The Newest Arbiter on the Block: Exploring LLMs as Evaluators through Transitivity

Rhea Acharya

December 2024

> "Justice consists not in being neutral between right and wrong, but in finding out the right and upholding it, wherever found, against the wrong."
>
> —Theodore Roosevelt, *Fear God and Take Your Own Part*

1 Introduction and Motivation

1.1 Agenda

This writeup aims to document the motivation for undertaking this research project, my results and work so far, and my future plans and questions to explore. It will feature my work completed as of Dec. 10, 2024, under Prof. Cynthia Dwork and Dr. Reid McIlroy-Young. I will first start out by outlining the theoretical foundations and motivations of this work, before transitioning into the structure and results of the applied experiments. I have gotten results on experiments conducted for concrete and abstract comparisons of numbers, difficulty of vocabulary words, and persuasiveness of Reddit posts, with more datasets in my ongoing and future work.

1.2 Motivation

AI is being increasingly used in evaluative processes. We see new startups like Mercor (AI hiring platform) and CoGrader (AI grading platform) disrupting the scene and producing new ways to evaluate candidates and student work. We also see existing companies like Linkedin or Canvas trying to incorporate AI workflows into its current offerings. This means that LLMs might be involved in the evaluation process for these high-stakes decisions like parsing resumes for hiring, or offering formative feedback to students while grading. New technology helps advance society, so it is important to embrace ways to streamline processes and improve efficiency, but we must be cognizant of the other implications of these decisions.

Can we trust LLMs to make fair, consistent decisions for us? These circumstances are high-stakes, yet often ambiguous, making it harder but all the most important for the evaluations to be well-motivated and aligned with human judgment and values. We hope in this case to ensure outcome indistinguishability, where we specifically cannot distinguish between our LLM-computed results and the real world ground truth values of what humans should / would prefer.

Many researchers have spent time thinking about how to judge LLM responses in ambiguous settings [1], how to calibrate models for intersectional fairness [3], and how to develop a better sense of a model's decision making frameworks through interpretability [5]8. In this research, I aim to focus on how model responses adhere to transitivity, especially for pairwise comparisons, using transitivity as a proxy for consistency.

1.3 High Level Introduction

In her 2020 paper, Christina Ilvento rigorously defined Ilvento's Algorithm [4], [2], a metric learning approach designed to produce fair and consistent orderings of elements within a dataset. The algorithm, rooted in notions of fairness, aims to mitigate biases that arise during the evaluation and ranking processes. Such concerns have become increasingly relevant as machine learning models, particularly large language models (LLMs), are deployed in domains where fairness is a critical value—such as hiring, grant proposal evaluations, resource allocation, and grading evaluations. As part of the process, Ilvento's Algorithm requires pairwise comparisons of some of the elements in order to create a distance-based ranking of the preferences.

A key aspect of fairness in these contexts is transitivity, a fundamental property of preference ordering. Transitivity ensures that if an evaluator prefers A over B and B over C, it should also prefer A over C. When transitivity is violated, it reveals inconsistencies in the evaluator's criteria or decision-making process, undermining the reliability and fairness of the system. For instance, consider a scenario where LLMs are used as arbiters in grant proposal evaluations. In such settings, the timing or context of an application submission should not influence its evaluation. If transitivity violations occur, it suggests that the LLM's evaluation framework is inconsistent, leading to unfair or arbitrary outcomes—a critical issue when decisions carry significant consequences for individuals or organizations.

LLMs have the potential to replace human arbiters as oracles in systems like these, particularly in large-scale evaluations requiring objectivity and efficiency. By serving as oracles, LLMs can streamline the evaluation process by generating consistent and unbiased rankings of candidates, proposals, or other elements. However, ensuring that LLMs respect fairness principles, including transitivity, is a prerequisite for their effective use in these roles. Without transitivity, the evaluation criteria become undefined, leading to unreliable decision-making that undermines the fairness these tools are meant to promote.

2 Theoretical Background

To begin, we introduce some background and related work that undergirds our algorithms and empirical analysis.

2.1 Ilvento's Algorithm

Before we discuss our LLM-minded extensions, let us define Ilvento's Algorithm in its original form. This algorithm, which aims to help create fair distance-based mapping when the metric is ambiguous or hard to compute at once across the entire universe, revolves around the human arbiter. This human arbiter's role is to provide consistent and unbiased pairwise judgments of similarity within a given task. These judgments form the foundation for constructing a task-specific similarity metric $D(u,v)$, essential for achieving Individual Fairness, which requires treating similar individuals similarly. Through relative distance queries (e.g., $O_{\text{TRIPLET}}(a,b,c)$), the arbiter helps establish approximate submetrics such as $D_r(u,v) = |D(r,u) - D(r,v)|$, which contract the original metric while maintaining fairness guarantees. Transitivity is a critical assumption, as it ensures logical consistency in preference orderings and avoids contradictory evaluations (e.g., if $u \succ v$ and $v \succ w$, then $u \succ w$ must hold). By leveraging the human arbiter's domain knowledge and enforcing transitivity, the algorithm enables scalable and fair decision-making systems, especially in contexts like hiring or other evaluations, where fairness is "a highly contextual property one can identify but not necessarily describe."

2.1.1 Fairness Constraint

The algorithm enforces Individual Fairness through the following condition:

$$D(u,v) \geq d(C(u), C(v)) \quad \forall u, v \in U,$$

where $D(u,v)$ is a task-specific similarity metric, and $d(C(u), C(v))$ measures divergence between output distributions.

2.1.2 Submetric Definition

To approximate D, submetrics are constructed. For a representative $r \in U$, the submetric is defined as:

$$D_r(u,v) = |D(r,u) - D(r,v)|,$$

ensuring:

$$D_r(u,v) \leq D(u,v).$$

2.1.3 Submetric Construction with Bounded Error

To reduce computational complexity, distances are rounded to a fixed granularity α. The resulting submetric is:

$$D'_r(u,v) = |f_r(u) - f_r(v)|,$$

where $f_r(u)$ is an α-consistent underestimator of $D(r,u)$, satisfying:

$$f_r(u) \le D(r,u),$$

$$f_r(u) \le f_r(v) \iff D(r,u) \le D(r,v),$$

$$|f_r(u) - f_r(v)| \le |D(r,u) - D(r,v)| + \alpha.$$

The final submetric satisfies:

$$D'_r(u,v) \le D(u,v) + \alpha.$$

2.1.4 Combining Submetrics

Multiple submetrics $D'_{r_1}, D'_{r_2}, \ldots, D'_{r_k}$ are merged using the max-merge operation:

$$D'_R(u,v) = \max_{r \in R} D'_r(u,v),$$

retaining the contractive property:

$$D'_R(u,v) \le D(u,v) + \alpha.$$

2.1.5 Fairness Enforcement

LLM outputs $C(u)$ are constrained to respect the fairness guarantees:

$$D'_R(u,v) \ge d(C(u), C(v)) \quad \forall u, v \in U.$$

2.2 The EXPLORE Mechanism

The EXPLORE mechanism focuses on learning a fairness-aware metric through human-provided pairwise comparisons, minimizing transitivity violations while iteratively refining the metric. It operates in two main phases:

2.2.1 Exploration Phase

Human feedback is collected in the form of triplets $\{(x_{i1}, x_{i2}, y_i)\}_{i=1}^n$, where $y_i \in \{0,1\}$ indicates whether the human arbiter considers x_{i1} and x_{i2} comparable ($y_i = 1$). The probabilistic model assumes:

$$y_i \mid x_{i1}, x_{i2} \sim \text{Ber}(2\sigma(-d_i)),$$

where d_i is the scaled distance between x_{i1} and x_{i2} in the learned representation space:

$$d_i = \|\phi_{i1} - \phi_{i2}\|_{\Sigma_0}^2 = (\phi_{i1} - \phi_{i2})^T \Sigma_0 (\phi_{i1} - \phi_{i2}),$$

and $\phi(x)$ represents the embedding of x in the learned feature space, while Σ_0 is a positive semi-definite (PSD) matrix encoding the task-specific metric. The logistic function σ is defined as:

$$\sigma(z) = \frac{1}{1+e^{-z}}.$$

2.2.2 Refinement Phase

Based on the gathered comparisons, the metric is updated by maximizing the log-likelihood of the observed data, which ensures alignment with human-provided feedback. The log-likelihood objective is:

$$\ell_n(\Sigma) = \frac{1}{n}\sum_{i=1}^n \left[y_i \log \frac{2\sigma(-d_i)}{1-2\sigma(-d_i)} + \log(1-2\sigma(-d_i)) \right].$$

Since $\ell_n(\Sigma)$ is concave over the space of PSD matrices $\mathbb{S}_+^d$, stochastic gradient descent (SGD) is used to optimize Σ. The update rule at iteration t is:

$$\Sigma_{t+1} = \text{Proj}_{\text{PSD}}\left(\Sigma_t + \eta_t \nabla \tilde{\ell}_n(\Sigma_t)\right),$$

where $\tilde{\ell}_n$ is the likelihood of a mini-batch, η_t is the step size, and Proj_{PSD} projects the updated matrix onto the cone of PSD matrices.

This iterative refinement ensures that the learned metric respects fairness constraints, minimizes transitivity violations, and aligns closely with human judgments.

EXPLORE's iterative design makes it particularly useful for dynamically adapting evaluations as new data becomes available. By applying these principles to LLMs, we aim to determine whether similar exploration and refinement methods can enforce transitivity in their outputs. This effort complements prior work on Ilvento's Algorithm by investigating how LLMs can act as consistent oracles for fair metric learning.

In practical applications, such as hiring or grant proposal evaluations, ensuring that similar candidates or proposals are treated equitably—regardless of external factors like timing—is paramount. By refining LLM outputs to respect transitivity, we aim to build systems that are both fair and reliable, paving the way for broader adoption of LLMs as arbiters in high-stakes decision-making. Through this work, we contribute to the growing body of research on fairness in AI, exploring how LLMs can be designed and deployed to uphold critical fairness principles in practice.

2.3 Integration of Ilvento's Algorithm and EXPLORE

Ilvento's Algorithm provides a framework for enforcing fairness in composed systems, while EXPLORE learns the metric Σ that underpins individual fairness. Together, they address key challenges in fairness-sensitive applications such as hiring, admissions, and resource allocation, ensuring both consistency and adaptability to real-world constraints.

3 Defining Transitivity and Using LLMs as Arbiters

3.1 Transitivity in Decision Systems

Transitivity is a fundamental property of preference ordering in decision-making systems. Formally, let X be a set of elements and $\succ$ a preference relation over X. The preference relation $\succ$ is said to satisfy transitivity if for all $x, y, z \in X$:

$$x \succ y \text{ and } y \succ z \implies x \succ z.$$

This property ensures that the ordering of preferences is consistent across all comparisons, eliminating cycles or contradictions. For example, if a system prefers A over B and B over C, transitivity requires that it also prefers A over C.

In the context of algorithmic fairness and decision systems, violations of transitivity can signal inconsistencies in the underlying evaluation criteria. This is particularly problematic in high-stakes scenarios such as hiring, admissions, or grant allocation, where fairness and consistency are highly valued.

3.2 Using Large Language Models (LLMs) as Arbiters

Traditionally, pairwise comparisons required for establishing preference orderings have been provided by human arbiters, whose judgments are based on domain expertise or agreed-upon criteria. However, as Large Language Models (LLMs) demonstrate increasing sophistication in understanding and generating natural language, I want to explore: *Can LLMs replace human arbiters to provide pairwise comparisons for decision-making systems?*

LLMs offer several advantages as arbiters:

- **Scalability**: LLMs can handle a vast number of pairwise comparisons efficiently, which would be infeasible for human arbiters in large datasets.
- **Consistency**: Unlike humans, whose judgments may vary due to fatigue or cognitive biases, LLMs can apply consistent reasoning patterns across comparisons.
- **Adaptability**: LLMs can be fine-tuned to specific domains, enabling them to encode domain-specific knowledge in their decision-making processes.

However,the use of LLMs also raises several concerns. Namely,

- **Fairness and Bias**: LLMs are trained on historical data, which may embed biases that affect their pairwise judgments.
- **Transitivity Violations**: There is no guarantee that the pairwise comparisons generated by LLMs will satisfy transitivity. For example, the LLM may prefer $A \succ B$, $B \succ C$, but $C \succ A$.
- **Interpretability**: The reasoning behind the LLM's judgments may not be transparent, making it difficult to audit or explain decisions.

3.3 Research Questions and Approach

Our work explores the feasibility of using LLMs as arbiters by addressing the following research questions:

1. **RQ1: Do LLMs satisfy transitivity in their pairwise comparisons?**
2. **RQ2: If transitivity violations occur, under what conditions do they arise?**
3. **RQ3: Can the text generation process of LLMs be modified to enforce transitivity?**

For this semester, I have been focused on questions 1 and 2, but I will do more work on these areas and branch into part 3 in the coming months. To address these questions, I have used the following approach.

First, I assemble a dataset that I can take pairwise comparisons of. In general, I try to have 100 trials of whatever is of interest, with each trial representing a triplet (A, B, C) of the elements, that I do three pairwise comparisons of (A with B, B with C, and C with A). I query the LLM with these pairwise comparisons (each as an individual query so memory is not retained) and then the LLM's rankings are analyzed to identify cycles or inconsistencies in the generated preference orderings. Specifically, we evaluate whether:

$$x \succ y \text{ and } y \succ z \implies x \succ z$$

holds for all comparisons. If transitivity violations are observed, we explore why this is the case, and later we will explore methods to enforce transitivity, such as fine-tuning the LLM with constraints or post-processing its outputs, such as with set-based prompting.

By rigorously analyzing the transitivity of LLM-generated comparisons and exploring methods to address violations, we aim to establish the feasibility of using LLMs as reliable and fair arbiters in decision-making systems.

3.4 LLM models

This research evaluates five variants of the Generative Pre-trained Transformer (GPT) models, developed by OpenAI, released between 2023 and 2024. Each model exhibits specific capabilities optimized for different trade-offs in performance, computational efficiency, and cost. Details of the evaluated models are as follows:

- **GPT-3.5 Turbo** (Released March 2023): A third-generation model variant, trained on a transformer architecture with approximately 175 billion parameters. It is optimized for cost efficiency and offers a balance of speed and performance. It is priced at 0.0015 USD per 1,000 tokens for input and 0.002 USD per 1,000 tokens for output. GPT-3.5 Turbo is designed for high-throughput, general-purpose applications.
- **GPT-4** (Released March 2024): The fourth-generation GPT model, significantly improved over GPT-3.5, with an estimated 1.8 trillion parameters (as disclosed in public approximations). It exhibits advanced reasoning, complex language understanding, and creativity. It is computationally intensive and priced at 0.03 USD per 1,000 tokens for input and 0.06 USD per 1,000 tokens for output. This model is used for tasks requiring high precision and contextual depth.
- **GPT-4 Turbo** (Released November 2024): A faster and more cost-effective variant of GPT-4, with comparable performance. It is optimized for efficiency, achieving reduced latency while maintaining similar reasoning and generative capabilities. GPT-4 Turbo is priced at 0.012 USD per 1,000 tokens for input and 0.024 USD per 1,000 tokens for output, representing a 60% cost reduction compared to GPT-4.
- **GPT-4o (Optimized)** (Released November 2024): An optimized implementation of GPT-4 designed for specific use cases requiring increased efficiency and fine-tuned performance. GPT-4o achieves superior throughput and computational resource utilization, making it suitable for resource-constrained environments. The parameter count and specific cost details are undisclosed but are reported to be lower than GPT-4 Turbo.
- **GPT-4o Mini** (Released November 2024): A lightweight version of GPT-4o with further reductions in computational requirements, targeting applications with significant resource constraints. This model is suitable for environments where memory and processing power are limited, offering a streamlined trade-off in performance while retaining key functionalities of the GPT-4 series.

- **LLaMa -2** (Released July 2023): This language model is from Meta, and is the main open-source model available. I can access through Harvard's cluster. Its downsides are that it takes longer to query, but the large positives are that I can finetune the model to fit my needs and propose changes to improve transitivity and consistency in evaluations.

3.5 Experimental Parameters and Configuration

In this section, I detail the key parameters and configurations used in the experiments, including the maximum token limit, temperature settings, and the nature of system and user inputs.

3.5.1 Max Tokens

The `max_tokens` parameter determines the maximum number of tokens the model can generate in response to a query. This parameter was set based on the nature of the task:

- For **number and word comparisons**, the `max_tokens` value was set to **5**, ensuring concise outputs such as a single number or word. This limitation was chosen to minimize verbosity and maintain focus on the direct comparison task.
- For **Reddit post comparisons**, the `max_tokens` value was increased to **50**. This allowed for more elaborate responses, including reasoning behind the selection of one argument over another. This adjustment was necessary due to the complexity of the task, which required nuanced analysis.

3.5.2 Temperature

The `temperature` parameter controls the randomness of the model's output. Lower values (closer to 0) make the output more deterministic, while higher values introduce variability. To explore the effect of this parameter on the model's consistency and performance, we conducted experiments with the following settings:

- **Temperature = 0.0**: This setting encourages deterministic outputs, where the model provides the same response for identical inputs. It was particularly useful for tasks requiring high consistency, such as number and word comparisons.
- **Temperature = 0.7**: This setting introduces randomness, allowing for a range of plausible outputs. While it provided more creative responses, it occasionally led to inconsistencies, especially in transitivity adherence.

The comparison of results across these two settings revealed that higher temperatures (0.7) led to increased cycles and order-dependence, highlighting a trade-off between consistency and diversity of responses.

3.5.3 System and User Inputs

The system and user inputs were tailored to each experimental context:

- For **number and word comparisons**, the system was prompted to return a single number or word relevant to the query. This ensured straightforward and interpretable outputs.
- For **Reddit post comparisons**, no specific constraints were placed on the system input. This allowed the model to generate more detailed explanations, providing insight into its reasoning process for selecting the more persuasive argument.

These configurations ensured that the model's outputs were appropriately tailored to the requirements of each task, enabling a thorough evaluation of its performance across diverse scenarios.

3.6 Codebase

My code for this project is contained to this Github Repository. This repository is split into four different subfolders: analysis, cmv, numbers, and words. I also have local folders called data and results that are only in my local files and in the lab's files, but that are not uploaded to Github, due to issues with pushing large data files. The organization of my files includes a .gitignore that prevents the data and the results from being pushed to the server.

4 Concrete Comparisons

Hiring is a challenging decision-making task due to the complexity of inputs (candidates with diverse attributes) and the ambiguity of the reward function (selecting the "best" candidate). However, there are some tasks where the inputs and the evaluation criteria are more clearly defined. For example, numerical comparisons allow for the evaluation of transitivity in scenarios with objective ground truth, such as comparing the magnitude of two integers. This section explores the performance of LLMs in such concrete settings.

4.1 Simpler Queries

Simpler queries test the LLM's ability to handle straightforward comparisons. For example, we asked the LLM to compare two randomly generated numbers and determine which is larger. Random numbers were generated using a custom function, `generate_number(digits)`, to produce integers of varying lengths. The LLM was then tasked with answering prompts such as, "Which number is larger, {X} or {Y}?"

Results: For these simpler queries, the LLM demonstrated near-perfect accuracy (transitivity adherence rate = 100%). This confirms that the model is highly effective at maintaining transitivity in direct numerical comparisons, aligning with its extensive training on arithmetic and numerical reasoning.

4.2 More Complicated Queries

To challenge the LLM further, we introduced comparisons based on derived metrics, such as the sum of prime factors of a number. Using the function `sum_of_prime_factors(n)`, we computed a secondary feature for each number. The LLM was then prompted with comparisons like "Which number has a larger sum of prime factors, {X} or{Y} ?"

Results: Once again, transitivity was upheld, and even more strongly, all comparisons were in line with the ground truth. Even for more nuanced comparisons like 20013442 (which is $10006721 \cdot 2$) with 90328468 ($2 \cdot 2 \cdot 409 \cdot 55213$), the LLM was able to pick up on the exact values and give accurate responses. This is reasonable, as the LLM has been trained extensively on close-ended, arithemtic comparisons and calculations, and the scope of computation for straightforward tasks can rival that of a human's.

I thought that we would potentially see a change around 10 digits, since that it is the length of a phone number, so maybe the LLM would be confused, but that was not the case.

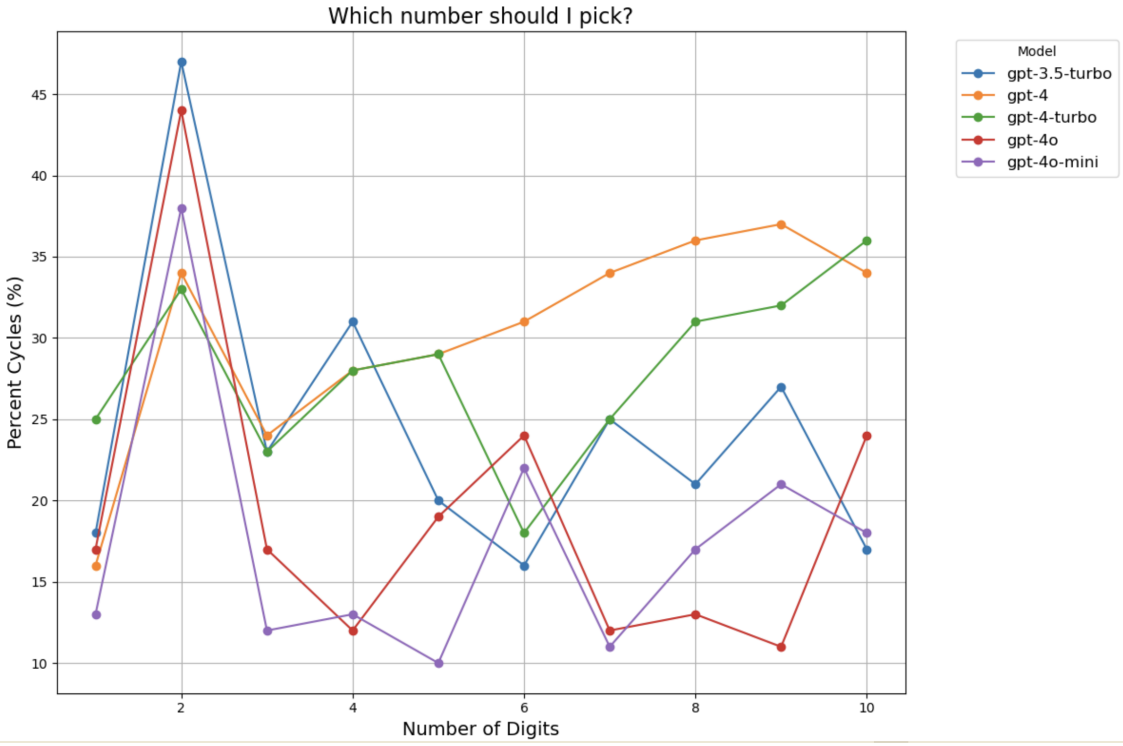
> **Future Extension:** For these results, I only looked at numbers through 12 digits, so potentially if I increased this to something like 20 digits, we might see some cycles. This is an extension I can explore in future work.

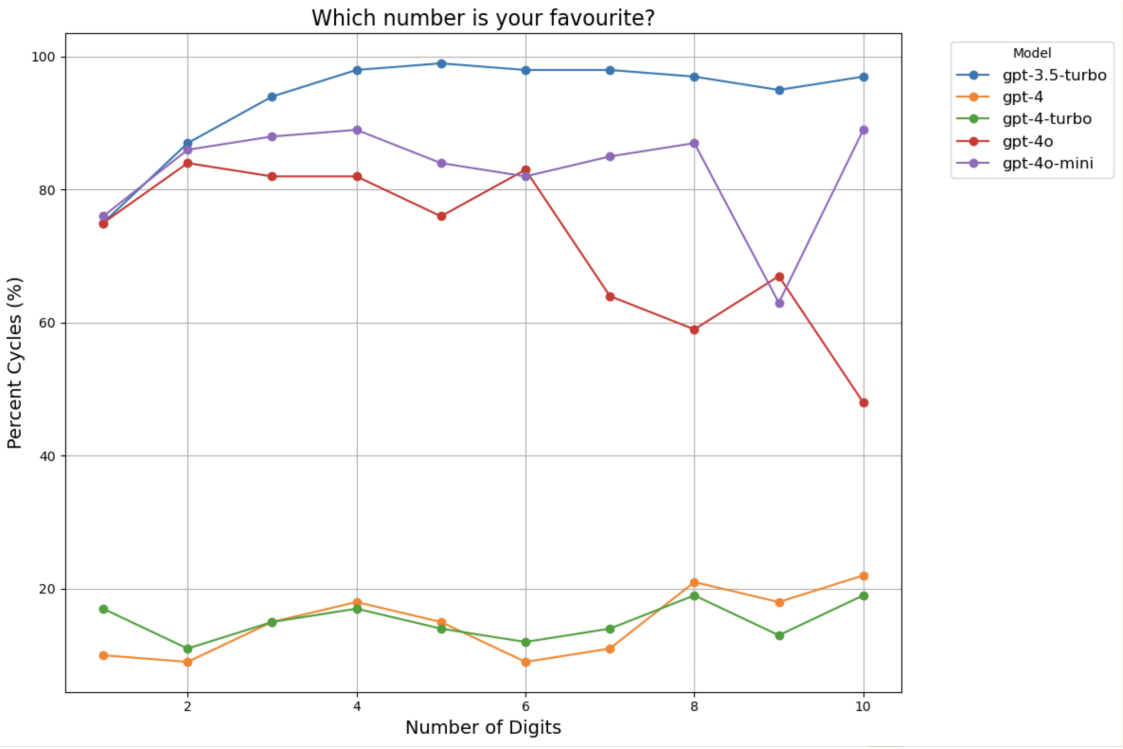
5 Abstract Comparisons

Abstract comparisons move beyond numerical reasoning to evaluate the LLM's ability to make consistent, subjective judgments. These tasks are designed to explore whether the model can apply coherent, and more importantly consistent, reasoning in settings where there is no objective ground truth.

5.1 Simple Questions

I started out asking the LLM basic questions like "Which number should I pick" or "Which number is your favourite?". I then noted down the percent of triplets that resulted in cyclic preference rankings.





For the first question, "which number should I pick?" we see that most of the models obey transitivity at least 1/3 of the time, but for the next question "which number is your favourite?",

we see that although the gpt-4 and gpt-4-turbo models obey transitivity most of the time, the gpt-3.5-turbo and both the gpt-4o models have incredibly high levels of cycles.

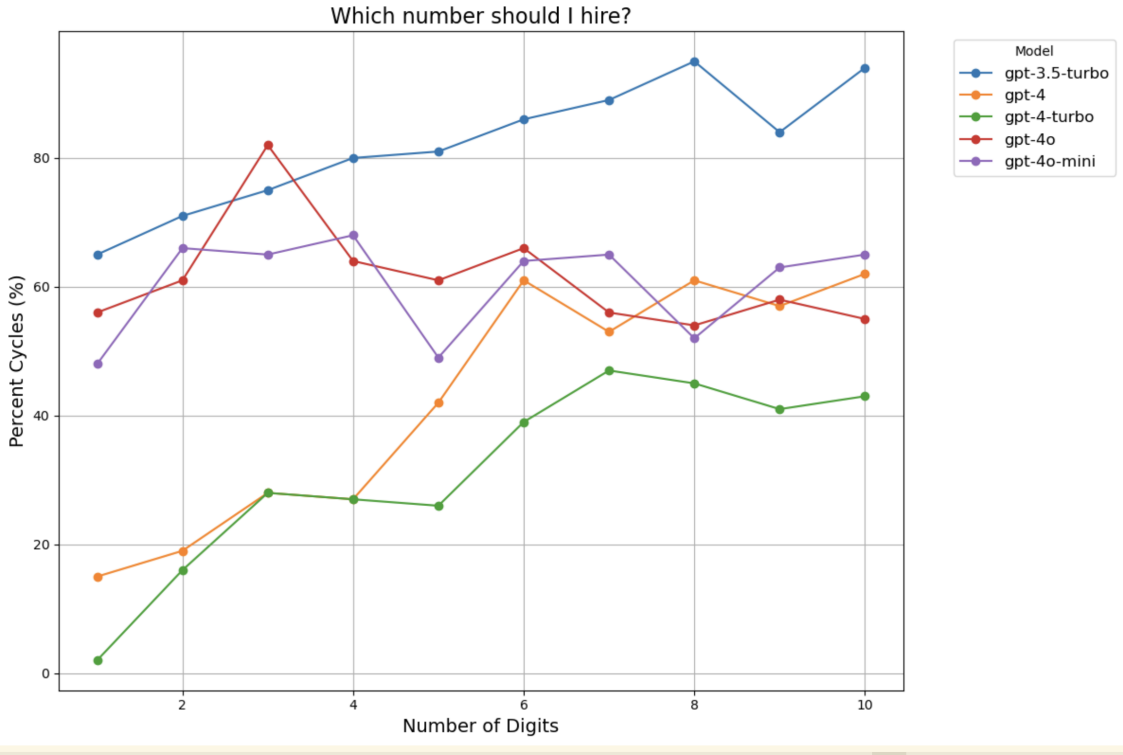
The high levels of the cycles are especially alarming, so I spent some time analyzing the data. I see that the results are order-dependent more than 80-90% of the time, meaning that the model selects the first number given as its favourite, leading to the high levels of cycles.

When I asked it to explain its reasoning for certain number selections, it would cite reasons like "repeated digits," "symmetry across the number," or "sum of digits" equally a well-known number like 64.

> **Future Extension:** See if set-based prompting makes the questions less order dependent, following the lab's previous work, and if this helps to reduce the number of cycles.

5.2 Which should I hire?

I give this question a bit of a fairness, real-world twist, by adding in an added element of having a hiring-based evaluation. Specifically, I prompt the model with "Which should I hire, {number 1} or {number 2}?



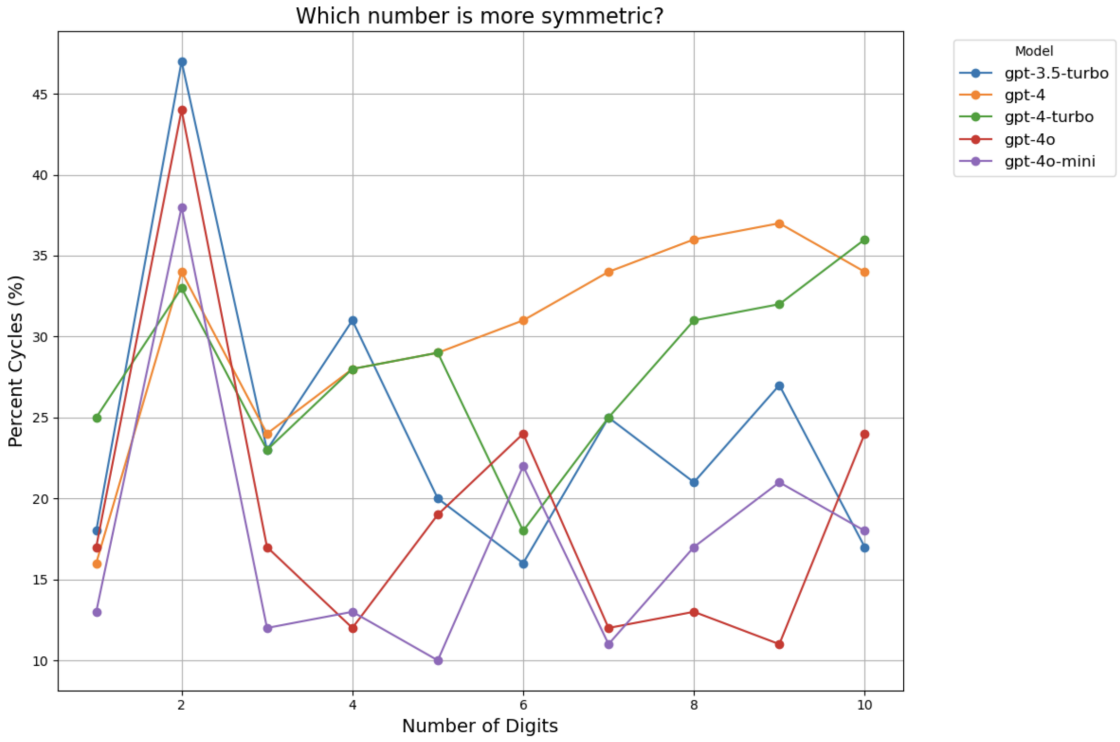
Here, we see that some of the models (gpt-3.5-turbo) consistently pick the larger number as the one to hire, but some other models 'gpt-4' and 'gpt-4-turbo' are actually making seemingly more nuanced decisions, resulting in fewer cycles.

This would be a good candidate for doing a deeper dive into directly asking about the model's decision making process.

5.3 Symmetric Numbers

Symmetric numbers are a concept that are a bit more defined than favourite numbers, so I was curious to see how the models evaluated them. My initial thought was that the numbers would look at the reverse of the number and calculate some metric based on how many of the digits were the same in the reverse (so a palindrome would have a full score).

Results:



We see a spike in the number of cycles at 2 digits. This makes sense, since the numbers are only 2 digits long, so it is hard to find symmetry if it is just following an AB pattern, otherwise we see that all the models fall below about 35% cycle count for all the digits, indicating a fairly more consistent question.

Looking into the model: I did a deep-dive into the following triplets of numbers: 5197722932, 6116796198, 2777095375. I queried the OpenAI API, in the 4o model, and asked it for each pair, "Which is more symmetric, {X} or {Y}?"

Overall, it was doing what I expected it to do, which is look at the number of matches in digits between the flipped and normal version of the number. For example, we have 5197722932, and its reverse 2392277915. Here, we see that the only place that matches is the 3rd digit, the 9, so there is a symmetry score of 1. Similarly, we can see that 6116796198 has a score of 2, and 2777095375 has a score of 1. However, the model not only takes into account this symmetry score but also the number and placement of the repeated digits.

However, the model does have some issues with computing the actual symmetry score in some instances, as is especially unable to pivot the 6 and 9 along the center, resulting in 6116796198 having an incorrect score of 4 in some comparisons.

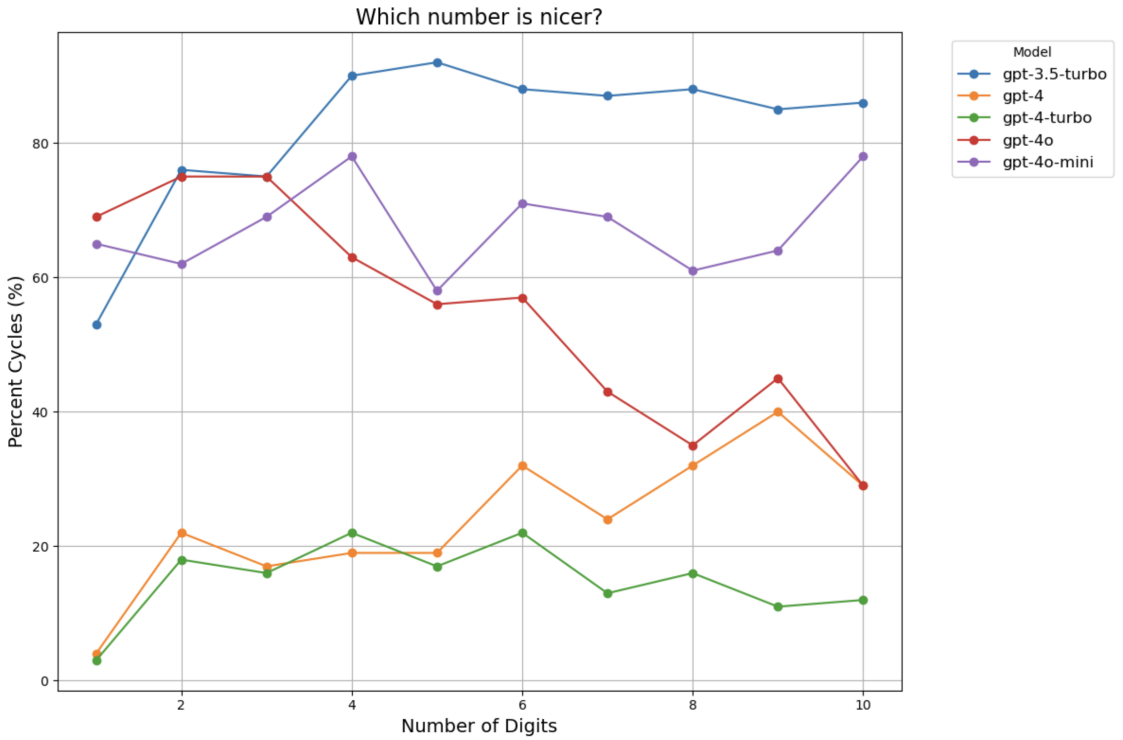
Thus, it makes the following claims:

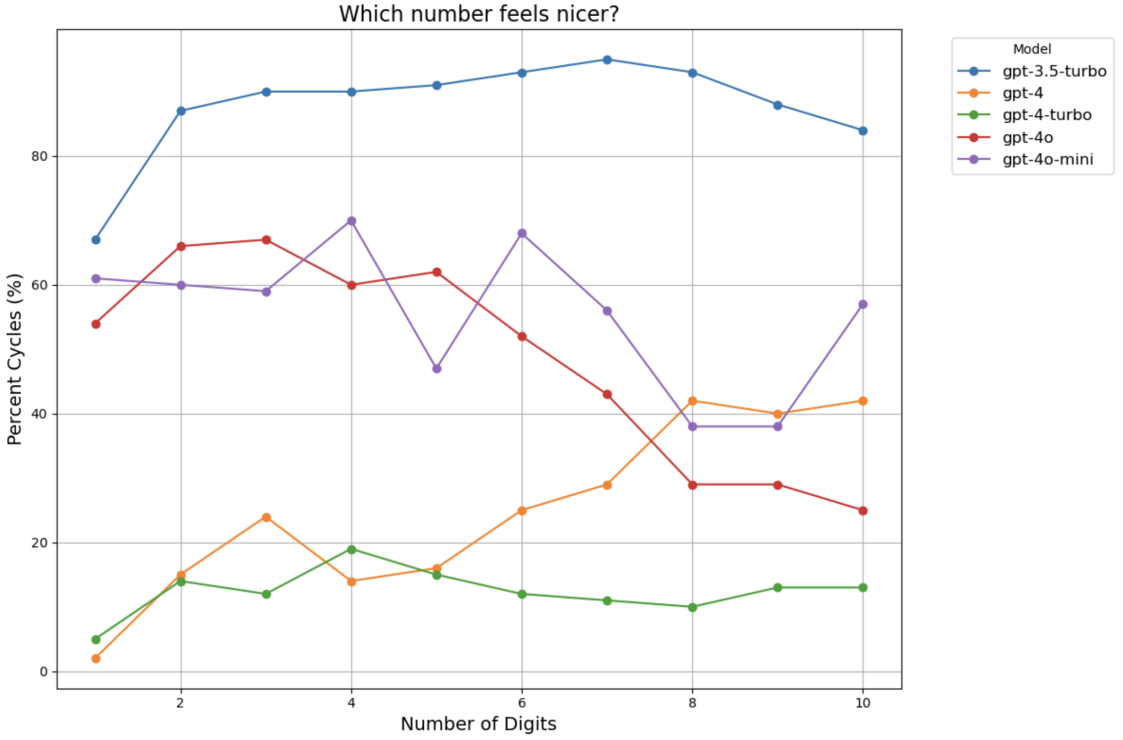
1. 6116796198 is more symmetric than 5197722932 because of the greater number of matched digits
2. 2777095375 is more symmetric than 6116796198 because of the greater number of repeated digits.
3. 5197722932 is more symmetric than 2777095375 because of the greater number of repeated digits centered in the middle.

Here we see that there are many conflicting views of what "symmetric" means, so if this is kept ambiguous, then the LLM might choose one at will and go with that, but alternate between queries, which leads to inconsistencies.

5.4 Nicer Numbers

I again played around with prompt engineering, asking about the LLM which number "feels nicer" and which "is nicer", resulting in the following results:

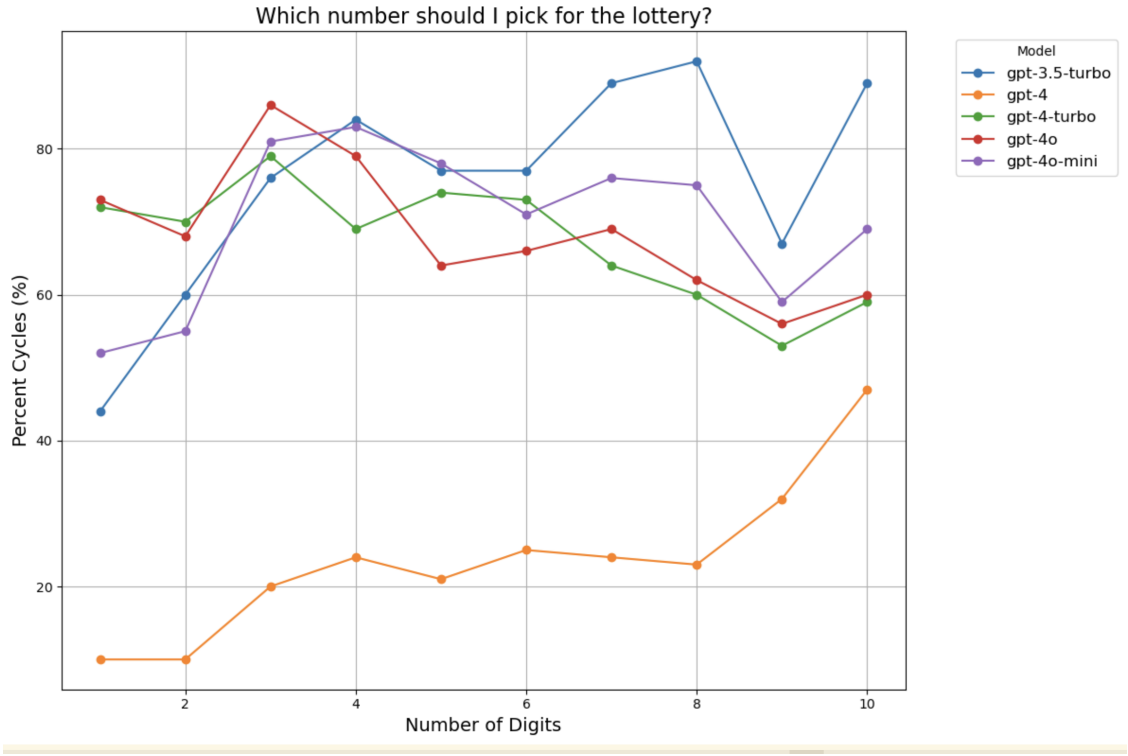


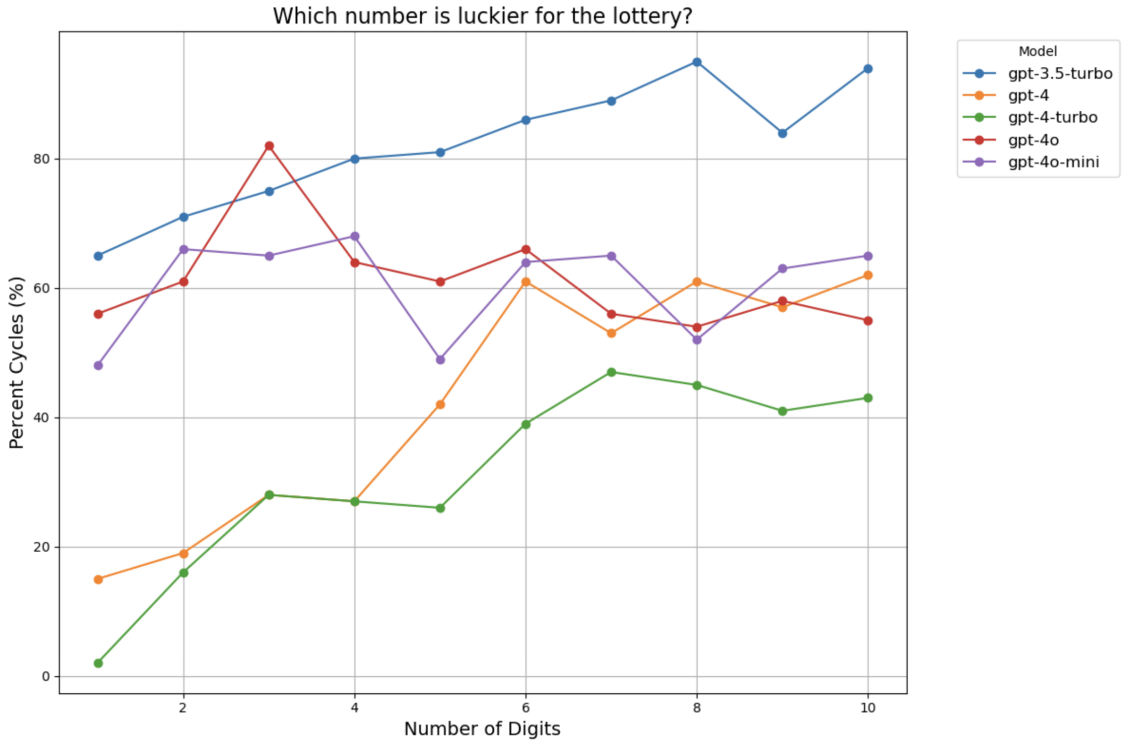


Overall, the results are relatively consistent for the same model between the two prompts, which is a promising sign for being robust to minor prompt changes.

5.5 Lottery Numbers

In this task, the LLM was asked to compare random lottery numbers. This again felt like a real-world tasks, grounded in monetary consequence, and one that I could play around with prompt engineering for. For this task, I was curious so I did 2 different prompts: "Which lottery number is luckier, {X} or {Y}?" and "Which lottery number should I choose, {X} or {Y}?"





Here, the performance of the gpt-3.5, gpt-4o, and gpt-4o-mini models are all consistent between the graphs. However, the number of cycles in the gpt-4 and gpt-4-turbo differs as we change the prompt. Namely, the gpt-4-turbo model is much less cyclic for the luckier query, but the gpt-4 model is less cyclic (by a large margin compared to the other models) for the which number should I pick? From doing some analysis of the dataset, it seems this is because it picks the larger number consistently.

6 Vocabulary Word Comparisons

Moving towards more practical applications, I was curious to see how LLMs quantify the difficulty of vocabulary words to learn in English. The Centre de traitement automatique du langage (CENTAL), an institute for language learning based out of Belgium, publishes a dataset called EFLLex that provides insights into vocabulary learning for English as a Foreign Language (EFL). This dataset is particularly useful for understanding the relative difficulty and usage frequency of words across different proficiency levels defined by the Common European Framework of Reference for Languages (CEFR).

6.1 Dataset Structure

The EFLLex dataset is organized by words or **lemmas**, which are the base forms of words such as "cat" or "Austrian." Each lemma is annotated with several attributes that help describe its

grammatical and frequency-related characteristics. These attributes include:

- **POS-tag**: Part-of-speech tag, indicating whether the lemma is a noun (NN), an adjective (JJ), or a verb (VB).
- **CEFR Levels**: The dataset reports normalized frequencies (per million words) for five levels of the CEFR: A1 (Beginner), A2 (Elementary), B1 (Intermediate), B2 (Upper Intermediate), and C1 (Advanced).
- **Total Frequency**: The cumulative normalized frequency of the lemma across all CEFR levels.
- **Document Frequency (nb_doc)**: The number of documents in which the lemma appears.
- **Multi-word Expressions**: Although the beta version primarily includes single words, it also plans to incorporate multi-word expressions in future iterations.

The frequencies in EFLLex are derived from a corpus of 13 EFL textbooks and 8 online resources, ensuring that the data reflects the vocabulary commonly encountered in English learning materials.

6.2 Comparison Metrics

To evaluate which word is "more advanced" in terms of usage and learning difficulty, we propose several metrics that leverage the normalized frequency data across CEFR levels:

1. **Concentration of Usage**: Identify the CEFR level where the word's frequency is highest. A word that peaks at a higher level (e.g., C1) could be considered more advanced.
2. **Weighted Average of CEFR Levels**: Assign weights to each CEFR level (e.g., A1 = 1, A2 = 2, ..., C1 = 5) and calculate a weighted average based on the word's frequency distribution. This provides a single numeric score representing the word's "average level."
3. **Proportion of Higher-Level Usage**: Compute the proportion of the word's frequency that occurs at higher levels (e.g., B2 and C1) compared to lower levels. This metric highlights whether a word is predominantly used by advanced learners.

6.3 Weighted Average Implementation

To implement the weighted average metric, the following steps can be used:

1. Assign numeric weights to each CEFR level: A1 = 1, A2 = 2, B1 = 3, B2 = 4, and C1 = 5.

2. Multiply each CEFR level's frequency by its corresponding weight.

3. Sum the weighted frequencies and divide by the total frequency to compute the weighted average level.

Mathematically, we can define this weighted average as follows:

$$w_{\text{score}} = \frac{w_{\text{A1}} \cdot 1 + w_{\text{A2}} \cdot 2 + w_{\text{B1}} \cdot 3 + w_{\text{B2}} \cdot 4 + w_{\text{C1}} \cdot 5}{3 \cdot w_{\text{total\_freq}}}$$

6.4 Dataset construction

Within this dataset, we take out words that have underscores or other punctuation in them, as they might be harder to compare other words to. I also excluded any words that have numbers in them, since this might be more of a term like the "1970s" rather than a true vocabulary word.

6.5 Queries

I asked a set of three prompts, in order to gain more insight into how the LLM would respond to evaluating the words under different premises:

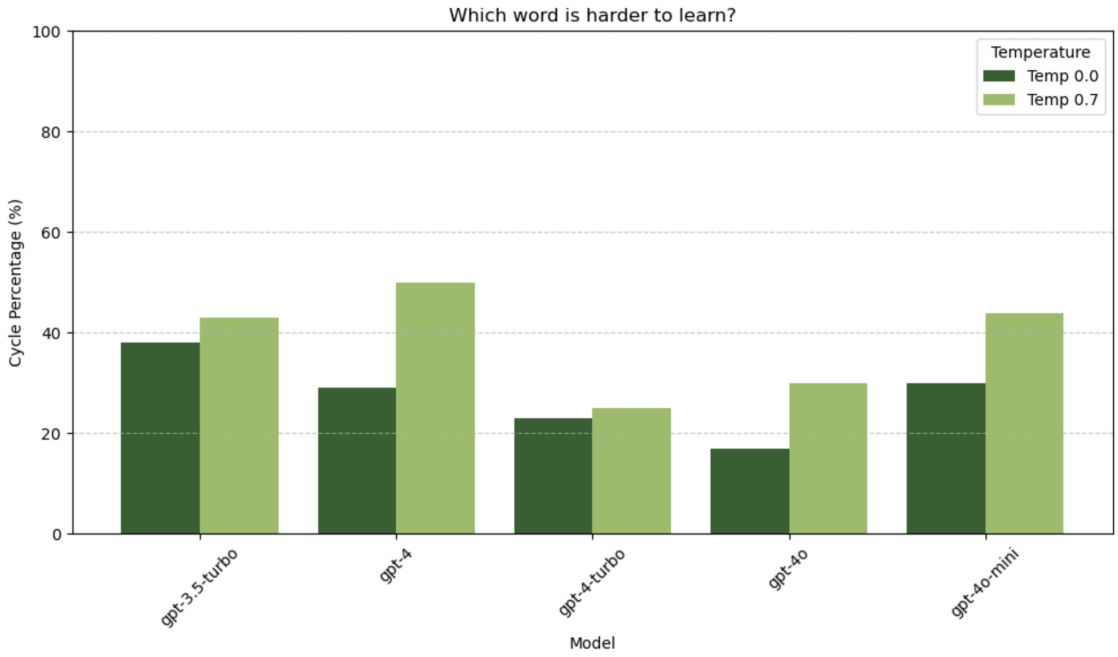
- "Which word is harder to learn?"
- "Which word is more advanced?"
- "Which word is at a higher vocabulary level?"

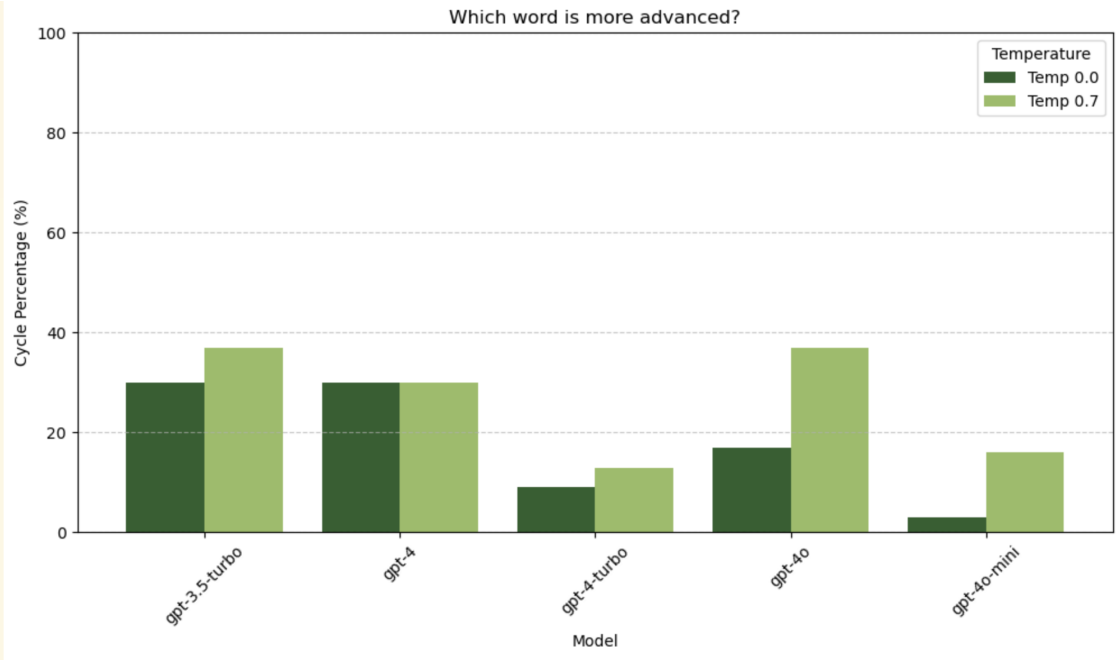
These gave a variety of positive sentiment questions like "more advanced" or "higher vocabulary level," whereas the first question has a negative seniment "harder to learn."

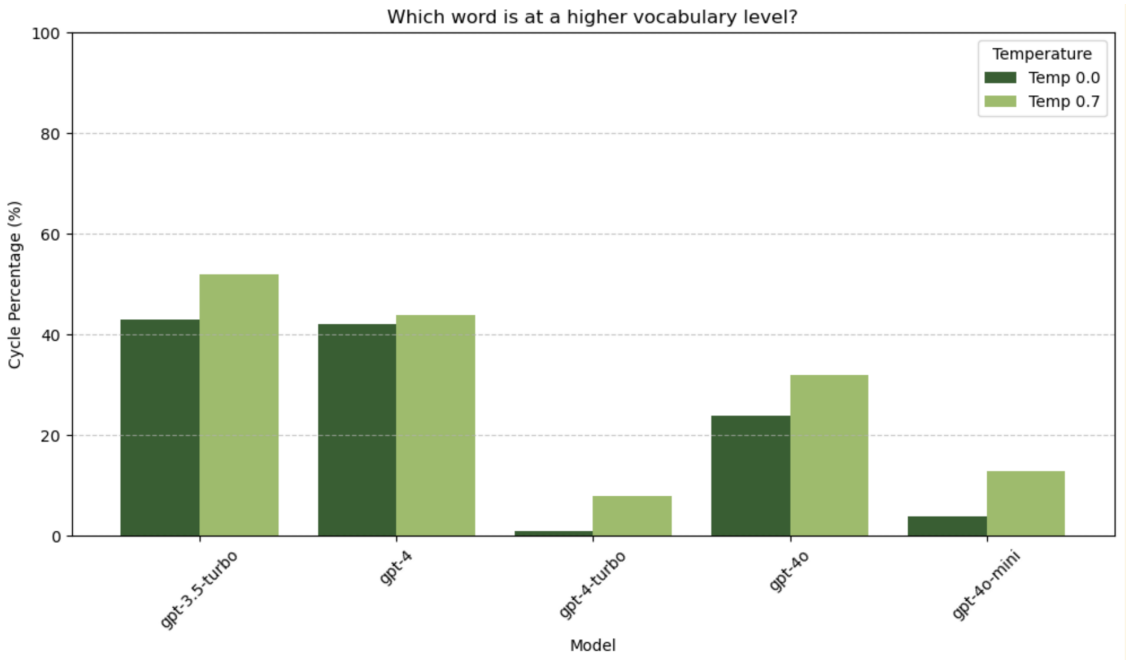
Future Extension: Look at whether the LLM is consistent in its responses with queries, and the negation of those queries. For example, if I ask "Which word is harder to learn, A or B?" and it responds with A, then if I ask "Which word is easier to learn, A or B?", it should respond with B.

6.6 Results:

The three graphs evaluate how different language models perform when given three types of prompts about word properties: difficulty, advancement, and vocabulary level. Importantly, a lower number of cycles indicates better model performance, as it reflects greater consistency in the model's responses.


Which word is harder to learn?
Temperature
Temp 0.0
Temp 0.7
Cycle Percentage (%)
100
80
60
40
20
0
gpt-3.5-turbo
gpt-4
gpt-4-turbo
gpt-4o
gpt-4o-mini
Model


Which word is more advanced?
Temperature
Temp 0.0
Temp 0.7
Cycle Percentage (%)
100
80
60
40
20
0
gpt-3.5-turbo
gpt-4
gpt-4-turbo
gpt-4o
gpt-4o-mini
Model



When comparing across the three graphs, GPT-4-turbo consistently has fewer cycles, showing it is the most reliable model for these tasks. The increase in cycles at temperature 0.7 indicates that higher temperatures make the models slightly less consistent. Unlike in some other scenarios, the gpt-4 does not perform as consistently here. Something that is interesting to me is that the gpt-4o-mini model outperforms the gpt-4o model in all the situations but "which is hardest to learn?" The gpt-4o-mini is meant for quicker tasks, so I wonder if it is just more familiar with short word-based tasks, or if it is making decisions too quickly regardly word difficulty. I also wonder what makes the hardest to learn question have different results than the other two – is it because it is a negatively-worded question or because difficulty is more subjective and more based on prior experiences of the person involved, whereas the other two questions are more intrinsic to the word. Interestingly, we also see that the LLM-generated preference only aligns with the ground truth of the weighted average about 50% of the time, indicating potential arbitrariness to the choices. We can perform more analysis on the words that were chosen to see if there are any underlying similarities, guiding the LLM's decision making.

Future Extension: Construct a model specific breakdown of where it aligns with the ground truth. Also, compare it against different metrics of ground truth, as previously defined.

7 Persuasiveness of Reddit Posts

Here, I look at

7.1 Original Dataset

The Winning Arguments (ChangeMyView) dataset was introduced in the study Winning Arguments: Interaction Dynamics and Persuasion Strategies in Good-faith Online Discussions (Chenhao Tan et al., WWW 2016). It was developed to analyze persuasive strategies in online discussions, specifically within the r/ChangeMyView subreddit, where users engage in constructive debates aimed at challenging or reconsidering opinions. The study focuses on understanding the dynamics of successful arguments, defined as those that convince the original poster (OP) to change their stance, as indicated by the subreddit's "Δ"system. The dataset was designed to enable paired analyses, comparing successful argument threads with the most similar unsuccessful threads in the same discussion.

The dataset spans from January 1, 2013, to May 7, 2015, and includes 3,051 conversations, 34,911 speakers, and 293,297 utterances. Conversations correspond to complete comment threads, each initiated by an OP's post. Metadata for each conversation includes the OP's username, post title, and body text, as well as pair IDs that link successful and unsuccessful threads. At the utterance level, it provides the full text of comments, binary success labels, unique pair IDs for matched threads, and reply structures that reconstruct the conversational hierarchy. Additional Reddit API metadata includes scores, timestamps, and controversiality, while speaker information is limited to usernames for privacy. The dataset's structure supports a detailed examination of linguistic and interactional patterns that contribute to persuasive success in online discussions.

7.2 Dataset Manipulation

To manipulate the Winning Arguments (ChangeMyView) dataset for analysis, I implemented a script to extract up to three arguments with upvote information from each thread. Using a JSON lines file compressed in bz2 format, I parsed each thread sequentially, focusing on the thread's title, text, and associated comments. Threads with "Unknown" in their titles or text (case-insensitive) were excluded to ensure data relevance. For each thread, I retrieved up to three comments containing arguments, prioritizing those with upvotes, and determined their success heuristically based on the presence of "delta" in the comment body. These arguments were organized into individual rows, capturing their text, success status, and upvotes, alongside the corresponding thread's title and text. This process was repeated until data from 100 valid threads were processed. The resulting dataset was structured into a pandas DataFrame and saved as a CSV file for further analysis. This approach allowed me to create a condensed subset of the dataset, on which I could do my later queries.

7.3 Comparison Mechanism

Here, I queried the models, asking it to determine which argument was more persuasive. In these instances, I allowed a max token amount of 50, which allows for more complex analysis

before the result is output as well as a longer answer. This means that instead of just saying "Argument 1" or "1", the result of the query will be some string along the lines of "On the whole, Argument 1 is more powerful because of its recognition of the healthcare implications of cutting taxes," where more reasoning is given rather than just the response.

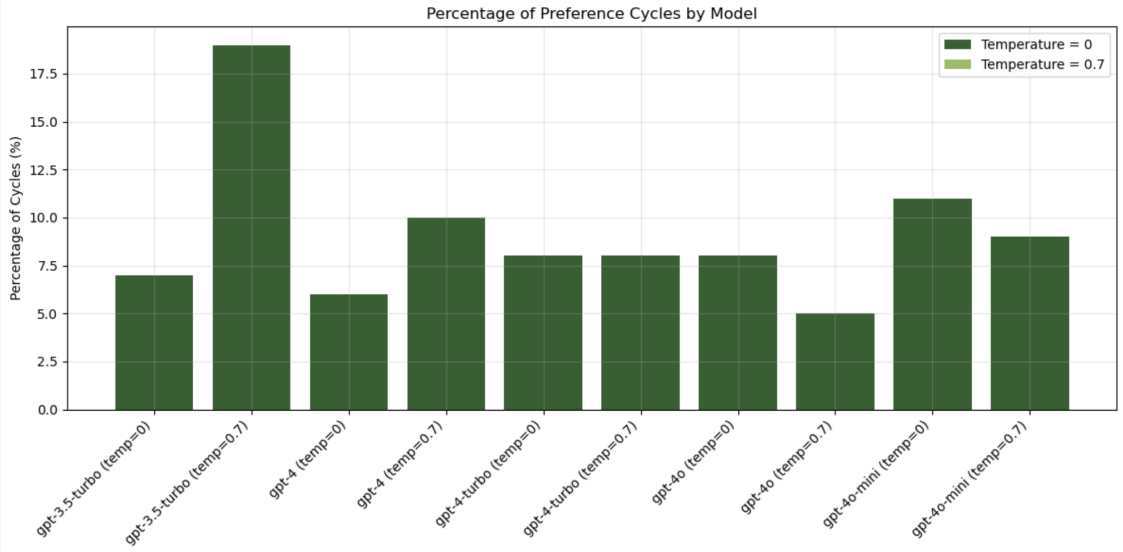
This raises a question of how to actually parse through this data in order to determine which argument the query is saying is better. Manually looking through all the data files would be tedious for a human, and potentially have room for error, so instead I turned to LLMs. Specifically, we created an LLM-evaluator python program, which takes in a string output, passes this through an LLM, asking it to determine which response the string evaluation is endorsing.

In order to implement this, I prioritize consistency, and pick just one model to evaluate all responses. Here, I picked gpt-4o as the model of choice to do the parsing, since it was the most recent full model released, which I took to imply it had the most sophisticated abilities to understand meaning from texts.

Future Extension: Conduct traditional machine learning analysis on the dataset of selected responses to see what shared features are the most prominent for those the model sees as most persuasive.

7.4 Results

Here, we see the percentage of cycles formed by the triples of responses for the various models and temperature pairings. Note that all of these percentages are quite low, especially compared to the other queries that we have been exploring. The least transitive model is by far the gpt-3.5-turbo with 0.7 temperature. This makes sense, because often increasing temperature increasing randomness, which could make consistency harder to achieve. Additionally, the gpt-3.5-turbo model is the least advanced, which could again make it more likely to have inconsistent results.

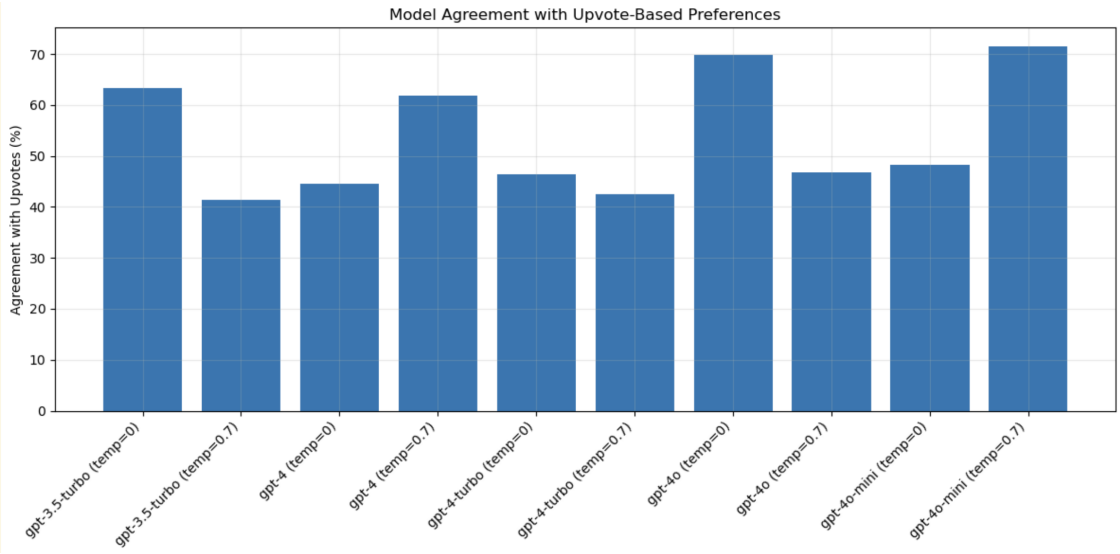


The exact results are included in this table:

Model	Temp=0	Temp=0.7
gpt-3.5-turbo	7.00%	19.00%
gpt-4	6.00%	10.00%
gpt-4-turbo	8.00%	8.00%
gpt-4o	8.00%	5.00%
gpt-4o-mini	11.00%	9.00%

Table 1: Cycle Percentages for Reddit Persuasiveness

Next, I wanted to compare this to the "ground truth" of the persuasiveness of the arguments, which is represented by the number of upvotes the post got. Here is a graph, split by model and temperature, of the percent of time that the model comparison agrees with the upvote-based preferences. This is a suprisingly low number – I would have expected it to agree more than 75 percent of the time, but the average is closer to 50. It is also interesting because there is no apparent pattern between the temperature and the model agreement as for some, the temperature = 0 trials are a lot higher than the temperature = 0.7, but for others, the opposite is true.



Overall, one point of fascination for me is that the model performance worsens when increasing temperature in some of the old models, but improves when increasing temperature in the new models. This means creativity and randomness might improve reasoning for the more advanced models, even if it was a hindrance and led to more inconsistency before.

8 Conclusion and Future Work

8.1 Conclusion

In my work so far, we have gained deeper insights into quantifying the transitivity of the GPT models under different circumstances, with various prompts.

I started with concrete situations, comparing magnitude of numbers. Then I transitioned into abstract comparisons of numbers, then going into the comparisons of the difficulty of different vocabulary words, and lastly I looked at the persuasiveness arguments of Reddit posts.

Each of the individual results is discussed in more detail above, but overall, I saw that the level of transivity changed a lot based on the specific prompts used. However, in certain circumstances, different models were more sensitive to change, such as the gpt-4 and gpt-4-turbo models for the lottery prompting, or the gpt-4-turbo and gpt-4o-mini for the vocabulary words. A high sensitivity to change in prompts come across as problematic, or at least an area to pay attention to, since the phrasing of questions defining the hiring / evaluation tasks will be crucial in getting the desired. It also makes me wonder if people can unfairly manipulate the results through subtle prompt-engineering.

It was interesting because the number of cycles did not move in a specific direction as the models became newer. The overall trend was that gpt-3.5 had the most cycles, but that gpt-4 or gpt-4-turbo had the fewest cycles. However, gpt-4o and gpt-4o-mini sometimes had a low number of cycles, but often had somewhere in the middle. This makes sense for gpt-4o-mini, as it is optimized primarily for quicker, more straightforward, rather than more ambiguous, evaluative tasks. But the gpt-4o result really surprised me. This made me think a lot more about what it means for an evaluation to be transitive, or not. Is it strictly better, or might we be overlooking an important question.

From doing this research, I landed on one pressing question: does consistency / transitivity truly equal fairness? There are some examples, such as with the favourite numbers scenario, where the model often chooses the first one because there is strong order dependence. This does not seem fair to me. If we broaden the context, we might say that other factors, that are as irrelevant as order of inputs might systemically cause the model to pick a certain option, even if this choice is not in line with the overall question that it is selecting for. For instance, in a hiring context, it might always pick the person with the higher GPA or from the richer home city, even if that is not likely to translate into better candidacy for the position. Still, if it had a consistent metric like this, it would be transitive, and pass this threshold. Perhaps, then in some situations, it might be better if the responses aren't consistent as it underscores the nuance in these decisions and makes sure there is no systemic bias, especially if the selection criteria is not well-defined. Alternatively, perhaps we should have three options, either the model has to pick option A, option B, or if absolutely necessary, can say there is a tie. But we should limit the occurrences of this final option.

Lastly, I want to put more emphasis on comparing the results to the ground truth of the data.

Again, it is not enough to be consistent, it must also, and perhaps more importantly, be accurate, so I want to fine-tune the model with this in mind.

8.2 Future Work

This project is still very much a work in progress, which I will continue to work on over the course of the winter break and the next semester. Throughout the paper, I've put some notes about areas I want to expand into. Here are some specific additional areas of expansion:

- **LLaMa -2 Model:** Over the course of the past two weeks, I have been working relentlessly to try to get data for the LLaMa models but due to a variety of technical difficulties, I haven't been able to get the full results yet, which is why they are omitted from this version of the paper.
- **Essay Dataset:** I want to conduct experiments using the dataset of essays that Reid gave me, asking the LLM to evaluate it on the basis of various points (like grammar, sentence structure, argument).
- **Asking LLM for reasoning:** I started asking the model to explain its reasoning for specific examples, especially those where the preferences are cyclic. I found this quite eye-opening, so I want to do this for all the components.
- **Fine-tuning:** With the LLaMa-2 Model, I will also be able to fine-tune my work, which I am very excited about. I can use insights from different prompting styles, as well as the traditional machine-learning analysis I do to make the models more consistent.

8.3 Thank you

Lastly, I would like to thank Reid McIlroy-Young for all of his work throughout the semester, and his prompt responses for all my questions, especially when I was struggling to figure out any technical systems. I feel very grateful to be mentored by someone who is not only so accomplished and knowledgeable (I've never met anyone who knows as many Visual Studio Code shortcuts as him), but also so patient and available for weekly meetings. I would also like to thank Prof. Dwork for her support in even allowing me to get involved in this research in the first place, and for all the insights and conversations over the course of this semester, both directly related to research and about the larger fairness questions in CS1260. It's been an eye-opening and fulfilling experience, and I'm excited to continue the work.

9 Personal Learnings

9.1 Overall Learnings

They say the best way to learn to swim is to be thrown into the deep end, and that's exactly how research has felt—exciting, overwhelming, and often messy. I've done research in the past, but my previous projects were either in the social sciences, primarily economics, or leaned heavily into theoretical computer science. Over the past year, I found myself increasingly interested in this question of how we can create more robust machine learning practices, along dimensions of safety, privacy, and fairness. And so, I find myself here.

Working on fairness in machine learning, I've realized the challenge isn't coming up with questions but figuring out where to focus. In a field like human-AI interaction, there are endless directions to explore—so many datasets, models, and methods that it's easy to get stuck in the cycle of exploring without ever digging deep. Balancing the explore/exploit tradeoff has been one of the hardest lessons to learn, especially when every new idea feels like it could lead somewhere groundbreaking. Research, I've come to understand, is as much about discipline and prioritization as it is about curiosity. Another component, slightly unique to fairness research, is reminding yourself to tie back what you are doing to the fairness questions at hand.

The less visible side of research has been just as challenging—figuring out how to run experiments on systems like the Harvard FASRC cluster, cleaning messy datasets, and troubleshooting infrastructure issues. These tasks can feel tedious and unproductive in the moment, but they're the foundation of meaningful work. I've spent hours trying to align datasets or debug scripts, only to realize these "detours" teach you just as much as the research itself. I'm now so much better at working within and finding creative ways to interact with datasets. Over time, I've come to appreciate that progress isn't always linear. It's about sticking with the process, learning from missteps, and trusting that each small breakthrough brings you closer to something impactful.

9.2 Technical Best Practices

Here are some best practices I learned along the way:

- In order to access the OpenAI API, each account has an API key, which indicates usage for the purposes of billing. Since this is your own personal API key, it is important that you protect it, without letting other people, especially those who you do not know or trust, gain access to it. But how do you do this if you need to use it in the code? In terminal, I did `'open -e /.zshrc'`. Then I added export `OPENAI_API_KEY="YOUR_API_KEY''` at the end. In my file, I did `client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))`
- During my first few runs of the experiments, I would query the OpenAI API 300 times, keeping a running count of the number of cycles seen cumulatively across the trials, and

then saving that final count at the end. My fatal mistake was not saving the data results from each trial as I was doing this, leaving me, weeks later with only a random "cycle count" and no other context or insight into why this might be the case or the trial results that led to this value. When conducting empirical experiments, it is best to seaprate the resulting data and the analysis. This means that a researcher should save the LLM's outputs in a CSV, and then use a separate file, perhaps a Jupyter Notebook, to actually conduct analysis on these results or compute statistics. As this analysis occurs, new points of research might occur, and you will probably want to look more closely at the results at the trial beyond the summary statistics, so it is helpful to have the raw data. From a more practical standpoint, if your laptop closes or stops running, then if you are saving to a CSV, it will simply later pick up where it left off, rather than having to restart the program, which is helpful.

- It is also super important to clean your data before you start to do experiments with, and make sure that it is in line with your expectations. Many times, I had to start over experiments because the data would be in a slightly unconventional format and not actually computing the value that I wanted. On that note, before doing loops of many trials, it is good to try to just do one trial of an experiment to make sure it is executing correctly and that the right question is being answers.

- What this project has made me most thankful for is the existence of .gitignore files. When conducting data-intense experiments like this one, it is hard to manage the files within a Github repository, since Git doesn't like large files over 100MB unless you specifically set up large file management. To work around this, you can place all data folders within a specific `/data` folder in your local directory and then add `/data` to your gitignore file. Make sure to push your gitignore file to the main branch you push the rest of the code though, otherwise you will, like me, have to learn a lot about how to unstage changes in Git. This is not an experience I would recommend.

9.3 How to Use the Cluster

1. Sign up for an account through Harvard FASRC. This should be approved within a couple business days, and will result in a new log-in, separate Harvard Key log in.

2. The cluster is accessed via SSH-ing through the terminal. Each time you access the cluster, you will have to put in your email, password, and verification code. I've found that the easier verification code platform is through this quaint Java-based authenticator.

3. To actually do commands, you need to `srun` the first command in the helpful commands list below, starting a session of a given time limit.

4. We run our jobs on "seas-gpu."

5. Often times, it can take a while to get allocation and for the jobs to load, so it is helpful to plan to run experiments overnight.

6. Here are some helpful SLURM commands:

    - `srun -pty -p test -mem 100 -t 0-01:00 /bin/bash`: This initializes a session, converting you from "login" status to "holy" status.
    - `source  /my_env/bin/activate`: Activate your environment, like normal
    - `sbatch run_experiment.slurm`: This submits your batch job, where the job is defined in the slurm file, and gives you a job id.
    - `squeue -u <user_id>`: This shows

References

[1] Henning Bartsch et al. Self-consistency of large language models under ambiguity. *arXiv preprint arXiv:2310.13439*, 2023.

[2] Cynthia Dwork and Christina Ilvento. Fairness under composition. *ArXiv*, abs/1806.06122, 2018.

[3] James Foulds, Rabeeh Karim Islam, and Kai-Wei Chang Keya. Bayesian modeling of intersectional fairness: The variance of bias. *arXiv preprint arXiv:1811.07255*, 2018.

[4] Christina Ilvento. Metric learning for individual fairness. *arXiv preprint arXiv:1906.00250*, 2019.

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