COMPENSATORY BIASES UNDER COGNITIVE LOAD: REDUCING SELECTION BIAS IN LARGE LANGUAGE MODELS

A PREPRINT

Jonathan E. Eicher
jonathan@elsworth.phd

Rafael F. Irgolič
cbuclrsbllm@irgolic.com

January 29, 2024

ABSTRACT

Large Language Models (LLMs) like gpt-3.5-turbo and claude-instant-1.2 have become instrumental in interpreting and executing semantic-based tasks. Unfortunately, these models' inherent biases, akin to human cognitive biases, adversely affect their performance. Particularly affected is object selection from lists; a fundamental operation in digital navigation and decision-making. This research critically examines these biases and quantifies the effects on a representative list selection task. To explore these biases, we conducted a series of controlled experiments, manipulating temperature, list length, object identity, object type, prompt complexity, and model. This enabled us to isolate and measure the influence of the biases on selection behavior. Our findings show that bias structure is strongly dependent on the model, with object type modulating the magnitude of the effect. With a strong primacy effect, causing the first objects in a list to be disproprotionately represented in outputs. Furthermore the usage of guard rails, a prompt engineering method of ensuring a response structure, can increase bias and decrease instruction adherence when combined with a selection task. The bias is ablated when the guard rail step is separated from the list sampling step, lowering the complexity of each individual task. The implications of this research are two-fold, practically providing a guide for designing unbiased LLM applications and theoretically suggesting that LLMs experience a form of cognitive load compensated for by increasing bias.

Keywords Large Language Models • Cognitive load • List Selection • Bias • Guard rails

1 Introduction

Bias is a quality exhibited by humans and AI alike (Li (2010), Nauts et al. (2014)). Humans have a tendency toward applying our biases inappropriately, such as in the case of multiple choice exams where item order affects the average for exams (Balch (1989)). This phenomenon of multiple choice bias is also well-documented in large language models (LLMs) (Zheng et al. (2023)). Bias is something central to the human experience and our ability to reduce cognitive load when making decisions (Allred et al. (2016)). That said, problems arise when we do not account for bias in the creation of our decision structures (Kirk et al. (2021), Weidinger et al. (2022)).

This is particularly true of LLMs, which inherit the structural bias of the sum total of human language (Kinniment et al. (2024)). As time progresses we will find LLMs in an increasing number of decision-making roles. Each of these new roles requires careful consideration to the risks and benefits of using LLMs, as well as bias mitigation strategies (Weidinger et al. (2022)). Of particular note are the inherent biases displayed by LLMs when analyzing the context for their responses: the "lost in the middle" phenomena wherein the LLM is unable to properly include information at the center of their context highlights this danger (Liu et al. (2023)). This risk is only compounded by the observed extant bias present in the training data, and by extension the output of the LLMs (Kirk et al. (2021), Touileb, Øvrelid, and Velldal (2022), Wolfe and Caliskan (2021)). While there are efforts to minimize bias in LLMs, there is a significant lack of research into the fundamental question of how bias itself manifests in LLMs (Liang et al. (2021))

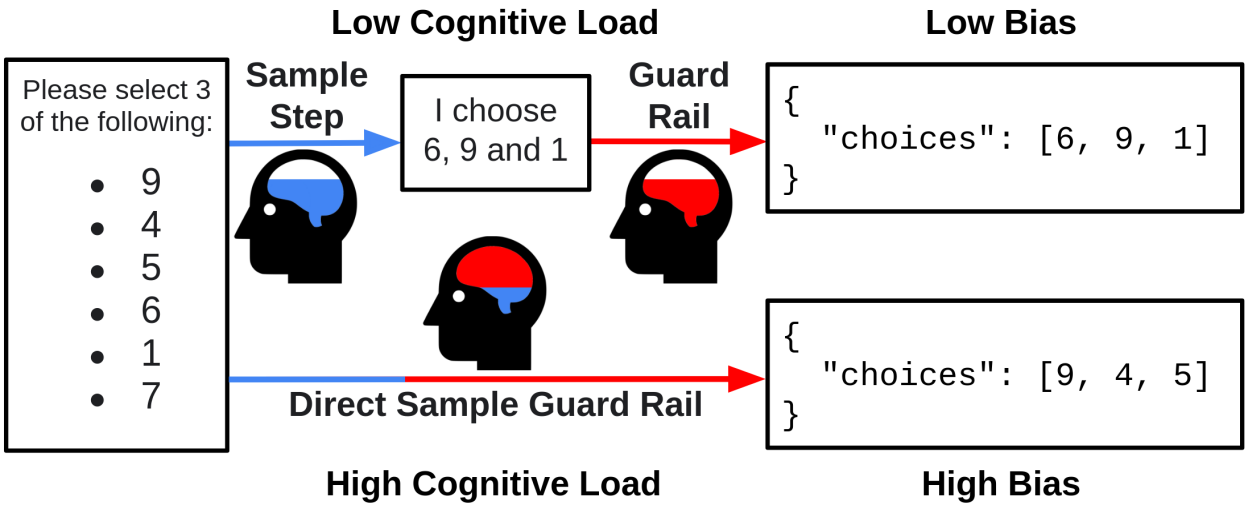


Figure 1: The two data collection methods used in this paper: a two-step method which separates the sample step and guard rail, and a direct guard rail method which combines them into a single step. Each method is annotated with its relative cognitive load and bias as a result of the method.

There have been some attempts to quantify the bias of LLMs dealing with selecting objects from lists (Zheng et al. (2023)). While intriguing, the results are not generalizable; moreover they focus on engineering concerns of LLM benchmarking rather than experimental exploration of bias profiles (Han et al. (2023)). Similarly, there has been work in output assurance via guard rails – a validator that assures a LLM's output structure (Rebedea et al. (2023)). Although, to our knowledge there has been no work on the effects of guard rails on LLM bias.

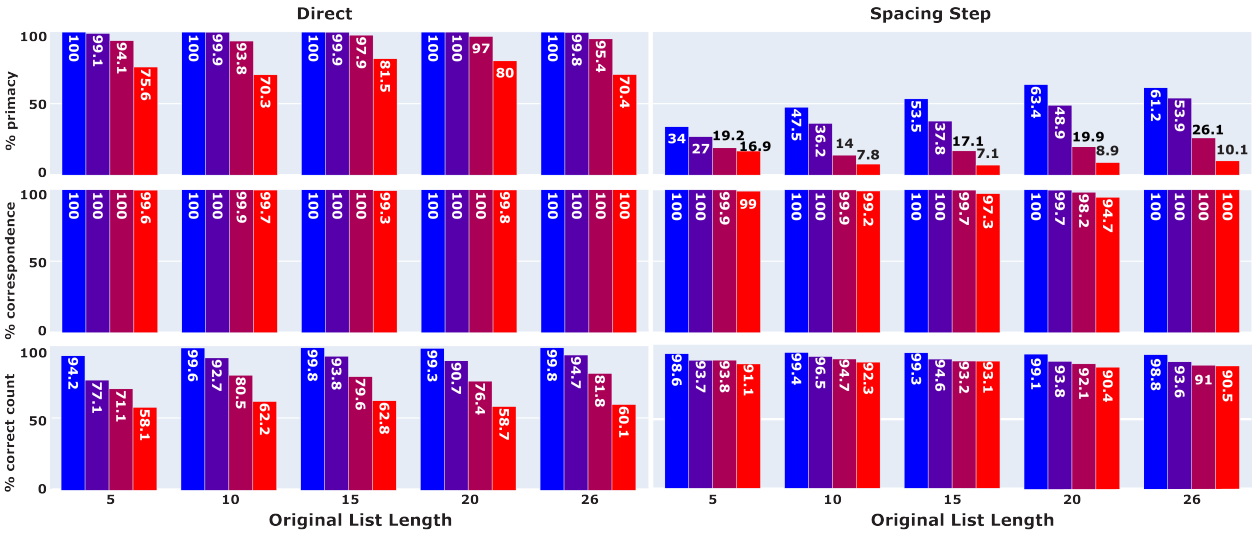


Figure 2: The percent of responses displaying primacy bias, correspondence, and correct count for both a direct guard rail and sample step sampling methodology when selecting numbers from a list. This was performed via gpt-3.5-turbo and for a variety of temperatures (blue 0 to red 1.5 with a step size of 0.5) and list lengths.

In order to address this gap we developed a simple model problem and rigorously analyzed the selection probabilities of two LLMs: gpt-3.5-turbo and claude-instant-1.2. The toy problem we devised was the selection of three objects from a variable-length list. This toy example can be used to extract guiding principles for the analysis of bias in LLMs and how to implement agentic frameworks that prevent exacerbation of these biases.

We selected several variables for our toy problem of selecting three objects from a list: the list length, object identities, temperature, and model. We analyzed the probabilities of the values being selected given an object's identity and its position; allowing us to systematically evaluate the effects of temperature modulation, list length modulation,

model type, object identity, object type, and position on the selection probabilities. We used guard rails derived from Guardrails AI ("Guardrails AI | Your Enterprise AI Needs Guardrails" (n.d.)) to parse our outputs and test their effects on selection biases.

We found that primacy bias is indeed a problem in our context, but is significantly model dependent (Figure 3). The identity and position of our objects significantly altered the selection probabilities (Figure 4). When comparing models we found that biases on position and letters were not consistent, with variation in every form of bias measured (Figure 5). A guard rail significantly alters the primacy bias and instruction adherence for all considered models (Figure 2). We hypothesize that this is a result of the cognitive load of the guard rail, which may be causing human-like compensatory behavior (de Jong (2010), Xu et al. (2023)).

2 Methods

Two LLMs were accessed via their APIs: anthropic-instant-1.2 and gpt-3.5-turbo. We tested a range of temperatures for each model: $[0, 0.5, 1, 1.5]$ for gpt-3.5-turbo, and $[-1, 0, 0.5, 1]$ for claude-instant-1.2. These models were presented with a list of letters or numbers of modular length, prepended with an instruction. The output of the first LLM call (Listing 1) was then fed into a guard rail to extract the choices (Listing 2), as opposed to the direct method, which implemented list selection and guard rail application in one LLM call (Listing 3). This process was repeated $N = 1000$ times for each temperature, input list length, and model. This allowed us to complete all analysis for a given model, temperature, input, and list length. If the output was not in a JSON format or the output was not the same length as the input the trial was discarded and noted as having failed the correct count condition.

The input list was selected from a pool of object $\mathbb{X}_p$ with a length of n_p, which in our study was 26, referring to the numbers 1-26 and the letters A-Z:

$$\mathbb{x}_p = \begin{bmatrix} 1 \\ 2 \\ ... \\ n_p \end{bmatrix}$$

$\mathbb{X}_p$ was then uniformly sampled with a list length, n_t, of 5, 10, 15, 20, or 26. The input list was ordered so when analyzing we considered it as an $n_t \times 2$ matrix where ℓ is the object and p is the position:

$$\mathbb{x}_t = \begin{bmatrix} a, 1 \\ d, 2 \\ q, 3 \\ ... \\ \ell, p_t \end{bmatrix}$$

The outputs were sent to a LLM through either a direct method or with a sample step. The direct method presents the input list alongside a guard rail to the LLM, ensuring the output is in a JSON format. The sample step queries the LLM to select objects from the list and the output is then sent to a guard rail for extraction. The number of objects requested from the list, n_s in our study, was always 3. As the output was also an ordered list, the position of the selected objects was also recorded, meaning the final output of a given LLM list sampling was a $n_s \times 3$ matrix.

$$\phi(\mathbb{x}_t) \rightarrow \mathbb{x}_s = \begin{bmatrix} a, 1, 1 \\ c, 4, 2 \\ d, 2, 3 \\ ... \\ \ell, p_t, p_u \end{bmatrix}$$

Where p_u is a position in the output list, p_t is a position in the input list, and ℓ is an object. In order to compute the probability of an object being selected we summed the number of times it was selected and divided it by the total number of possible selections that could have occurred, $\ell_i \in \mathbb{X}_t$. Similarly we computed the probability of a position being selected by summing the number of times that position was selected and dividing it by the total number of possible selections that could have occurred, $p_{t,i} \in \mathbb{X}_t$.

$$P(p_{t,i} \in \mathbb{x}_s) = \frac{\#\ p_{t,i} \in \mathbb{X}_s}{\#\ p_{t,i} \in \mathbb{X}_t}$$

$$P(\ell_i \in \mathbb{x}_s) = \frac{\#\ \ell_i \in \mathbb{X}_s}{\#\ \ell_i \in \mathbb{X}_t}$$

The outputs were then processed to extract relevant features. First, we tested for primacy bias; whether the outputs were of the first three positions, returned in the exact same order. Primacy can be expressed as:

$$\%\text{Primacy} = \frac{\#\mathbb{p}_t \subset \mathbb{x}_s : \mathbb{p}_t = \{1, 2, 3\}}{N}$$

Next, they were tested for correspondence hallucinations; whether the output objects were present in the input list. We can define this correspondence mathematically as:

$$\%\text{Correspondence} = \frac{\#\,\mathbb{l}_s \subset \mathbb{l}_t}{N}$$

Finally, they were examined for adherence to instructions; whether the exact number of objects specified were selected from the input list (Correct Count). We can define this adherence mathematically as:

$$\%\text{Correct count} = \frac{\#n_{s,target} = n_s}{N}$$

We are assuming that the selections of objects and positions are independent of each other. In the case that they are not, we can more weakly assume the random distribution of the objects and positions should produce an average probability. In order to compute the probability and error robustly we performed bootstrapping using 3000 samples with replacement.

While this tells us information about the probability of an object or position being selected it does not tell us about the probability of an object being selected given a position. To compute the joint probability $P(\ell_i \cap p_i, t)$, we count the number of times an object was selected from a position and divide it by the total number of times that position and object occurred concurrently.

$$P(\ell_i \cap p_i, t) = \frac{\#\ (\ell_i, p_{i,t}) \in \mathbb{x}_s}{\#\ (\ell_i, p_{i,t}) \in \mathbb{x}_t}$$

Mutual information, a measure of how much information the identity of one variable provides on another variable, was computed using the following equation:

$$I(L_s; P_t) = \sum_{\ell \in L_s} \sum_{p_t \in P_t} P_{(L_s, P_t)}(\ell, p) \log \left(\frac{P_{(L_s, P_t)}(\ell, p)}{P_{L_s}(\ell) P_{P_t}(p)} \right)$$

Where L_s is the distribution of inputs in the selected list, and P_t is the distribution of positions from the input list that were selected. In this case the mutual information describes how knowledge of an input being selected gives us certainty on which position it was in.

3 Results

3.1 Primacy, Hallucination and Adherence

Primacy bias, for the purpose of this paper, is defined as the bias for a LLM to select the first three objects in a list regardless of their identity. Correspondence refers to the ability of the LLM to return only objects in the input list, a measure of how common hallucination was. Correct count refers to the ability of the LLM to return the correct number of objects from the list, a measure of the LLM's ability to adhere to instructions.

For all conditions gpt-3.5-turbo showed a significantly increased primacy bias compared to claude-instant-1.2 (Figure 3). This trend was consistent across all temperatures and list lengths. Primacy bias with temperature is correlated negatively for gpt-3.5-turbo and positively for claude-instant-1.2. Both showed similar levels of correspondence and correct counts, and gpt-3.5-turbo showed a slightly lower accuracy in both cases. When altering the object type of a list, numbers vs letters, the usage of numbers appears to reduce the primacy bias in most cases while simultaneously reducing instruction following behavior, correct count, for gpt-3.5-turbo (Figure 10). Only in a couple isolated conditions for claude-instant-1.2 (Figure 11) did the list length and expected probability of primacy correspond with each other (Table 1).

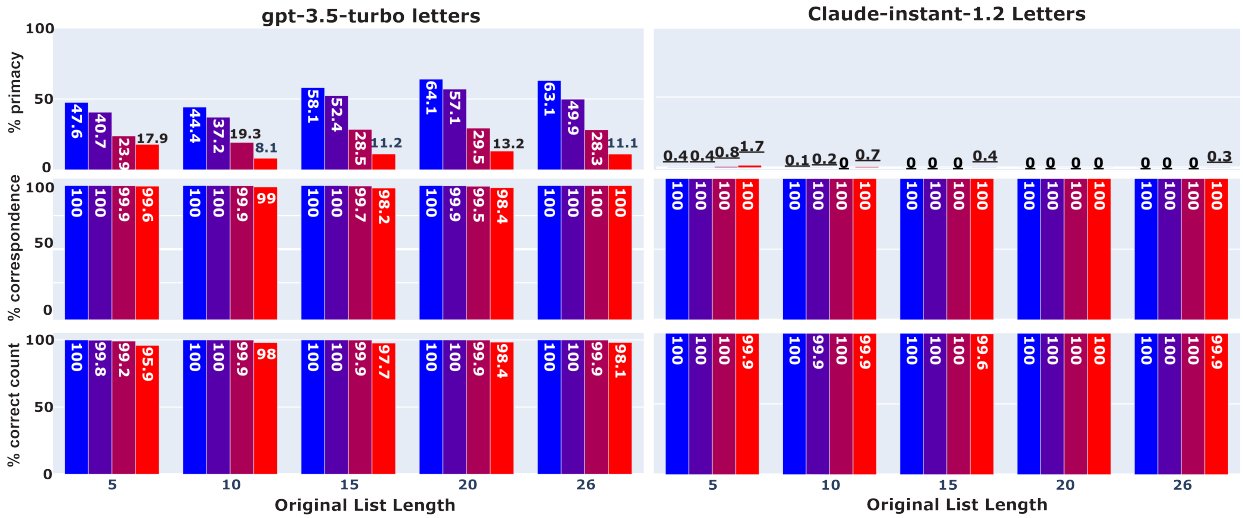


Figure 3: The percent of responses displaying primacy bias, correspondence, and correct count The temperature is denoted by color for gpt-3.5-turbo and claude-instant-1.2 respectively from blue (0, -1), purple (0.5, 0), dark red (1, 0.5) and red (1.5, 1).

Table 1: Given a uniform distribution of selecting three objects from a list the chance that you will return the first three objects in order.

List Length	Probability of Primacy
5	1.7%
10	0.14%
15	0.034%
20	0.015%
26	0.0064%

List length did seem to have some minor effects on primacy bias, such as at high temperatures the minima were between a list length of $(10 - 15)$ for gpt-3.5-turbo (Figure 3). Which was observed in both the number and letter condition (Figure 10), although low list lengths were associated with a reduced primacy bias at low temperatures.

3.2 Positional Selections

The probability that a position will be selected was calculated with respect to a variety of temperatures, list lengths, models, and temperatures. The probability was then split by primacy, so we could detect differences in the selections. Claude-instant-1.2 had negligible primacy bias and as such saw little by way of modulation.

As primacy is a feature defined by position, the first three positions were equally affected by the proportions of the results. In the case of list length 5, the results were restorative at all temperatures to a seemingly linear decrease in probability that a position will be selected. In this case, temperature largely affected the steepness of the linear trend, with $T : 0$ resulting in a probability of 0.05 for position 5 and $T : 1.5$ resulting in a probability of 0.15 (Figure 4).

As the list length increases, the non-primacy probability of a positional selection forms a basin where the start and end of a list are more likely to be selected. Even without primacy bias the most frequently selected position tends to be the first, second, or third. Increasing temperature tends to minimize this effect and increases the probability of the last few positions being selected as well. This effect was specific to gpt-3.5-turbo, with claude-instant-1.2 showing only a return to the expected uniform distribution as a function of temperature (Figure 5).

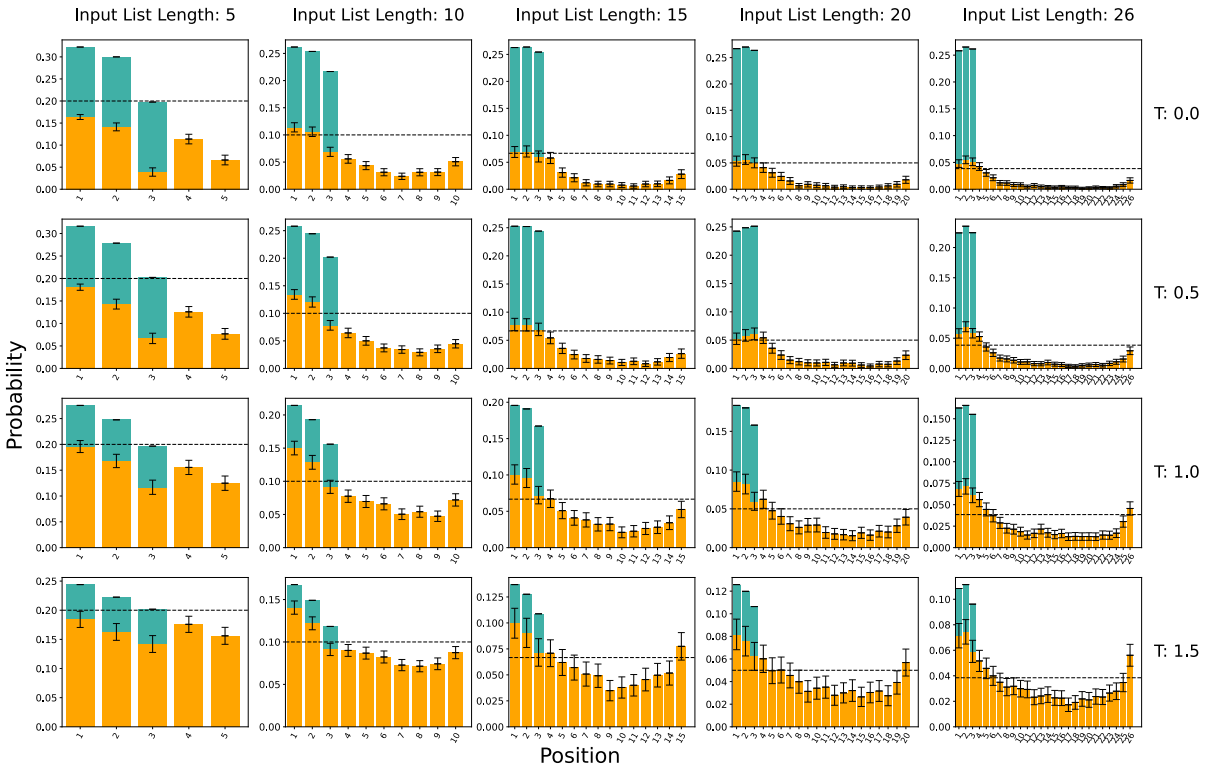


Figure 4: Using gpt-3.5-turbo, the scaled probability that a position of a letter will be selected given a temperature and original list length. The dotted black line is the expected probability given random sampling of a uniform distribution for a list length. The orange bars are the probability that a position without primacy bias will be selected, while blue represents the probability that a position with primacy bias will be selected. Error bars are the standard error from 3000 bootstrap replicates.

Claude-1.2-instant had a significant bias against the first position at low list lengths, with a compensatory gain towards the third position. The relative bias toward the third position was retained, and even magnified as the input list length increased (Figure 5). There was also a significant bias against the last position in the list that appears insensitive to the temperature, while having a complex relationship with list length.

While in the case of gpt-3.5-turbo the probability of a position being selected was qualitatively invariant of object type (letter vs. number), claude-instant-1.2 showed significant modulations in profile (Figure 14). The bias against the first position was higher for numbers than letters, and never reached the expected uniform distribution. The bias for the third position, while present, was weaker. Overall, the numbers appear to have more specific positional preferences that were invariant to temperature.

3.3 Input Object Selection

The probability of an input being selected given that it appeared within a list was computed. Primacy was considered separately from non-primacy in order to clarify their effect magnitudes on selection. The probability of a given input being selected was then compared to the expected probability of a uniform distribution.

Primacy played a significant role in contributing to the total probability for gpt-3.5-turbo, but showed few sensible effects. Several letters showed consistently low probability of being selected via primacy such as the letter I and the letter Z (Figure 6). What was interesting was the variation in the non-primacy selection probabilities. There was significant bias toward or away from certain letters that produced distributions that grew increasingly non-uniform as the list length increased. Temperature had a positive correlation with returning toward the uniform distribution, but was unable to fully restore it within our study parameters. These effects were qualitatively similar in the case of number based inputs, but of a smaller magnitude (Figure 16).

Claude-instant-1.2 was extremely biased when it came to object selection. At low list lengths the uniformity of the sampling was relatively high, but showed that several letters were highly selected against (Figure 7). As the list length

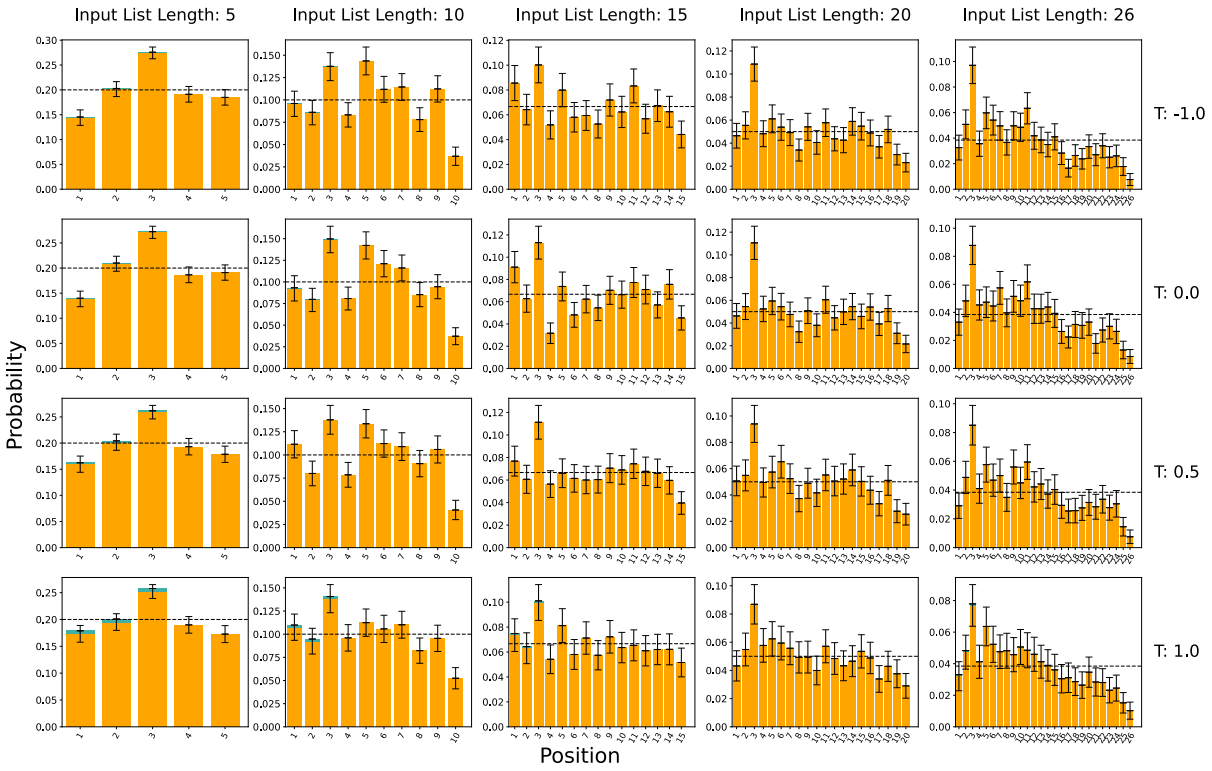


Figure 5: Using claude-instant-1.2, the scaled probability that a position of a letter will be selected given a temperature and original list length. The dotted black line is the expected probability given random sampling of a uniform distribution for a list length. The orange bars are the probability that a position without primacy bias will be selected, while blue represents the probability that a position with primacy bias will be selected. Error bars are the standard error from 3000 bootstrap replicates.

increased, the bias against the letters increased in magnitude until several objects showed effectively zero probability of being selected. Temperature modulation alleviated the bias, but could not restore the uniformity of the distribution. When considering numbers, the bias only grew worse. At low list lengths the probability distribution resembles a unimodal symmetric form where the first and last positions are selected against (Figure 17). As the list length increases, multimodality appears and multiple high probability peaks and valleys occur where certain numbers are heavily selected for or against, similarly to the letter input selection probabilities.

3.4 Mutual Information

Mutual information allows for the quantification of how two random variables are linked. If there is a high amount of mutual information, knowledge of one variable will give significant information on another.

In the case of our work, we computed the mutual information of a selected object's identity and position on the input list. The higher the mutual information, the more we learn about the possible position of a chosen object if we learn the object's identity. Claude-instant-1.2 showed significantly lower mutual information overall when compared to gpt-3.5-turbo (Figure 9). Importantly, the mutual information for gpt-3.5-turbo was negatively correlated with temperature.

Mutual information for gpt-3.5-turbo shows a clear temperature correlation at all list lengths (Figure 8). A negative correlation between list length and mutual information is observed for gpt-3.5-turbo, especially at low temperatures. Claude-instant-1.2 has a positive correlation between the maximum of served mutual information and list length. While there is a weak negative correlation between list length and mutual information (Figure 9), the effect is only on letters that do not display mutual information between input identity and position. Therefore, while we do not observe temperature dependence at low list lengths, as list length increases, several letters gain a temperature dependence for mutual information. These effects are invariant to the object's class (letter or number), with only the specific identity of the object changing the magnitude of the mutual information (Figure 18).

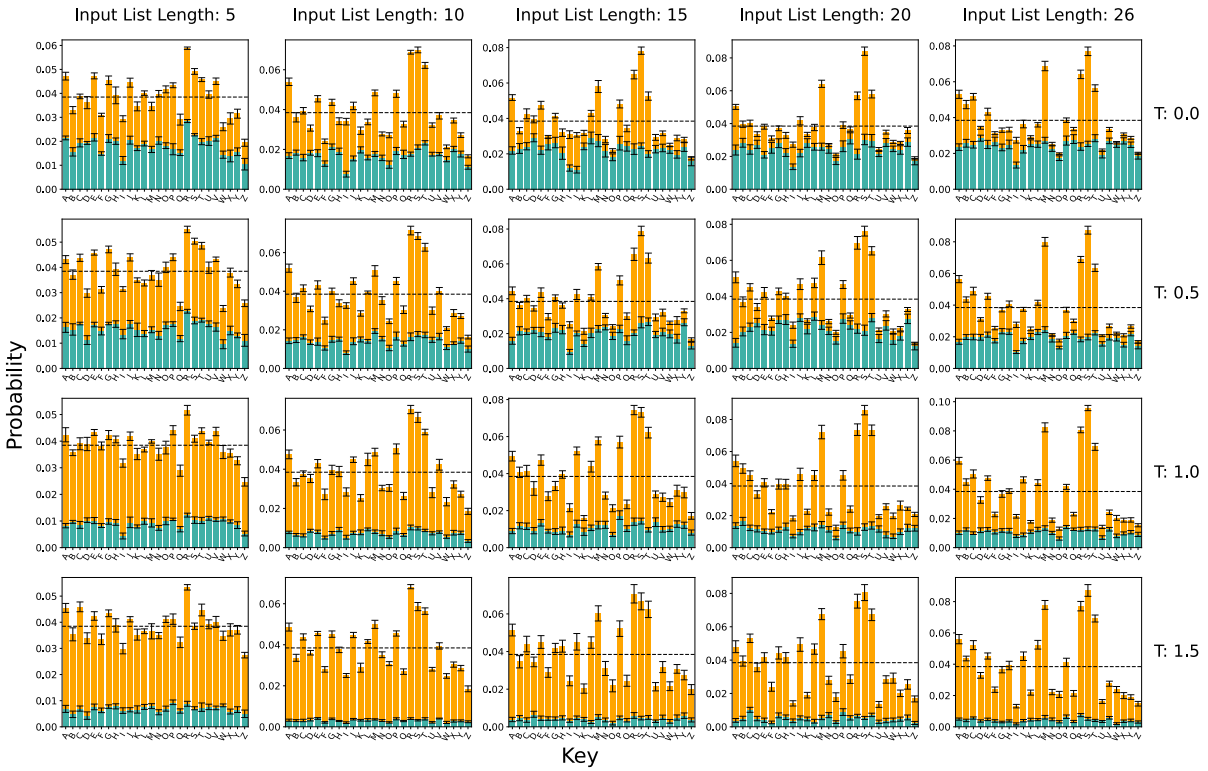


Figure 6: Using gpt-3.5-turbo, the scaled probability that a letter will be selected given a temperature and original list length. The dotted black line is the expected probability given random sampling of a uniform distribution for a list length. The orange bars are the probability that a position without primacy bias will be selected, while blue represents the probability that a position with primacy bias will be selected. Error bars are the standard error from 3000 bootstrap replicates.

3.5 Guard Rails With and Without a Sample Step

Guard rails are methods to ensure a LLM returns output in a requested format, which is useful for quantifying the results of our study. When sampling for experimental purposes we used a sample step between the sampling query and the guard rail. To verify our methodology we included a direct guard rail method for study, where we do not separate the two steps (Figure 1).

The largest change affected by a sample step is a significant reduction in the primacy bias (Figure 2), with all conditions displaying a minimum of a 37% reduction in primacy bias up to 81%. The direct method helped ensure correspondence between the input and output list, although this only proved to be a minor problem at higher temperatures and intermediate list lengths for the sample step. Following instructions by responding with the correct number of objects from the list was greatly increased across the board via the introduction of a sample step. These effects were seen similarly in letter selection, with slight differences in effect magnitude (Figure 12).

Claude-instant-1.2 was more robust against the application of guard rails (Figure 13), with the largest change being in primacy, as the direct primacy values were similar to gpt-3.5-turbo despite claude-1.2-turbo having significantly lower incidence of primacy initially. There is a weaker effect on the correct count metric for numbers, with high list lengths fully ablating the effect of the direct guard rail. For letters the effect on correct count was negligible.

When considering the mutual information, we see that the direct application of guard rails causes a large spike for all studied conditions (Figure 21); to be expected given the high primacy of direct guard rail applications reducing the number of possible positions (Figure 2). The effect of temperature on mutual information is reduced for claude-instant-1.2 as compared to gpt-3.5-turbo, with an extremely chaotic profile for input identities (Figure 19). The effect of list length on mutual information is consistently negative, with larger list lengths having lower mutual information across models and temperatures (Figure 20). That said, there does appear to be a lower limit to the mutual information, dependent on the object identity and model.

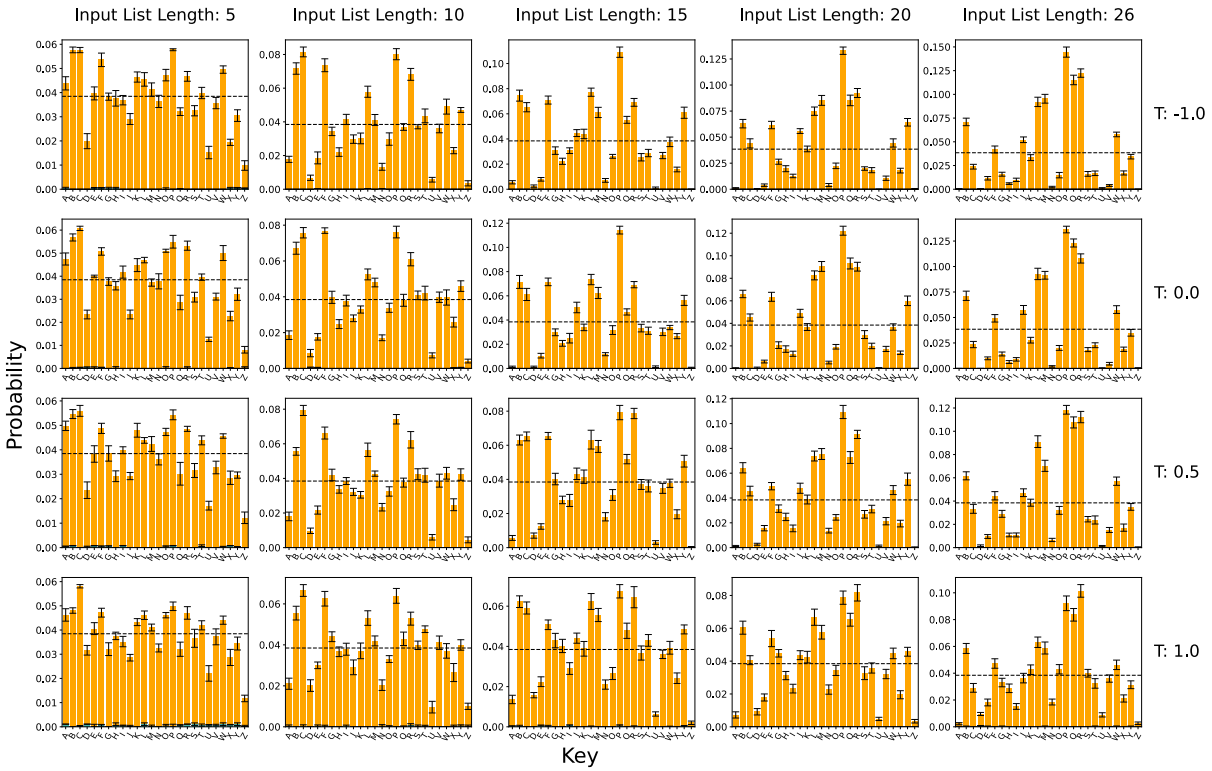


Figure 7: Using claude-instant-1.2, the scaled probability that a letter will be selected given a temperature and original list length. The dotted black line is the expected probability given random sampling of a uniform distribution for a list length. The orange bars are the probability that a position without primacy bias will be selected, while blue represents the probability that a position with primacy bias will be selected. Error bars are the standard error from 3000 bootstrap replicates.

4 Discussion

4.1 Comparisons Between Models

For the task of list selection, claude-instant-1.2 outperforms gpt-3.5-turbo in terms of low bias (Figure 3) and robustness under the application of a guard rail (Figure 13). The mutual information between position and object identity highlights this disparity, where, except at high temperatures, claude-instant-1.2 has lower mutual information across the board (Figure 8).

While there is little public information about the internal structure of the two models, our results highlight an important consideration: even similarly performing models may have wildly divergent bias profiles. For example, gpt-3.5-turbo shows a positional bias structure, Figure 4, similar to the that reported by Wang et al. (2023), while the positional bias structure of claude-instant-1.2 is inverted, with the central positions being the most common at low list lengths. This is a trend that, as list length increases, morphs into a favoring of the third position and a disfavoring of the first two and last positions. This result is at odds with the predicted behavior derived from Liu et al. (2023), which would point to a favoring of information in the start and end of a context. When dealing with bias, generalizing behavior is risky and may not be appropriate even for similar tasks and models.

While it would be easy to state that claude-instant-1.2 is the superior model, this is not necessarily the case. While claude-instant-1.2 has significantly lowered incidence of primacy bias, it has its own bias which is relatively insensitive to temperature (Figure 14). Moreover there are a number of conditions where certain objects are not selected at all. Furthermore there are a variety of biases we may not be appropriately detecting. One such candidate would be a bias towards the selection of the average value of a given list of numbers (Figure 17). Unfortunately we did not study this in enough detail to comment extensively, but this is indicative of emergent bias patterns that blindly trusting any given LLM may blindside us with.

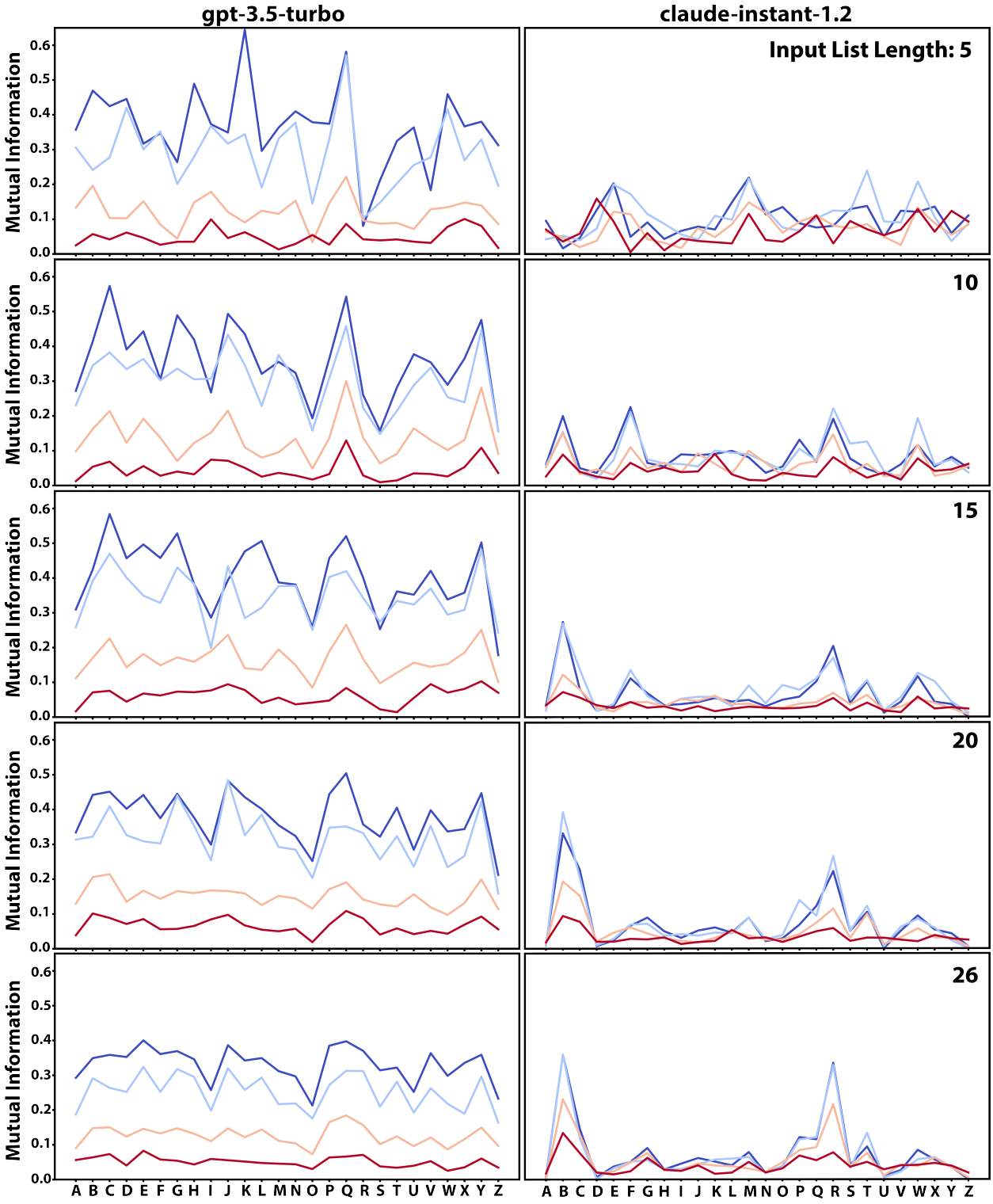


Figure 8: Mutual information between input letter and position. The temperature is denoted by color for gpt-3.5-turbo and claude-instant-1.2, respectively being dark blue $(0, -1)$, light blue $(0.5, 0)$, light red $(1, 0.5)$ and dark red $(1.5, 1)$. Each row represents a different initial list length from 5-26 letters.

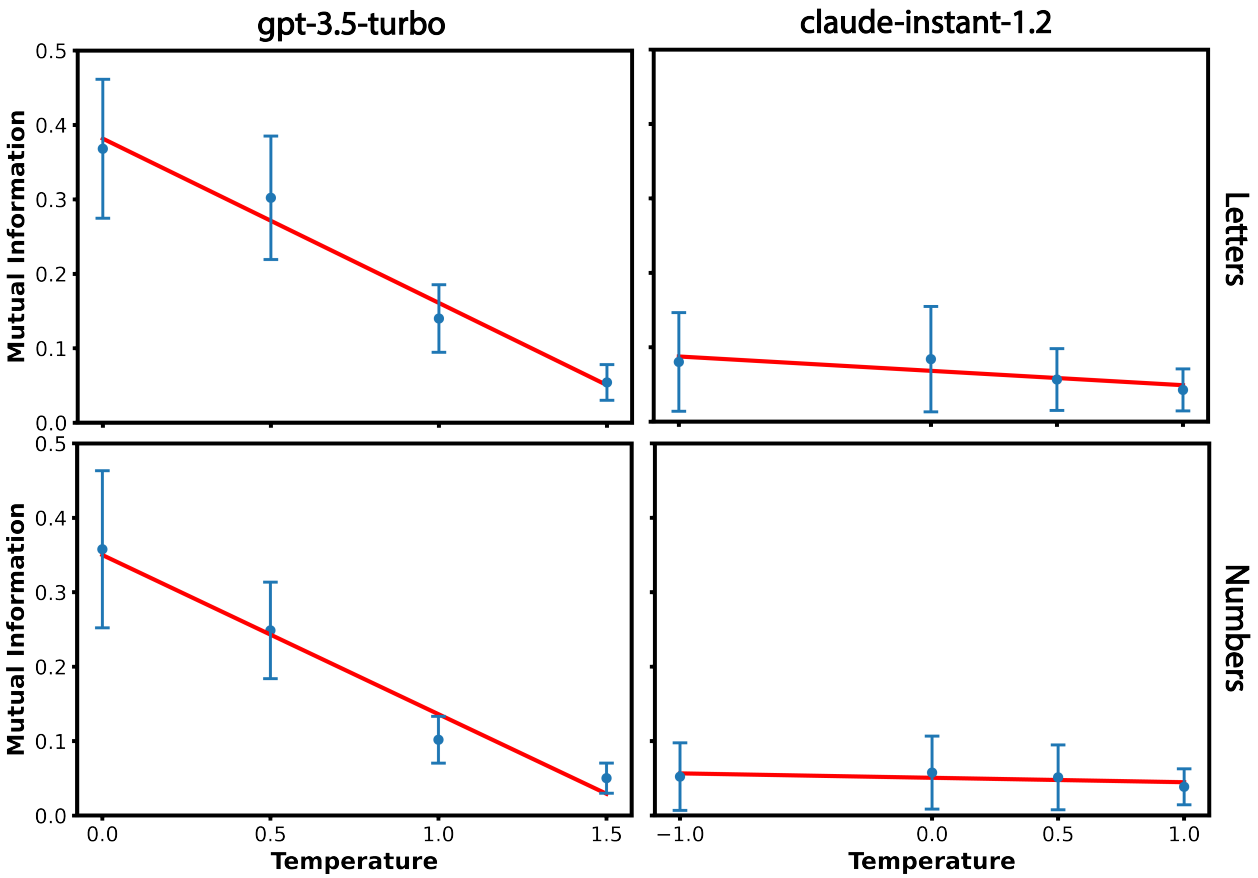


Figure 9: Average mutual information between an input and position with standard deviations. Averages were computed over all list lengths and inputs of a given type. A linear regression was computed for letter inputs (A), (gpt-3.5-turbo, $R^2 = 0.971$ and claude-instant-1.2, $R^2 = 0.703$), and number inputs (B), (gpt-3.5-turbo, $R^2 = 0.975$ and claude-instant-1.2, $R^2 = 0.394$).

4.2 Guard Rails and Cognitive Load

Guard rails are potent methods for assuring the output of a LLM complies with user defined parameters (Rebedea et al. (2023)). While this technology constitutes a great tool for many situations, the effect of these guard rails on LLM decision-making has not been fully examined (Shankar et al. (2024), Abdelkader et al. (2024)). The introduction of a guard rail resulted in a model agnostic gain in primacy bias (Figure 12, Figure 13), while decreasing the correct count, a metric of instruction-following behavior. Without guard rails, only gpt-3.5-turbo, while selecting number objects, showed any issues with instruction-following (Figure 10). When guard rails are applied, both letters and numbers experienced a similar magnitude increase in primacy and failure to follow instructions. There is model specificity to guard rail induced bias, as while there was still some effect on instruction adherence, it was qualitatively different from claude-instant-1.2 (Figure 13). Furthermore, the direct guard rail had a significant effect on the mutual information between position and object identity for gpt-3.5-turbo and claude-instant-1.2 both (Figure 19, Figure 20), with the averages for both models being significantly higher in the direct guard rail application than the two-step method (Figure 21).

Asking a human to select three objects from a list and write them down is a trivial task. Asking them to format the response in a JSON, a task guard rails have been developed to tackle in LLMs, will result in a much more significant expenditure of effort. To handle the extra cognitive load, humans will compensate by allowing bias to reduce the effort (de Jong (2010)). In the same way, LLMs follow similar trends by increasing primacy bias (Figure 2) and the mutual information between objects and positions (Figure 8, Figure 19). The relative semantic complexity of the task may indicate that LLMs experience a form of cognitive load that causes a compensatory raise in bias when a guard rail is applied. This would align with a novel prompt injection technique termed “cognitive overload” (Xu et al. (2023)). In our study, the LLM compensates by increasing the primacy bias and reducing instruction adherence. This compensation implies that the use of any sort of structural assurance in output for a LLM will introduce some form

of compensation. While in our case this involved the adherence to instruction and uncreative list selection, the exact nature may vary greatly depending on the model and task just as a human might (Paas and van Merriënboer (2020)). Therefore the distribution of outputs of a cognitively taxing structure for a LLM should be empirically analyzed given our current theoretical grounding.

The proposed theory of LLM cognition implies an interesting future direction of research, where the cognitive load of a LLM is measured in order to determine the appropriate compensation for a given task. If done properly models for predicting the effect of cognitive load can be made, allowing for rapid deployment of complex structures in LLMs.

5 Limitations and Future Directions

The scope of this study focused exclusively on the patterns inherent in selecting naive numbers and letters from lists. Conceptually this is identical to the usage of more complex objects such as words, phrases, or randomized tokens. In common use these effects may show great mitigation or enhancement due to the semantic content of the objects being selected. This connection between the semantic content of the objects and the selection patterns is a proposed area of future research. Future research should delve into how LLMs bias more meaningful use cases, such as real-world agentic systems and code generation.

We only selected three objects from the list at a time, other works should expand on this to see if the patterns hold for larger or smaller selection pools. We also noted some patterns in the objects that were selected together, this is an area of future research and should be expanded to help understand LLM list construction tasks.

The prompt we used was a simple request to "Please select 3 of the following:"; our goal was to minimize prompt bias while still retaining the spirit of directed object selection. There are numerous options for prompt construction such as including "randomly" or even attempting to instruct the LLM to select things in an unbiased manner. Although from our preliminary work, we believe this will just lead to new probability distributions rather than ablating them fully.

Of note, this is not accounting for the possibility that the presence of other objects in a list may affect the probability of an object being selected: $P(\ell_i \in \aleph_s | \ell_j \in \aleph_s) \neq P(\ell_i \in \aleph_s)$. Looking into how the presence of other objects in the list affect the selection probabilities will allow for better list debiasing methods.

The models used are closed source, using open source models such as Pythia would allow for close inspection of bias evolution as a function of training and size (Biderman et al. (2023)). This would also allow us to more accurately analyze the modulation in bias as the functionality of a model changes to see if phase transitions in ability are associated with bias modulation (Chang (2023)).

6 Conclusion

The application of LLMs requires a careful eye to the prompt structure and implementation. If, for example, one were to construct a list of actions a LLM agent could undertake in response to some contextual data, the appropriate generated answer would be in a tug-of-war with the object-position bias (Figure 23). Ideally, the model used will have a low mutual information between the position and object, such as in claude-instant-1.2, allowing for agnostic object order. That is not always the case (Figure 9), and needs to be tested empirically for the use case at hand, especially due to the object identities and hyperparameters creating wildly different mutual information (Figure 21). A variety of methods could be employed to handle this, the simplest being to sample the positional bias and object bias of the LLM, and optimizing the order such that it approaches the uniform distribution when no context is provided. Such a method would also be amenable to the use of a thesaurus to retain semantic meaning of actions while minimizing the bias from selection.

When implementing a LLM, one must consider what tools and structures are being imposed on the output and how that will bias the final outputs (Figure 2). While the intrinsic bias for a model may be low, complex prompt structures will strongly alter the probability distribution of the output. In our study we were able to circumvent this by inserting a sample step between the selection task and the JSON structuring task. We propose a model of cognitive load to interpret this result, with the two-step method reducing the total cognitive load of our initial prompt, which fosters creativity and reduces bias while taking advantage of the functionality of a guard rail for our work. Our work was significantly sped up via the application of a guard rail over attempting to perform arbitrary text matching. We achieved a promising result for the use of guard rails in LLMs, but more work is needed to understand the cognitive load of guard rails and other prompt structures.

The application of LLMs requires a careful eye to the prompt structure and implementation. If, for example, one were to construct a list of actions a LLM agent could undertake in response to some contextual data, the appropriate generated answer would be in a tug-of-war with the object-position bias (Figure 23). Ideally, the model used will have a low mutual information between the position and object, such as in claude-instant-1.2, allowing for agnostic object

order. That is not always the case (Figure 9), and needs to be tested empirically for the use case at hand. Especially due the prompting methodology creating wildly different mutual information (Figure 21). A variety of methods could be employed to handle this, the simplest being to sample the positional bias and object bias of the LLM, and optimizing the order such that it approaches the uniform distribution when no context is provided. Such a method would also be amenable to the use of a thesaurus to retain semantic meaning of actions while minimizing the bias from selection.

When implementing a LLM, one must consider what tools and structures are being imposed on the output and how that will bias the final outputs (Figure 2). While the intrinsic bias for a model may be low, complex prompt structures will strongly alter the probability distribution of the output. In our study we were able to circumvent this by inserting a spacing step between the selection task and the structuring task. We propose a model of cognitive load to interpret this result, with the two-step method reducing the total cognitive load of our initial prompt, which fosters creativity and reduces bias while taking advantage of the functionality of a guard rail for our work. We have achieved a promising result for the use of guard rails in LLMs, but more work is needed to understand the cognitive load of guard rails and other prompt structures.

7 Competing Interests

Rafael F. Irgolič has contributed to Guardrails AI's open-source project, and has been involved in contractual work with Guardrails AI. The other authors declare that they have no competing interests.

8 Authors' Contributions

J.E. forumlated the problem, designed the experiments, analyzed the data, and wrote the manuscript. R.F.I. designed and implemented the experiments. All authors read, edited, and approved the final manuscript.

References

Abdelkader, Hala, Mohamed Abdelrazek, Scott Barnett, Jean-Guy Schneider, Priya Rani, and Rajesh Vasa. 2024. "ML-On-Rails: Safeguarding Machine Learning Models in Software Systems A Case Study." arXiv. https://doi.org/10.48550/arXiv.2401.06513.

Allred, Sarah R., L. Elizabeth Crawford, Sean Duffy, and John Smith. 2016. "Working Memory and Spatial Judgments: Cognitive Load Increases the Central Tendency Bias." *Psychon Bull Rev* 23 (6): 1825–31. https://doi.org/10.3758/s13423-016-1039-0.

Balch, William R. 1989. "Item Order Affects Performance on Multiple-Choice Exams." *Teaching of Psychology* 16 (2): 75–77. https://doi.org/10.1207/s15328023top1602_9.

Biderman, Stella, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, et al. 2023. "Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling." In *Proceedings of the 40th International Conference on Machine Learning*, 2397–2430. PMLR.

Chang, Cheng-Shang. 2023. "A Simple Explanation for the Phase Transition in Large Language Models with List Decoding." arXiv. https://doi.org/10.48550/arXiv.2303.13112.

de Jong, Ton. 2010. "Cognitive Load Theory, Educational Research, and Instructional Design: Some Food for Thought." *Instr Sci* 38 (2): 105–34. https://doi.org/10.1007/s11251-009-9110-0.

"Guardrails AI | Your Enterprise AI Needs Guardrails." n.d. https://guardrailsai.com/docs/. Accessed January 28, 2024.

Han, Ridong, Tao Peng, Chaohao Yang, Benyou Wang, Lu Liu, and Xiang Wan. 2023. "Is Information Extraction Solved by ChatGPT? An Analysis of Performance, Evaluation Criteria, Robustness and Errors." arXiv. https://doi.org/10.48550/arXiv.2305.14450.

Kinniment, Megan, Lucas Jun Koba Sato, Haoxing Du, Brian Goodrich, Max Hasin, Lawrence Chan, Luke Harold Miles, et al. 2024. "Evaluating Language-Model Agents on Realistic Autonomous Tasks." arXiv. https://doi.org/10.48550/arXiv.2312.11671.

Kirk, Hannah Rose, Yennie Jun, Filippo Volpin, Haider Iqbal, Elias Benussi, Frederic Dreyer, Aleksandar Shtedritski, and Yuki Asano. 2021. "Bias Out-of-the-Box: An Empirical Analysis of Intersectional Occupational Biases in Popular Generative Language Models." In *Advances in Neural Information Processing Systems*, 34:2611–24. Curran Associates, Inc.

Li, Cong. 2010. "Primacy Effect or Recency Effect? A Long-Term Memory Test of Super Bowl Commercials." *Journal of Consumer Behaviour* 9 (1): 32–44. https://doi.org/10.1002/cb.291.

Liang, Paul Pu, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. "Towards Understanding and Mitigating Social Biases in Language Models." In *Proceedings of the 38th International Conference on Machine Learning*, 6565–76. PMLR.

Liu, Nelson F., Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. "Lost in the Middle: How Language Models Use Long Contexts." arXiv. https://doi.org/10.48550/arXiv.2307.03172.

Nauts, Sanne, Oliver Langner, Inge Huijsmans, Roos Vonk, and Daniël H. J. Wigboldus. 2014. "Forming Impressions of Personality." *Social Psychology* 45 (3): 153–63. https://doi.org/10.1027/1864-9335/a000179.

Paas, Fred, and Jeroen J. G. van Merriënboer. 2020. "Cognitive-Load Theory: Methods to Manage Working Memory Load in the Learning of Complex Tasks." *Curr Dir Psychol Sci* 29 (4): 394–98. https://doi.org/10.1177/0963721420922183.

Rebedea, Traian, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. "NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails." arXiv. https://doi.org/10.48550/arXiv.2310.10501.

Shankar, Shreya, Haotian Li, Parth Asawa, Madelon Hulsebos, Yiming Lin, J. D. Zamfirescu-Pereira, Harrison Chase, Will Fu-Hinthorn, Aditya G. Parameswaran, and Eugene Wu. 2024. "SPADE: Synthesizing Assertions for Large Language Model Pipelines." arXiv. https://doi.org/10.48550/arXiv.2401.03038.

Touileb, Samia, Lilja Øvrelid, and Erik Velldal. 2022. "Occupational Biases in Norwegian and Multilingual Language Models." In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, 200–211. Seattle, Washington: Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.gebnlp-1.21.

Wang, Yiwei, Yujun Cai, Muhao Chen, Yuxuan Liang, and Bryan Hooi. 2023. "Primacy Effect of ChatGPT." arXiv. https://arxiv.org/abs/2310.13206.

Weidinger, Laura, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, et al. 2022. "Taxonomy of Risks Posed by Language Models." In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 214–29. FAccT '22. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3531146.3533088.

Wolfe, Robert, and Aylin Caliskan. 2021. "Low Frequency Names Exhibit Bias and Overfitting in Contextualizing Language Models." arXiv. https://doi.org/10.48550/arXiv.2110.00672.

Xu, Nan, Fei Wang, Ben Zhou, Bang Zheng Li, Chaowei Xiao, and Muhao Chen. 2023. "Cognitive Overload: Jailbreaking Large Language Models with Overloaded Logical Thinking." arXiv. https://doi.org/10.48550/arXiv.2311.09827.

Zheng, Chujie, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2023. "Large Language Models Are Not Robust Multiple Choice Selectors." arXiv. https://arxiv.org/abs/2309.03882.

9 Supplemental Information

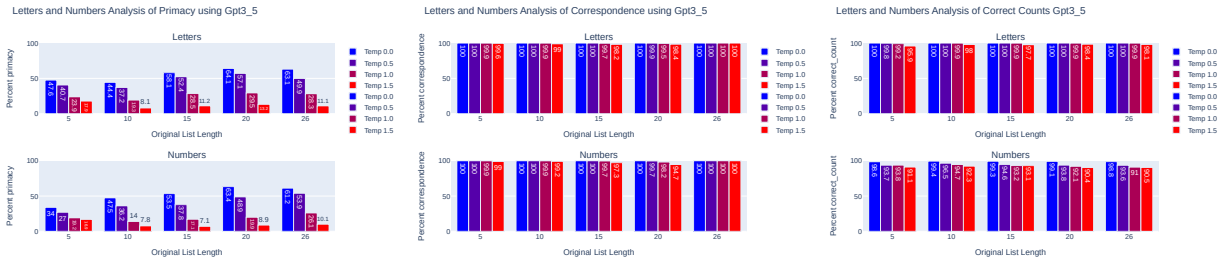


Figure 10: The full results of the gpt-3.5-turbo model with a spacing step marked on primacy, correspondence, and correct counts. This is compared for all list lengths analyzed and all temperatures.

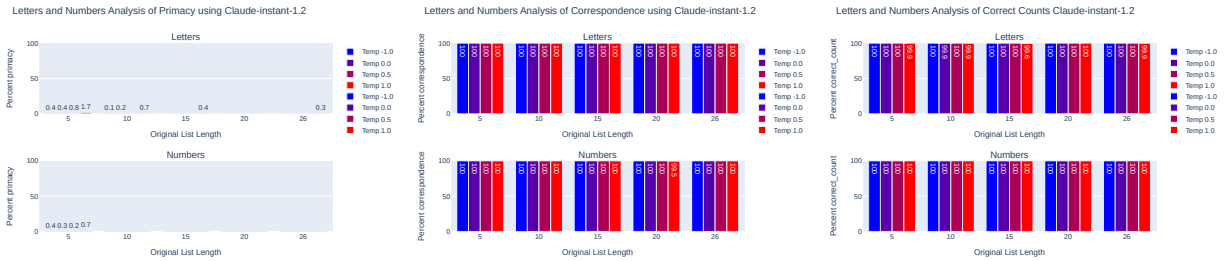


Figure 11: The full results of the claude model with a spacing step marked on primacy, correspondence, and correct counts. This is compared for all list lengths analyzed and all temperatures. For primacy, all empty values are 0 as there was no occurance of primacy.

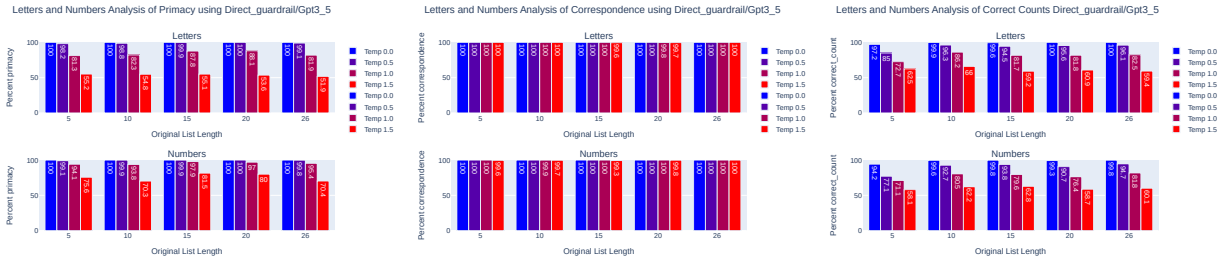


Figure 12: The full results of the gpt-3.5-turbo model without a spacing step marked on primacy, correspondence, and correct counts. This is compared for all list lengths analyzed and all temperatures.

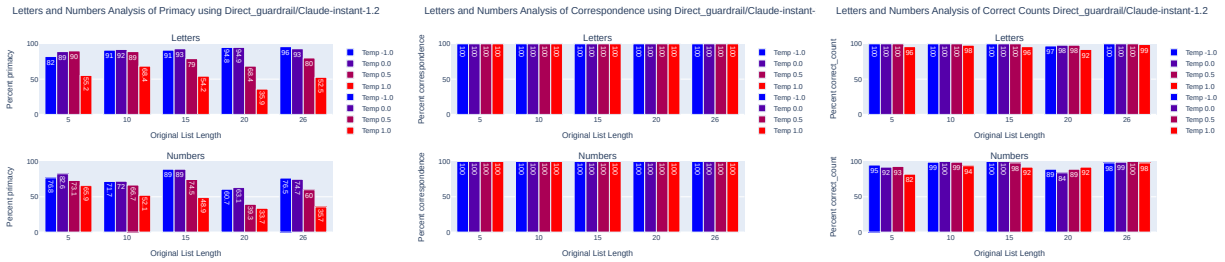


Figure 13: The full results of the claude model without a spacing step marked on primacy, correspondence, and correct counts. This is compared for all list lengths analyzed and all temperatures. For primacy, all empty values are 0 as there was no occurrence of primacy.

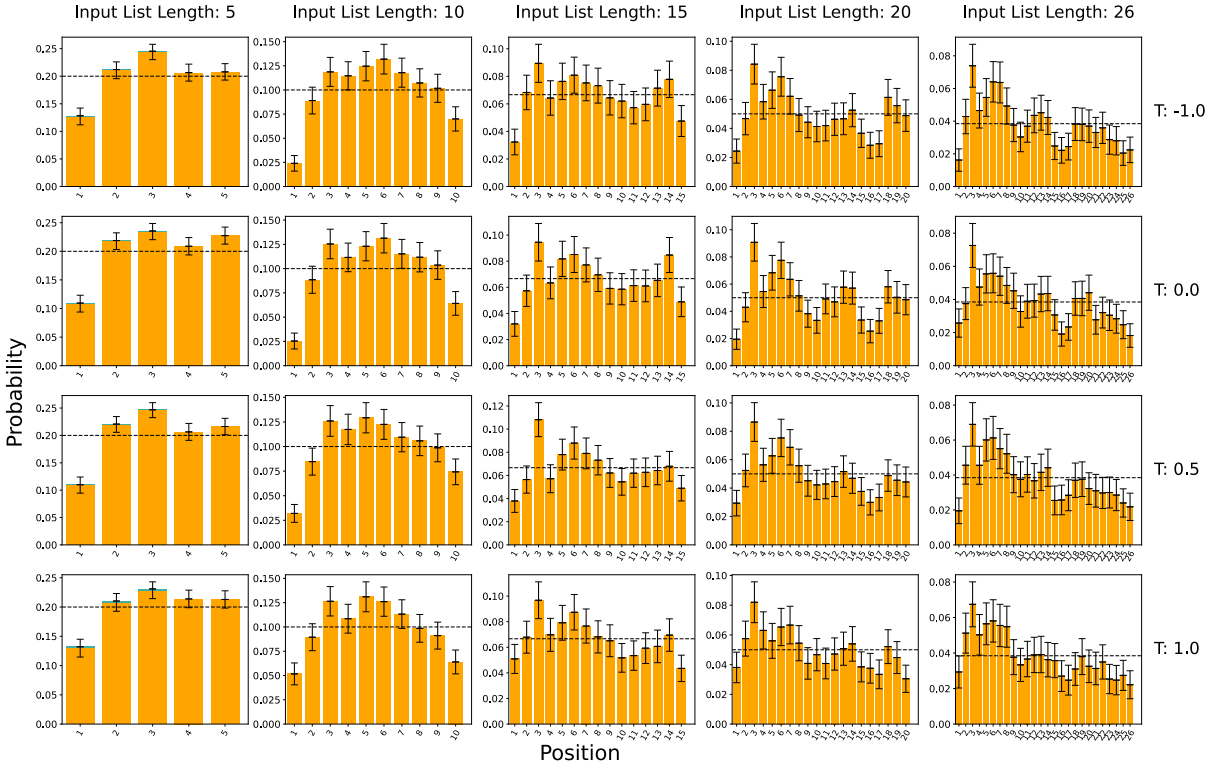


Figure 14: Using claude-instant-1.2 the scaled probability that a position of a number will be selected given a temperature and original list length was computed. The dotted black line is the expected probability given random sampling of a uniform distribution for a list length. Orange bars are the probability that a position without primacy bias will be selected, while blue represents the probability that a value with primacy bias will be selected. Error bars are standard error from 3000 bootstrap replicates.

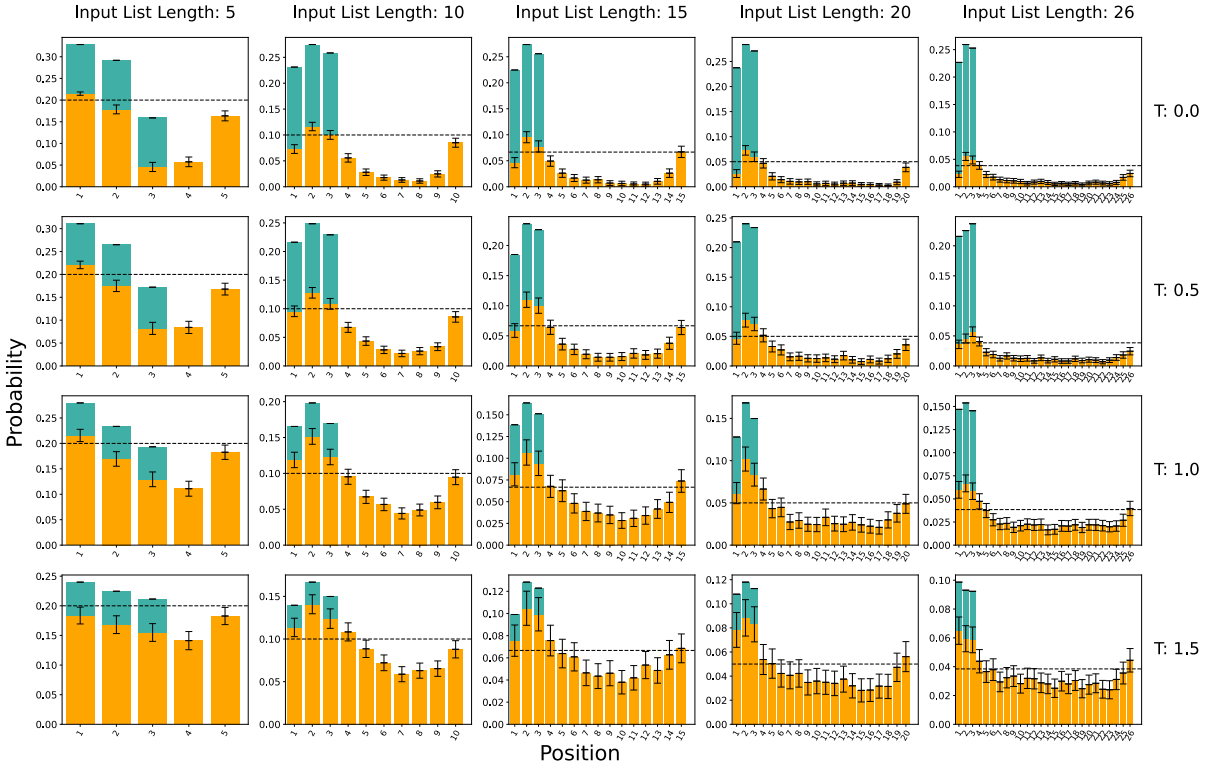


Figure 15: Using gpt-3.5-turbo the scaled probability that a position of a number will be selected given a temperature and original list length was computed. The dotted black line is the expected probability given random sampling of a uniform distribution for a list length. Orange bars are the probability that a position without primacy bias will be selected, while blue represents the probability that a value with primacy bias will be selected. Error bars are standard error from 3000 bootstrap replicates.

Listing 1 Code to construct initial prompt for data collection

```
def construct_prompt(choices: list[str], choice_count: int):
    prompt = f"""Please select {choice_count} of the following:"""
    for i, choice in enumerate(choices):
        prompt += f"\n- {choice}"

    return prompt
```

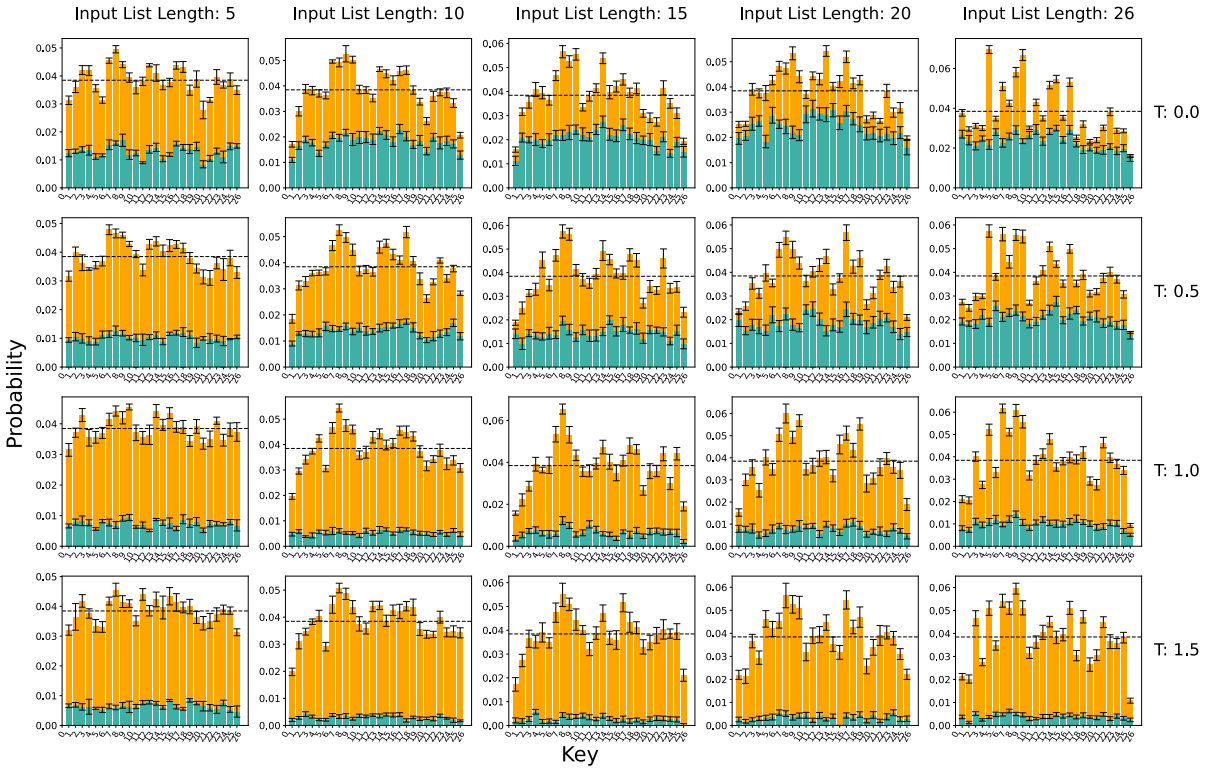


Figure 16: Using gpt-3.5-turbo the scaled probability that a number will be selected given a temperature and original list length. The dotted black line is the expected probability given random sampling of a uniform distribution for a list length. Orange bars are the probability that a position without primacy bias will be selected, while blue represents the probability that a value with primacy bias will be selected. Error bars are standard error from 3000 bootstrap replicates.

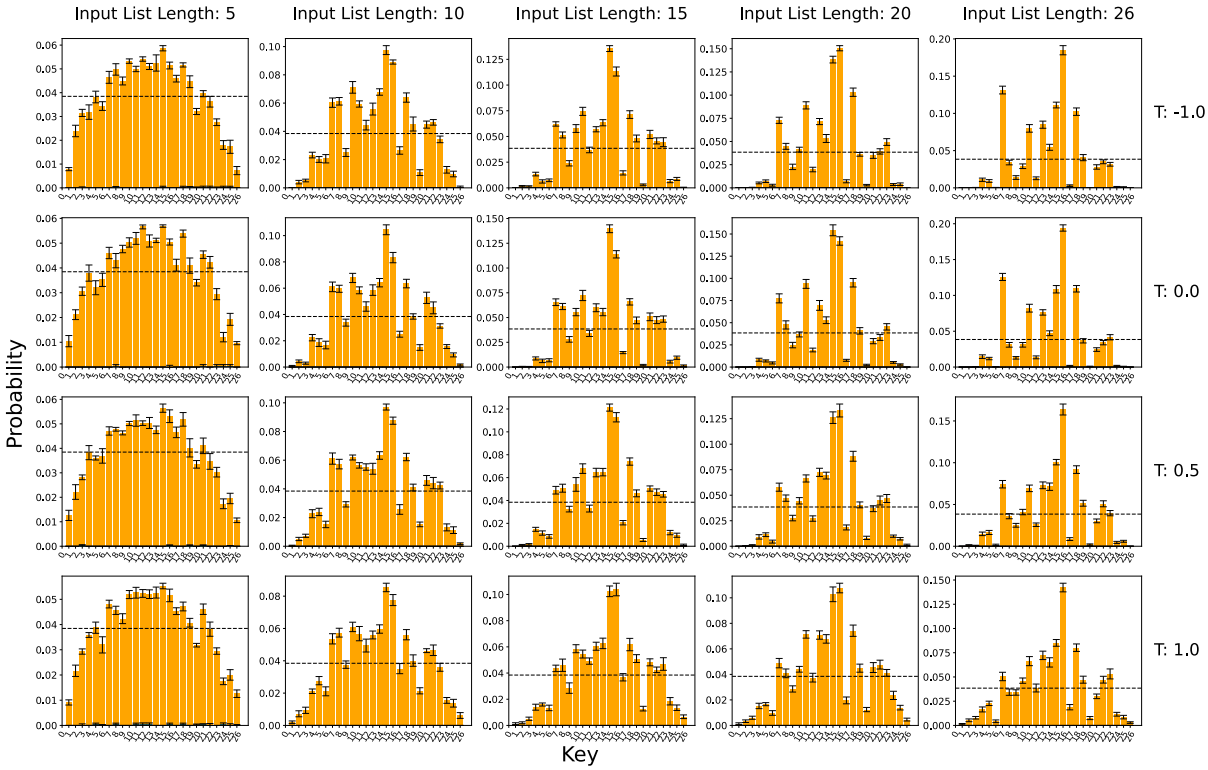
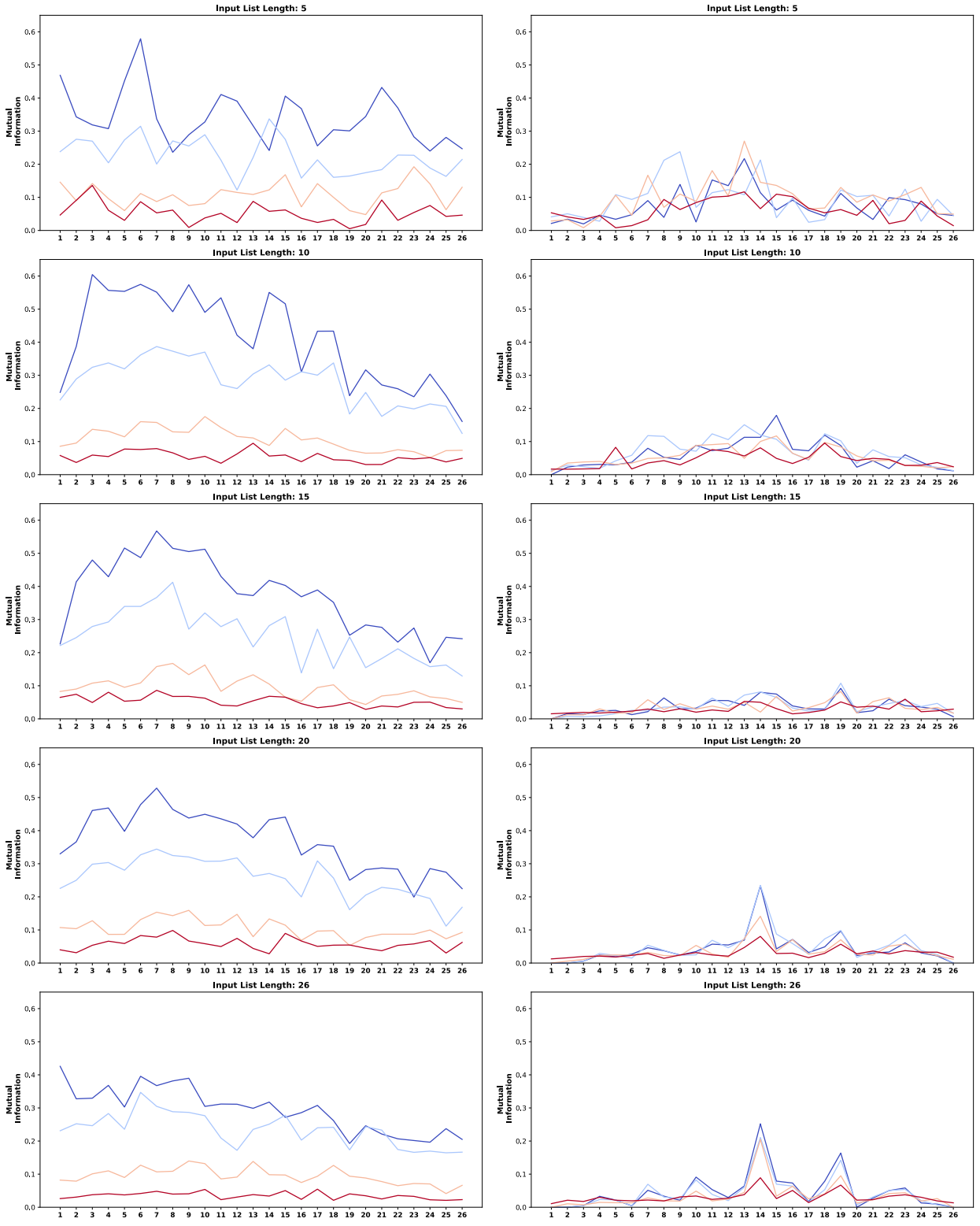


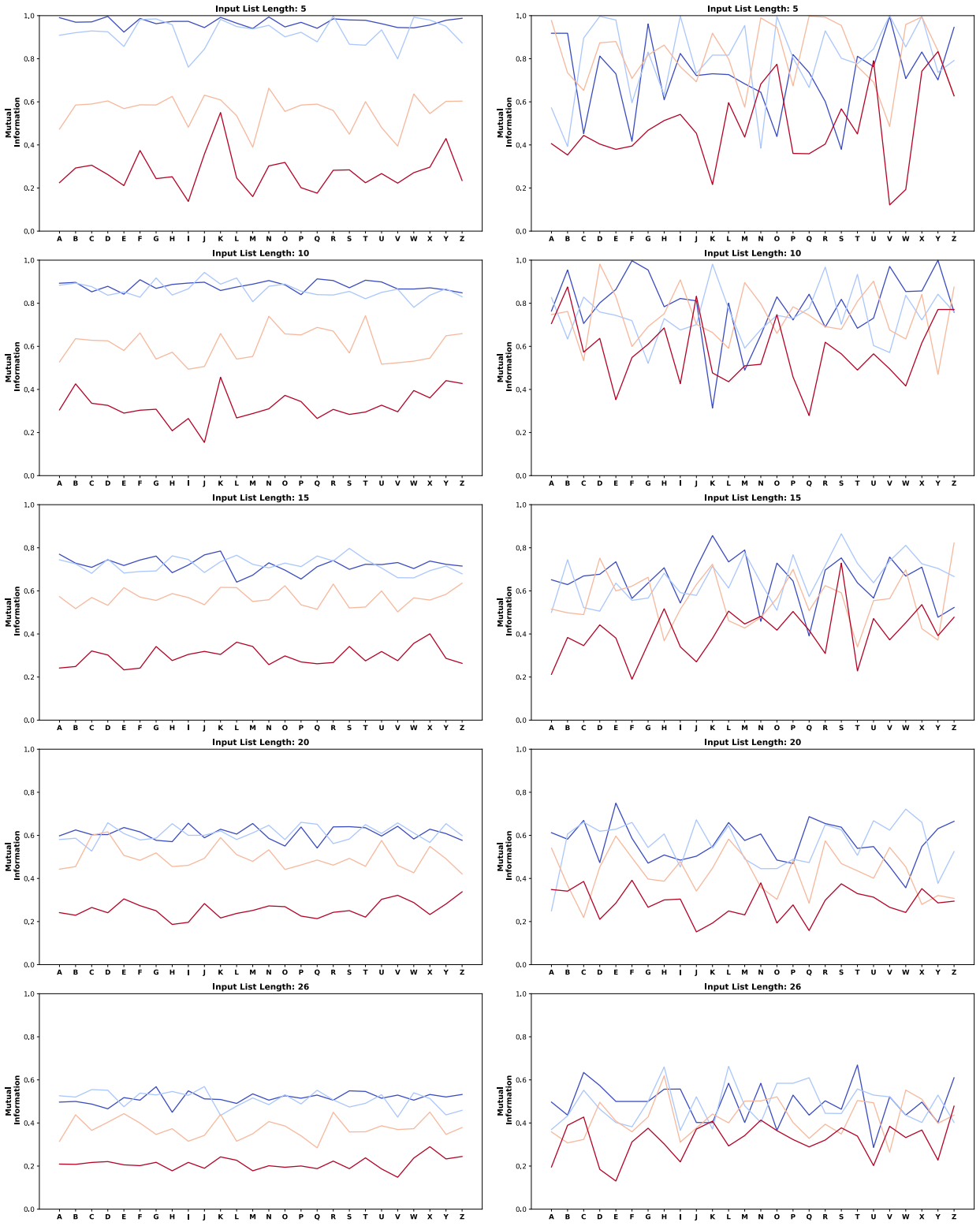
Figure 17: Using claude-instant-1.2 the scaled probability that a number will be selected given a temperature and original list length. The dotted black line is the expected probability given random sampling of a uniform distribution for a list length. Orange bars are the probability that a position without primacy bias will be selected, while blue represents the probability that a value with primacy bias will be selected. Error bars are standard error from 3000 bootstrap replicates.



(a) gpt-3.5-instant

(b) claude-instant-1.2

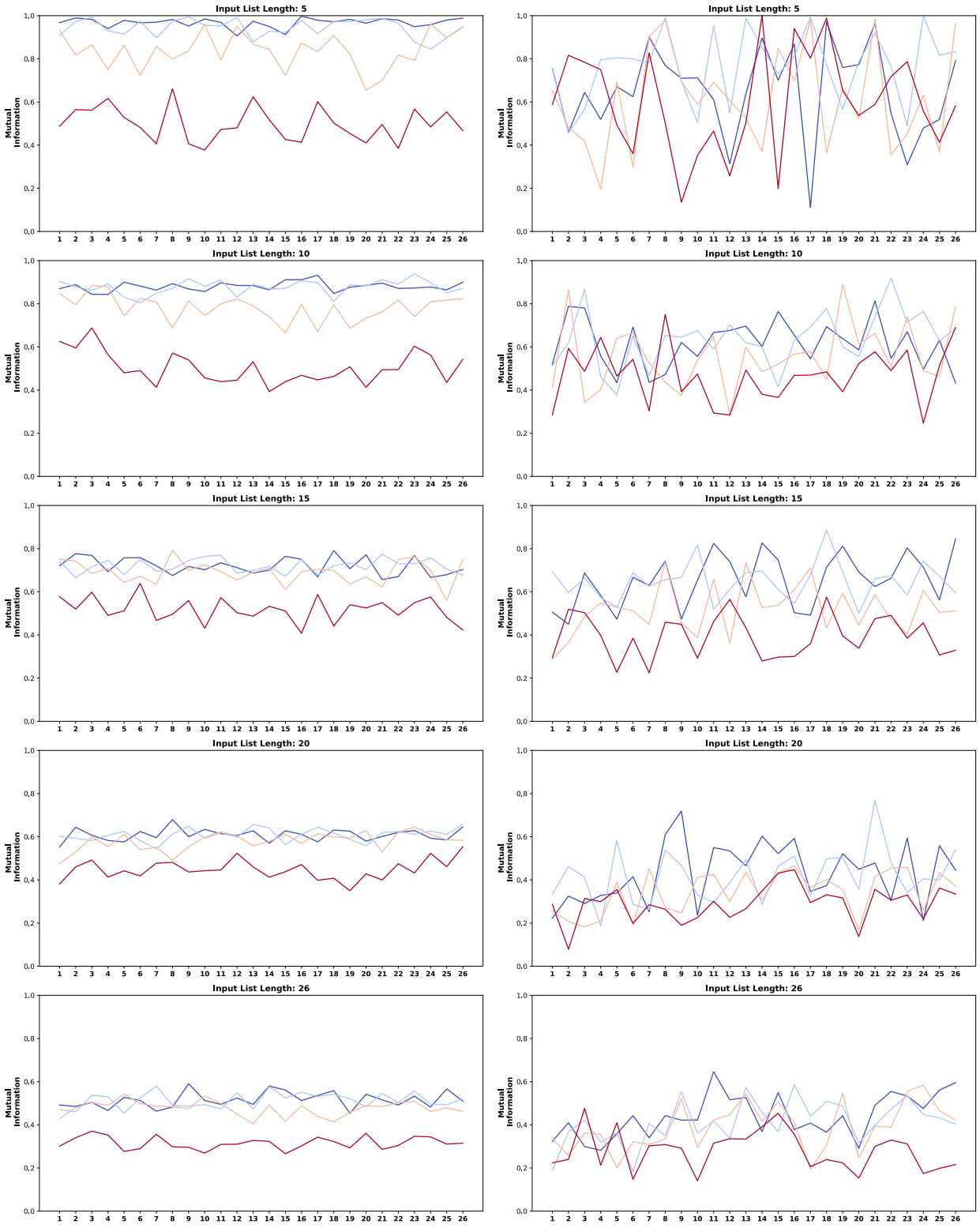
Figure 18: The mutual information between the input and output of the gpt-3.5-turbo and claude-instant-1.2 models. The mutual information is computed for each number in the list and then averaged across all numbers.



(a) gpt-3.5-instant

(b) claude-instant-1.2

Figure 19: The mutual information between the input and output of the direct guard rails method for gpt-3.5-turbo and claude-instant-1.2 models. The mutual information is computed for each letter in the list and then averaged across all letters.



(a) gpt-3.5-instant

(b) claude-instant-1.2

Figure 20: The mutual information between the input and output of the direct guard rails method for gpt-3.5-turbo and claude-instant-1.2 models. The mutual information is computed for each number in the list and then averaged across all numbers.

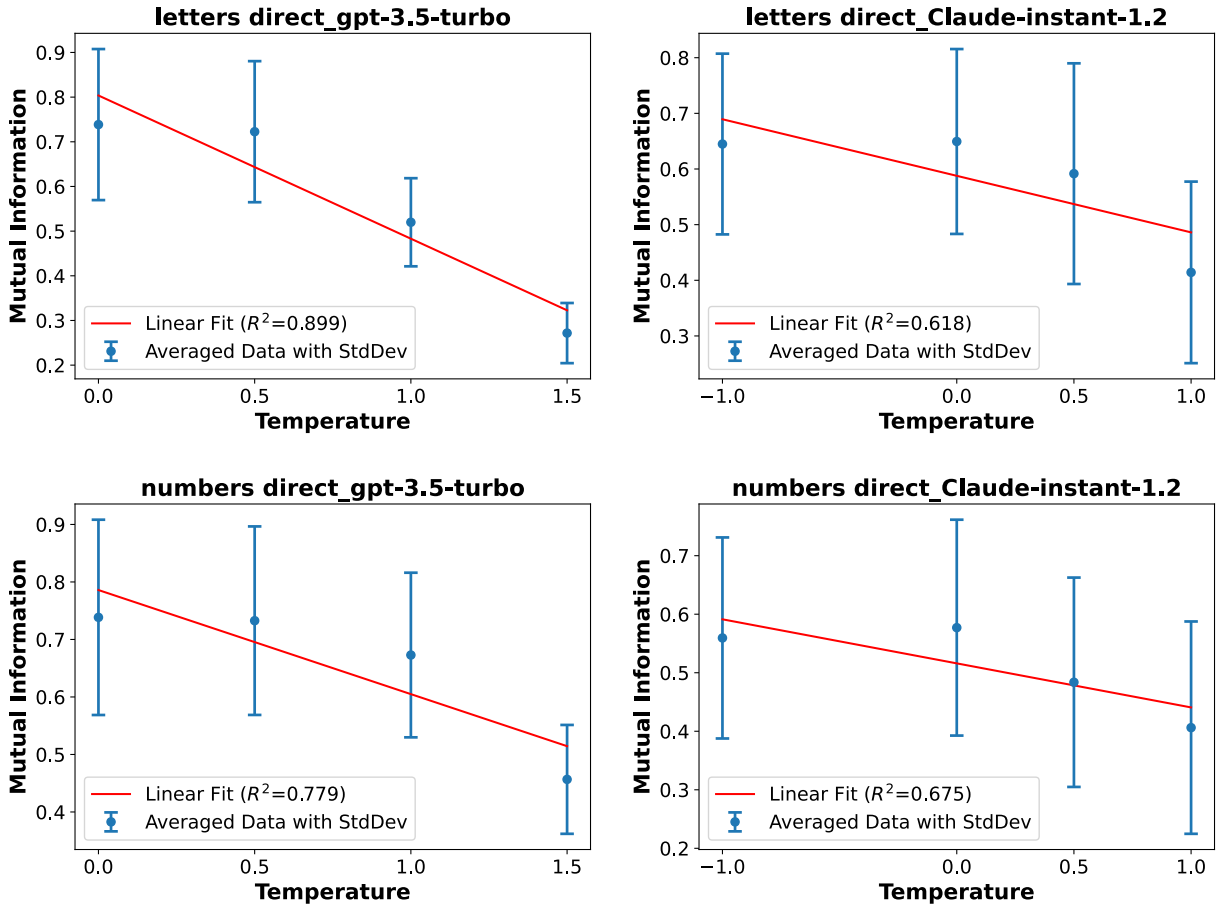
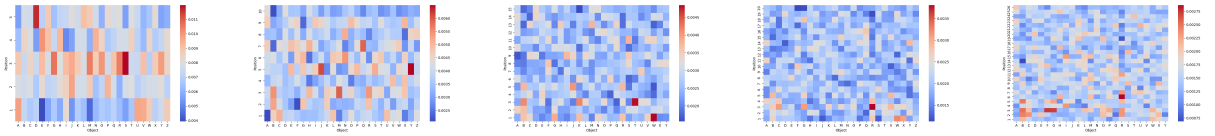
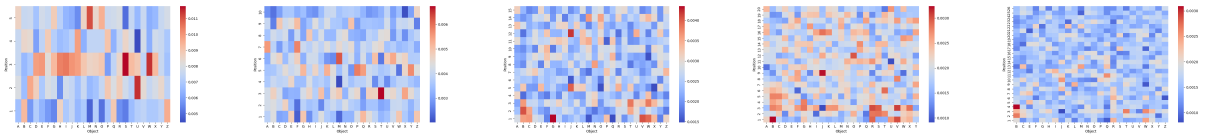


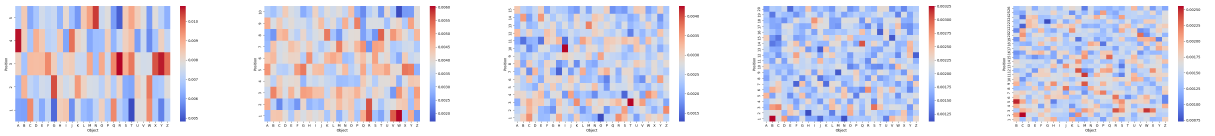
Figure 21: Average mutual information between an input and position with standard deviations. Averages were computed over all list lengths and inputs of a given type. A linear regression was computed for letter inputs (gpt-3.5-turbo $R^2 = 0.899$, Claude-instant-1.2 $R^2 = 0.618$) and number inputs (gpt-3.5-turbo $R^2 = 0.779$, Claude-instant-1.2 $R^2 = 0.675$).



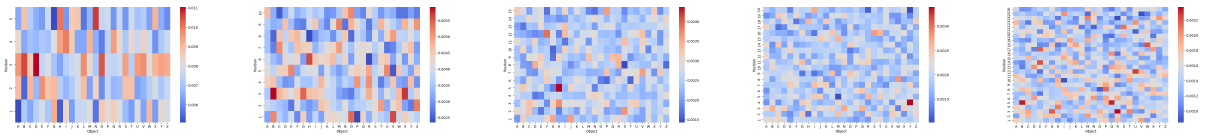
(a) T = -1, Count = 5 (b) T = -1, Count = 10 (c) T = -1, Count = 15 (d) T = -1, Count = 20 (e) T = -1, Count = 26



(f) T = 0, Count = 5 (g) T = 0, Count = 10 (h) T = 0, Count = 15 (i) T = 0, Count = 20 (j) T = 0, Count = 26

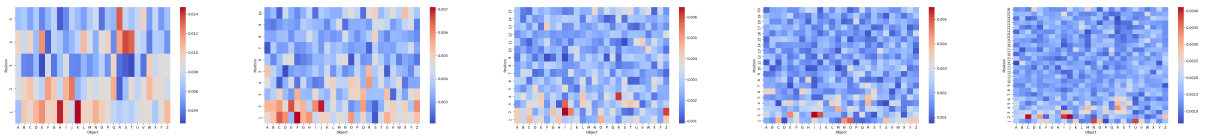


(k) T = 0.5, Count = 5 (l) T = 0.5, Count = 10 (m) T = 0.5, Count = 15 (n) T = 0.5, Count = 20 (o) T = 0.5, Count = 26

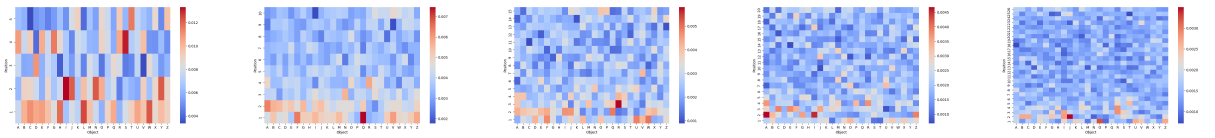


(p) T = 1, Count = 5 (q) T = 1, Count = 10 (r) T = 1, Count = 15 (s) T = 1, Count = 20 (t) T = 1, Count = 26

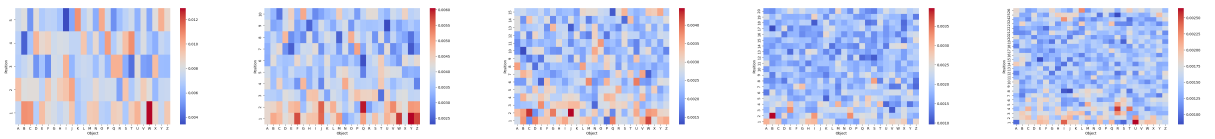
Figure 22: The joint probability of claude-instant-1.2 for letters and positions



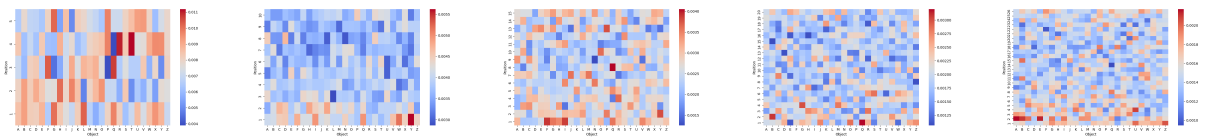
(a) T = 0, Count = 5 (b) T = 0, Count = 10 (c) T = 0, Count = 15 (d) T = 0, Count = 20 (e) T = 0, Count = 26



(f) T = 0.5, Count = 5 (g) T = 0.5, Count = 10 (h) T = 0.5, Count = 15 (i) T = 0.5, Count = 20 (j) T = 0.5, Count = 26



(k) T = 1, Count = 5 (l) T = 1, Count = 10 (m) T = 1, Count = 15 (n) T = 1, Count = 20 (o) T = 1, Count = 26



(p) T = 1.5, Count = 5 (q) T = 1.5, Count = 10 (r) T = 1.5, Count = 15 (s) T = 1.5, Count = 20 (t) T = 1.5, Count = 26

Figure 23: The joint probability of gpt-3.5-turbo for letters and positions

Listing 2 Code to perform spacing step data collection via LLM call, here {prompt} refers to the constructed prompt from Listing 1, while {initial_response} refers to the result of the first prompt.

```
<rail version="0.1">

<output>
<list
    name="choices"
>
</list>
</output>

<instructions>
You are a helpful assistant only capable of communicating with valid JSON,
and no other text.

${{gr.json_suffix_prompt_examples}}
</instructions>

<prompt>
+++
{initial_response}
+++

${{gr.xml_prefix_prompt}}

${{output_schema}}

Your returned value should be a dictionary with a single "choices" key,
whose value contains a list of values chosen in the above response enclosed in +++.

</prompt>


</rail>
```

Listing 3 Code to perform direct data collection via LLM call, here {prompt} refers to the constructed prompt from Listing 1

```
<rail version="0.1">

<output>
<list
    name="choices"
    format="length: {choice_count} {choice_count}"
    on-fail-format="noop"
>
<choice>
{"".join(f'''<case name="{choice}">
</case>''' for choice in choices)}
</choice>
</list>
</output>

<instructions>
You are a helpful assistant only capable of communicating with valid JSON,
and no other text.

${{gr.json_suffix_prompt_examples}}
</instructions>

<prompt>
{prompt}

${{gr.xml_prefix_prompt}}

${{output_schema}}

</prompt>

</rail>
```