances. Therefore, achieving an *optimal prompt* is a rather daunting task. (Sclar et al., 2023) has highlighted the high sensitivity of LLMs to subtle changes in prompt formatting, giving accuracy ranges from 4%-88% for a given task with LLaMA-2-70B and 47%-85% with GPT-3.5 (Liu et al., 2023b) has demonstrated that LLMs struggle to read and comprehend longer prompts. Instead, they tend to focus on words at the beginning and end, often neglecting those in between. They call this phenomenon '*lost in the middle*'. In Fig. 1, the prompt provided on the left-hand side is not effectively read by the LLM, resulting in a hallucinated generation. However, a paraphrased version of the same prompt, incorporating [PAUSE] tokens, is read and comprehended well by the same LLM, thereby eliminating hallucinations. Continuing along the same line, the identical prompt may be read and comprehended differently by different LLMs, as depicted in Fig. 2.

[1] https://huggingface.co/spaces/aisafe/SCA

Among the myriad potential paraphrases of a given prompt, a specific one may emerge as optimal for a particular LLM to effectively read and comprehend, as illustrated in **??**. Conversely, a different paraphrase version may be more suitable for other LLMs (see Fig. 3).

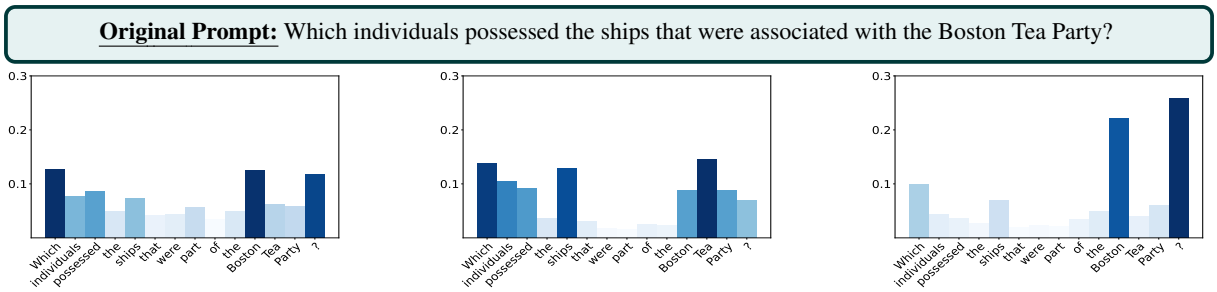
The premise of this work posits that improved comprehension can lead to reduced hallucination. "*Sorry, Come Again?*" (SCA henceforth) is a common expression in human communication, indicating difficulty in understanding the previous statement. In response, the speaker typically rephrases their utterance for better clarity. LLMs cannot seek clarification or ask follow-up questions for better understanding. This study presents SCA, an innovative approach in optimal prompt engineering aimed at finding the clearest prompt for a given LLM, leading to a decrease in hallucination occurrences.

2 Dissecting an LLM's Comprehension

Due to the blackbox nature of deep neural networks, deducing the internal process of how an LLM comprehends an input prompt presents a significant challenge. Integrated Gradients (Sundararajan et al., 2017) serve as the cornerstone among explainability methods, calculating the gradient of the model's prediction output with respect to its input features. Following the approach outlined by (Liu et al., 2023b), we investigate which input words are effectively comprehended by LLMs, which forms our working hypothesis of comprehension. However, this hypothesis could be subject to further scrutiny, and we engage in self-criticism. We have utilized the following SoTA explainability methods such as Discretized Integrated Gradients (DIG) (Sanyal and Ren, 2021), and Sequential Integrated Gradients (SIG) (Enguehard, 2023) in this study. Developing new methods for explainability is an evolving area of research, and we have yet to determine the best-performing method among IG, DIG, and SIG. Therefore, in our study, we utilize all of them and calculate an average score obtained from them at the word level.

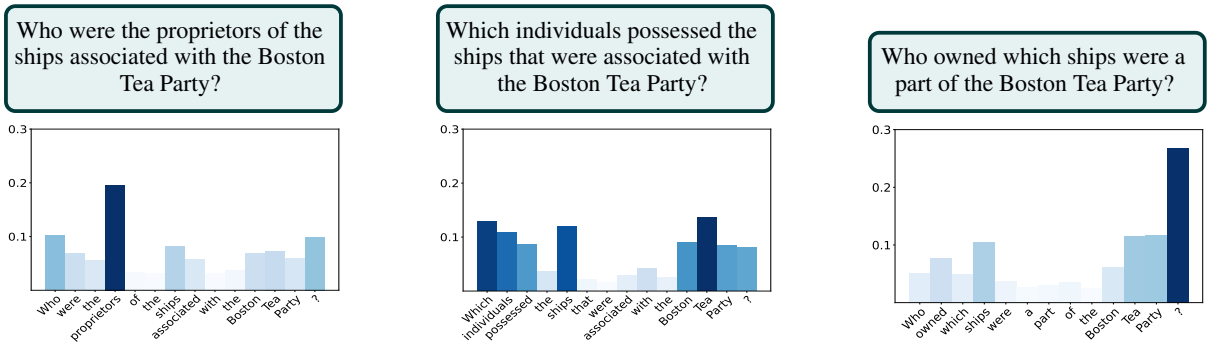
3 Linguistic Nuances of Prompts

Numerous practitioners advocate that proficient prompt engineering could serve as an effective method to mitigate hallucination (Kelly, 2023; Gheorghiu; Jr., 2023; MacManus, 2023; Greyling, 2023). However, such assertions require empirical testing conducted with scientific rigor. To the



(a) Falcon (b) BLOOM (c) Dolly

Figure 2: The same prompt is read by different LLMs differently.



(a) Optimal Prompt for Falcon (b) Optimal Prompt for BLOOM (c) Optimal Prompt for Dolly

Figure 3: Paraphrased versions of the aforementioned prompt with a focus on suitability for different LLMs.

best of our knowledge, there is scarce research (except one (Rawte et al., 2023b)) on the linguistic properties of prompts and their resultant impact on hallucination in generated content. In this study, we delve into an examination of three pivotal linguistic features: *readability (Flesch, 1948), formality (Heylighen and Dewaele, 1999), and concreteness (Paivio, 2013)* of a prompt, and their consequential effects on hallucination.

Readability (R) assesses the ease with which a text can be read and comprehended, taking into account factors such as complexity, familiarity, legibility, and typography. The widely recognized measure of readability is the Flesch Reading Ease Score (FRES) (Flesch, 1948), which provides a numerical representation of a text's readability. It is computed based on sentence length and word complexity using the formula: FRES $= 206.835 - 1.015 \cdot (\text{total words}/\text{total sentences}) - 84.6 \cdot (\text{total syllables}/\text{total words})$. For instance, a simple sentence yields a high score, while a complex one results in a lower score, reflecting the ease or difficulty of comprehension, as shown below.

Easily readable **FRES score = 75.5**
Sentence: The sun rises in the east every morning.

Challenging readability **FRES score = 11.45**
Sentence: The intricacies of quantum mechanics, as expounded upon by renowned physicists, continue to baffle even the most astute scholars.

Formality (F) in language is characterized by detachment, accuracy, rigidity, and heaviness; an informal style is more flexible, direct, implicit, and involved, but less informative.

Informal sentence **Formality score = 54.5**
The big thing in the corner dates from the 18th century.

Formal sentence **Formality score = 62**
In the right corner, next to the entrance, stands a 2 meter high wooden cupboard with gold inlays, that dates from the 18th century.

The widely accepted method for measuring formality, proposed by (Heylighen and Dewaele, 1999), is calculated as follows: Formality $= (\text{freq}_{noun} + \text{freq}_{adjective} + \text{freq}_{preposition} + \text{freq}_{article} - \text{freq}_{pronoun} - \text{freq}_{verb} - \text{freq}_{adverb} - \text{freq}_{interjection} + 100)/2$, where $\text{freq}_{part\ of\ speech}$ represents the frequency of the respective part of speech.

Concreteness (C) measures how well a word represents a tangible concept, with concrete words being easier to process than abstract ones (Paivio, 2013). The degree of concreteness is rated on a 5-point scale (1-5) from abstract to concrete. Concrete words relate to tangible, sensory experiences, while abstract words involve concepts not di-

rectly sensed. Concreteness ratings for over 39,000 English words are available in (Brysbaert et al., 2014). In this work, to compute the concreteness of a sentence with n words, an average of concreteness ratings is calculated using the formula: $\sum_{i=1}^{n}$ concreteness rating$_i/n$.

Examples of *concrete* words
Apple 5 , Dog 4 , Chair 4 , Book 5 , Water 5 , Car 5

Examples of *abstract* words
Justice 1 , Love 1 , Happiness 1 , Courage 1 , Wisdom 1

We analyze the impact of linguistic characteristics on LLM hallucination by establishing specific score ranges (see Table 1) and provide a detailed examination in Figs. 4, 10 and 11.

Range → Linguistic Aspect ↓	Low	Mid	High	Std. dev.
Readability	0-13.68	13.69-52.42	52.42-100	19.37
Formality	0-45.65	45.66-70	70.051-100	12.1
Concreteness	1-3.03	3.03-3.47	3.47-5	0.22

Table 1: Range(s) for prompt's three linguistic aspects.

4 Types of Hallucination

The phenomenon of generating factually incorrect or imaginary responses by LLMs is commonly called *hallucination* (Augenstein et al., 2023; Xu et al., 2024; Wang et al., 2024). Recent studies (Ladhak et al., 2023; Varshney et al., 2023) have categorized various types of hallucinations. (Rawte et al., 2023a) defined two fundamental types of hallucination: when an LLM hallucinates despite being given a factually correct prompt, it is termed as a *factual mirage*, whereas when an LLM hallucinates given a factually incorrect prompt, it is termed as a *silver lining*. This study confines its investigation solely to the phenomenon of factual mirage hallucination. In this study, we adopt a simplified approach by utilizing the *four* distinct categories of hallucination proposed Additionally, we furnish descriptions and examples for each category, emphasizing the hallucinated text in red.

1. Person (P): This occurs when an LLM invents a fictional personality without any tangible proof.

Original: The three people who were killed in the shooting at Michigan State University were all students, the police said on Tuesday morning.

AI-generated: The three students who died were identified as 17 y.o. Diva Davis, 20 y.o. Thomas McDevitt and 19 y.o. Jordan Eubanks.

Fact: Three students — Alexandria Verner of Clawson; Brian Fraser of Grosse Pointe; and Arielle Anderson of Grosse Pointe - lost their lives.

2. Location (L): This issue arises when LLMs produce an inaccurate location linked to an event.

Original: A wooden boat carrying 130 migrants broke apart against rocks near a beach town in southern Italy.

AI-generated: ...it ran aground at dawn on Sunday near the beach town of Punta Imperatore, in the province of Salerno, in Campania.

Fact: Many of the bodies were reported to have washed up on a tourist beach near Steccato di Cutro...

3. Number (N): This happens when an LLM produces imaginary numbers (such as age, etc.).

Original: In 1944, when the Nazis killed 643 people in a French village, Robert Hebras was one of a handful who lived to tell the story.

AI-generated: Robert Hebras was one of seven men who managed to escape the massacre.

Fact: Only six wounded survived, hidden under corpses.

4. Time (T): This issue involves LLMs generating text about events from various timelines.

Original: After a Chinese spy balloon was shot down this month, the U.S. has brought down at least three UFOs...

AI-generated: April 3, 2020: U.S. military shot down a Chinese spy balloon.

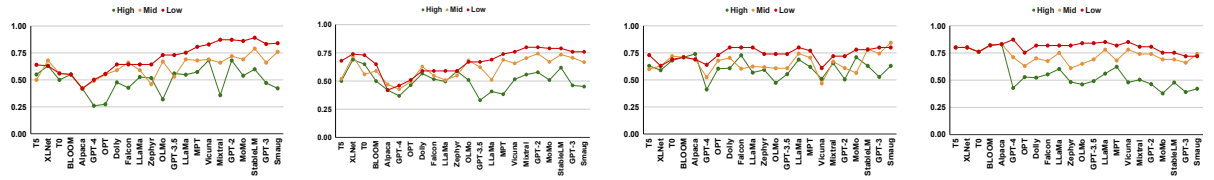
Fact: Feb. 4 2023: A U.S. fighter plane shoots down the balloon.

5 Selection of LLMs

We have selected 21 contemporary LLMs that have consistently demonstrated outstanding performance across a wide spectrum of NLP tasks, per the Open LLM Leaderboard (Beeching et al., 2023). These models include: (i) GPT-4 (OpenAI, 2023), (ii) GPT-3.5 (OpenAI, 2022), (iii) LLama2 (Touvron et al., 2023), (iv) GPT-2 (Radford et al., 2019), (v) MPT (Wang et al., 2023), (vi) OPT (Zhang et al., 2022), (vii) LLaMA (Meta, 2023), (viii) BLOOM (Scao et al., 2022), (ix) Alpaca (Taori et al., 2023), (x) Vicuna (Chiang et al., 2023), (xi) Dolly (databricks, 2023), (xii) StableLM (Liu et al., 2023a), (xiii) XLNet (Yang et al., 2019), (xiv) T5 (Raffel et al., 2020), (xv) T0 (Deleu et al., 2022), (xvi) Falcon (Almazrouei et al., 2023), (xvii) Zephyr (Tunstall et al., 2023), (xviii) Mixtral (Jiang et al., 2024), (xix) OLMo (Groeneveld et al., 2024), (xx) MoMo (Chada et al., 2023), (xxi) Smaug (AI).

6 Dataset

To construct the $\mathcal{SCA}{-}90\mathcal{K}$ dataset, we utilized NYTimes tweets (NYT) primary sources of data as prompts. We selected 21 LLMs, based on the criteria delineated in Sec. 5, and used them to generate a total of 52,500 text passages, with each LLM producing 2,500 text prose entries. We follow a similar approach to (Rawte et al., 2023a) for annotating our data. More details are in Appendix C. Table 2 provides detailed dataset statistics.



(a) Person (b) Location (c) Number (d) Time

Research Questions on Concreteness

① How does the level of concreteness in a prompt impact the probability of hallucination in LLMs?
② How does concreteness affect different kinds of hallucination? and which LLM is more sensitive to concreteness vs. hallucination types?
② Are LLMs more prone to hallucination when given abstract or vague prompts compared to concrete and specific prompts?

Effects on LLM's hallucination

① Based on empirical observations - prompts with concreteness scores falling in the range of 2.2 to 3.3 are most effective in preventing hallucinations. Prompts with concreteness scores exceeding 3.3 are not processed well by LLMs.
② The level of concreteness in a prompt has a similar impact as formality. This implies that elevating the concreteness score of a prompt can help prevent hallucinations related to persons and locations.

Figure 4: Percentage of hallucination for four different categories of hallucination for three levels of concreteness.

Hallucination Category	# Sentences
Person	9,570
Location	32,190
Number	11,745
Time	36,105
Total	89,610

Table 2: Statistics of $\mathcal{SCA}-90\mathcal{K}$.

7 Can Paraphrasing Help in Better Comprehension?

As discussed, it is apparent that enhanced prompt comprehension correlates with reduced hallucination. Therefore, it is necessary to determine the optimal comprehensible prompt. This premise has led to our experiments with paraphrasing, in which we generate up to 5 paraphrases for a given prompt.

7.1 Automatic Paraphrasing

When choosing automatic paraphrasing, there are many other factors to consider for e.g., a model may only be able to generate a limited number of paraphrase variations compared to others, but others can be more correct and/or consistent. As such, we consider three major dimensions in our evaluation: *(i)* ***coverage****: a number of considerable generations, (ii)* ***correctness****: correctness in those generations, and (iii)* ***diversity****: linguistic diversity in those generations.*

Model	Coverage	Correctness	Diversity
Pegasus	32.46	94.38%	3.76
T5	30.26	83.84%	3.17
GPT-3	35.51	88.16%	7.72

Table 3: Experimental results of automatic paraphrasing models based on three factors: *(i) coverage, (ii) correctness, and (iii) diversity*. GPT-3 (`text-davinci-003`) is the most performant considering all three aspects.

We conducted experiments with three models: (a) Pegasus (Zhang et al., 2020), (b) T5-Large (Raffel et al., 2020), and (c) GPT-3 (`text-davinci-003`) (Brown et al., 2020). Based on empirical observations, we concluded that GPT-3 outperformed all the other models. To offer transparency around our experimental process, we detail coverage, correctness, and diversity, along with the experimental paraphrasing setup, in D.

7.2 Choosing a Prompt's Optimal Paraphrase

Suppose the top-performing paraphraser generates the following *five* rephrasings for the prompt *"Which individuals possessed the ships that were part of the Boston Tea Party?"*. The objective is to acquire the most comprehensible paraphrase tailored to a specific LLM.

LLM comprehension is determined by two factors: (i) whether all the words in a given prompt are well-read, indicated by having an IG score above a threshold and (ii) whether all the topic words are well-read by the LLM. The overall approach is illustrated in Algorithm 1. This process employs a two-step method, as described below. Further details are available in Appendix E.

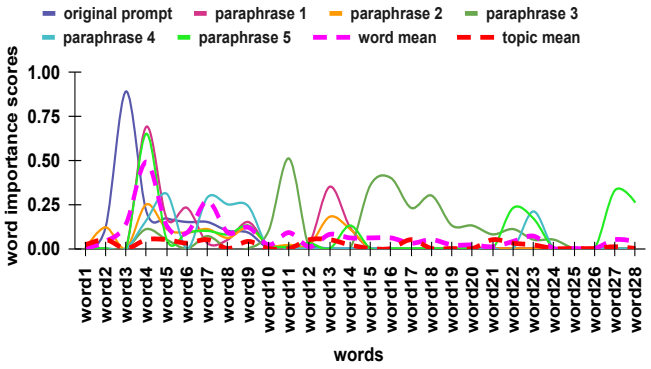
Distance We compute integrated gradients for paraphrased prompts, calculate their mean and measure the distance of each paraphrased prompt from the mean using cosine similarity.

Topic Modeling To address potential oversights in hidden word patterns, we include topic modeling using LDA (Blei et al., 2003). This involves

Original Prompt: Which individuals possessed the ships that were part of the Boston Tea Party?

Para §1: Who were the owners of the ships associated with the Boston Tea Party?
Para §2: Which individuals possessed the ships that were associated with the Boston Tea Party?
Para §3: Who owned which ships were a part of Boston Tea Party?
Para §4: What was the identity of those who owned the ships that were associated with the Boston Tea Party?
Para §5: Can you identify the shipkeepers of Boston Tea Party?

(a) Five paraphrases generated for the original prompt using the T5 paraphrasing model.



(b) Word importance scores distribution for the original prompt and its five paraphrases. The purple dashed line shows the mean of the IGs while the red dashed line shows the topic mean.

Figure 5: (a) Paraphrased versions for a given prompt; (b) Per-word importance score distribution for each paraphrase.

Algorithm 1 Finding the optimal paraphrased prompt

1: Find out the topics for the original prompt
2: **for** i in 1...5 **do**
3: a: Compute the IG, DIG, and SIG and b: an **average gradient** = $\frac{IG+DIG+SIG}{3}$ for $paraphrased\_prompt_i$
4: Compute the mean of all the gradients across various tokens
5: Find out the topics for $paraphrased\_prompt_i$
6: Calculate the distance of the mean prompt from the $paraphrased\_prompt_i$
7: Calculate the topic similarity between the original prompt and the $paraphrased\_prompt_i$
8: **end for**
9: Calculate a weighted average **Comprehension Score** = $(w_1 \times \text{distance} + w_2 \times \text{topic similarity})$ where, w_1 and w_2 are equal weights.
10: Select the $paraphrased\_prompt_i$ with the highest weighted average as the **optimal** $paraphrased\_prompt$

identifying topics for both the original prompt and paraphrases. Topic similarity scores are then employed to determine the most similar topics between a paraphrase and the original prompt. The final selection is determined by calculating distance and topic similarity for these two steps and then computing a weighted average. *Having spent much of my career studying various combination methods, it has been somewhat frustrating to consistently find that the simple average performs so well empirically.* (Clemen, 2008). The optimal paraphrase is chosen based on the highest average score. It is crucial to highlight that the original prompt itself may be the optimal prompt.

8 LLMs Need to Breathe While Reading!

The '*lost in the middle*' phenomenon, as introduced by (Liu et al., 2023b), illustrates that a substantial amount of information contained in the middle section of lengthy input prompts is overlooked during the comprehension process by LLMs. Recently, the introduction of [PAUSE] tokens demonstrated improvements in reasoning tasks (Goyal et al., 2023). Based on these findings and the 'lost in the middle' phenomenon, we propose that inserting [PAUSE] tokens may enhance LLM comprehension of longer prompts, potentially minimizing hallucination. Empirical results support this hypothesis.

Our Contributions related to [PAUSE] tokens

- **Where to inject [PAUSE] token(s)?** We propose clause boundary aka injecting [PAUSE] after conjunction.
- **How many [PAUSE] token(s)?** We propose a content-based method for [PAUSE] injection.
- **Best fine-tuning method(s)?** We introduce a novel fine-tuning paradigm named reverse proxy tuning.

8.1 Where to Inject [PAUSE] Tokens?

In their work, (Goyal et al., 2023) suggested an overall insertion of 10% [PAUSE] tokens; however, they did not provide specific guidelines or methods for determining the optimal positions for inserting [PAUSE]. We posit that the most effective location for injecting the [PAUSE] token should be at clause boundaries. However, identifying these boundaries comes with its own set of challenges. As a simple approach, we have opted to insert the [PAUSE] token after conjunctions, illustrated in Fig. 6.

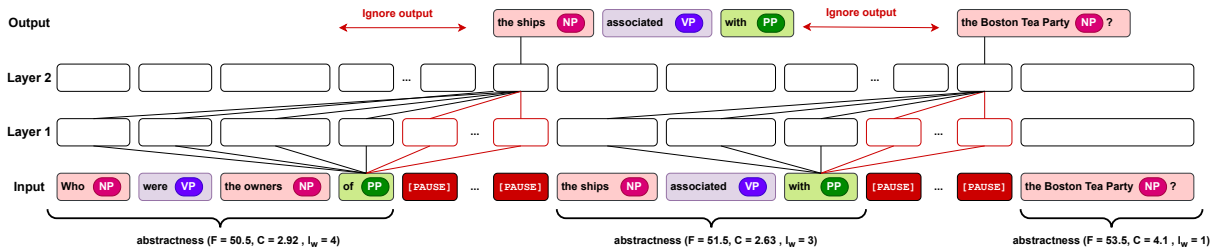


Figure 6: We use conjunct PP to split the long prompt. We use standard POS tagging (Akbik et al., 2018). Two [PAUSE] tokens are appended after PP based on the concreteness score of the chunk before the [PAUSE] tokens. Hence, it ignores, meaning it *breathes* for the next two tokens, as shown by Ignore output.

8.2 How Many [PAUSE] Tokens?

The study by (Goyal et al., 2023) did not definitively assert the ideal quantity of [PAUSE] tokens. Their experimentation ranged from 2 to 50 tokens, with a general conclusion that around 10 tokens were optimal, though this determination varied depending on the specific task. In contrast, we propose a content-based approach.

Our assessment of their impact on LLM comprehension revealed that readability provides a weaker signal compared to formality and concreteness. We define a combined measure called *abstractness*: $abs = \frac{\delta_1 * F + \delta_1 * C}{l_w}$, where δ_1, and δ_2 are coefficients. F is the formality measure, C is the concreteness measure, and l_w is the length of text in terms of the words. Additionally, we divided abstractness into three ranges—high, mid, and low—based on the overall distribution, mean, and standard deviations. Our method involves utilizing the abstractness score of the text preceding a [PAUSE] token to determine the appropriate number of tokens required. Higher abstractness scores suggest a lower (2) necessity to pause, whereas lower scores indicate a greater need for the language model to pause for comprehension, necessitating more (10) tokens. For the mid range abstractness we decide to insert five [PAUSE] tokens. The mechanism for inserting [PAUSE] has been illustrated in Fig. 6. Please refer to Appendix H for more details.

8.3 Reverse Proxy-Tuning

(Goyal et al., 2023) did not extensively explore a range of state-of-the-art (SoTA) fine-tuning techniques, such as LoRA, QALoRA, or ReLoRA, particularly regarding the injection of [PAUSE] tokens. These techniques fall into three broad categories: **1. Prompt Modifications:** Examples include Soft Prompt Tuning, Soft Prompt vs. Prompting, Prefix Tuning, and Hard Prompt Tuning. **2. Adapter Methods:** Such as LLaMA-Adapters.

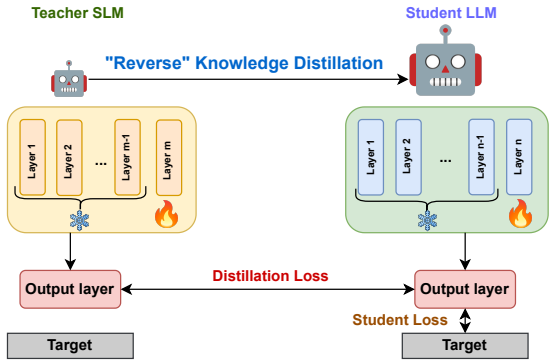
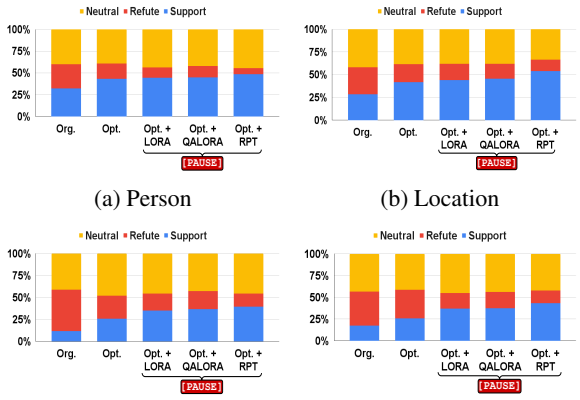


Figure 7: **Reverse Proxy-Tuning:** SLM is used to fine-tune LLM. First, SLM is fine-tuned on SQuAD where all the hidden layers except the last one are frozen. This fine-tuned SLM is further used to distill knowledge to the LLM, where all hidden layers of the LLM except the last one are frozen.



(a) Person (b) Location

(c) Numeric (d) Time

Figure 8: Empirical results for reverse proxy tuning using optimal prompt and [PAUSE] token for *four* different hallucination categories. . **Org.**: Original Prompt and **Opt.**: Optimal Paraphrase + LDA topics. These results indicate an overall average for all the 21 LLMs.

3. Reparameterization: Including Low Rank Adaptation (LoRA) (Hu et al., 2021), Quantized Low-Rank Adaptation (QLoRA) (Dettmers et al., 2023), Quantization-Aware Low-Rank Adaptation

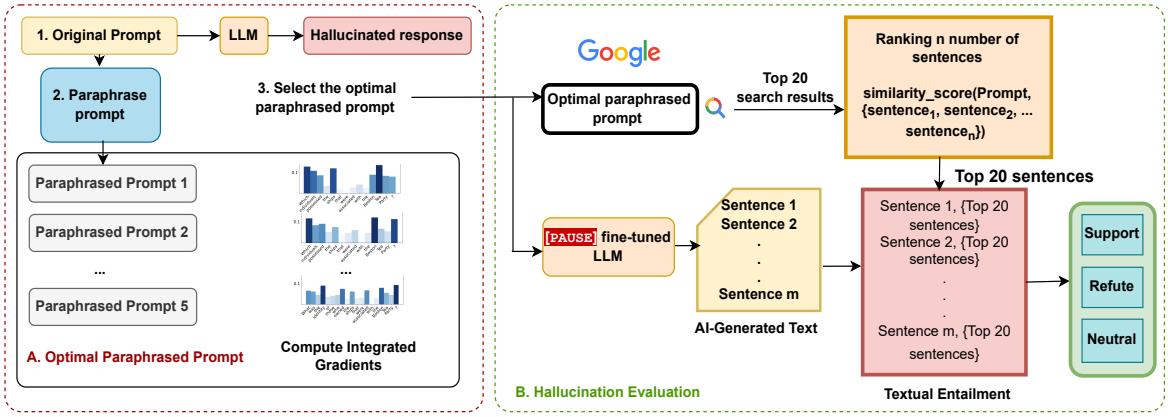


Figure 9: ***ACTIVATOR*** is a two-part end-to-end pipeline: **1. Optimal Paraphrased Prompt selection:** Using the Algorithm 1, an optimal prompt is selected by computing the average IG. **2. Hallucination Evaluation:** With the chosen optimal prompt, textual entailment is done to verify whether the AI-generated response is correct.

(QALoRA) (Xu et al., 2023), and Refined Low-Rank Adaptation (ReLoRA) (Lialin et al., 2023).

Although the above-mentioned fine-tuning methods are much more efficient for fine-tuning LLMs, they are still computationally expensive for our purpose – single modification to the prompt – adding [PAUSE] token(s). So in this work, we use the small language model (SLM) to fine-tune the larger language model. We adopt this idea from Knowledge Distillation (KD) (Hinton et al., 2015; Gu et al., 2023; Hsieh et al., 2023). The core concept in KD is distilling the knowledge from a larger model (Teacher) to a smaller model (Student). In this process, the Student not only learns from the expected labels but also from the Teacher. During this distillation, all the layers are updated using a loss function. However, changing weights for all layers is also computationally expensive. Therefore, in our case, we only choose the last output layer for fine-tuning and freeze all the layers. Additionally, we use an SLM to fine-tune the LLM which is reverse KD. We were further inspired by (Liu et al., 2024), which presents *Proxy Tuning*. Here, we introduce a novel approach called **Reverse Proxy-Tuning** (RPT), depicted in Fig. 7, where the SLM serves as a proxy model. This method is computationally efficient as it involves updating only the last layer and utilizing an SLM to fine-tune an LLM. Experimental results are illustrated in Fig. 8.

8.4 Dataset and Experimental Setup for [PAUSE] finetuning

For all our fine-tuning experiments, we use the CommonsenseQA dataset (Talmor et al., 2019). We have implemented two baselines: QLoRA (Dettmers et al., 2023) and QALoRA (Xu et al., 2023). The proposed novel reverse proxy-tuning yielded better performance than these two baselines. Further details regarding the experimental setup, including hyperparameters and other specifics, can be found in the Table 4 in the Appendix.

9 Does Better Comprehension Guarantee Lesser Hallucination?

This question is likely to captivate the reader's attention significantly. Enhancing comprehension and mitigating hallucinations in LLMs may initially appear as two distinct considerations. The subsequent query that naturally arises is how we discern hallucinations after furnishing an optimal prompt to the LLM. We have chosen the entailment approach to empirically evaluate whether overall support scores improve following the implementation of SCA. Support scores signify factual entailment.

While there's no assurance that the most comprehensible prompt will completely eliminate hallucinations, the results depicted in Fig. 8 provide empirical evidence of improvement in overall entailment support scores across all the hallucination classes. Additional details on entailment-based fact verification are provided in Appendix G.

Takeaways related to Reverse Proxy Tuning

- Optimal paraphrase + LDA yields better results for both Number and Time categories.
- We see marginal betterment for the Person and Location categories with Lora and QALora and a significant boost for the Number and Time categories.
- Among all other fine-tuning techniques, reverse proxy-tuning performs the best across all four categories.

10 *ACTIVATOR* - A Reprompter

We propose the ***ACTIVATOR*** pipeline to automatically rephrase and evaluate the prompt as shown in Fig. 9. Activator is an end-to-end pipeline that accepts a prompt as input and outputs an entailment score. This process involves pre-processing the input prompt to add [PAUSE] tokens, paraphrasing the input prompts to identify the most optimal prompt which maximizes comprehension by minimizing distance to the mean prompt and maximizing topic similarity based on the original prompt based on a mean of the integrated gradients score. This optimal prompt undergoes sentence-level entailment based on a web lookup to yield final entailment scores.

11 Conclusion

In this preliminary research study, we begin by categorizing the primary types of hallucinations present in LLMs. Subsequently, we compile our dataset by utilizing New York Times news tweets, aligning with these established categories. Language intricacies assume a crucial role in the comprehension of language. Therefore, we delve into the examination of three significant linguistic dimensions: readability, formality, and concreteness, and their potential influence on the occurrence of hallucinations in LLMs.

12 Discussion and Limitations

Discussion: On June 14th, 2023, the European Parliament successfully passed its version of the EU AI Act (European-Parliament, 2023). Following this, many other countries began discussing their stance on the evolving realm of Generative AI. A primary agenda of policymaking is to protect citizens from political, digital, and physical security risks posed by Generative AI. While safeguarding against misuse is crucial, one of the biggest concerns among policymakers is the occurrence of unwanted errors by systems, such as hallucination (source: https://cetas.turing.ac.uk/publications/rapid-rise-generative-ai).

Limitations: In this paper, we present several key findings: (i) LLM comprehension, (ii) paraphrasing can improve LLM comprehension, (iii) optimal paraphrasing, (iv) [PAUSE] injection, and (v) finally empirically show that the overall hallucination is reducing due to better LLM comprehension. We believe the following aspects require critical attention in future endeavours.

Limitation 1: The three linguistic properties are NOT independent. Certainly, these factors are not mutually exclusive. Our assessment of their impact on LLM comprehension revealed that readability provides a weaker signal compared to formality and concreteness. As a result, we have chosen to prioritize concreteness as the actionable feature.

Limitation 2: Which explainability method is the best? Integrated Gradient (IG) has long served as a fundamental principle governing explainability methods in deep neural networks. Despite recent advancements such as DIG and SIG, which have shown improved performance in various contexts, we were uncertain about their effectiveness for our specific use case of hallucination detection. Therefore, we opted for a more cautious approach and decided to average the results obtained from all three methods. A suitable explainability method for hallucination could be a nice future direction to explore.

Limitation 4: Is fine-tuning the ONLY method? One could argue that instead of fine-tuning, we could have explored techniques like In-Context Learning (ICL), Zero-Shot, and Few-Shot learning for [PAUSE] insertion. Some team members believe that ICL might yield competitive results compared to fine-tuning. However, due to time constraints, we were unable to conduct these

experiments. Nevertheless, we acknowledge that exploring these techniques could be a valuable direction for future research.

13 Ethical Considerations

Through our experiments, we have uncovered the susceptibility of LLMs to hallucination. While emphasizing the vulnerabilities of LLMs, our goal is to underscore their current limitations. However, it's crucial to address the potential misuse of our findings by malicious entities who might exploit AI-generated text for nefarious purposes, such as designing new adversarial attacks or creating fake news that is indistinguishable from human-written content. We strongly discourage such misuse and strongly advise against it.

References

Abacus AI. Smaug.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, et al. 2023. Factuality challenges in the era of large language models. *arXiv preprint arXiv:2310.05189*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46:904–911.

Rakesh Chada, Zhaoheng Zheng, and Pradeep Natarajan. 2023. Momo: A shared encoder model for text, image and multi-modal representations. *arXiv preprint arXiv:2304.05523*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Robert T Clemen. 2008. Comment on cooke's classical method. *Reliability Engineering & System Safety*, 93(5):760–765.

databricks. 2023. Dolly.

Kevin J. Delaney. 2023. Bringing a.i. tools to the workplace requires a delicate balance.

Tristan Deleu, David Kanaa, Leo Feng, Giancarlo Kerg, Yoshua Bengio, Guillaume Lajoie, and Pierre-Luc Bacon. 2022. Continuous-time meta-learning with forward mode differentiation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Lydia DePillis and Steve Lohr. 2023. Tinkering with chatgpt, workers wonder: Will this take my job?

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Joseph Enguehard. 2023. Sequential integrated gradients: a simple but effective method for explaining language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7555–7565, Toronto, Canada. Association for Computational Linguistics.

European-Parliament. 2023. Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts.

R Flesch. 1948. A new readability yardstick journal of applied psychology 32: 221–233.

Andrei Gheorghiu. 4 ways to treat a hallucinating ai with prompt engineering.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2023. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*.

Cobus Greyling. 2023. Preventing llm hallucination with contextual prompt engineering — an example from openai.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. *Interner Bericht, Center "Leo Apostel", Vrije Universiteit Brüssel*, 4(1).

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Tom Huddleston Jr. 2023. This is the no. 1 'most important' ai skill you need to know, says mit expert: 'you can learn the basics in 2 hours'.

Patrick Kelly. 2023. 10 best practices to reduce ai hallucinations with prompt engineering.

Faisal Ladhak, Esin Durmus, Mirac Suzgun, Tianyi Zhang, Dan Jurafsky, Kathleen Mckeown, and Tatsunori B Hashimoto. 2023. When do pretraining biases propagate to downstream tasks? a case study in text summarization. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3198–3211.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. 2023. Stack more layers differently: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023a. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023b. Lost in the middle:

How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Richard MacManus. 2023. Stopping ai hallucinations for enterprise is key for vectara.

AI Meta. 2023. Introducing llama: A foundational, 65-billion-parameter large language model. *Meta AI. https://ai. facebook. com/blog/large-language-model-llama-meta-ai*.

NYT. The new york times.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report.

Allan Paivio. 2013. Dual coding theory, word abstractness, and emotion: a critical review of kousta et al.(2011).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Vipula Rawte, Swagata Chakraborty, Agnibh Pathak, Anubhav Sarkar, S.M Towhidul Islam Tonmoy, Aman Chadha, Amit Sheth, and Amitava Das. 2023a. The troubling emergence of hallucination in large language models - an extensive definition, quantification, and prescriptive remediations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2541–2573, Singapore. Association for Computational Linguistics.

Vipula Rawte, Prachi Priya, SM Tonmoy, SM Zaman, Amit Sheth, and Amitava Das. 2023b. Exploring the relationship between llm hallucinations and prompt linguistic nuances: Readability, formality, and concreteness. *arXiv preprint arXiv:2309.11064*.

Soumya Sanyal and Xiang Ren. 2021. Discretized integrated gradients for explaining language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10285–10299, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *ArXiv*, abs/2310.11324.

Craig S. Smith. 2023. Mom, dad, i want to be a prompt engineer.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy

Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Georgi Georgiev, Rocktim Jyoti Das, and Preslav Nakov. 2024. Factuality of large language models in the year 2024.

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. 2023. Multitask prompt tuning enables parameter-efficient transfer learning. In *The Eleventh International Conference on Learning Representations*.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhensu Chen, Xiaopeng Zhang, and Qi Tian. 2023. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Frequently Asked Questions (FAQs)

✻ Why do you select those 21 large language models?

➠ We want to select several language models with varying parameter sizes for our experiments - ranging from large to small. Hence, the above chosen models consist of both large models like GPT-3, LLaMa and smaller ones like T5 and T0.

✻ Why only three linguistic properties are selected for this study?

➠ As far as we know, formality, readability, and concreteness appear to be the most obvious criteria for assessing LLM comprehension.

✻ What is the purpose of calculating integrated gradients? Why not simply use attention scores?

➠ Integrated Gradient provides an explanatory score at the word level, indicating how the LLM interprets each word and generates output. In contrast, attention scores only reveal the encoding side of processing.

✻ Why do you only generate five paraphrases?

➠ We conducted a study to assess the limit of how many ways a single sentence could be paraphrased. Our findings suggest that there is indeed a limit, as generating too many paraphrases can disrupt diversity. Through experimentation, we have observed that five paraphrases is the optimal number.

✻ What are the broad implications of the *ACTIVATOR* framework for hallucination mitigation?

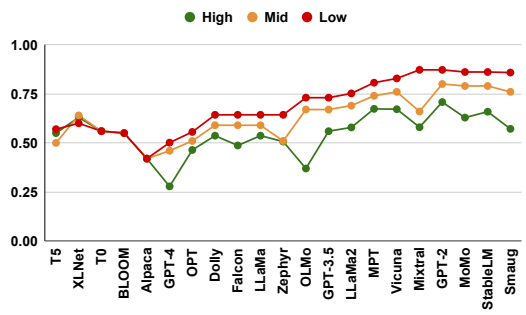
➠ The primary aim of ACTIVATOR is automation. End users might lack proper training and understanding of linguistic properties like formality, readability, or concreteness. Additionally, the functioning of LLMs is often a black box for end users. ACTIVATOR serves to assist end users in obtaining the best non-hallucinated output from LLMs.

A Appendix

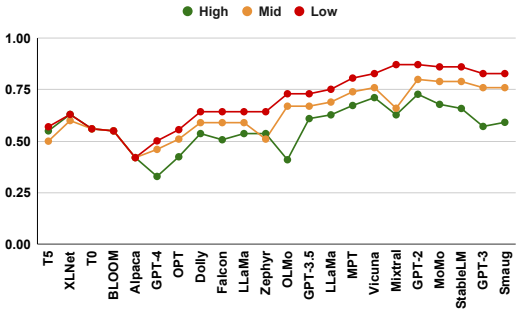
This section provides supplementary material in the form of additional examples, implementation details, etc. to bolster the reader's understanding of the concepts presented in this work.

B Linguistic Nuances

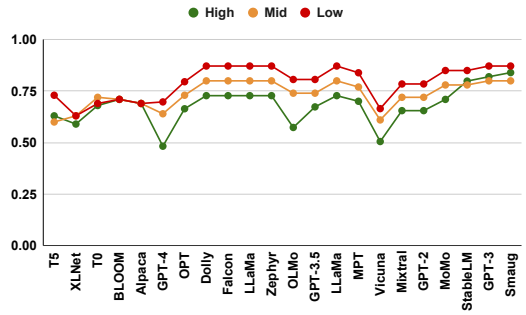
Linguistic nuances refer to subtle variations in language that convey additional meaning or context beyond the literal interpretation. **Readability** pertains to how easily text can be understood, often influenced by sentence structure and vocabulary. **Formality** involves the level of politeness or professionalism in language, ranging from casual to formal expressions. **Concreteness** relates to the degree of specificity and tangible details in language, with concrete language being more explicit and tangible than abstract language. These nuances contribute to the overall tone, clarity, and effectiveness of communication.



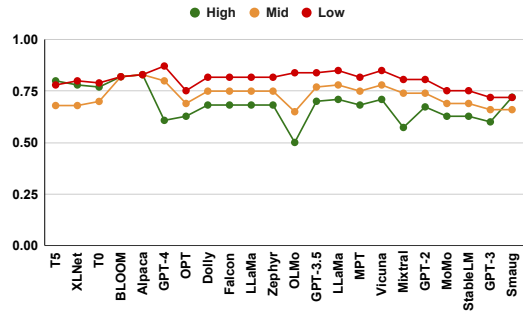
(a) Person



(b) Location



(c) Number



(d) Time

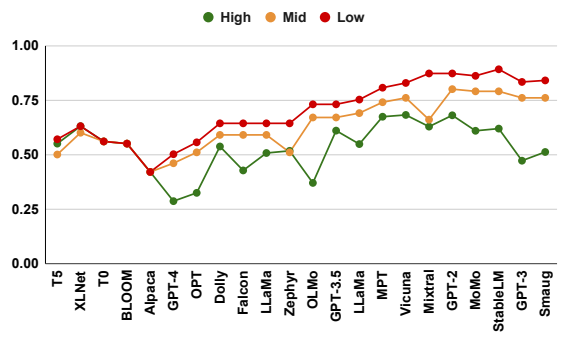
Research Questions on Readability

① How does the complexity of a prompt's language or vocabulary affect the likelihood of hallucination in LLM-generated responses?
② Does the length of a prompt impact the potential for hallucination, and how does the readability of a long versus a short prompt affect LLM behavior?
③ How do different LLMs (e.g., GPT-3, GPT-4, etc.) respond to prompts of varying linguistic readability, and do they exhibit differences in hallucination tendencies?

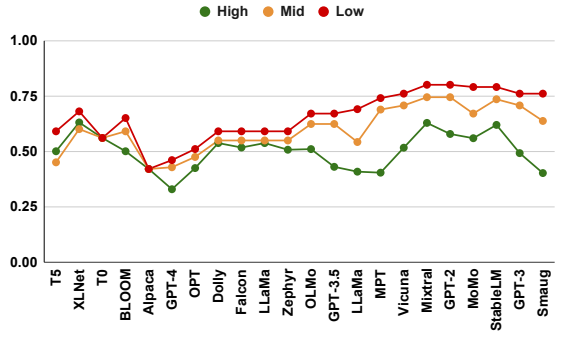
Effects on LLM's hallucination

① Prompts that are easier to read tend to have fewer instances of hallucinations.
② Some difficult-to-read prompts, but more formal also hallucinate less.
③ Hence, the results regarding readability are somewhat uncertain, displaying a combination of findings.

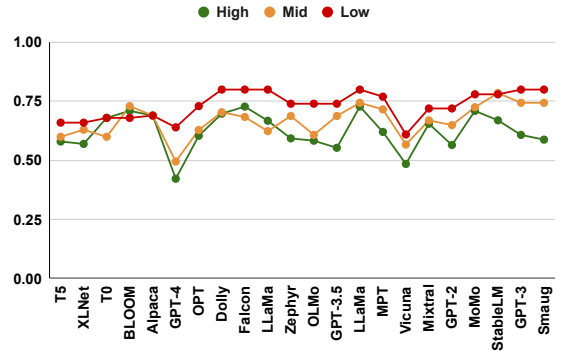
Figure 10: Readability



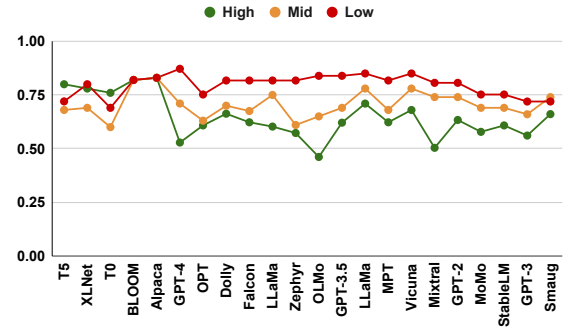
(a) Person



(b) Location



(c) Number



(d) Time

Research Questions on Formality

① How does the level of formality in prompts influence the likelihood of hallucination in responses generated by LLMs?
② Are there specific categories of hallucination that are more prevalent in responses prompted with formal versus informal language?

Effects on LLM's hallucination

① A decrease in the occurrence of hallucination is noticeable as the formality score increases, but LLM stopped responding to prompts having formality scores > 70.
② Hallucinations pertaining to personalities and locations show a partial reduction, but those involving numbers and acronyms largely persist without significant change.

Figure 11: Formality

C Dataset Annotation

Crowdsourcing platforms are widely acknowledged for their efficiency and cost-effectiveness in annotation tasks. However, it is crucial to acknowledge that they may introduce inaccuracies or noise in annotations. To address this, we conducted an in-house annotation process involving 1,000 samples before employing crowdsourcing services. This internal process involved prompts and generated text snippets from five different LLMs, serving the dual purpose of formulating comprehensive annotation guidelines and creating a tailored annotation interface. The internal annotation aimed to ensure the quality and reliability of annotations before transitioning to crowdsourcing. We follow the similar annoatation guidelines as (Rawte et al., 2023a) to generate the $\mathcal{SCA}-90\mathcal{K}$ dataset.

D Paraphrasing

Paraphrasing entails the process of rephrasing or altering the wording of a text while preserving its initial meaning. This practice aims to present the content differently to improve clarity, prevent plagiarism, and tailor the language for a particular audience or purpose. Successful paraphrasing demands a thorough grasp of the source material, involving the reorganization of sentences, alteration of word selections, and retention of core ideas without replicating the exact wording from the original text. Following are the three characteristics of paraphrasing methods.

Coverage: Our goal is to create up to 5 paraphrases for each claim. After generating the claims, we use the Minimum Edit Distance (MED) (Wagner and Fischer, 1974) measure (in words) for comparison. If the MED exceeds ± 2 for any paraphrase candidate (e.g., $c - p_1^c$) with the original claim, we include it; otherwise, we discard it. The evaluation is based on determining which model produces the highest number of meaningful paraphrases under this criterion.

Correctness: Following the initial filtration, we conducted pairwise entailment, retaining only paraphrase candidates endorsed as entailed by (Liu et al., 2019) (Roberta Large), the state-of-the-art model trained on SNLI (Bowman et al., 2015).

Diversity: Our focus was on selecting a model capable of producing linguistically diverse paraphrases. To assess this, we examined dissimilarities among generated paraphrase claims. For instance, we calculated dissimilarity scores for pairs like $c - p_n^c$, $p_1^c - p_n^c$, $p_2^c - p_n^c$, and so on, using the inverse of the BLEU score (Papineni et al., 2002). This process was repeated for all paraphrases, and the average dissimilarity score was computed. Our experiments revealed that `gpt-3.5-turbo-0301` performed the best in terms of linguistic diversity, as shown in the table. Furthermore, `gpt-3.5-turbo-0301` excelled in maximizing linguistic variations, as indicated in the diversity vs. models plot in Fig. 12.

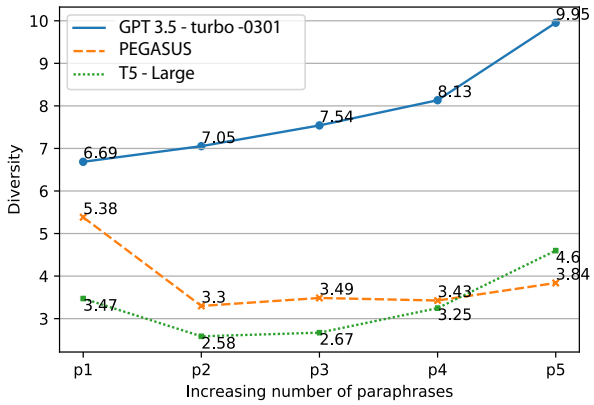


Figure 12: This figure shows the various parameters for generating paraphrases.

E Selecting the optimal paraphrase

E.1 Cosine Similarity

Cosine similarity is a metric used to measure the similarity between two vectors, often in the context of high-dimensional spaces. It calculates the cosine of the angle between the two vectors, providing a numerical value that indicates how closely related the vectors are.

In the context of natural language processing, cosine similarity is often employed to assess the similarity between two documents represented as vectors in a high-dimensional space, where each dimension corresponds to a term or word. The cosine similarity ranges from -1 (completely dissimilar) to 1 (completely similar), with 0 indicating orthogonality (no similarity).

The formula for cosine similarity between two vectors A and B is given in Eq. (1).

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{1}$$

E.2 Topic Modeling

Topic modeling is a statistical technique that aims to identify topics present in a collection of text documents. The goal is to uncover the hidden thematic structure within the text data. One common algorithm used for topic modeling is Latent Dirichlet Allocation (LDA).

In the process of topic modeling, each document in the corpus is considered as a mixture of various topics, and each topic is represented as a distribution of words. The algorithm analyzes the co-occurrence

patterns of words across documents to identify these latent topics. It helps in understanding the main themes or subjects present in a large collection of textual data without the need for manual annotation.

Topic modeling has applications in various NLP tasks, including document categorization, information retrieval, and content recommendation. It enables researchers and practitioners to gain insights into the underlying themes and structures within large textual datasets, making it a valuable tool for text analysis and understanding.

E.2.1 Topic Similarity

To overcome the issue of lengthy prompts, (Goyal et al., 2023) introduce, the idea of inserting `[PAUSE]` tokens. However, it is not clear where these tokens can be added. Since they follow a rather random approach, in this work, we use a more deterministic approach.

F Experimental Details

For different fine-tuning techniques, the list of hypermaters is provided in Table 4.

Parameter	Value
FC1 size	768
FC2 size	600
Number of epochs	5
Learning rate	1E-03
Optimizer	AdamW
Dropout probability	0.1
Batch size	1

Table 4: Hyperparameters for different fine-tuning techniques.

G Factuality based entailment

In this approach, the prompt is submitted to the Google Search API to retrieve the top 20 relevant search results. From these 20 results, we assess a total of n sentences for their pertinence to the prompt using a similarity metric. The top 20 sentences most akin to the prompt are chosen. For each of the m sentences in the AI-generated text and the selected top 20 sentences, we utilize a textual entailment model to individually evaluate their credibility. Based on the entailment scores, we classify the AI-generated text into three categories: (i) *support*, (ii) *refute*, and (iii) *not enough information*.

H Results after Adding [PAUSE] tokens

In the Table 5 below, we show the experimental results for adding [PAUSE].

Fine-tuning technique	Person			Location			Numeric			Time		
	Support	Refute	Neutral	Support	Refute	Neutral	Support	Refute	Neutral	Support	Refute	Neutral
Original Prompt	0.63	0.54	0.78	0.52	0.55	0.77	0.22	0.89	0.77	0.29	0.65	0.72
Optimal Paraphrase + LDA topics	0.65	0.26	0.59	0.59	0.28	0.54	0.36	0.36	0.66	0.44	0.56	0.7
		[PAUSE] Injection										
Optimal Paraphrase + LDA topics + w/ [PAUSE] token LoRA	0.7	0.19	0.69	0.61	0.25	0.53	0.53	0.29	0.69	0.59	0.29	0.72
Optimal Paraphrase + LDA topics + w/ [PAUSE] token QALoRA	0.72	0.21	0.67	0.62	0.22	0.52	0.58	0.32	0.67	0.62	0.31	0.73
Optimal Paraphrase + LDA topics + w/ [PAUSE] token Reverse Proxy Tuning	0.86	0.12	0.79	0.77	0.18	0.48	0.69	0.26	0.79	0.68	0.23	0.66

Table 5: Empirical results for Reverse Proxy Tuning with [PAUSE] tokens.

I Selecting the optimal paraphrased prompt

The detailed explanation of our algorithm to identify the optimal paraphrased prompt is provided in the illustration in Fig. 13.

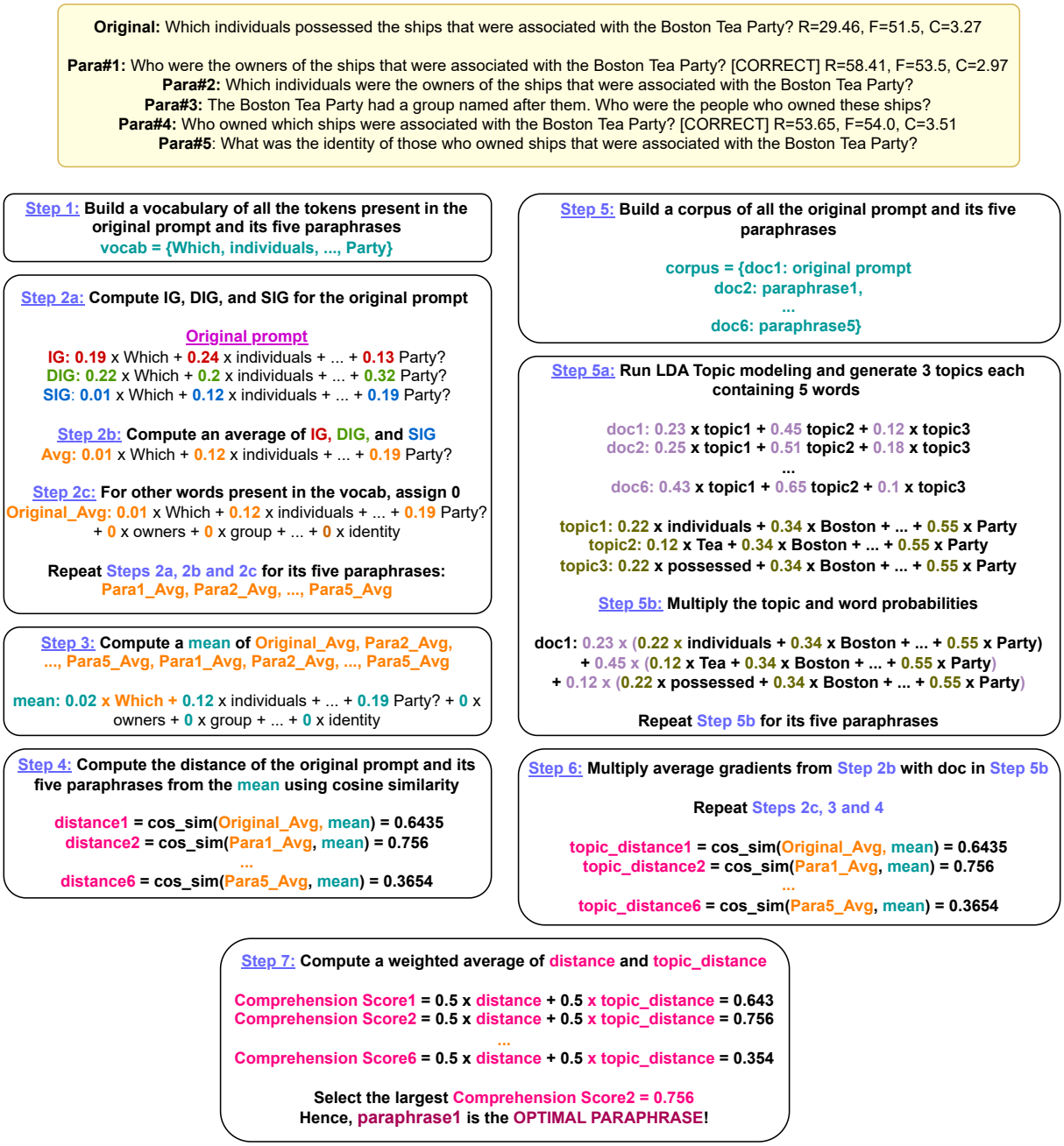
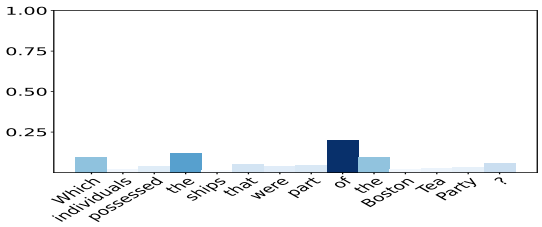


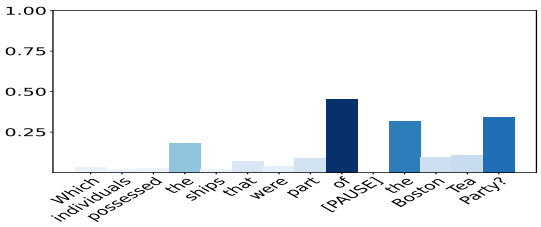
Figure 13: A walkthrough of our optimal paraphrase selection process.

J Before and after adding [PAUSE] token

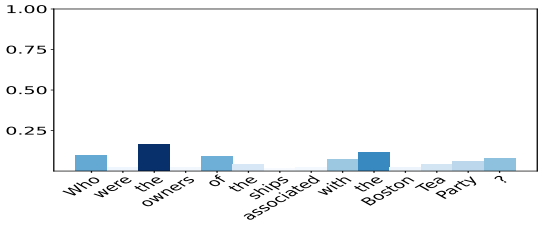
In the Figs. 14 to 24 below, we show the impact of adding [PAUSE] token to understand the longer prompts.



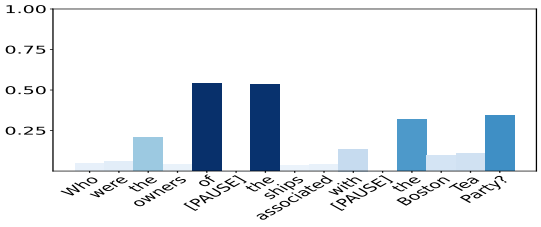
(a) Before adding [PAUSE] tokens to original prompt.



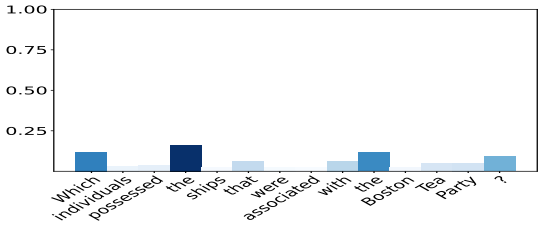
(b) After adding [PAUSE] tokens to original prompt.



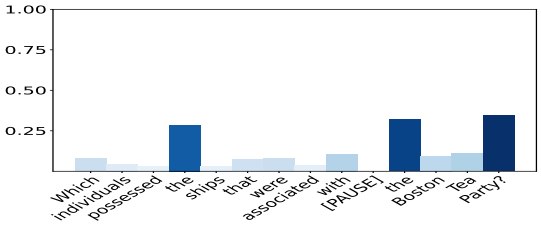
(c) Before adding [PAUSE] tokens to paraphrase 1.



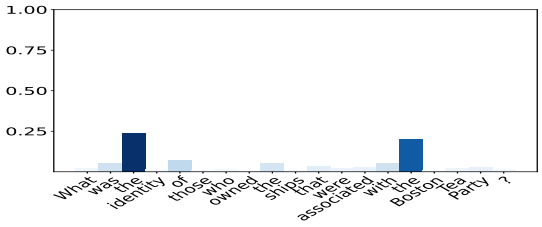
(d) After adding [PAUSE] tokens to paraphrase 1.



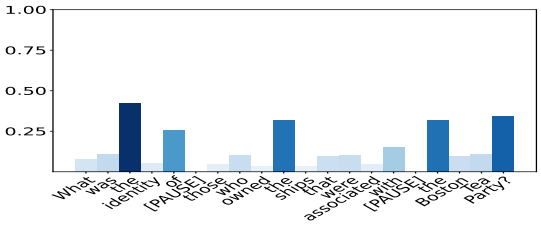
(e) Before adding [PAUSE] tokens to paraphrase 2.



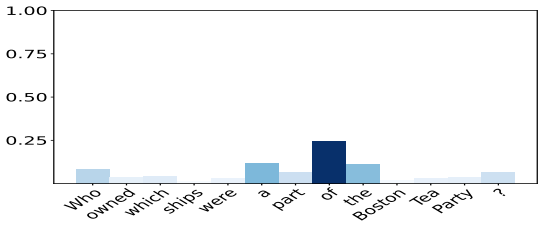
(f) After adding [PAUSE] tokens to paraphrase 2.



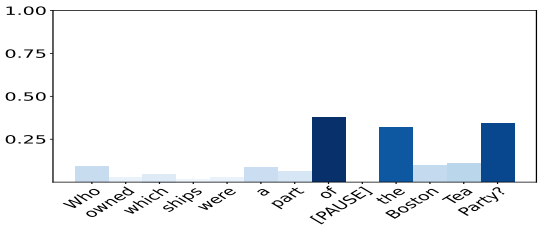
(g) Before adding [PAUSE] tokens to paraphrase 3.



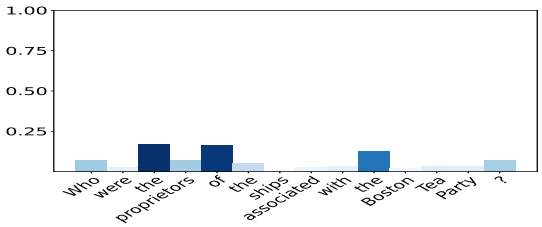
(h) After adding [PAUSE] tokens to paraphrase 3.



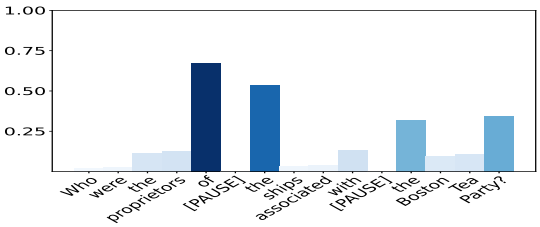
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

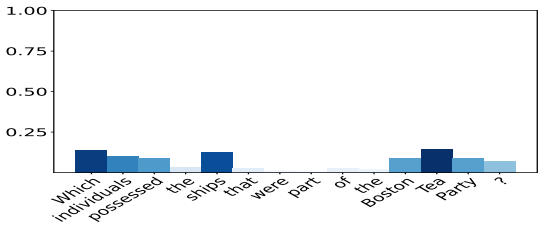


(k) Before adding [PAUSE] tokens to paraphrase 5.

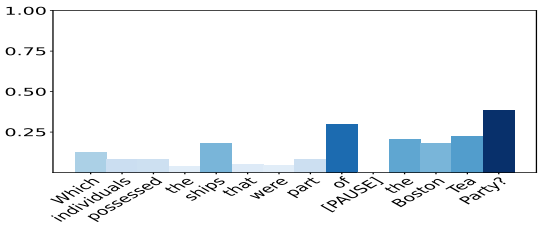


(l) After adding [PAUSE] tokens to paraphrase 5.

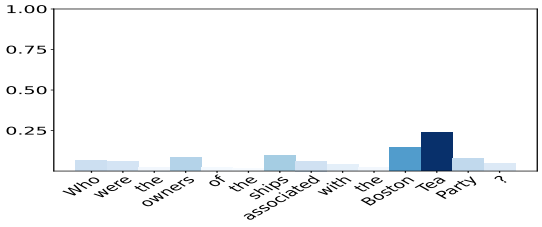
Figure 14: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for alpaca.



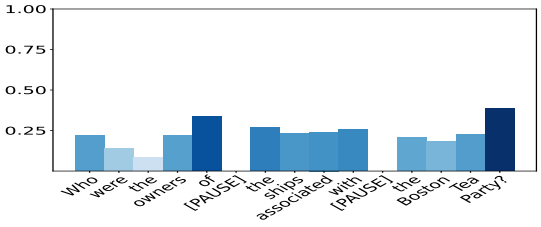
(a) Before adding [PAUSE] tokens to original prompt.



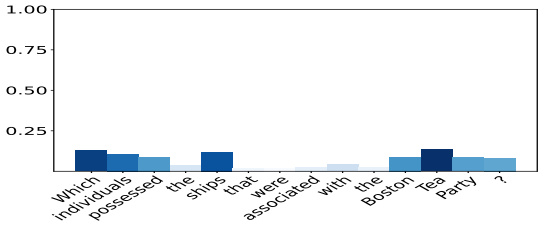
(b) After adding [PAUSE] tokens to original prompt.



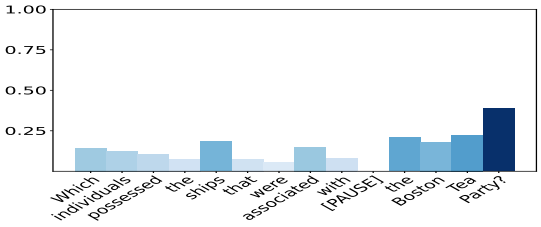
(c) Before adding [PAUSE] tokens to paraphrase 1.



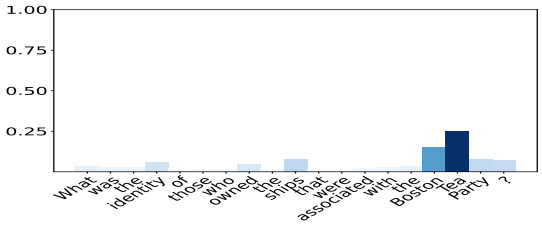
(d) After adding [PAUSE] tokens to paraphrase 1.



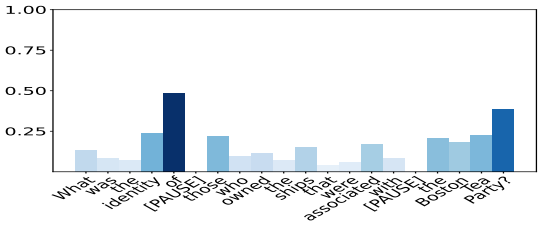
(e) Before adding [PAUSE] tokens to paraphrase 2.



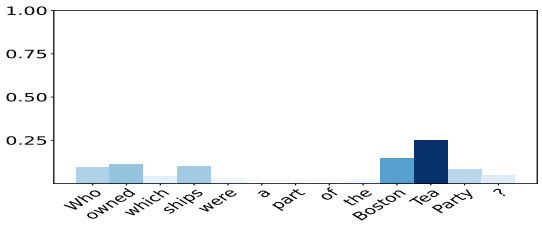
(f) After adding [PAUSE] tokens to paraphrase 2.



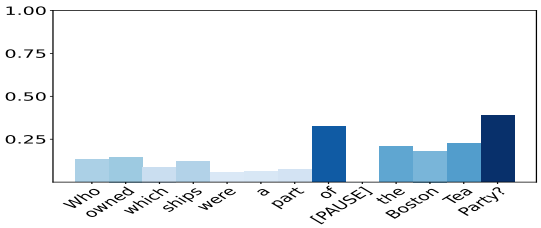
(g) Before adding [PAUSE] tokens to paraphrase 3.



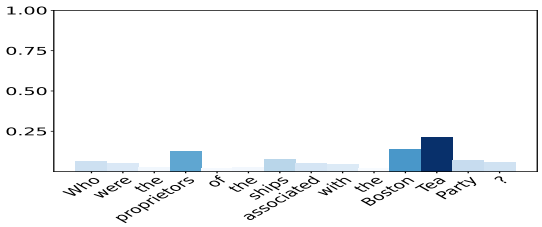
(h) After adding [PAUSE] tokens to paraphrase 3.



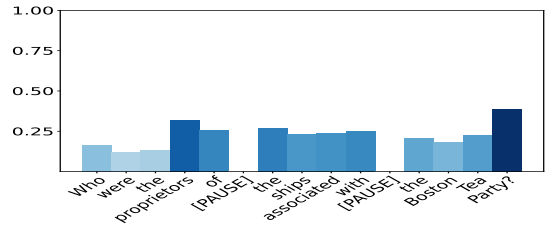
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

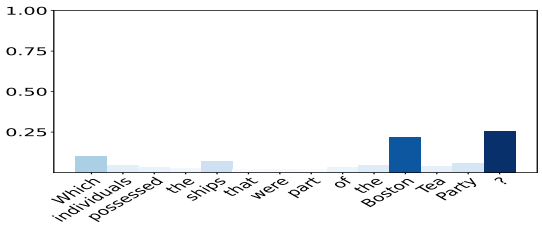


(k) Before adding [PAUSE] tokens to paraphrase 5.

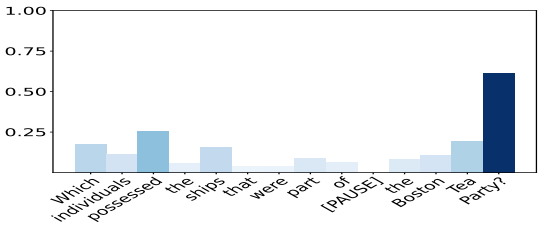


(l) After adding [PAUSE] tokens to paraphrase 5.

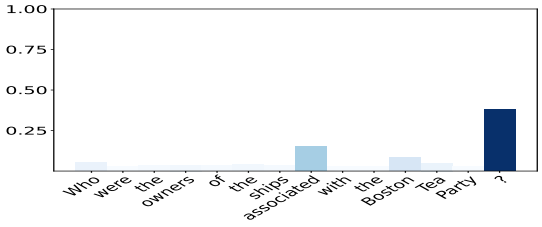
Figure 15: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for bloomz.



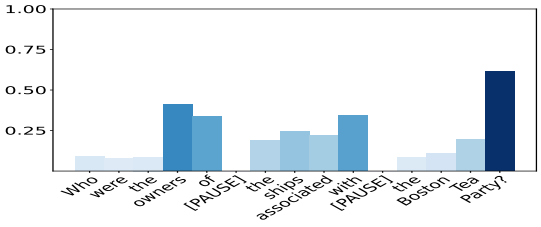
(a) Before adding [PAUSE] tokens to original prompt.



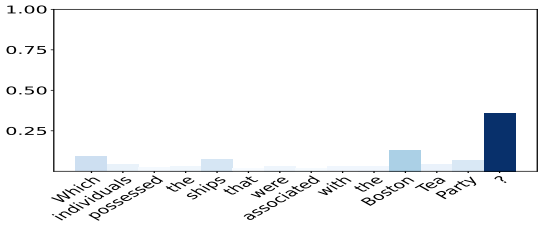
(b) After adding [PAUSE] tokens to original prompt.



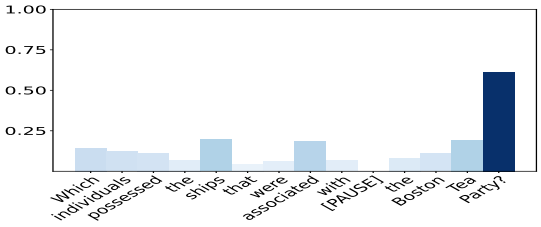
(c) Before adding [PAUSE] tokens to paraphrase 1.



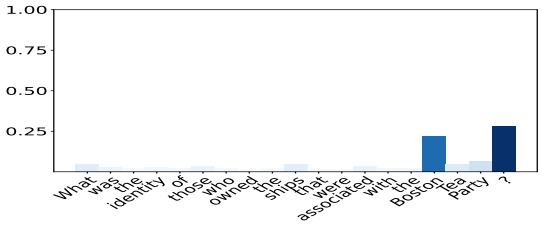
(d) After adding [PAUSE] tokens to paraphrase 1.



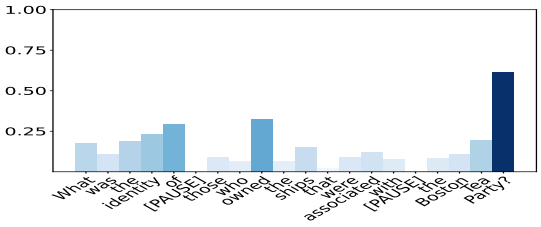
(e) Before adding [PAUSE] tokens to paraphrase 2.



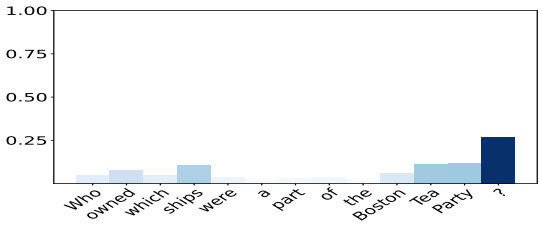
(f) After adding [PAUSE] tokens to paraphrase 2.



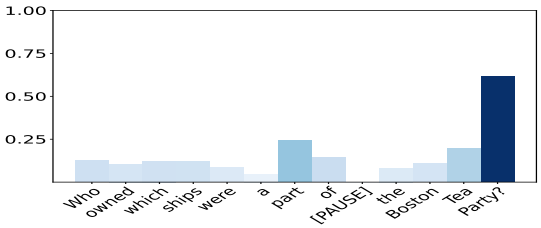
(g) Before adding [PAUSE] tokens to paraphrase 3.



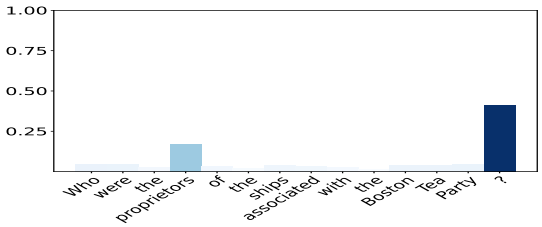
(h) After adding [PAUSE] tokens to paraphrase 3.



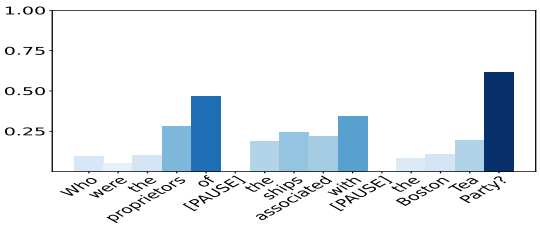
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

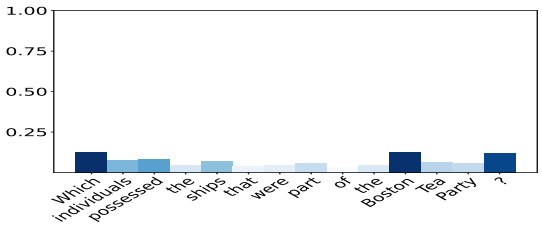


(k) Before adding [PAUSE] tokens to paraphrase 5.

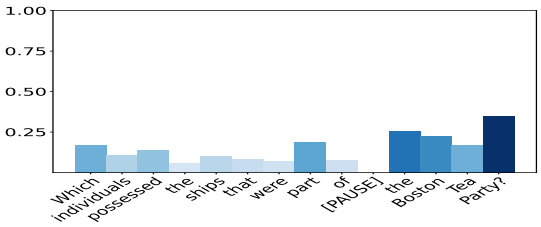


(l) After adding [PAUSE] tokens to paraphrase 5.

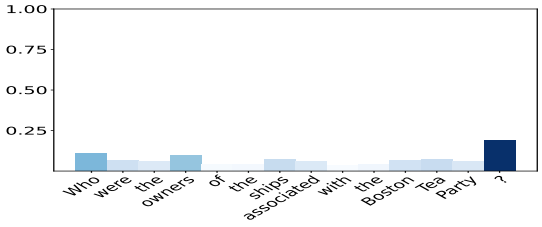
Figure 16: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for dolly.



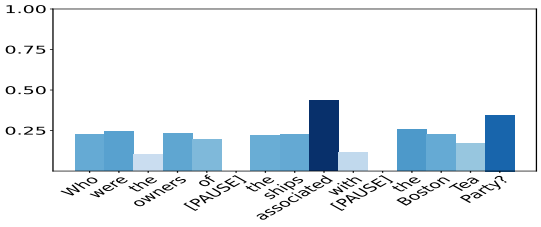
(a) Before adding [PAUSE] tokens to original prompt.



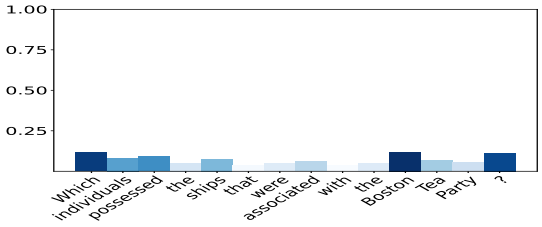
(b) After adding [PAUSE] tokens to original prompt.



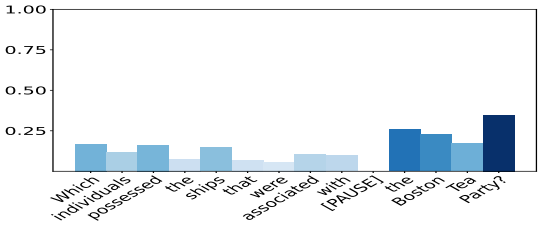
(c) Before adding [PAUSE] tokens to paraphrase 1.



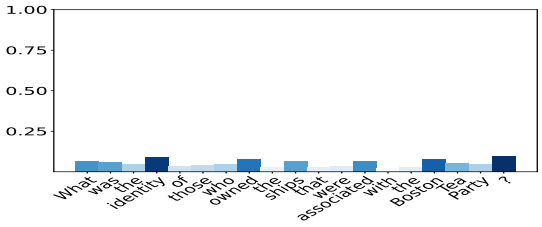
(d) After adding [PAUSE] tokens to paraphrase 1.



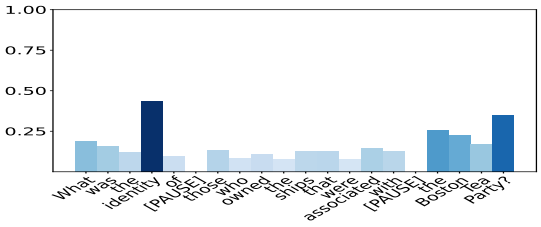
(e) Before adding [PAUSE] tokens to paraphrase 2.



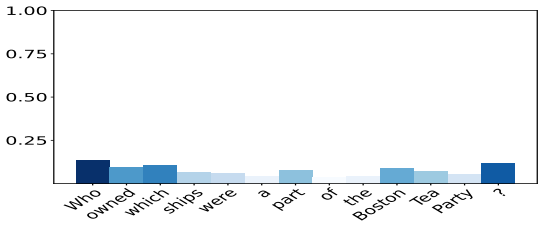
(f) After adding [PAUSE] tokens to paraphrase 2.



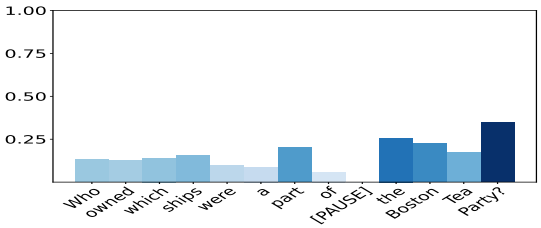
(g) Before adding [PAUSE] tokens to paraphrase 3.



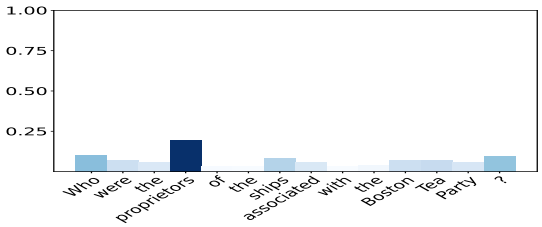
(h) After adding [PAUSE] tokens to paraphrase 3.



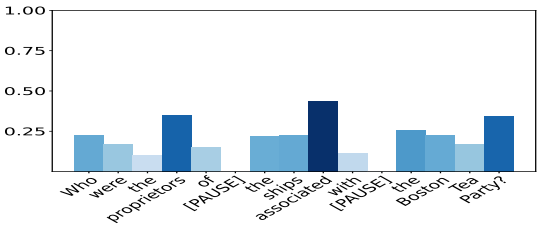
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

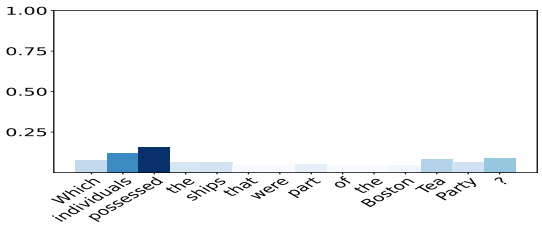


(k) Before adding [PAUSE] tokens to paraphrase 5.

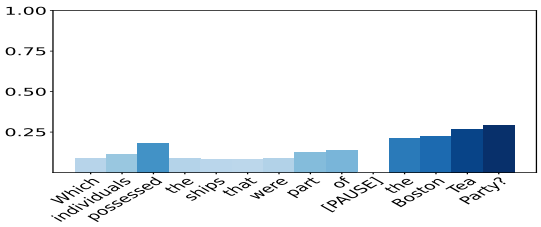


(l) After adding [PAUSE] tokens to paraphrase 5.

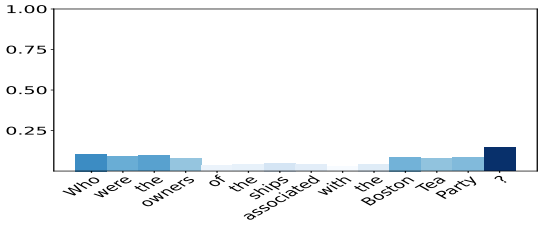
Figure 17: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for Falcon.



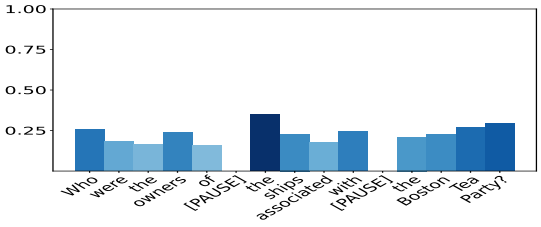
(a) Before adding [PAUSE] tokens to original prompt.



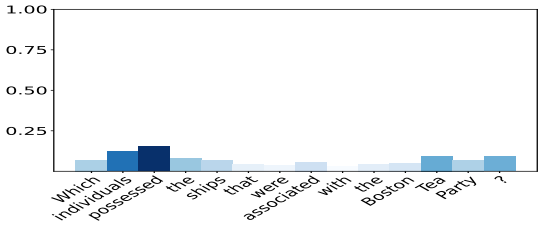
(b) After adding [PAUSE] tokens to original prompt.



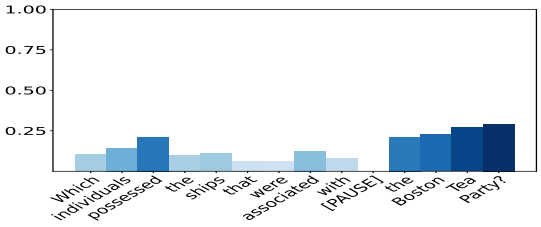
(c) Before adding [PAUSE] tokens to paraphrase 1.



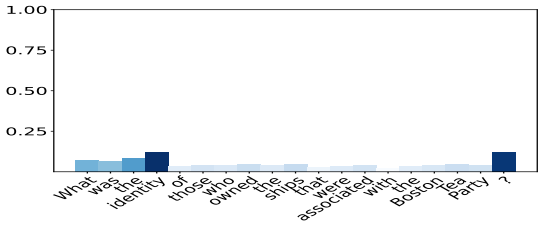
(d) After adding [PAUSE] tokens to paraphrase 1.



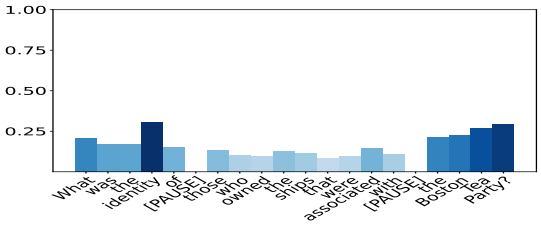
(e) Before adding [PAUSE] tokens to paraphrase 2.



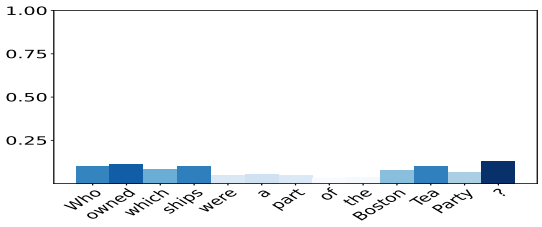
(f) After adding [PAUSE] tokens to paraphrase 2.



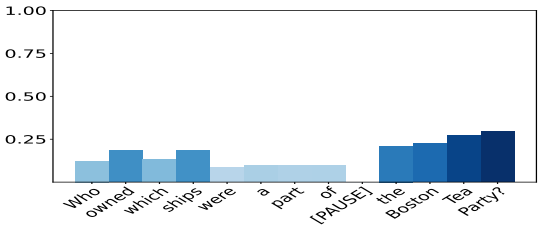
(g) Before adding [PAUSE] tokens to paraphrase 3.



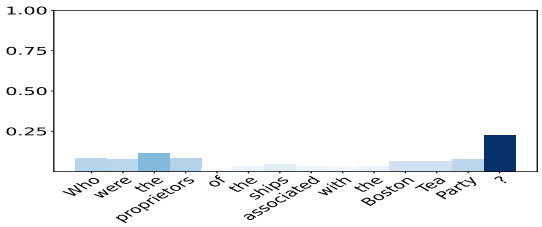
(h) After adding [PAUSE] tokens to paraphrase 3.



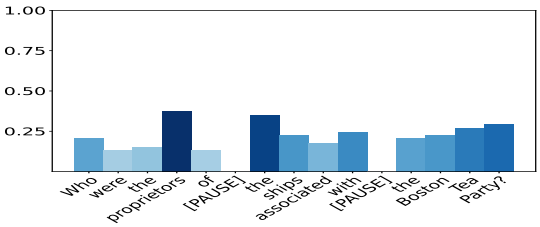
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

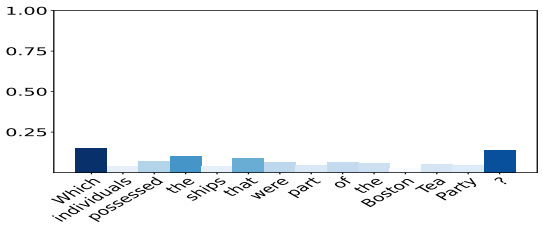


(k) Before adding [PAUSE] tokens to paraphrase 5.

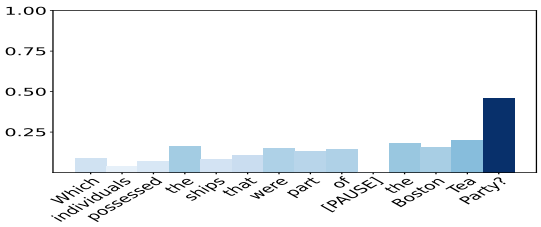


(l) After adding [PAUSE] tokens to paraphrase 5.

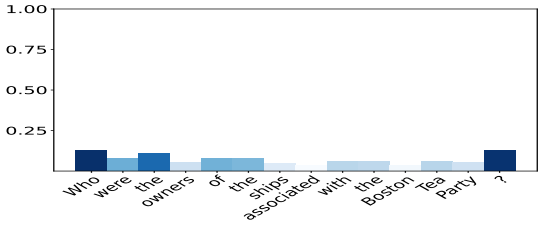
Figure 18: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for FLAN-T5.



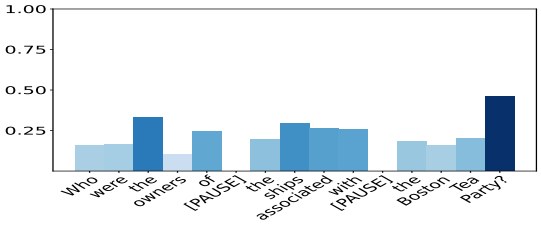
(a) Before adding [PAUSE] tokens to original prompt.



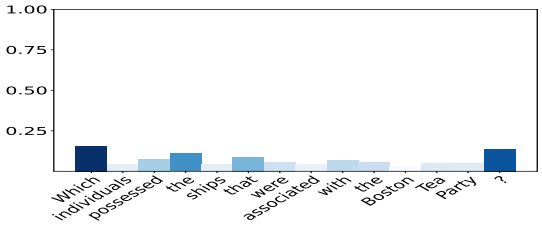
(b) After adding [PAUSE] tokens to original prompt.



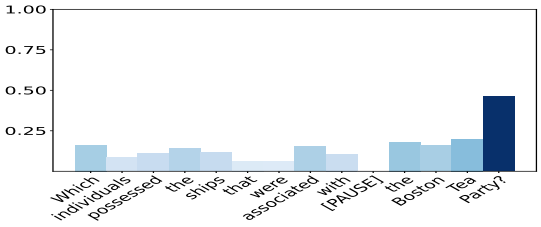
(c) Before adding [PAUSE] tokens to paraphrase 1.



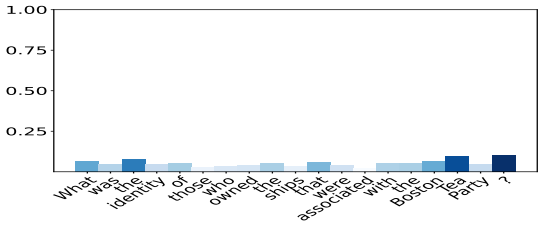
(d) After adding [PAUSE] tokens to paraphrase 1.



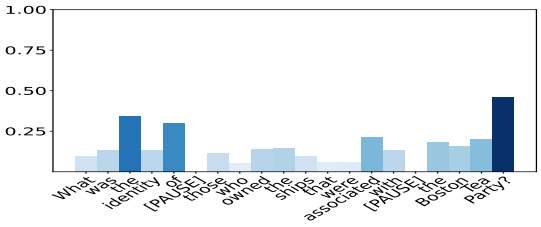
(e) Before adding [PAUSE] tokens to paraphrase 2.



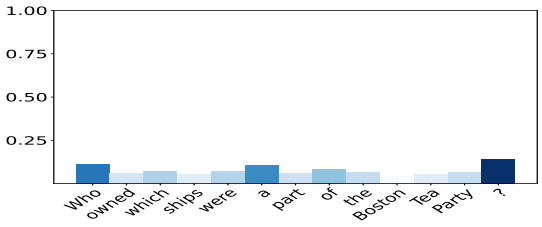
(f) After adding [PAUSE] tokens to paraphrase 2.



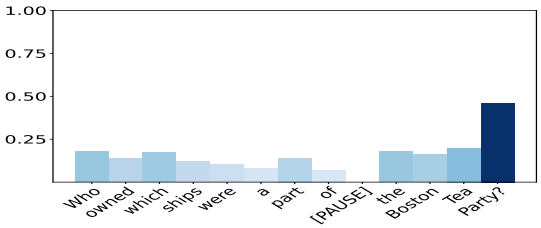
(g) Before adding [PAUSE] tokens to paraphrase 3.



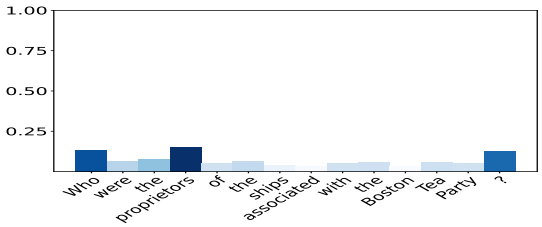
(h) After adding [PAUSE] tokens to paraphrase 3.



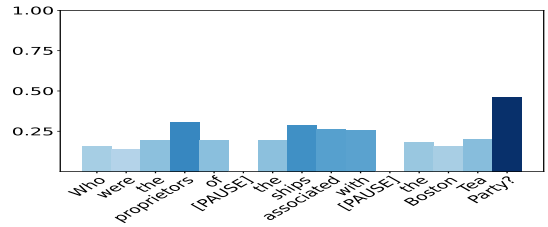
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

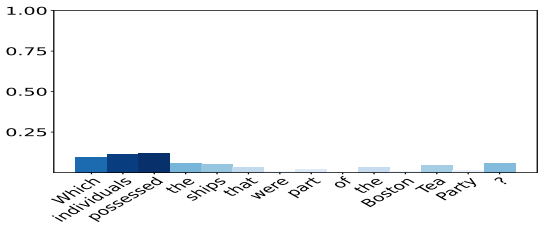


(k) Before adding [PAUSE] tokens to paraphrase 5.

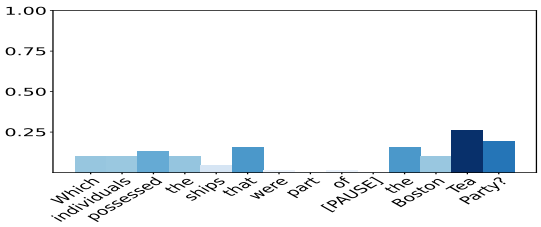


(l) After adding [PAUSE] tokens to paraphrase 5.

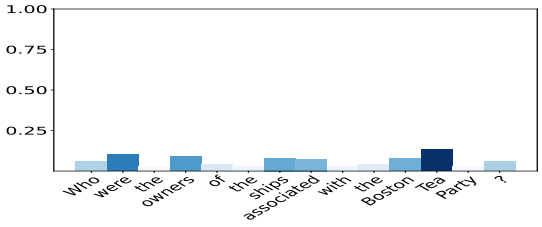
Figure 19: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for GPT Neo.



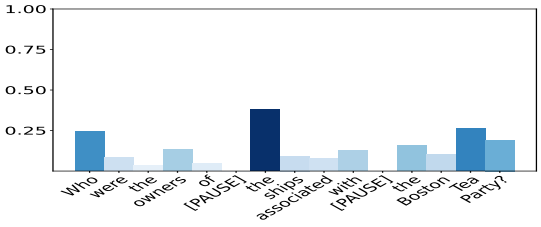
(a) Before adding [PAUSE] tokens to original prompt.



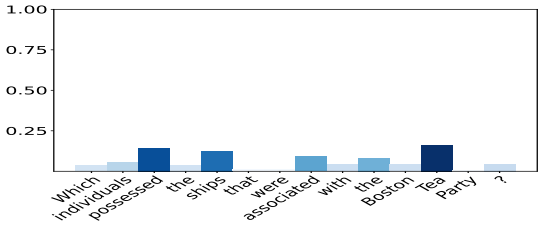
(b) After adding [PAUSE] tokens to original prompt.



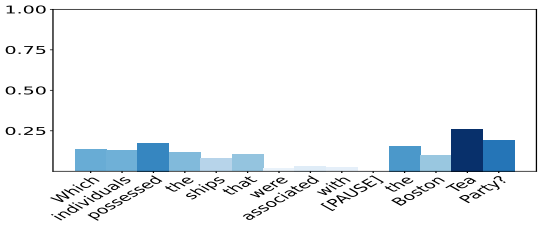
(c) Before adding [PAUSE] tokens to paraphrase 1.



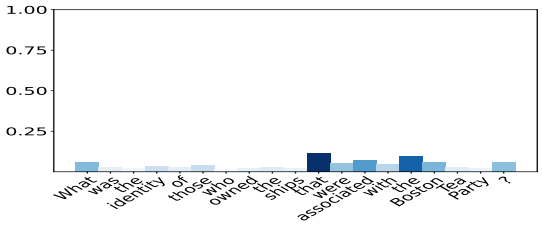
(d) After adding [PAUSE] tokens to paraphrase 1.



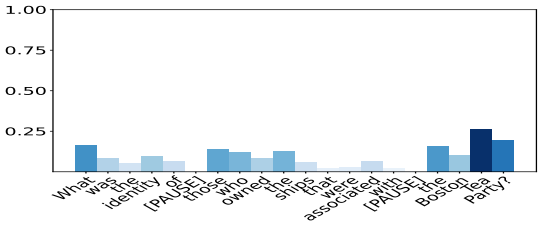
(e) Before adding [PAUSE] tokens to paraphrase 2.



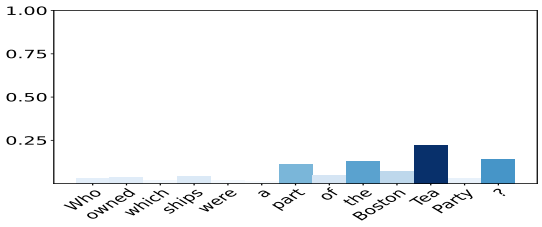
(f) After adding [PAUSE] tokens to paraphrase 2.



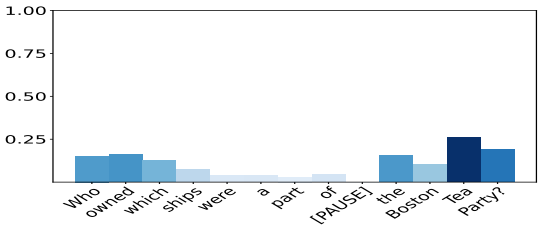
(g) Before adding [PAUSE] tokens to paraphrase 3.



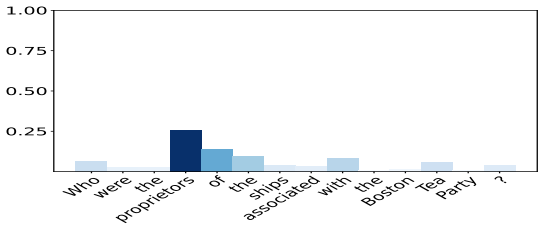
(h) After adding [PAUSE] tokens to paraphrase 3.



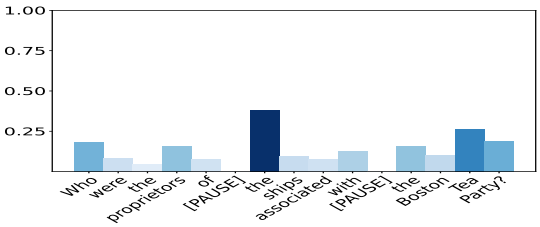
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

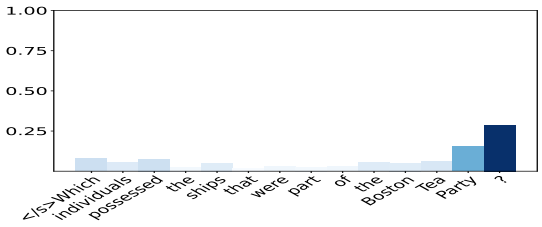


(k) Before adding [PAUSE] tokens to paraphrase 5.

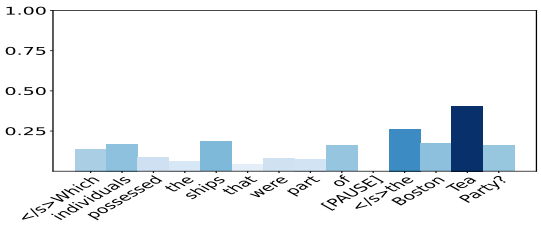


(l) After adding [PAUSE] tokens to paraphrase 5.

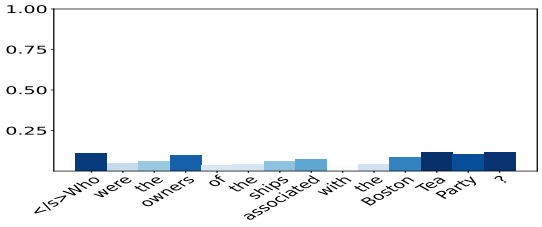
Figure 20: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for Llama2.



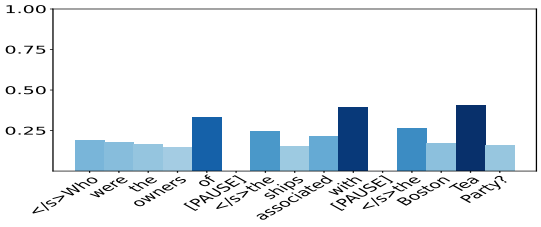
(a) Before adding [PAUSE] tokens to original prompt.



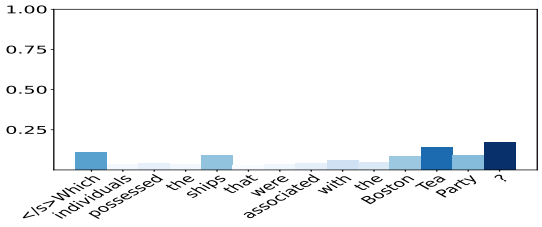
(b) After adding [PAUSE] tokens to original prompt.



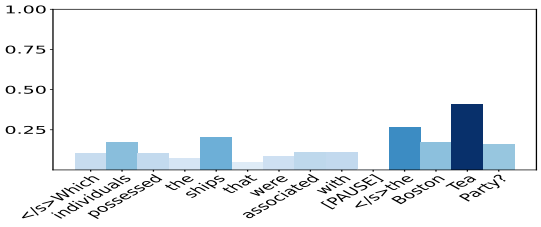
(c) Before adding [PAUSE] tokens to paraphrase 1.



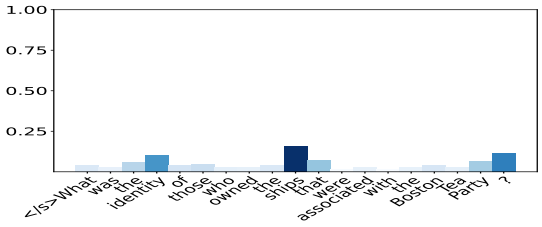
(d) After adding [PAUSE] tokens to paraphrase 1.



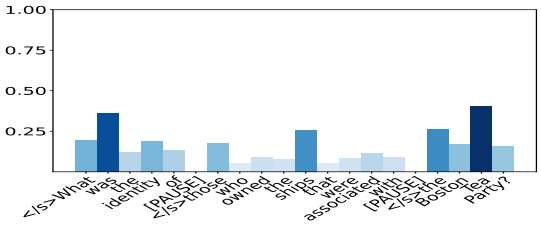
(e) Before adding [PAUSE] tokens to paraphrase 2.



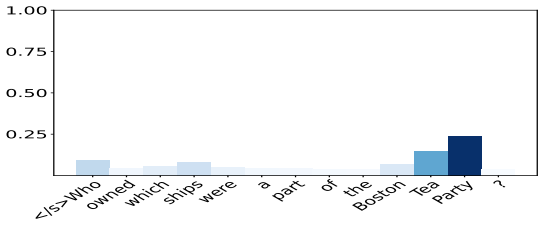
(f) After adding [PAUSE] tokens to paraphrase 2.



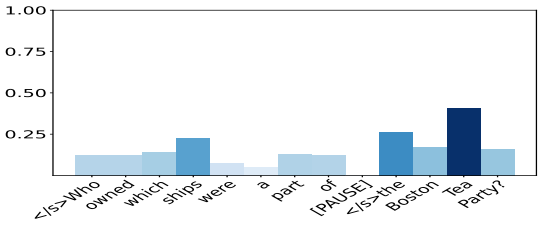
(g) Before adding [PAUSE] tokens to paraphrase 3.



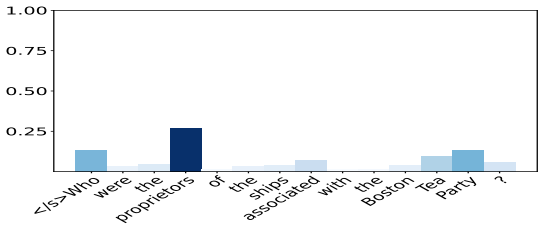
(h) After adding [PAUSE] tokens to paraphrase 3.



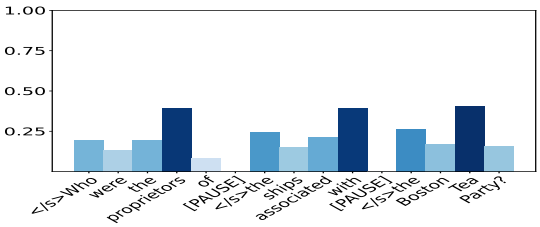
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

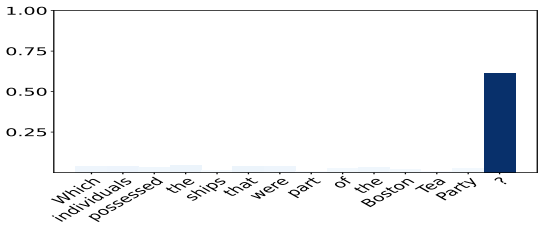


(k) Before adding [PAUSE] tokens to paraphrase 5.

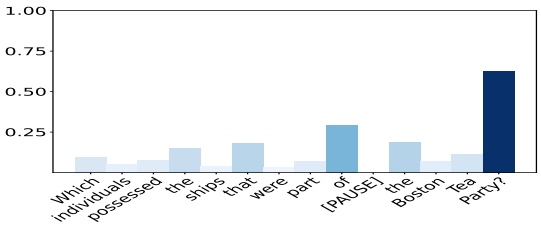


(l) After adding [PAUSE] tokens to paraphrase 5.

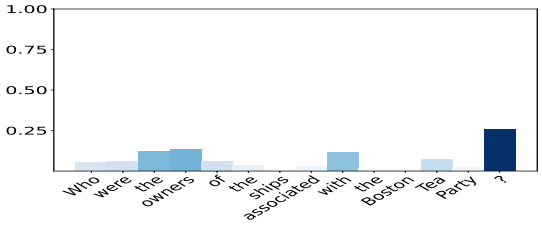
Figure 21: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for OPT.



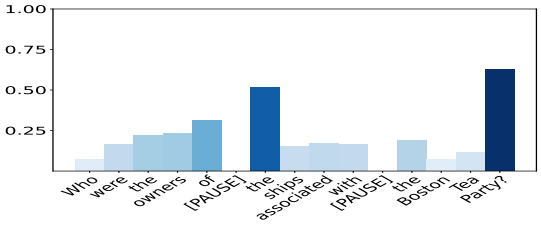
(a) Before adding [PAUSE] tokens to original prompt.



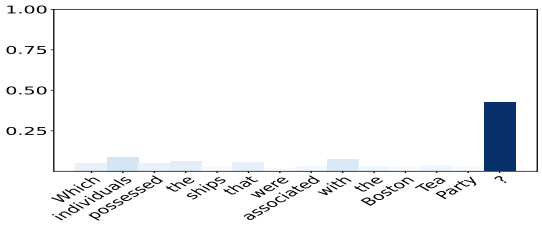
(b) After adding [PAUSE] tokens to original prompt.



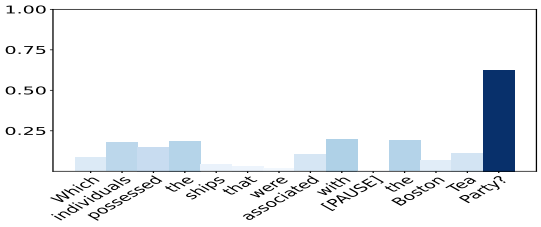
(c) Before adding [PAUSE] tokens to paraphrase 1.



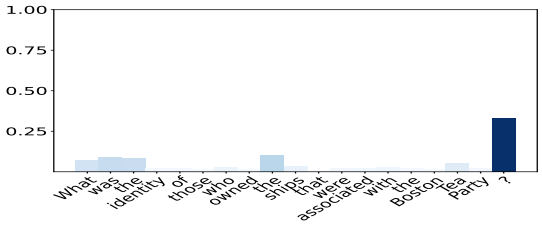
(d) After adding [PAUSE] tokens to paraphrase 1.



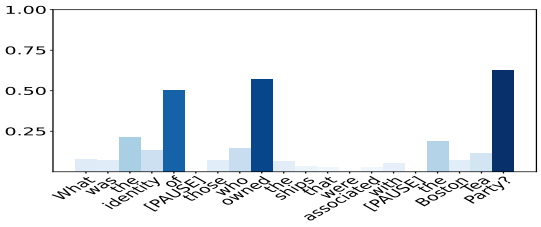
(e) Before adding [PAUSE] tokens to paraphrase 2.



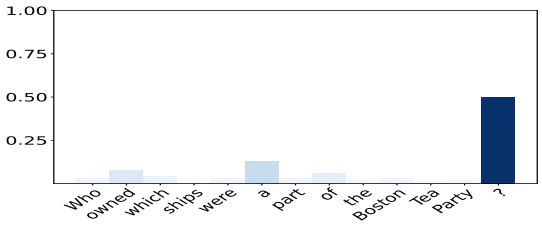
(f) After adding [PAUSE] tokens to paraphrase 2.



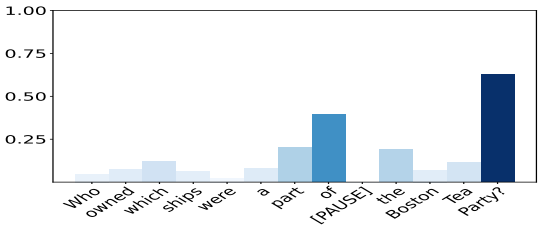
(g) Before adding [PAUSE] tokens to paraphrase 3.



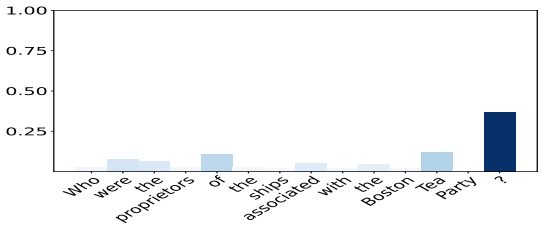
(h) After adding [PAUSE] tokens to paraphrase 3.



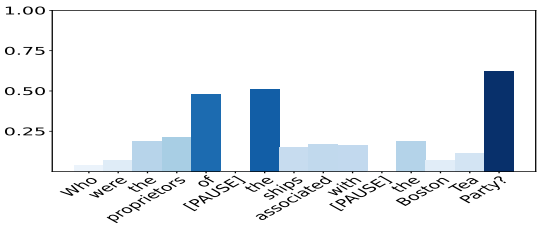
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

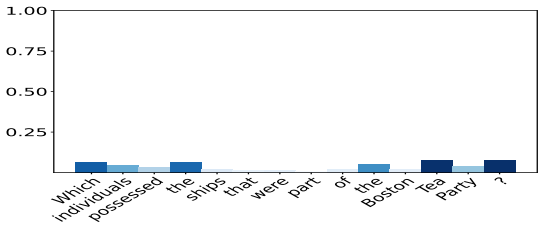


(k) Before adding [PAUSE] tokens to paraphrase 5.

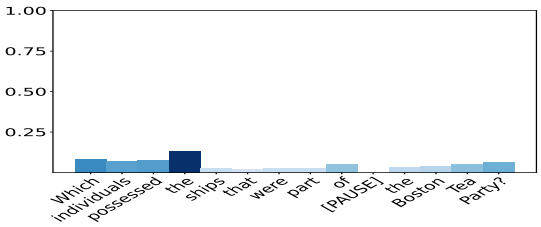


(l) After adding [PAUSE] tokens to paraphrase 5.

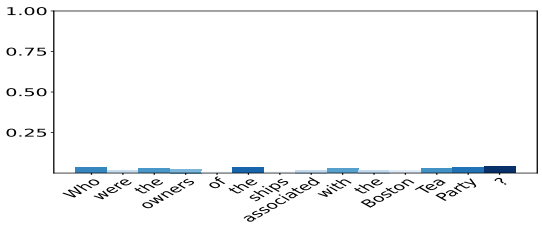
Figure 22: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for phi-2.



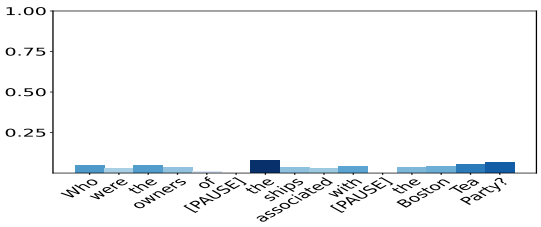
(a) Before adding [PAUSE] tokens to original prompt.



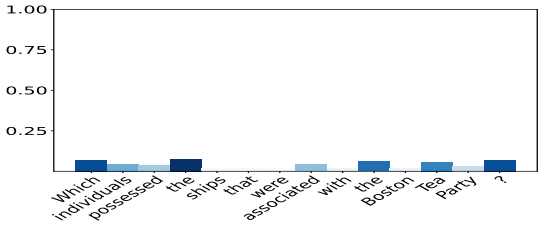
(b) After adding [PAUSE] tokens to original prompt.



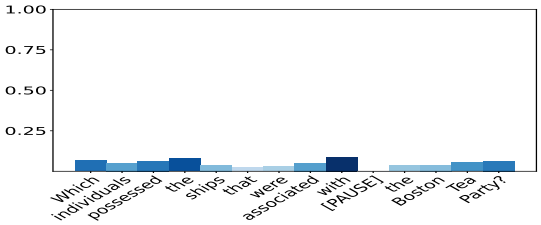
(c) Before adding [PAUSE] tokens to paraphrase 1.



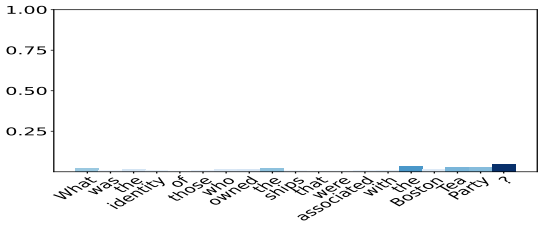
(d) After adding [PAUSE] tokens to paraphrase 1.



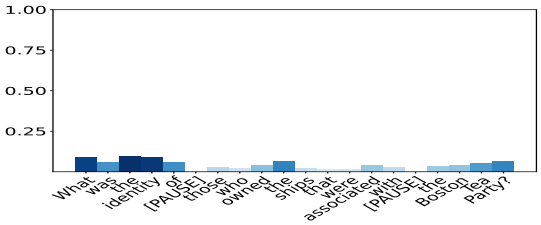
(e) Before adding [PAUSE] tokens to paraphrase 2.



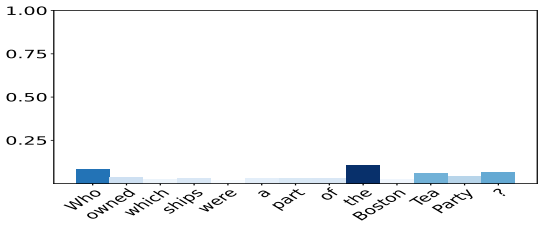
(f) After adding [PAUSE] tokens to paraphrase 2.



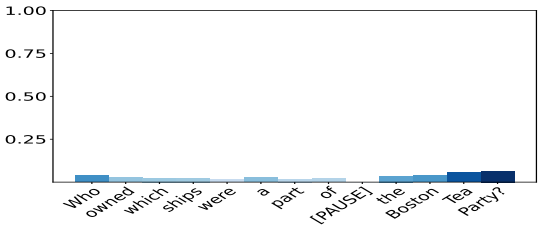
(g) Before adding [PAUSE] tokens to paraphrase 3.



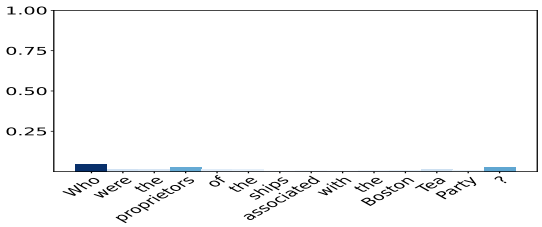
(h) After adding [PAUSE] tokens to paraphrase 3.



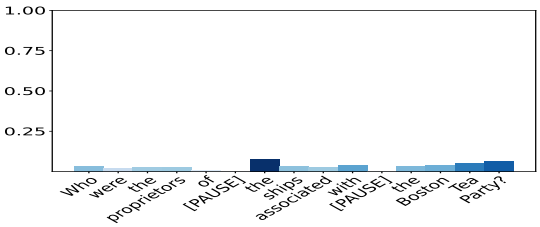
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.

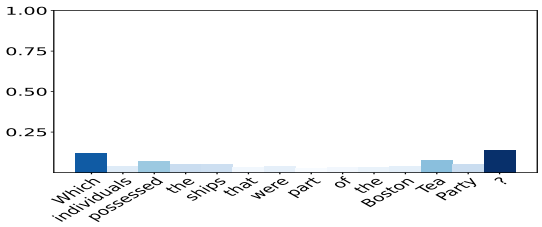


(k) Before adding [PAUSE] tokens to paraphrase 5.

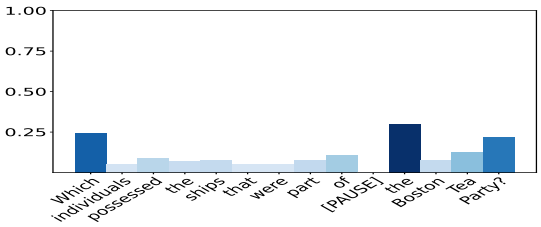


(l) After adding [PAUSE] tokens to paraphrase 5.

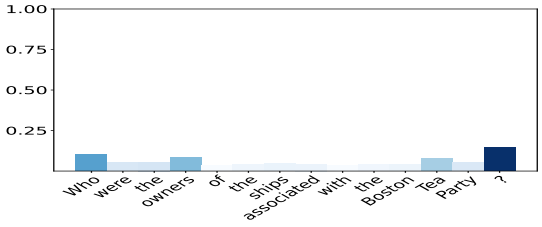
Figure 23: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for Vicuna.



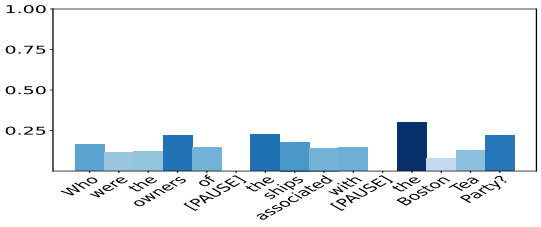
(a) Before adding [PAUSE] tokens to original prompt.



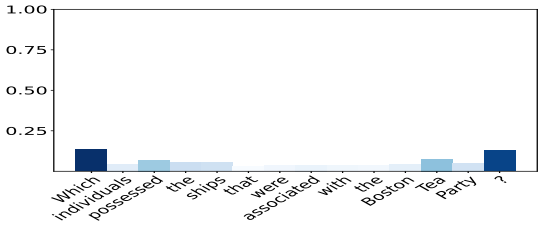
(b) After adding [PAUSE] tokens to original prompt.



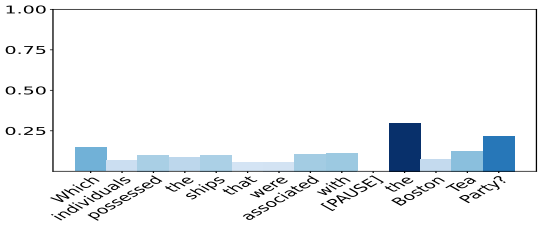
(c) Before adding [PAUSE] tokens to paraphrase 1.



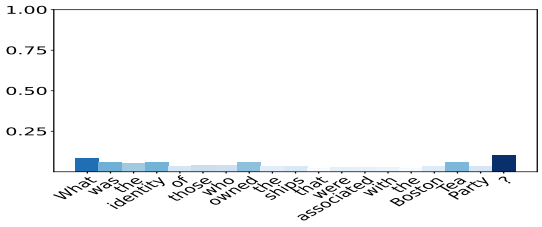
(d) After adding [PAUSE] tokens to paraphrase 1.



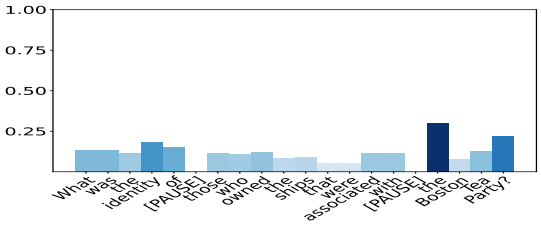
(e) Before adding [PAUSE] tokens to paraphrase 2.



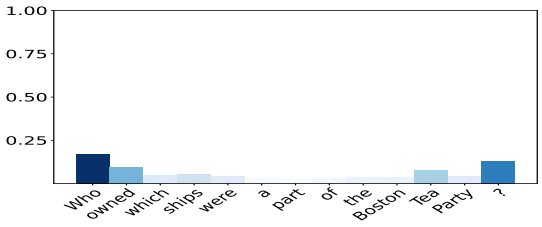
(f) After adding [PAUSE] tokens to paraphrase 2.



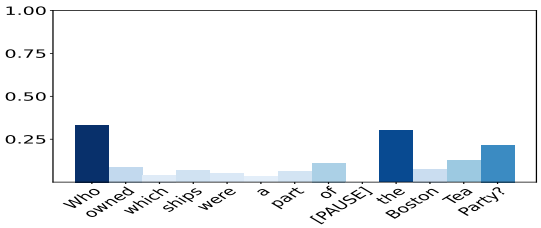
(g) Before adding [PAUSE] tokens to paraphrase 3.



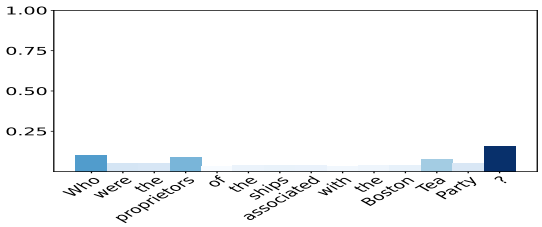
(h) After adding [PAUSE] tokens to paraphrase 3.



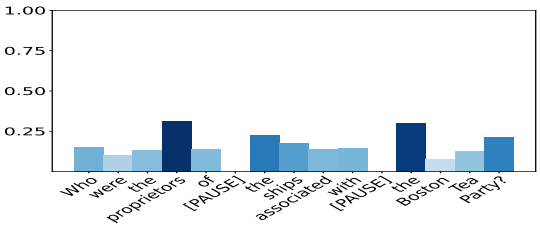
(i) Before adding [PAUSE] tokens to paraphrase 4.



(j) After adding [PAUSE] tokens to paraphrase 4.



(k) Before adding [PAUSE] tokens to paraphrase 5.



(l) After adding [PAUSE] tokens to paraphrase 5.

Figure 24: The phrase **Boston Tea** gets more importance score after adding [PAUSE] token for Zephyr.