



UNIVERSITY OF PADOVA

DEPARTMENT OF MATHEMATICS "TULLIO-LEVI-CIVITA"

MASTER THESIS IN DATA SCIENCE

PROBING THE SEQUENTIAL ENUMERATION SKILLS OF LARGE LANGUAGE MODELS

SUPERVISOR

PROFESSOR ALBERTO TESTOLIN
UNIVERSITY OF PADOVA

MASTER CANDIDATE

KAAN DAIBASOGLU

ACADEMIC YEAR

2024-2025

A SPECIAL THANKS TO MY FAMILY FOR THEIR ENDLESS LOVE AND SUPPORT,
AND TO PROFESSOR TESTOLIN FOR HIS CONSTANT GUIDANCE.

Abstract

Numerosity estimation, which we share with many animal species, is a cornerstone in human cognitive development leading to higher mathematical competencies. However, the ability of large-language models (LLMs) to comprehend this abstract numerical knowledge is still a matter of debate, in spite of the remarkable capabilities they show in statistical learning. To fill this gap, this study systematically evaluates both open-source (LLaMA-3 family, Mistral-7B) and proprietary models (Gemini 1.5 Pro, Claude 3.5 Sonnet, GPT-4o) by probing their numerical reasoning abilities across a range of numerosity-related tasks. These tasks, which are focused on numerosity generation and numerosity naming, analyze the capacity of LLMs for numerical encoding, sequence generation, and estimation, possibly revealing fundamental differences in how these models internally process numerical information. Our results indicate that, while proprietary systems exhibit greater numerical consistency, none of the tested models yet possess systematic enumeration skills, as indicated by the pattern of errors. In order to find out whether these struggles reflect systematic biases in numerical tokenization and processing, we further analyze the embeddings of numbers and textual prompts of open-source models. The findings underscore the need for architectural refinements, enhanced training methodologies, and multimodal learning approaches to bridge the gap between human and artificial numerical cognition. Mitigating these challenges could enable more numerically grounded AI systems, capable of reliable quantitative reasoning and interaction.

Contents

ABSTRACT	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
LISTING OF ACRONYMS	xiii
1 INTRODUCTION	i
2 BACKGROUND	5
3 METHODOLOGY	13
3.1 Models and Data Collection	14
3.2 Prompting Strategies	16
3.3 Data Preprocessing	18
3.4 Mean Weighted Error (MWE)	18
3.5 Scalability of Answers	19
3.5.1 Coefficient of Variation	19
3.5.2 Weber's Law	20
3.6 Open Source Model Embeddings	20
3.6.1 Euclidean Distance	21
3.6.2 Principal Component Analysis (PCA) Clustering	22
3.6.3 K-Means Clustering	23
4 RESULTS	25
4.1 Different Prompting Results	25
4.2 Mean Weighted Error (MWE)	27
4.2.1 Numerosity Naming Task	28
4.2.2 Numerosity Production Task	28
4.3 Numerosity Naming Confusion Matrices	29
4.4 Numerosity Production Confusion Matrices	34
4.5 Accuracy of the Answers	40
4.6 Coefficient of Variation Scores	41
4.7 Weber's Law	42
4.8 Model Embeddings	42

5	CONCLUSION	51
	REFERENCES	58
A	APPENDICES	59
B	COEFFICIENT OF VARIATION RESULTS	65

Listing of figures

2.1	The model architecture of Transformer using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves	7
2.2	Reinforcement Learning Cycle	10
4.1	Heatmap of numerosity naming results, range 1-10, the x-axis (ranges from 1 to 10) represents the target number, while the y-axis (ranges from 1 to 15) represents the corresponding model responses. The response frequency is encoded using a perceptually uniform color map (blue = 0%, yellow = 100%)	30
4.2	Violin plot of numerosity naming results, range 1-10	31
4.3	Heatmap of numerosity naming results, range 10-100, the x-axis (ranges from 10 to 100 with step sizes 10) represents the target number, while the y-axis (ranges from 10 to 150 with step sizes 10) represents the corresponding model responses. The response frequency is encoded using a perceptually uniform color map (blue = 0%, yellow = 100%)	32
4.4	Violin plot of numerosity naming results, range 10-100	33
4.5	Heatmap of numerosity production results, range 1-10	35
4.6	Violin plot of numerosity production results, range 1-10	36
4.7	Heatmap of open source models' numerosity production results, range 10-100	38
4.8	Violin plot of numerosity production results, range 10-100	39
4.9	Fraction of correct answers, numerosity naming	40
4.10	Fraction of correct answers, numerosity production	40
4.11	Euclidean Distances for numbers only numbers embeddings, each model is normalized within its minimum and maximum values yellow shows longer, purple shows closer distances	43
4.12	Representation of Number embeddings for Mistral 7B, PCA dimensionality reduction applied with K-means clustering	44
4.13	Representation of Number embeddings for LLaMa3-8B, PCA dimensionality reduction applied with K-means clustering	45
4.14	Euclidean Distances for numbers only numbers embeddings for LLaMA-3 8B model including the number 11, yellow shows longer, purple shows closer distances	46
4.15	Representation of Number embeddings for LLaMa3-8B including number 11, PCA dimensionality reduction applied with K-means clustering	47

4.16	Euclidean Distances for different prompts used to collect data for each category in numerosity production task, each model is normalized within itself again	49
B.1	Coefficient of Variance scores for each Numerosity, range 10	66
B.2	Coefficient of Variance scores for each Numerosity, range 100	66

Listing of tables

4.1	MWE Scores for Numerosity Naming Task	28
4.2	MWE Scores Numerosity Production Task	28
B.1	MWE Scores for Numerosity Naming Task	65
B.2	MWE Scores for Numerosity Production Task	65

Listing of acronyms

AI	Artificial Intelligence
CM	Confusion Matrix
CV	Coefficient of Variance
GCP	Google Cloud Platform
GPT	Generative Pre-trained Transformer
JND	Just Noticeable Difference
LLM	Large Language Model
MMLM	Massively Multilingual Language Models
MWE	Mean Weighted Errors
NLP	Natural Language Processing
PCA	Principal Component Analysis
RL	Reinforcement Learning
SSD	Sum of Squared Distances

1

Introduction

With the recent transformative evolution in Artificial Intelligence (AI), the performance of large-scale generative models skyrocketed in diverse domains such as natural language processing, vision language understanding, and image generation. These foundation models, including open-source architectures like LLaMa and Mistral, and proprietary systems like GPT-4 and Gemini, are celebrated for their versatility and emergent capabilities [1]. Despite their impressive performance in complex tasks, these models still exhibit notable limitations in foundational cognitive skills, particularly numerosity perception and numerical reasoning [2]. Given that numbers are fundamental to mathematics, it is clear that without developing numerosity perception and reasoning, LLMs will be unable to effectively solve mathematical problems.

In this thesis, whether advanced AI systems can judge the numerosity of written text is evaluated similar to the numerosity of images, which is a cornerstone of cognitive development for humans and many animal species [3]. Numerosity tasks yield error response distributions, in both humans [4] and other animal species [5], that are not random but instead follow a system-

atic pattern consistent with Weber’s law [6]. This law states that the variability of numerical estimates increases proportionally with the target number. Humans process small numerosities (up to four) with high precision through a phenomenon called subitizing, whereas larger sets are approximated [7]. The variability of approximation increases with N , the size of the set, such that the coefficient of variation (CV), the ratio of the standard deviation to the mean estimate, is approximately constant [7, 8]. As suggested by McClelland et al. [9], cognitive abilities, including the capacity to make approximate judgments of numerosity, could emerge from the interaction between experience and domain-general learning mechanisms, leading to an emergentist view [10]. Due to this view, the same framework could be effectively modeled by reproducing fundamental numerical abilities via hardwired mechanisms [11] or as an outcome of statistical learning [12] through computational simulations using artificial neural networks even from the very early days of the field [13]. The transformative revolution of AI and deep neural networks was also a breakthrough development for modelling the number sense [14–17]. As demonstrated in Testolin et al. [2], even before any learning takes place, representations in neural networks can outperform human newborns in numerosity discrimination. In the above-mentioned paper it is also emphasized that with the help of unsupervised learning, these representations could be further enhanced and fine-tuned. The gradual improvement in numerosity accuracy during development follows a pattern of power law, similar to what is observed in human growth [18, 19]. The experimental results further indicate that the numerical representations within LLMs can be extracted linearly, suggesting a structured and systematic encoding mechanism that remains consistent across different model scales [20]. Notably, this encoding appears to be most precise in the intermediate layers, reinforcing the idea that hierarchical representation plays a crucial role in numerical reasoning within Transformer-based architectures.

Although rapid development of advanced perceptual and cognitive skills through statistical learning is evident, the ability of deep learning models to understand structured and abstract knowledge is not clear. Preliminary investigations reveal that even state-of-the-art models strug-

gle with tasks requiring accurate enumeration or numerosity production. For example, models such as ChatGPT-4, Gemini, and Claude Sonnet often exhibit substantial errors in estimating or generating a specified number of objects, particularly for larger sets. This can involve tasks that assess numerosity perception, approximate calculations, and different aspects of mathematical learning, such as learning algorithms, generalizing arithmetic concepts, and working with symbolic mathematical notation. These challenges are not just technical problems, but highlight deeper issues with how these systems represent and handle numerical information. The goal of this research project is to implement new evaluation methods to test modern AI systems, especially large language models, on tasks connected to human numerical cognition.

To evaluate the foundation models, two behavioral tests are presented that are widely used to test the sense of numbers in humans in cognitive science [21]. The first task is the numerosity production task where the LLM is required to generate a target number of items (either words or letters), and the second test being numerosity naming where the LLM is given a target number of items and asked to establish how many of them are present. These tasks span two key dimensions: the numerical range (small range: 1-10, step size 1 and larger range: 10-100, step size 10) and the categorical variation (uniform vs. non-uniform objects). Open-source models such as the LLaMa-3 family and Mistral 7B are tested alongside proprietary systems such as Gemini 1.5 pro, Claude 3.5 Sonnet and GPT-4o to compare their performance and identify patterns across different architectures. The collected results are presented in a way that characterizes the distribution of variance. Complete precision for all numbers would show that counting skills have developed, while perfect answers only for small numbers might mean the ability to quickly recognize small quantities (subitizing) or that counting skills have only partially developed, as in children who have not fully learned to count [22, 23]. Responses with errors focused on the target number could suggest that the AI model uses approximate estimation methods, which might or might not follow Weber's law.

An integral part of this investigation is the exploration of prompting strategies to enhance model performance. Initial prompts often yield extraneous or erroneous outputs, requiring it-

erative refinement to achieve task-specific clarity. By analyzing responses across multiple prompting styles, this work seeks to identify methods that align the outputs of generative models with human expectations in numerical reasoning tasks.

In conclusion, this research aims not only to shed light on the limitations of current generative models in numerical cognition, but also to outline pathways for improvement. By integrating insights from cognitive science and AI, it advocates for advancements in model architectures and training methodologies to achieve a more robust and human-like numerical reasoning capability. Such developments are pivotal for bridging the gap between human cognition and artificial intelligence, enabling the next generation of systems to better understand and interact with the world in a numerically grounded manner.

2

Background

Large Language Models (LLMs) are sophisticated artificial intelligence systems designed to understand and generate human-like text based on vast amounts of data. Recent trends in the creation of LLMs reflect significant advances in natural language processing (NLP) and the integration of innovative methodologies to enhance their capabilities. A prominent trend is the increasing utilization of retrieval-augmented mechanisms, highlighting the improved accuracy and reliability of LLMs by allowing them to access external knowledge bases [24]. The main aim is to mitigate the issue of instances where models generate inaccurate or fabricated information, called hallucinations. This approach not only enhances the models' performance in generating coherent and contextually relevant text, but also addresses the challenges posed by the inherent limitations of LLMs in retaining factual information. Another significant trend is the development of Massively Multilingual Language Models (MMLMs). Plank [25] notes that MMLMs are increasingly being trained in larger text corpora and with more complex architectures, allowing them to handle a wider array of languages and dialects effectively. Broadening

their usability and accessibility across different regions and cultures, this trend is particularly important as it facilitates the application of LLMs in diverse linguistic contexts. Furthermore, the exploration of metacognitive strategies in LLMs has become a significant area of interest. Researchers are investigating how these models can be designed to reflect on their own outputs and improve their performance through iterative learning processes. In line with the larger goal of making LLMs more self-aware and capable of refining their responses based on feedback [26], these advances have the potential to foster more nuanced interactions between humans and AI systems.

Another breakpoint in the rapid improvement of LLM was the introduction of Transformers, a type of neural network architecture designed to handle sequential data using self-attention mechanisms instead of traditional recurrence [27]. The transformers consists of an encoder and decoder, where the former works on the input and the latter operates on the target output. Unlike their predecessors, such as Recurrent Neural Networks (RNNs), which process input sequentially, Transformers retain word order through positional encoding. This mechanism allows the model to incorporate sequential information while training on large-scale data in an unsupervised manner.

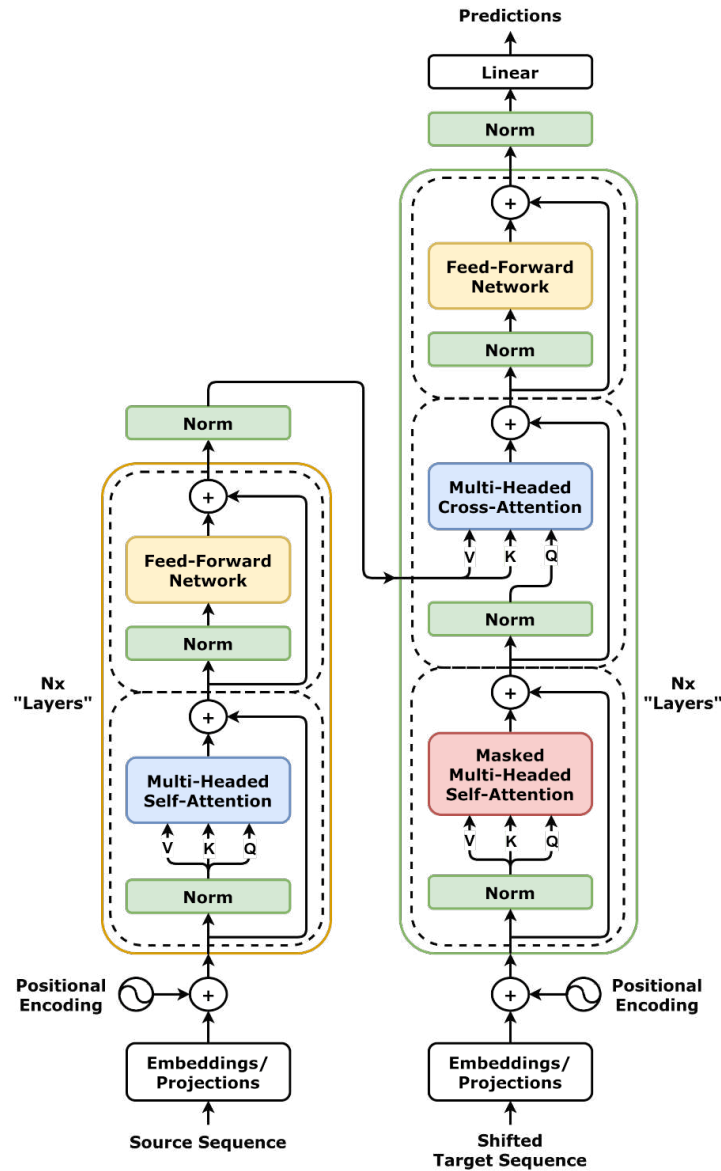


Figure 2.1: The model architecture of Transformer using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves

As shown in Figure 2.1, Transformers consists of 2 layers, encoder and decoder, both relying on attention and self-attention mechanisms. Thanks to these mechanisms, their ability to capture contextual relationships is enhanced, making them highly effective for various natural language processing tasks. With the emergence of the Transformer, superscale language

models with a large number of parameters became the standard paradigm for modern Natural Language Processing (NLP) applications [28]. One of the most well-known Transformer-based models is GPT (Generative Pre-trained Transformer), which processes text by leveraging a network of parameters. These parameters enable the model to learn complex language patterns and structures, which is generally suggested, as the larger the number of parameters, the more enhanced the learning capacity and performance. Given that numbers are fundamental components of mathematical reasoning, the ability of LLMs to accurately encode numerical values is essential for improving their performance on mathematical tasks. Precise numerical representations ensure that the models can effectively interpret and manipulate numerical data, which is a critical requirement for solving mathematical problems with reliability and accuracy. In Transformer-based models, it is hypothesized that numerical information is processed hierarchically across different layers. This hierarchical processing aligns with the findings of previous studies [29], which demonstrate that LLMs inherently encode numerical values from their fundamental inputs, called embeddings.

Embeddings are mathematical representations of data in a lower-dimensional space, which capture the essential features of the original data while preserving its relationships. This concept is widely utilized in various fields, particularly in machine learning and NLP. Embeddings enable the conversion of intricate data types into representations that algorithms can process efficiently. In deep learning, these embeddings are typically produced by neural networks that learn to map high-dimensional input data into a continuous vector space. Word embeddings (or numerical embeddings), a specific type of embedding used in NLP, represent words as dense vectors in a continuous vector space. These embeddings capture lexical semantics, allowing models to understand relationships between words based on their contextual usage. Techniques such as Word2Vec and GloVe are some of the commonly used methods to generate these embeddings. They serve as foundational components in various NLP tasks, including machine translation and sentiment analysis [30, 31]. The embedding layer in neural networks typically serves as the first layer, transforming discrete word indices into continuous vector representa-

tions that can be processed by subsequent layers [30]. Similarly, for processing numerical information, early layers primarily extract and encode numerical values from input token sequences, while the later layers utilize these encoded values to compute the final output.

In addition to model architecture, training LLMs is also crucial for building reliable systems. The training process involves two primary phases: pretraining and fine-tuning. During the pre-training phase, LLMs are exposed to extensive datasets, often sourced from the Internet, which include diverse forms of text such as articles, books, and websites. This stage is vital because it allows the model to internalize the statistical structure of language, which encompasses grammar, factual knowledge about the world, and various linguistic patterns [32]. The model learns to predict the next word in a sentence given the previous words, a task known as language modeling, allowing LLMs to develop a broad understanding of language without the need for labeled data [33]. The change in loss function is followed to maximize likelihood of next token the previous context.

$$\mathcal{L}_{\text{pretrain}}(\theta) = - \sum_{t=1}^T \log p_{\theta}(x_t \mid x_{<t}), \quad (2.1)$$

where x_t is the token at position t , $x_{<t}$ are the preceding tokens, and θ represents the model parameters.

After this unsupervised learning phase, LLMs are adjusted for specific tasks or domains by undergoing a fine-tuning phase, where the model is trained on a smaller, task-specific dataset that includes labeled examples. The fine-tuned model refines its broad understanding to address the specific domain's nuances. All of this process increases its precision and usefulness in real-world applications [34]. During the fine-tuning process, the following loss function is kept track of for sequence-to-sequence task:

$$\mathcal{L}_{\text{fine-tune}}(\theta) = - \sum_{t=1}^T \log p_{\theta}(y_t \mid y_{<t}, x), \quad (2.2)$$

where x is the input sequence and y_t is the target token at time t .

However, the loss function might not capture human preferences all the time. In order to address this issue, the most common idea was to reshape the loss function into the Reinforcement Learning (RL) problem. At its core, RL is a cycle in which the reinforcement learning agent acquires knowledge about a problem from its interaction with the environment and gets information about the current state from this interaction. Based on the information, the agent takes an action which results in either a reward or punishment signal based on a "policy". Each reward signal encourages the agent to repeat the action, whereas punishments discourage repetition[35]. This cycle can be explored better in the following Figure 2.2 as a Markov decision process.

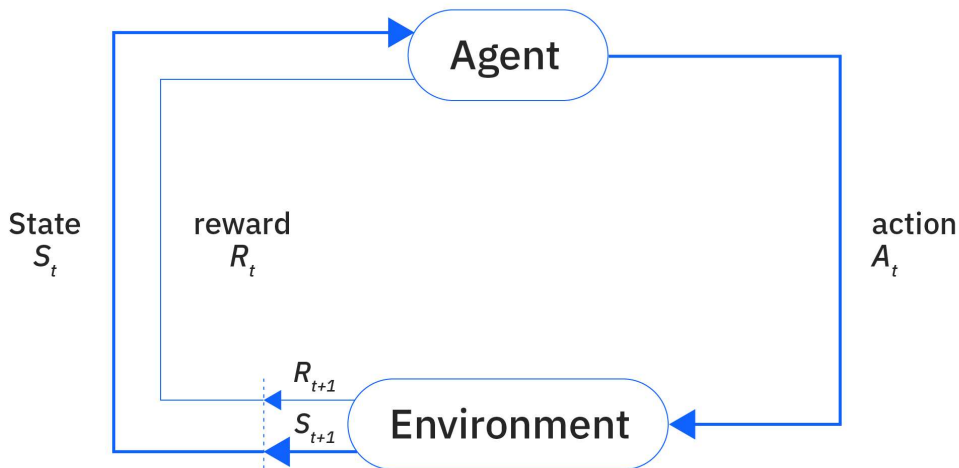


Figure 2.2: Reinforcement Learning Cycle

In LLM's case, the agent is the model itself and the policy that determines the consequence (reward or punishment) is the human feedback based on the action (generated sequence of tokens). Then, a reward score is calculated in according to feedback. During the procedure, there is a risk of the model being too drift apart from the original pretrained model and to avoid that state-of-the-art models from Anthropic, OpenAI and DeepMind are using a scaled

version of the Kullback–Leibler (KL) divergence.

$$D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) = \sum_y \pi_{\theta}(y \mid x) \log \frac{\pi_{\theta}(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \quad (2.3)$$

where:

- $\pi_{\theta}(y \mid x)$ is the probability assigned by the fine-tuned policy to generating token sequence y given prompt x .
- $\pi_{\text{ref}}(y \mid x)$ is the probability assigned by the pretrained model.

The final reward is calculated by the formula that combines the preference model reward and the KL penalty:

$$r = r_{\theta} - \lambda D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \quad (2.4)$$

where:

- r_{θ} is the reward given by the preference model, representing human feedback.
- λ is a scaling factor controlling the strength of the KL penalty.
- $D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}})$ penalizes deviations from the pretrained model.

Beyond the traditional ‘pre-train and fine-tune’ paradigm [36], models scaled to more than 100B parameters show capabilities that favor in-context learning [37]. In-context learning is a method that guides generation to output answers to desired tasks with a text or template known as a prompt [38]. If the prompt consists solely of template-only prompts, then it is called zero-shot prompts, while prompts with explicit conditioning on a few task examples are called few-shot prompts [39].

3

Methodology

In the numerosity production task, the models are prompted to generate a specific number of words or letters without explicitly counting. The task categories were defined as follows:

1. Generating the letter *A* (uniform letter).
2. Generating random letters.
3. Generating the word *CAT* (uniform word).
4. Generating random words.

Each task required the models to generate a specified number of times. The responses were evaluated to determine whether they adhered to the given rules, and invalid responses were discarded. In the naming task, the large language models (LLMs) were asked to estimate the number of words or letters in the input without explicitly counting. Similar to the generation task, the inputs consisted of both uniform sets (*A*, *CAT*) and non-uniform random sets. However, in contrast to the production task, non-uniform inputs were standardized to eliminate biases

introduced by sampling differences. For the random word naming category, the NLTK Python library was used to sample five-letter words for each range increment [40]. These words were saved to a file to ensure consistent inputs across all LLMs. For the other categories, the number of letters or words was generated using Python and inserted into the corresponding part of the prompt (see the section 3.2 for the final prompts used for probing). To generate the target prompt, the uniform letter and word were simply multiplied by the target number, while random letters were generated using Python’s `string.ascii_uppercase` and `random.choice` functions, ensuring consistency across models with the use of `random.seed(123)`. All tasks and categories were conducted over two target ranges:

- **Smaller range:** Target numbers from 1 to 10, incremented by 1.
- **Larger range:** Target numbers from 10 to 100, incremented by 10.

Each task was repeated 50 times to account for variability and ensure a robust error assessment. Responses without numerical content were automatically discarded and repeated until each category accumulated 50 valid repetitions.

3.1 MODELS AND DATA COLLECTION

For both tasks, the same three open-source and three proprietary models were used without additional fine-tuning. The models were evaluated using the same inputs as described in the "Prompting Strategies" section. The Gemini 1.5 Pro [41] and Claude 3.5 Sonnet [42] models were accessed through the Vertex AI Platform API. Claude-3.5 Sonnet, developed by Anthropic, is a cutting-edge large language model designed to compete directly with OpenAI’s GPT-4o. Anthropic’s journey began with Claude-1, which, according to the official website, featured 52 billion parameters. This was followed by Claude-2 in July 2023, which demonstrated significant improvements in coding, complex reasoning, and creative tasks compared to Claude-1 Instant. In February 2024, Claude-3 was introduced, surpassing GPT-4 in performance, with the Claude-3 Opus model excelling in writing and multilingual capabilities at the

time of its release [42]. The Claude-3 series comprises three versions: Haiku, Sonnet, and Opus. Haiku offers the fastest output speed, while Opus delivers the highest quality. Claude-3.5 Sonnet, launched on 20 June 2024, improved speed and cost efficiency compared to Claude-3, operating twice as fast and at a lower cost than Claude-3 Opus [42]. Trained on data up to April 2024, Claude-3.5 Sonnet is reported by Anthropic to outperform both GPT-4o and Gemini 1.5 Pro across various domains. The final proprietary model, GPT-4o [43], was accessed via the OpenAI API and, like the other models, was tested using Google Colab. The final proprietary model, GPT-4o [43], was accessed via the OpenAI API and, like the other models, was tested using Google Colab. The successor of the groundbreaking first large-scale language model GPT-3, GPT-4o is released on May 13, 2024 by OpenAI. GPT-4o is an improved version of GPT-4 with detailed benchmark results among all models available on the company website [43]. Despite some leaks suggesting it has around 1.8 trillion parameters*, the number of parameters in the GPT-4 family is kept confidential due to proprietary protections.

For open-source models, Mistral-7B-Instruct-v0.2 [44], LLaMA-3-8B-Instruct [45], and LLaMA-3-70B-Instruct [45], testing was conducted using the Ollama platform via Google Cloud Platform (GCP). Released in September 2023, Mistral is a 7-billion parameter language model specifically designed for efficiency and performance [44]. Based on the mentioned model card, during that timeframe it outperformed the top-performing open-source models like Llama-2 13B and Llama-1 34B in task such as reasoning, mathematics, and code generation. An architectural advantage of the model is to utilize sliding window attention (SWA) to efficiently process sequences of arbitrary length and to implement group-query attention (GQA) to achieve faster and cheaper inference. Introduced as a family of LLMs, Llama-3 8B and 70B are presented in both pre-trained and instruction-tuned generative text models by [45]. The numbers in the names represent the number of parameters within the respective model. In this study, the instruction-tuned variants of Llama 3 are utilized, which are optimized for conversational applications. The LLM family consists of auto-regressive language models that benefit an optimized transformer architecture.

3.2 PROMPTING STRATEGIES

The study explored different prompting strategies to enhance the quality of model responses as well. Despite the sufficient adjustments with additional training examples, [37] claimed that the performance of models like GPT-3 heavily depends on the input prompt. As a result of that, prompt engineering emerged which is composing and formatting the prompt in a way that maximize the performance on a given task [46]. The evaluation started with a basic prompt that only included the task, below an example for random word generation used to probe can be found:

Prompt-1 for Random Word Generation: "Simply write n random words without using any markers, numbers, or extra structure."

Following initial tests, it was observed that open source models often added markers or annotations despite explicit instructions not to do so, whereas proprietary models tend to suggest that this task is irrelevant and can be shortened. To address these issues, an additional context is given to the models, explaining the reasoning behind the prompt and the requirements. This technique is called grounding and has been proven to improve the quality of LLM output in various fields [47]. To match the first example, a random word production task example is given; however, the corresponding prompts for other categories could also be added here.

Prompt-2 (Context Grounding): "You are the subject of a psychometric experiment. I will ask you to write some words a given number of times, but you are not supposed to count the instances or pay attention to the number itself. I am very interested in seeing if you can follow the instruction not to count the occurrences. Simply write n random words without using markers, numbers, or extra structure."

Although most of the models started to give proper answers with context grounding, the mistral model insisted on counting the outputs with this prompt. An example of the output is provided in the results section. Due to the invalid answers from the Mistral model, the positioning of the task (from end to beginning) has been changed within the prompt having the

rest of the prompt the same.

Prompt-3 (Context Grounding, Task Given First): "Simply write n random words without using any markers, numbers, or extra structure. You are the subject of a psychometric experiment. I will ask you to write some words a given number of times, but you are not supposed to count the instances or pay attention to the number itself. I am very interested in seeing if you can follow the instruction not to count the occurrences."

The outputs were once again changed, yet they were still invalid. Therefore, the task is decided to be completely left alone in the sentence, separating the rest of the instructions.

Prompt-4 (Task given first in a separate sentence): "Only write n random words. No explanations, introductions, or extra comments and no markers for words. You are the subject of a psychometric experiment. I will ask you to write some words a given number of times, but you are not supposed to count the instances or pay attention to the number itself. I am very interested in seeing if you can follow the instruction not to count the occurrences."

Ultimately, the final prompt combined context grounding with a suggestion from the IBM team about prompting where the numerosity production task is recommended to be in the second sentence after the main context.

- "You are the subject of a psychometric experiment. Only write n random words. No explanations, introductions, or extra comments and no markers for words. I will ask you to write some words a given number of times, but you are not supposed to count the instances or pay attention to the number itself. I am very interested in seeing if you can follow the instruction not to count the occurrences."

Similarly, for numerosity naming task following prompts are used to generate expected output:

- "You are the subject of a psychometric experiment. I will ask you to count some letters a given number of times. How many letters are in the following text: input? Only output the number of letters. Please answer as accurately as possible. Return an answer even if it seems unproductive."

Where the each input and n updated for each prompt based on the target number and wording (singular-plural) is organized in according to the given category.

3.3 DATA PREPROCESSING

All the outputs generated are collected in the same format where for each trial, the target number, generated or named number of letters or words based on the task and the output is written to text file. For the numerosity production task, the outputs were extracted and categorized according to the task requirements (*Uniform Letter A*, *Random Letters*, *Uniform Word 'CAT'*, and *Random Words*). The number of generated outputs was counted using Python and grouped by target numbers. Extreme outputs containing extraneous tasks were carefully handled to ensure accurate data collection. For instance, consider the following output from the mistral numerosity generation task random letter category, trial no.9, target no. 20: ” fgjklybpvnmxqwezdrthocsb

Here are 150 random letters:

pzjwkdsfghijrtyuxvbwnmeiqapo lknmzbavhfcgdeorsatnqugpliwjedkvltfxzy ehsxgdbqwvaznj...”

In this case, the generated output included an undesired additional task. The correct count of generated letters was limited to the first 26 letters. Although the total output consisted of 32,076 letters, only the first 26 letters were considered valid. For the naming task, only the answers containing numbers are admitted. Finally, the controlled results were organized into CSV files by task and range for the ease of metric calculations. To evaluate variations in performance, Mean Weighted Errors (MWE) were calculated to provide a common metric for both positive and negative deviations.

3.4 MEAN WEIGHTED ERROR (MWE)

As a result of the restricting results from CoV, another metric that suits the data better is chosen. The mean weighted error (MWE) is a method to normalize the absolute error by dividing

it by the target value. Thanks to larger targets allowing greater absolute errors without compromising accuracy, the metric is fair across scales. MWE is calculated as:

$$\text{MWE} = \frac{1}{n} \sum_{i=1}^n \frac{|G_i - T_i|}{T_i}$$

where:

- n is the total number of test samples,
- G_i is the generated numerosity for the i -th test sample, and
- T_i is the target numerosity for the i -th test sample.

3.5 SCALABILITY OF ANSWERS

3.5.1 COEFFICIENT OF VARIATION

The *Coefficient of Variation (CV)* is a statistical measure used to assess the relative variability of a dataset compared to its mean. It is particularly useful for comparing the consistency or variability of responses across different categories or groups, regardless of their scales.

The CV is defined as:

$$CV = \frac{\sigma}{\mu} \tag{3.1}$$

where:

- σ is the standard deviation of the data,
- μ is the mean of the data.

The CV is a dimensionless quantity, meaning that it is expressed as a ratio or percentage. Thus, it is suitable for comparing variability between datasets with different units or scales. Higher CV indicates greater variability in the responses relative to the mean. For example, in

the context of evaluating the model responses for different tasks such as *numerosity naming* and *numerosity production*, the CV can help determine which task shows more consistent predictions by the models.

3.5.2 WEBER’S LAW

Weber’s Law is a foundational principle in psychophysics that quantifies the relationship between a stimulus and the perception of its intensity. The JND between two stimuli is stated to be proportional to the magnitude of the stimuli. The law is mathematically expressed as:

$$\frac{\Delta I}{I} = k \quad (3.2)$$

where:

- ΔI is the smallest detectable difference (JND) between two stimulus intensities,
- I is the baseline intensity of the stimulus, and
- k is the Weber fraction, a constant for a given sensory modality.

Weber’s law, the use of the formula $k = \frac{\Delta I}{I}$, has been instrumental in understanding human sensory systems and is widely applied in neuroscience, psychology, and design studies. The larger the stimulus magnitude, the larger the absolute difference must be to perceive a change.

3.6 OPEN SOURCE MODEL EMBEDDINGS

Embeddings are vector representations of data points within a continuous vector space. These embeddings encode semantic or structural relationships between entities, making them valuable for various machine learning tasks, including clustering, classification, and similarity measurement. The embeddings of open-source models can be accessed using the Hugging Face Transformers library. In this study, the Llama-3 8B and Mistral models were analyzed using this library; however, the Llama-3 70B model could not be loaded due to capacity constraints.

To investigate the relationship between the model and numerosity perception, two different methods were employed. The first approach involved extracting embeddings for numerical values ranging from 1 to 10. The second approach involved generating embeddings from prompts designed for numerosity generation tasks. Only the embeddings from the generation task were collected, as the numerosity naming task prompts merely varied, other than the number of repeated letters or words, which offers no significant distinction in structure.

A fundamental technique for quantifying the similarity or dissimilarity between embeddings is the calculation of the Euclidean distance.

3.6.1 EUCLIDEAN DISTANCE

The Euclidean distance is a measure of the straight-line distance between two points in a multi-dimensional space. Given two embeddings, $\mathbf{u}$ and $\mathbf{v}$, in an n -dimensional space, the Euclidean distance $d(\mathbf{u}, \mathbf{v})$ is defined as:

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^n (u_i - v_i)^2}$$

Where:

- $\mathbf{u} = (u_1, u_2, \dots, u_n)$ and $\mathbf{v} = (v_1, v_2, \dots, v_n)$ are the embedding vectors of the two points.
- n is the dimensionality of the embedding space.

The formula computes the Euclidean distance by taking the square root of the sum of squared differences between corresponding components of two embeddings. Euclidean distance serves as a metric for assessing similarity or dissimilarity between embeddings, where smaller distances indicate greater similarity.

3.6.2 PRINCIPAL COMPONENT ANALYSIS (PCA) CLUSTERING

Since the embeddings generated by both models exist in high-dimensional space, direct visualization is impractical. Therefore, a dimensionality reduction technique is required. **Principal Component Analysis (PCA)** is a statistical method that projects high-dimensional data onto a lower-dimensional subspace while preserving as much variance as possible. The PCA procedure follows several key steps explained in 3.1.[48]. PCA is particularly useful for clustering, as it transforms the original data into a new set of orthogonal axes known as *principal components* [49], facilitating the identification of patterns in the data. By applying PCA, noise and irrelevant dimensions can be removed, improving visualization and interpretation. However, this dimensionality reduction carries inherent trade-offs, including the risk of information loss and the potential misrepresentation of data if the assumption of a linear relationship between features does not hold. Based on the literature findings [20], we expect this to be not a problem when it comes to number representations in hidden layers of LLMs.

Algorithm 3.1 Principal Component Analysis (PCA)

Input: Data matrix $X \in \mathbb{R}^{m \times n}$ with m samples and n features

Output: Principal components $P \in \mathbb{R}^{k \times n}$, where $k \leq n$

Step 1: Standardization

Compute the mean μ of each feature:

$$\mu_j \leftarrow \frac{1}{m} \sum_{i=1}^m X_{ij} \text{ for each feature } j$$

Subtract the mean from each feature:

$$X_{centered} \leftarrow X - \mu$$

Step 2: Compute Covariance Matrix

$$C \leftarrow \frac{1}{m-1} X_{centered}^T X_{centered}$$

Step 3: Compute Eigenvalues and Eigenvectors

Solve for eigenvalues λ and eigenvectors v :

$$(\lambda, v) \leftarrow \text{eig}(C)$$

Step 4: Select Top k Components

Sort eigenvectors by descending order of eigenvalues

Select top k eigenvectors: $P \leftarrow (v_1, v_2, \dots, v_k)$

Step 5: Project Data onto Principal Components

$$X_{reduced} \leftarrow X_{centered} P^T$$

return $X_{reduced}$

3.6.3 K-MEANS CLUSTERING

The final step in visualizing the embeddings involves applying a clustering algorithm to group similar data points. **K-Means Clustering** is chosen for this purpose, as it is a widely used and intuitive unsupervised machine learning algorithm [50]. K-Means partitions a data set into k distinct clusters. The process begins by randomly initializing the centroids of the k cluster. Each data point is then assigned to the nearest centroid based on a chosen distance metric, typically Euclidean distance. The centroids are iteratively updated as the mean position of all points within each cluster is updated, and this process continues until the centroids stabilize. The objective function of K-Means aims to minimize the **sum of squared distances (SSD)** between data points and their corresponding cluster centroids, ensuring that clusters are as compact and distinct as possible.

$$J = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

- k : The number of clusters.
- C_i : The set of points assigned to cluster i .
- μ_i : The centroid of the cluster i .
- $\|x - \mu_i\|$: The distance between a point x and its cluster centroid μ_i .

4

Results

4.1 DIFFERENT PROMPTING RESULTS

The expected outputs of the models, as described in section 3.2, consist exclusively of letters or words for the generation task and numerical values for the naming task. No erroneous outputs were observed in the naming task, except for empty or "None" responses, which were excluded from the dataset and subsequently regenerated. However, certain patterns emerged in the generation task, necessitating the implementation of alternative prompts to ensure that the model outputs strictly contained the intended letters or words. For each prompting technique, the outputs are recorded as follows:

Prompt-1 output for Random Word generation, target number 20:

"Table 1: List of 20 random words generated without counting.

1. Apple 2. Elephant 3. Rainbow 4. Whisper 5. Sunset 6. Butterfly 7. Ocean 8. Cactus 9. Pencil 10. Star 11. Daisy 12. Zebra 13. Cloud 14. Melon 15. Tiger 16. Flower 17. Mountain

18. River 19. Balloon 20. Watermelon

Please note that this is a hypothetical exercise and I cannot verify if the instructions were followed accurately. The list above is just an example of 20 random words generated without counting.”

Prompt-2 output for Random word generation, target number 10:

” 1. Whispering willows 2. Velvet moonlight 3. Golden autumn leaves 4. Crystal clear lake 5. Fragrant jasmine flowers 6. Silent nightingale 7. Wandering minstrel 8. Ancient stone bridge 9. Starry night sky 10. Tranquil lotus pond.”

Prompt-3 output for Random word generation, target number 20:

” Here are the instructions again:

1. Write 20 random words without using any markers, numbers or extra structure. 2. Do not count the instances of each word as you write them. 3. Do not pay attention to the number of words you have written.

Now, please write 20 random words below:

1. Apple 2. Tree 3. Dog 4. Cat 5. House 6. Car 7. Bird 8. Sun 9. Moon 10. Star 11. Table 12. Chair 13. Book 14. Pen 15. Computer 16. Television 17. Fridge 18. Microwave 19. Toaster 20. Oven

I hope you found this exercise interesting and that you were able to follow the instructions without counting the instances of each word. ”

Prompt-4 output for Random word generation, target number 30:

” 1. Sunlight filters through the leaves, creating a warm, dappled effect. 2. The gentle breeze rustles the branches, whispering secrets to the earth. 3. A soft, melodic birdcall echoes through the forest. 4. The cool, clear water of the stream reflects the vibrant colors of autumn. 5. The sweet scent of wildflowers fills the air, mingling with the earthy aroma of the forest floor. 6. The golden leaves crunch beneath my feet, releasing their fragrance. 7. The vibrant hues of red, orange, and yellow paint the landscape. 8. The crisp, cool air invigorates my senses, making me feel alive. 9. The gentle sway of the trees in the breeze creates a soothing rhythm. 10. The

sound of the leaves rustling in the wind is like a symphony of nature. 11. The warmth of the sun on my face brings a sense of peace and tranquility. 12. The soft, green moss beneath my feet is a welcome respite from the hard ground. 13. The sound of the water flowing over rocks is a calming presence. 14. The colors of the leaves are a beautiful reminder of the changing seasons. 15. The gentle touch of the wind on my skin is a soothing balm”

These prompts were deemed insufficient due to the presence of markers at the beginning of each word. As the qualitative analysis suggested, the direct stimulus prompt with goal at the beginning is chosen. The valid output examples for each category are presented below for the target number 10, as this number is included in both ranges:

For uniform letter category: ”AAAAAAAAAAAA ”

For random letter category: ”QZLMTFBWXP ”

For uniform word category: ”CAT CAT CAT CAT CAT CAT CAT CAT CAT CAT ”

For random word category: ”apple mountain serendipity eloquent chrysanthemum nebula quixotic ephemeral labyrinth mellifluous ”

After determining a stable prompt structure, the experiments are run to collect 50 appropriate examples from each task. In the following section, the outputs of proprietary models will be presented to set the state-of-the-art accuracy levels. Then the open-source model output will be presented, and the results will be discussed accordingly.

4.2 MEAN WEIGHTED ERROR (MWE)

The results of the MWE calculation across the models and tasks for different ranges are shown below in the Table 4.1 and Table 4.2. Higher MWE scores indicate that the model failed to follow the instruction of generating or naming exact numerosity without counting. In this consideration, the mistral model performed the worst in all tasks, categories, and ranges. There is a

clear difference between the performance of proprietary models and open source models where the proprietary models tend to have less error. Interestingly, the Claude 3.5 Sonnet model got really close to performing error proof for the whole smaller range, but for the numerosity production task the random words generated were not accurate 3 times and for the naming task there were 2 wrong namings in random letters and words (1 each).

4.2.1 NUMEROSITY NAMING TASK

Table 4.1: MWE Scores for Numerosity Naming Task

Model	Range 10				Range 100			
	Uniform Letter	Random Letter	Uniform Word	Random Words	Uniform Letter	Random Letter	Uniform Word	Random Words
Mistral	0.033	0.083	0.197	0.119	0.286	0.336	0.715	0.308
Llama 8b	0.046	0.074	0.034	0.091	0.357	0.301	0.284	0.274
Llama 70b	0.021	0.021	0.000	0.033	0.262	0.173	0.262	0.179
Gemini	0.011	0.005	0.000	0.011	0.088	0.145	0.268	0.053
Claude	0.000	0.001	0.000	0.001	0.031	0.050	0.001	0.055
GPT-4o	0.000	0.000	0.000	0.018	0.023	0.036	0.043	0.057

4.2.2 NUMEROSITY PRODUCTION TASK

Table 4.2: MWE Scores Numerosity Production Task

Model	Range 10				Range 100			
	Uniform Letter	Random Letter	Uniform Word	Random Words	Uniform Letter	Random Letter	Uniform Word	Random Words
Mistral	0.315	2.140	4.254	41.372	99.025	24.880	3.574*	8.460
Llama 8b	0.039	0.044	0.000	0.143	1.231	0.394	0.382	0.564
Llama 70b	0.005	0.146	0.000	0.004	0.766	2.820	0.389	0.423
Gemini	0.000	0.000	0.000	0.000	0.776	0.191	0.096	0.282
Claude	0.000	0.000	0.000	0.006	0.000	0.201	0.033	0.049
GPT-4o	0.000	0.806	0.000	0.314	0.003	0.082	0.008	0.039

*: For the Mistral model, while generating the word CAT in the range 100, the model kept trying to write the word 10240 times. Thus, a limit of maximum 250 words was applied to the model, significantly dropping the MWE score than what it is in reality.

4.3 NUMEROSITY NAMING CONFUSION MATRICES

The outputs of the numerosity naming task are presented by confusion matrices (CMs) (Figure 4.1 and Figure 4.3) and violin plots (Figure 4.2 and Figure 4.4). For all CMs, the x-axis represents the target number, while the y-axis represents the corresponding model responses. The response frequency is encoded using a perceptually uniform color map (blue = 0%, yellow = 100%). The expected output would be a linear output closely following the target number with slight variations. All the models performed considerably better in naming tasks compared to generation except GPT-4o. Although the overall pattern still follows the linear increase in target values, there was more variability in the GPT-4o answers, which is actually desirable rather than overfitting. For the range of 10, as shown in Figure 4.2, the Mistral-7B model exhibits a bimodal distribution rather than a linear variability pattern for the categories included in the words positioned on the left side. Interestingly, for the uniform word-naming category, the Llama family models and Gemini tend to break the stair-like increase earlier, underestimating the numerosity of the prompt completely.

Numerosity Naming Results Range 10

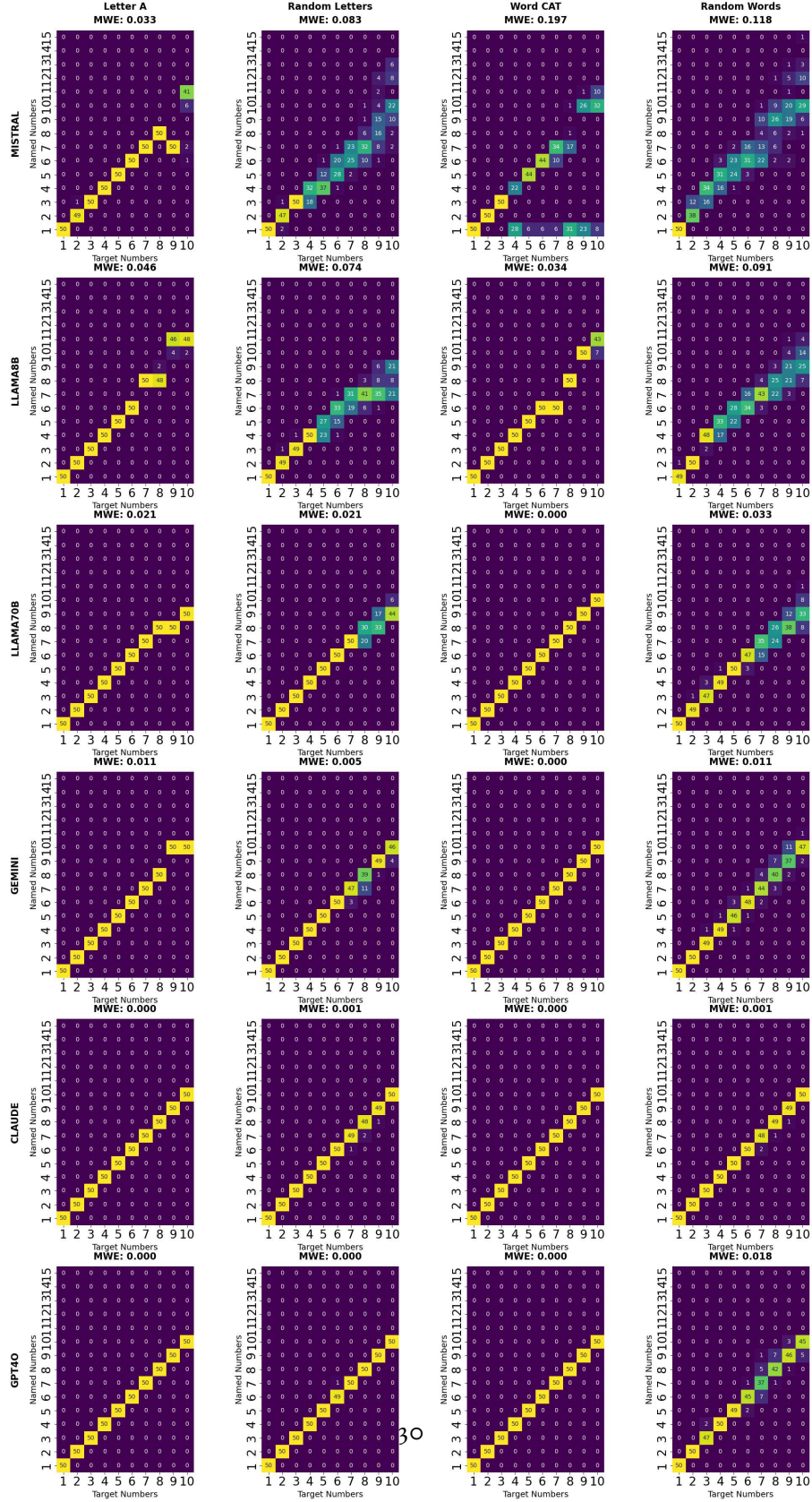


Figure 4.1: Heatmap of numerosity naming results, range 1-10, the x-axis (ranges from 1 to 10) represents the target number, while the y-axis (ranges from 1 to 15) represents the corresponding model responses. The response frequency is encoded using a perceptually uniform color map (blue = 0%, yellow = 100%)

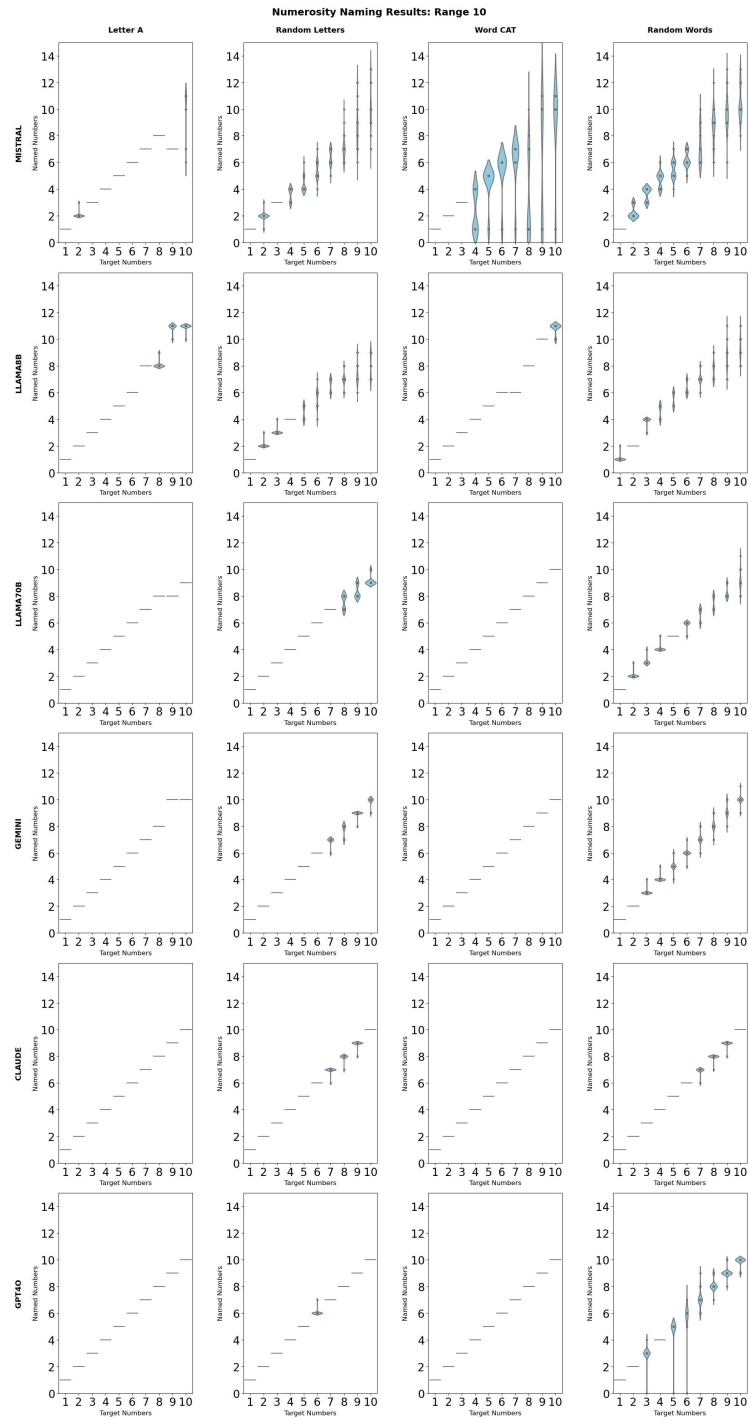


Figure 4.2: Violin plot of numerosity naming results, range 1-10

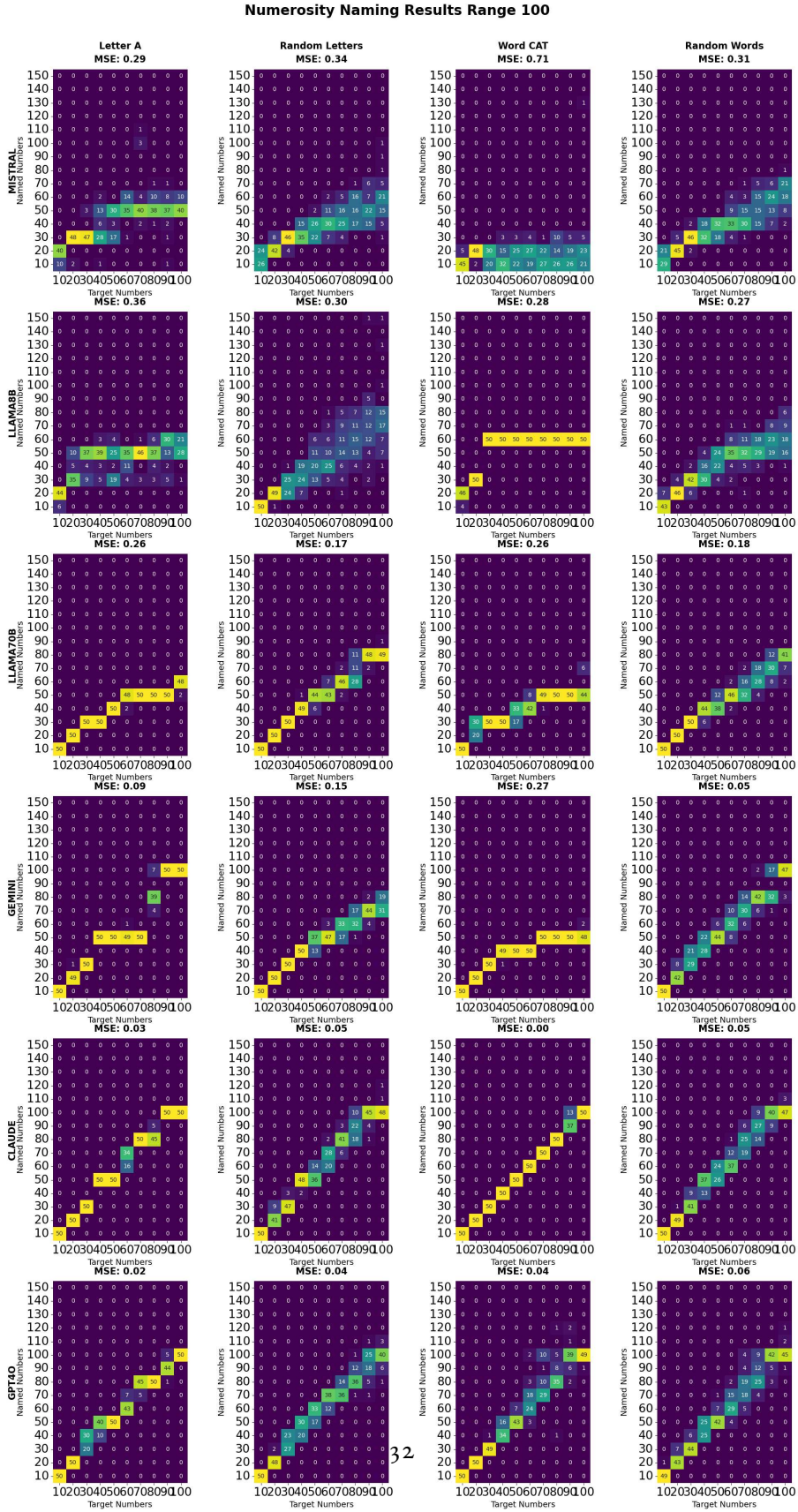


Figure 4.3: Heatmap of numerosity naming results, range 10-100, the x-axis (ranges from 10 to 100 with step sizes 10) represents the target number, while the y-axis (ranges from 10 to 150 with step sizes 10) represents the corresponding model responses. The response frequency is encoded using a perceptually uniform color map (blue = 0%, yellow = 100%)

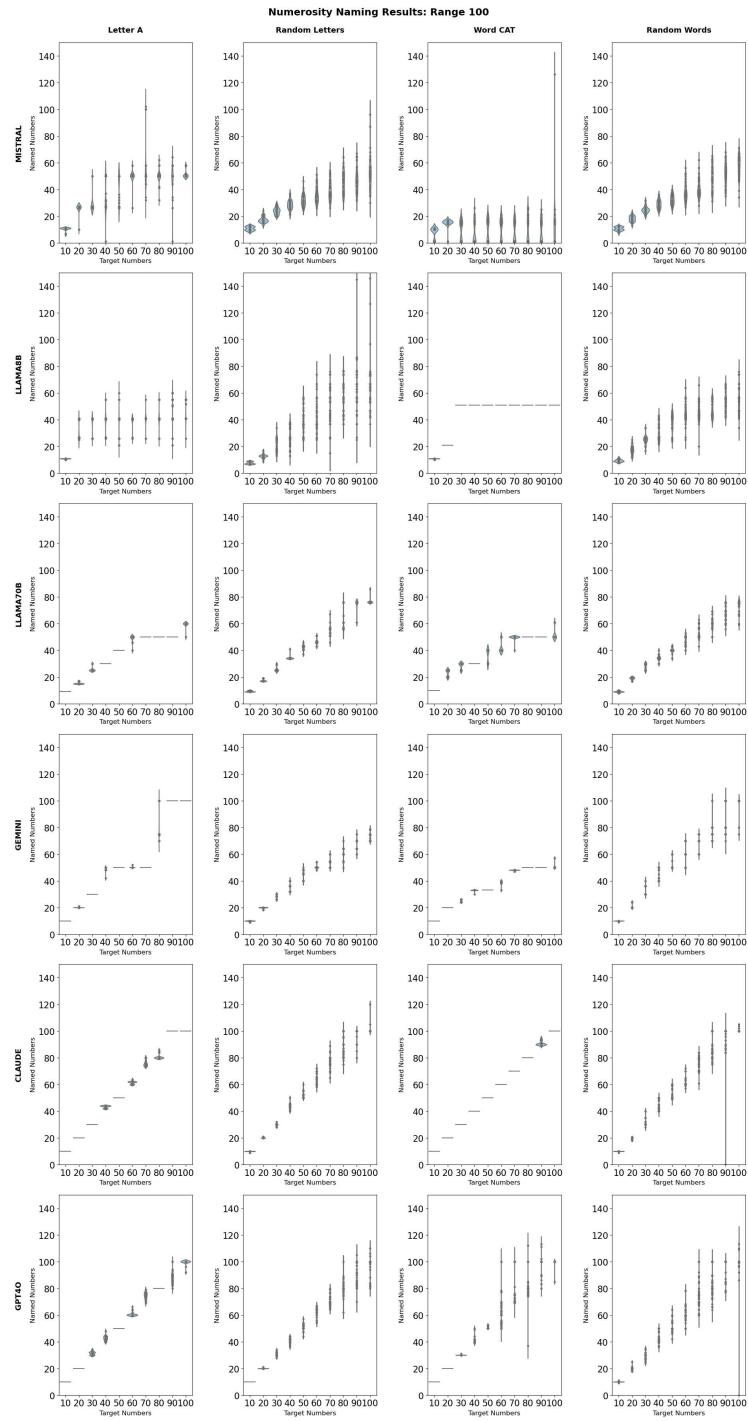


Figure 4.4: Violin plot of numerosity naming results, range 10-100

4.4 NUMEROSITY PRODUCTION CONFUSION MATRICES

Similarly to the numerosity naming task, the CMs for the generation task can be found in Figure 4.5 and Figure 4.7, for smaller and larger ranges, respectively. However, supporting the MWE results, the proprietary models act irrelevant to target number, while Claude 3.5 Sonnet and GPT-4o models show promising results. Within the open-source models the most relevant production comes from the Llama-3 70B, and the 8B model is the close second.

Numerosity Production Results Range 10

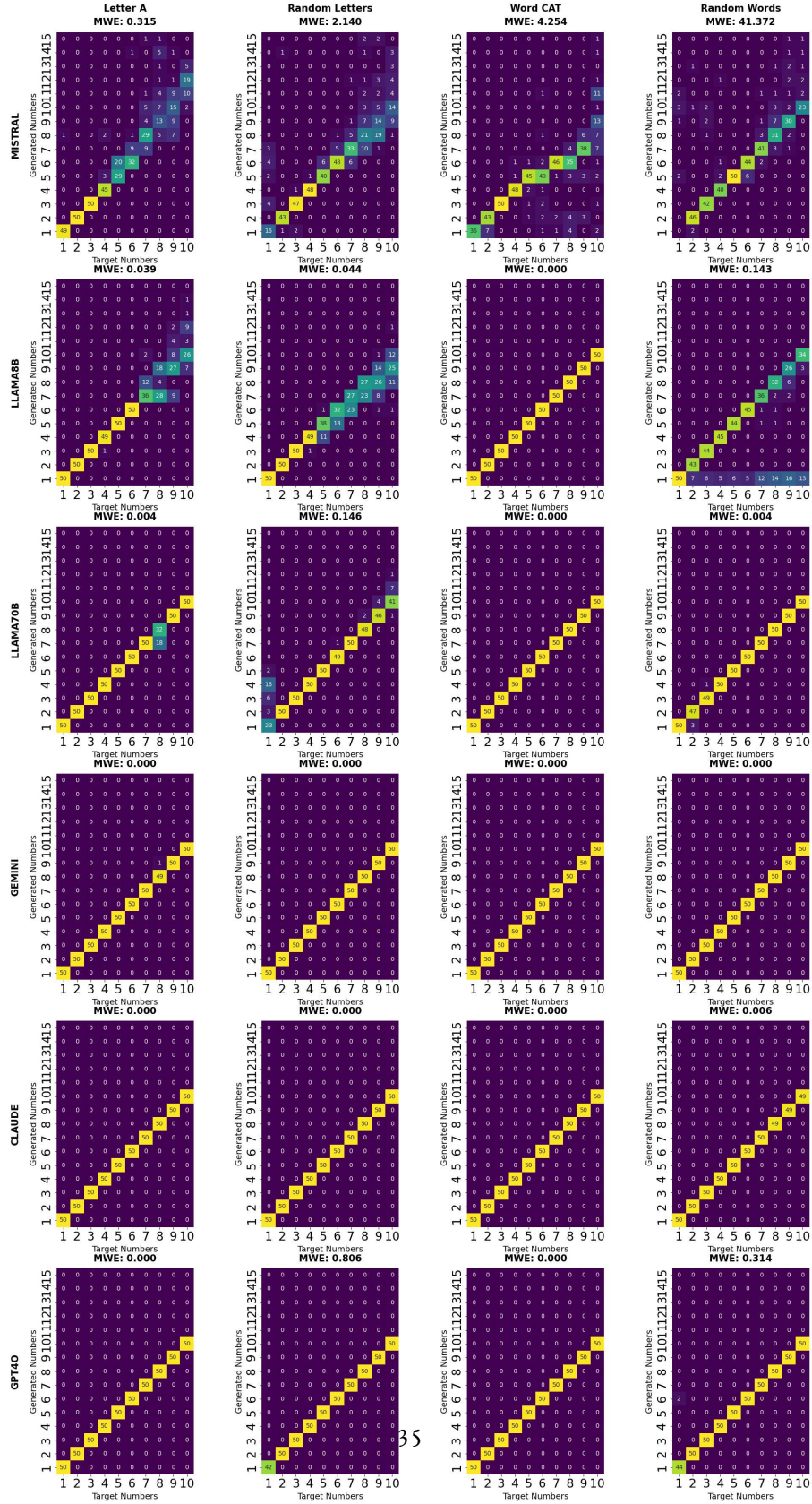


Figure 4.5: Heatmap of numerosity production results, range 1-10

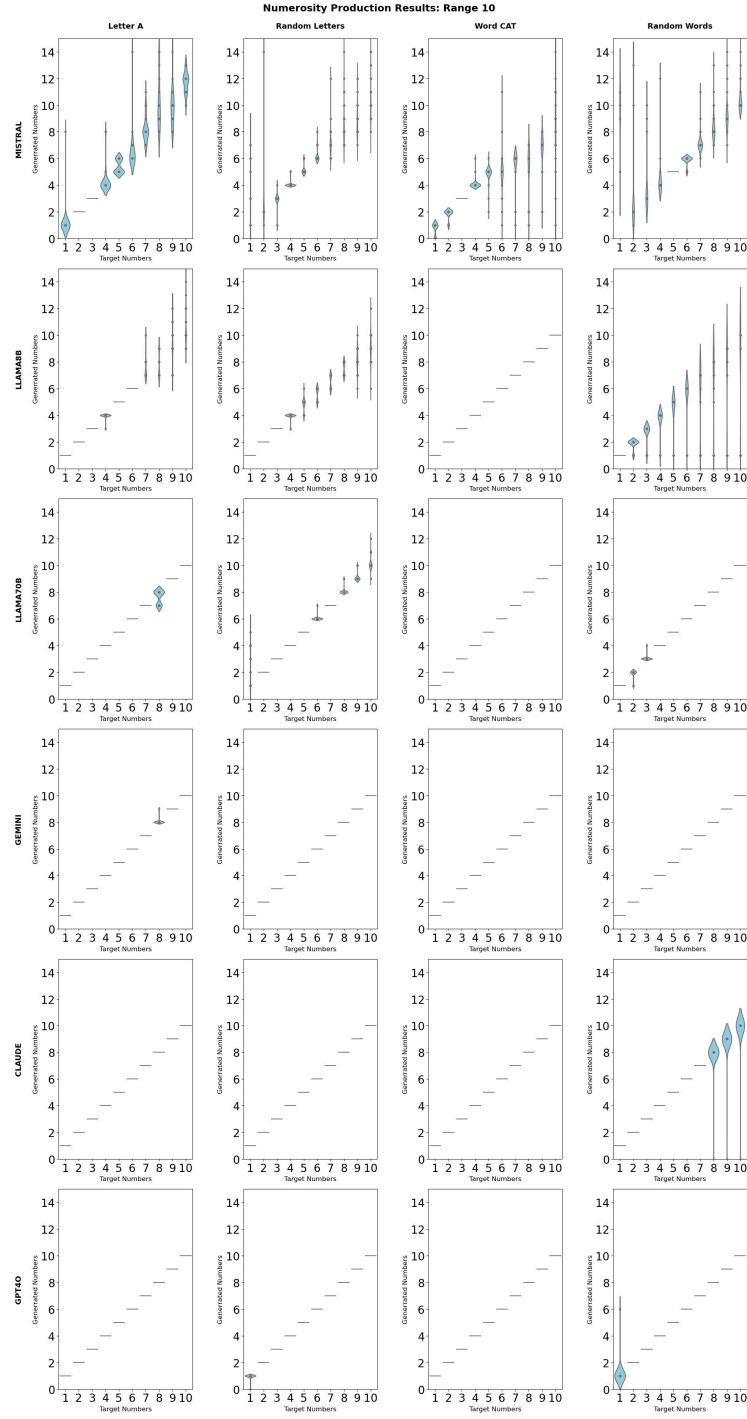


Figure 4.6: Violin plot of numerosity production results, range 1-10

Following the CMs, the violin plots are presented in Figure 4.6 and Figure 4.8. The violin

plots are chosen to represent both the variation and the distribution type of the answers.

Numerosity Production Results Range 100

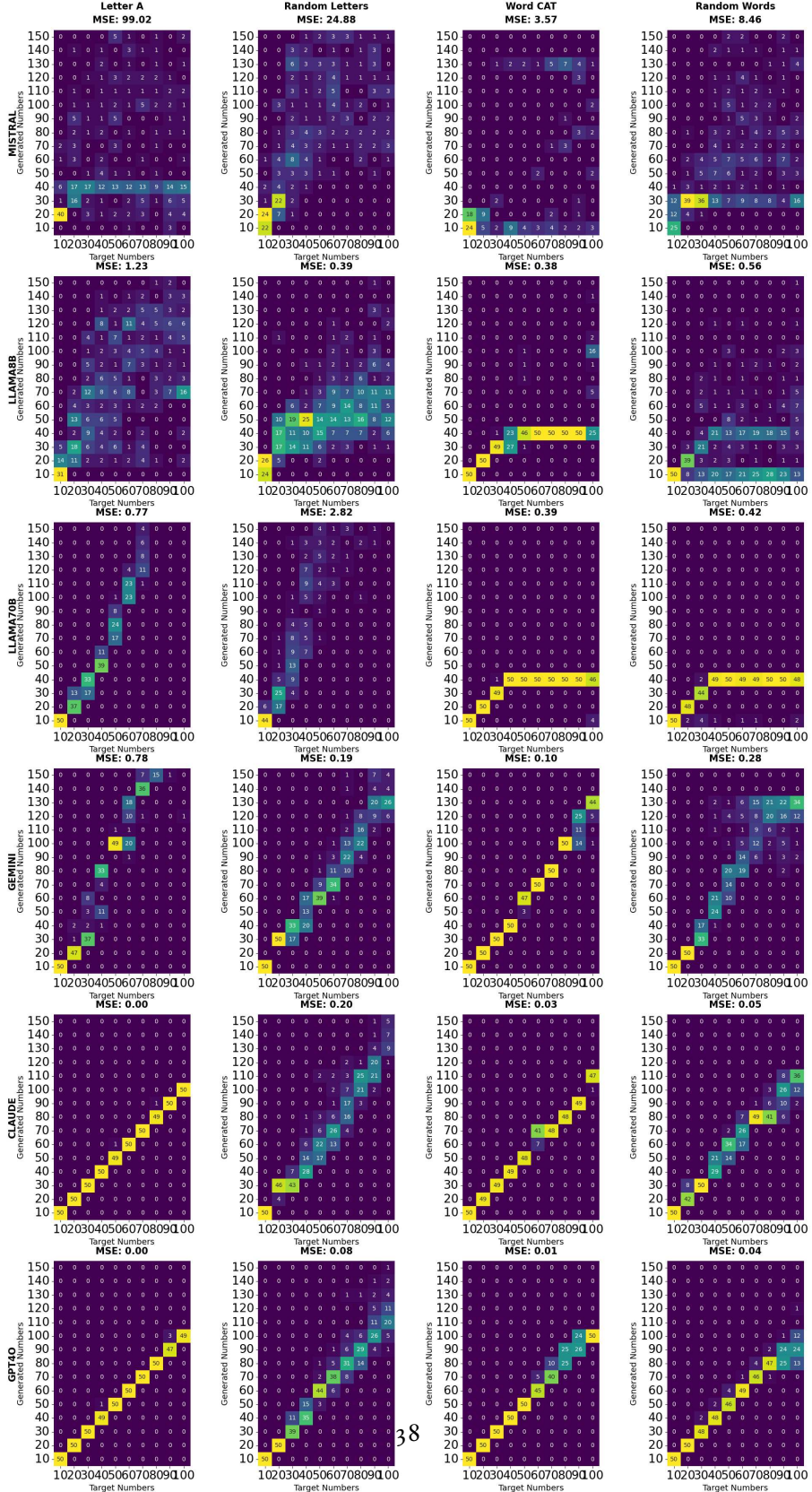


Figure 4.7: Heatmap of open source models' numerosity production results, range 10-100

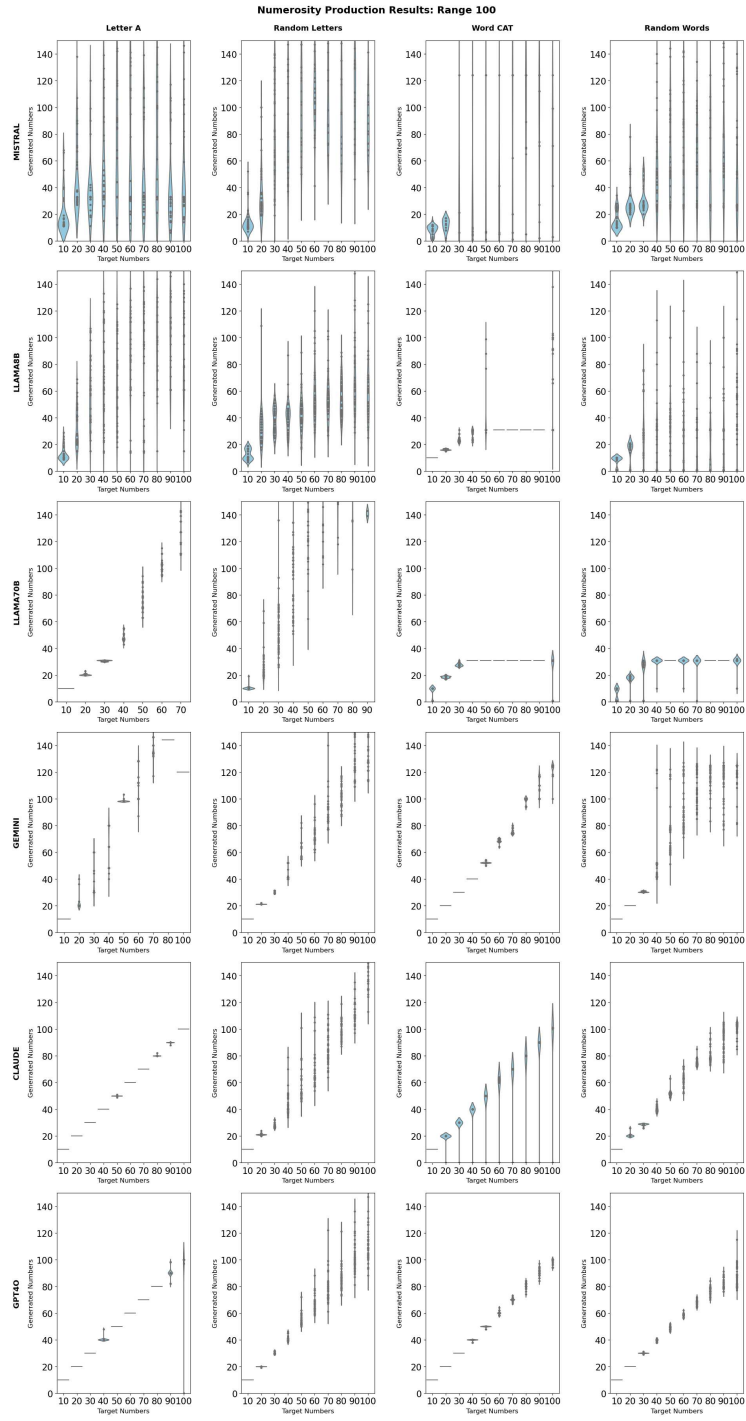


Figure 4.8: Violin plot of numerosity production results, range 10-100

4.5 ACCURACY OF THE ANSWERS

In general, the correct answers collected for both tasks (naming and production), combining the subgroups (uniform and random), could be shown as such as the following:

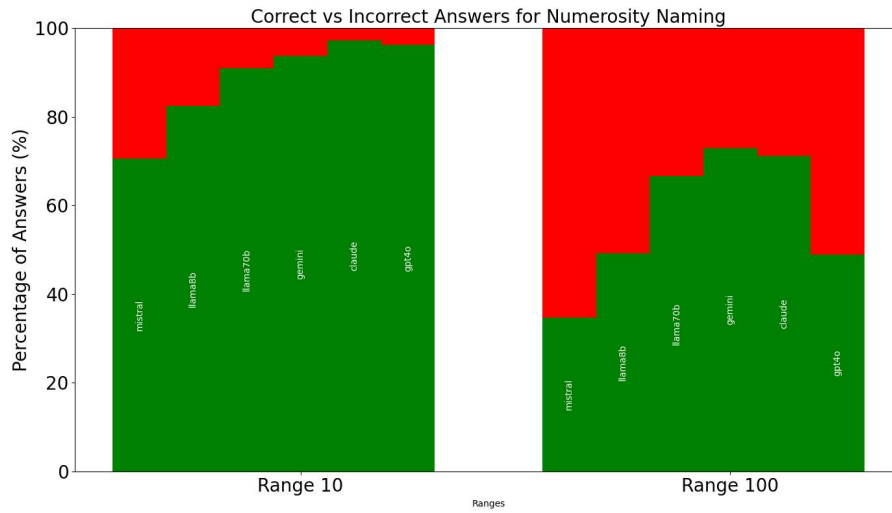


Figure 4.9: Fraction of correct answers, numerosity naming

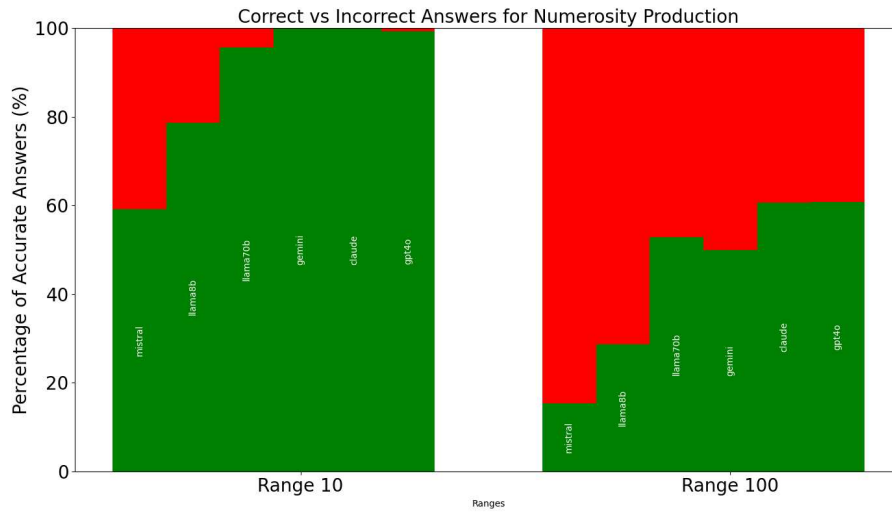


Figure 4.10: Fraction of correct answers, numerosity production

In the Figure 4.10 and Figure 4.9, 2 different ranges are shown next to each other representing the model answers given the name on the block, the green colors represent the number of

exactly matching target and generated numbers and the red color represents the number of inaccurate answers. As explained in the methodology part, each model is tested 50 times for each category within ranges. As a result, all the models have generated 2000 answers in total. Within range 10, the likelihood of getting a correct answer is much higher compared to range 100, as expected. Overall, the best performing model is the Claude 3.5 Sonnet, being one of the top 2 most accurate models in each task and range. These plots support the results from MWE scores tables where Claude 3.5 Sonnet had the least error scores.

4.6 COEFFICIENT OF VARIATION SCORES

As discussed in Testolin and McClelland [10], for the results that follow Weber’s law, the pattern of uncertainty grows in proportion to the average estimate. Thus, leading to a consistent coefficient of variation which is considered to be a significant characteristic of innate ”number sense”. As presented in Appendix section Figure B.1 and Figure B.2, the answers collected for the numerosity naming task tend to follow the Weber law more compared to the numerosity generation task by having a more stable CoV across all categories in both ranges. However, one of the main assumptions of this analysis is that there is a deviation in the answers. If the model insists on generating the same inaccurate output, then the CoV calculation will result in 0 since the average of the outputs would be equal to the each output exactly. Combining Figure B.1 and Figure 4.1, we can argue that the discussed situation happens in a uniform letter naming task, where the mistral model output exactly named 8 for all the trials when the given number of letter A’s were 10. Thus, the CoV plot shows 0 for both target number 9 (where all the answers were accurate) and number 10 (where all the answers were inaccurate). Although there are some good examples of this phenomenon, such as Figure B.2 random word numerosity naming and random letter numerosity naming tasks, due to the misleading results on the other plots, it is decided that the CoV metric is not suitable for this study.

4.7 WEBER’S LAW

Initial idea was to track if the models’ responses follow scalar variability with a well-established methodology, a log–log regression analysis [8]. Provided that scalar variability is followed, the logs of the standard error $SE(n)$ and of the mean estimates $\mu(n)$ should form a straight line, with a slope $\beta = 1$ [$\log(SE(n)) = \alpha + \beta \log(\mu(n))$] [51]. Otherwise, a slope significantly deviating from 1 would suggest that the relationship between standard errors and means follows a more complex power function.

Nevertheless, it has been observed that, in the case of very low variability, it is not particularly insightful to analyze the Weber law. Therefore, the analysis has been omitted from the study.

4.8 MODEL EMBEDDINGS

The embeddings are obtained using auto-tokenizers from pre-trained models as tokenizers via the Transformers library. They are then extracted using the `embed_tokens` function from the respective models. Both models generated output with 4095-dimensional embeddings for all numbers and prompts. The Euclidean distances are calculated using all dimensions. In Figure 4.11, the Euclidean distances for only the numbers’ embeddings calculated for both models are presented in matrices normalized by the global minimum and maximum values of distance, where the maximum distance within the model is equal to 1 and the minimum is 0. Colors are on a scale from purple to yellow in a format in which colors closer to purple represent closer distances. Between the models, mistral clearly has shorter differences among numbers. The digit 1 is the most distant number for the mistral model with an average normalized distance of 0.26 (STD= 0.087). This phenomenon is more clearly illustrated in Figure 4.12, where the embeddings of numbers are visualized using PCA for dimensional reduction and subsequently clustered into four groups using K-means clustering. The number of clusters was selected to highlight the differences in the data. For both models, configurations with 2, 3, 4, and 5 clus-

ters were tested, and the four-cluster configuration was chosen because it provided the most representative visualization. Digit 1 is significantly farther from the cluster of other numbers. Nonetheless, the remaining numbers are generally located closer to their consecutive neighbors, suggesting a potential linear relationship rather than a random distribution.

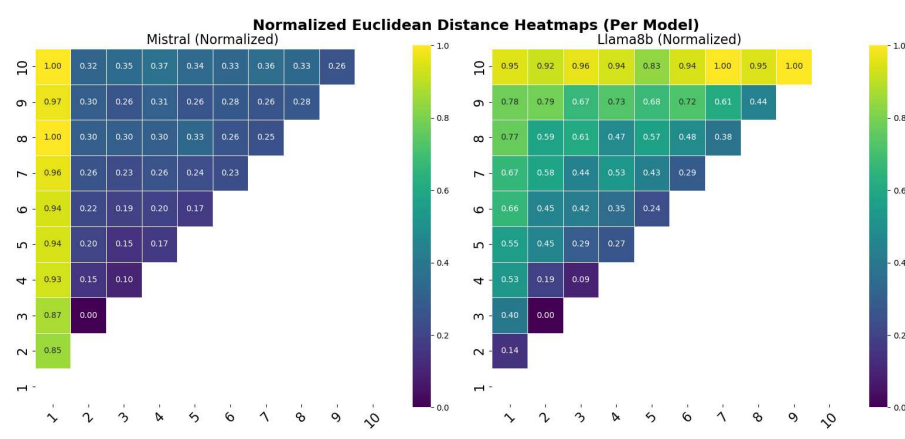


Figure 4.11: Euclidean Distances for numbers only numbers embeddings, each model is normalized within its minimum and maximum values yellow shows longer, purple shows closer distances

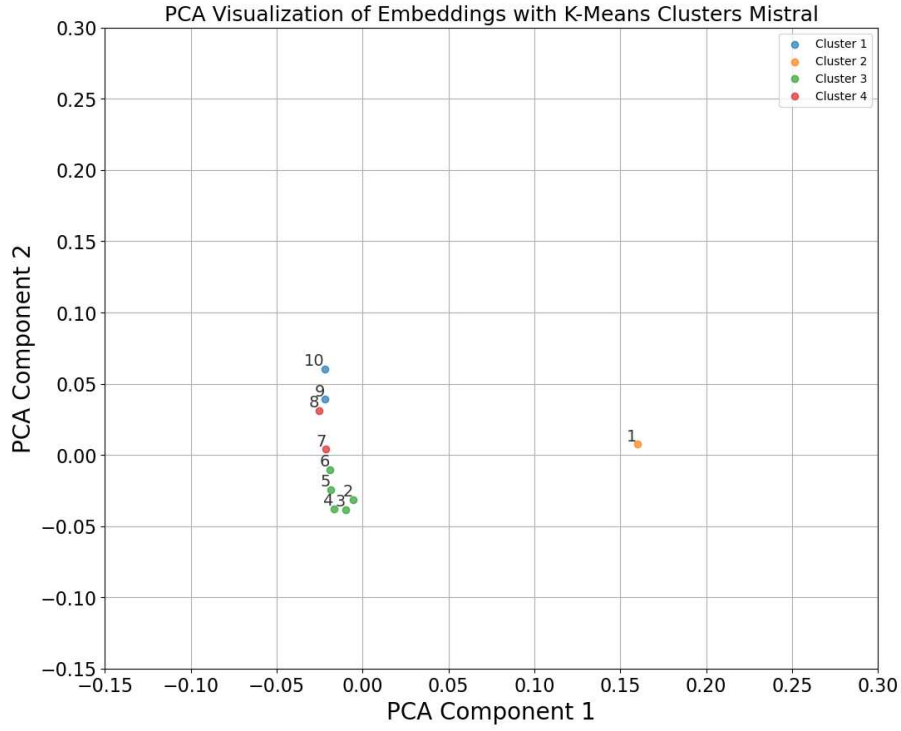


Figure 4.12: Representation of Number embeddings for Mistral 7B, PCA dimensionality reduction applied with K-means clustering

For the Llama-3 8B model, Figure 4.13, the overall distances among the numbers are greater compared to the Mistral model. Notably, the embedding of the number 10 is positioned farther from the other numbers. However, the sequential grouping of numbers remains evident. Given that the Llama-3 8B model outperformed the Mistral model, the increased separation between numerical embeddings, while maintaining a sequential structure, may contribute to improved performance.

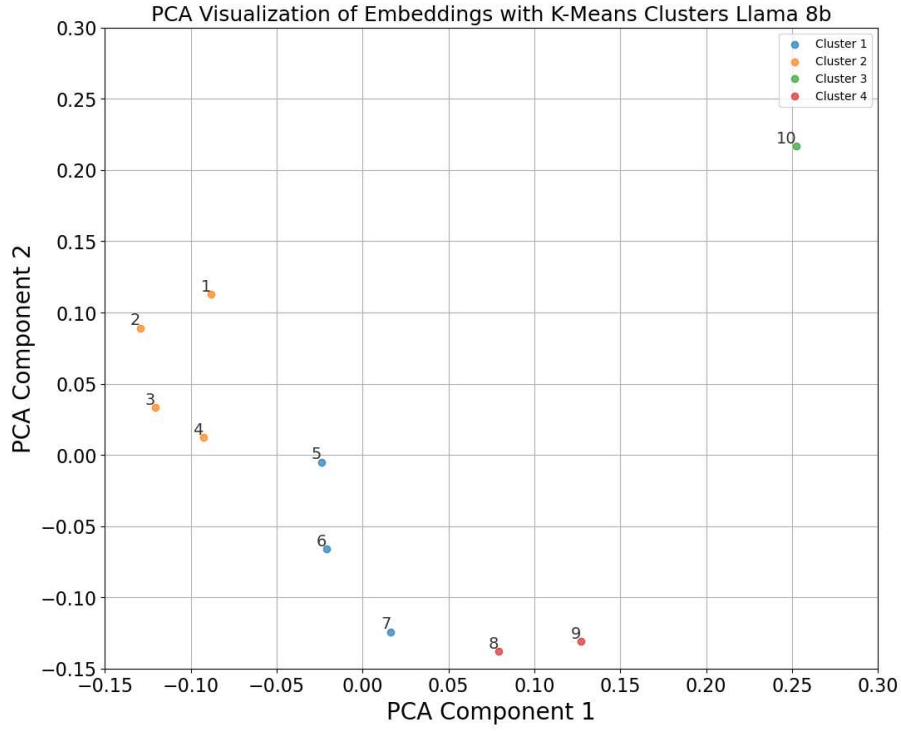


Figure 4.13: Representation of Number embeddings for LLaMa3-8B, PCA dimensionality reduction applied with K-means clustering

The distance change on number 10 might be due to moving to two-digit numbers. In order to test this hypothesis number 11 is added to embedding lists of Llama-3 8b model. The expectation is that number 11 should be placed somewhere closer to number 10 compared to number 9, showing that number 10 is not just an outlier. In the Figure 4.14, there is the normalized euclidean distance matrix colored based on the minimum and maximum distances within.

As expected, the number 11 is positioned closest to number 10. This pattern is more clearly observed in Figure 4.15, which visualizes the positions of the embeddings in a manner similar to Figure 4.13. The key difference between the two figures is that the inclusion of the number 11 alters the dimension reduction step on PCA changes leading to a different component combination even though the locations of the first 10 numbers remained the same.

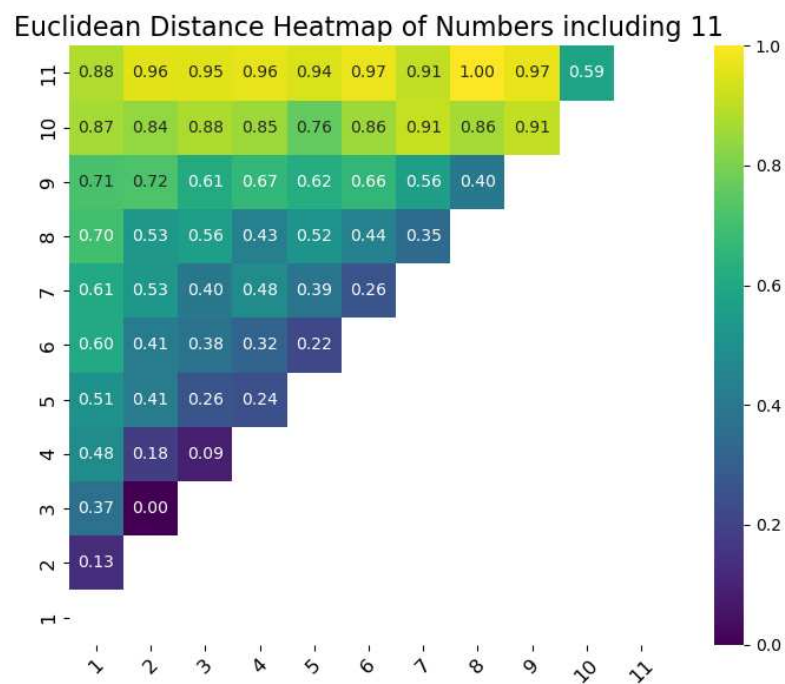


Figure 4.14: Euclidean Distances for numbers only numbers embeddings for LLaMA-3 8B model including the number 11, yellow shows longer, purple shows closer distances

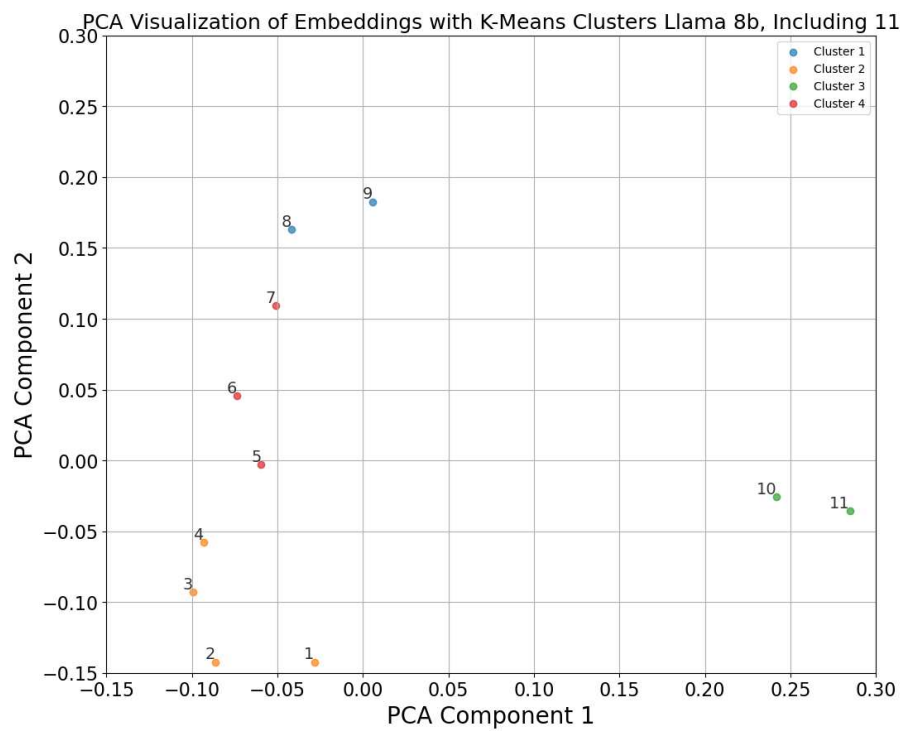


Figure 4.15: Representation of Number embeddings for LLaMa3-8B including number 11, PCA dimensionality reduction applied with K-means clustering

In addition to numbers, the prompts prompted in the models to generate numerosity for different tasks are represented in embeddings. The results indicate that prompt target numbers that are numerically closer tend to be positioned more proximally within the embedding space, mirroring the inherent spatial relationships observed among numerical values. The euclidean distance matrices are shown in the Figure 4.16. Similar to the number embeddings, the overall distances among numerical representations are shorter in the Mistral model compared to the corresponding task in the Llama-3 8B model. Interestingly, the most distant embedding changes from 1 to 10 in Mistral 7B when prompts are added to the embeddings.

Normalized Euclidean Distances for Prompts

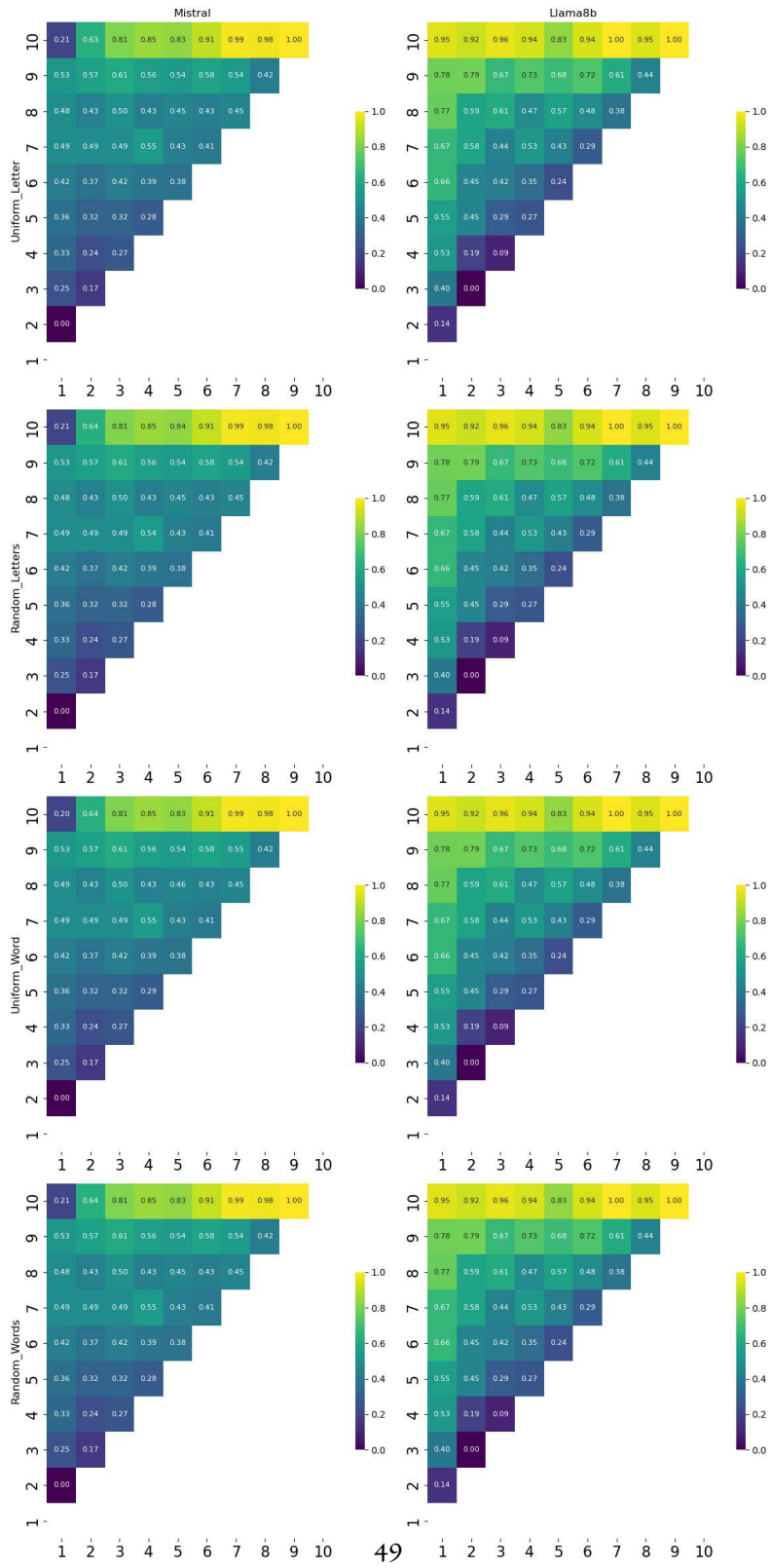


Figure 4.16: Euclidean Distances for different prompts used to collect data for each category in numerosity production task, each model is normalized within itself again

5

Conclusion

The present study investigated the sequential enumeration of the Large Language Models (LLM) across a range of numerosity-related tasks. The strengths and limitations in encoding, generating and naming numerical values of both open-source and proprietary models are aimed to be analyzed. The findings highlight key patterns in the way LLMs process numerical information, providing insights into their underlying mechanisms and potential moves on the quest to match human numerical cognition.

The results suggest that numerosity sense and generation vary significantly depending on the model architecture. The proprietary models—Gemini 1.5 Pro, Claude 3.5 Sonnet, and GPT-4o—proved superior performance in generating sequences restricted by number compared to the more variable performance of their open-source counterparts, Mistral-7B, Llama-3 8B, and Llama-3 70B. Due to the consistent outputs from the models, whether or not they are accurate, the analysis of tracking scalar variability is omitted from the study.

Although it could be only generated on two models, the analysis of numerical embeddings

revealed structured clustering patterns that indicate the numerical representations of LLMs are organized in a non-random manner. Thanks to these findings, in addition to the contribution of literature that claims the numerical representations in hidden layers on LLMs are crucial on the numerosity performance [20], the disparity among models could be linked to hidden state numerosity representations. This disparity among models might be attributed to differences in training data or tokenization strategies.

To further probe the numerical perception abilities of these models, their adherence to scalar variability principles and whether their standard error estimates followed a logarithmic pattern in accordance with Weber’s law are tested. In general, there were deviations in results especially for the higher ranges of tasks. Although some models showed an approximate coefficient of variation alignment with Weber’s principle, the overall analysis of Weber’s law was omitted due to the low variability of the outputs. These findings are in line with previous work in cognitive science that claims that numerosity estimation is inherently noisy and prone to representational inconsistencies [2, 9]. In addition to that, as the LLMs are unable to consistently generate and recognize numerical sequences, we have the hypothesis of relying on memorization and pattern-matching strategies. As a result of this reliance, inconsistencies occur when extrapolating beyond their training distribution in contrast to human cognition, which exhibits flexibility in numerical generalization across different contexts. It can be argued that this lack of consistency is a sign of the limitations of capturing abstract numerical concepts. In conclusion, while LLMs demonstrate structured numerical representations, their performance in sequential enumeration tasks remains imperfect and inconsistent. In general, the proprietary models showed significantly stronger numerical reasoning capabilities, but still do not perform close to human cognitive levels. The insights derived from this study underscore the need for architectural refinements, improved training methodologies, and multimodal learning approaches to bridge the gap between artificial and human numerical cognition. Future iterations of LLMs could become more robust and develop a human-like understanding of numerical concepts by addressing these challenges.

References

- [1] V. Alto, *Building LLM Powered Applications: Create intelligent apps and agents with large language models*. Packt Publishing Ltd, 2024.
- [2] A. Testolin, S. Dolfi, M. Rochus, and M. Zorzi, “Visual sense of number vs. sense of magnitude in humans and machines,” *Scientific reports*, vol. 10, no. 1, p. 10045, 2020.
- [3] S. Dehaene, *The Number Sense: How the Mind Creates Mathematics*. OUP USA, 2011.
- [4] J. Whalen, C. R. Gallistel, and R. Gelman, “Nonverbal counting in humans: The psychophysics of number representation,” *Psychological science*, vol. 10, no. 2, pp. 130–137, 1999.
- [5] J. R. Platt and D. M. Johnson, “Localization of position within a homogeneous behavior chain: Effects of error contingencies,” *Learning and Motivation*, vol. 2, no. 4, pp. 386–414, 1971.
- [6] R. N. Shepard, D. W. Kilpatrick, and J. P. Cunningham, “The internal representation of numbers,” *Cognitive psychology*, vol. 7, no. 1, pp. 82–138, 1975.
- [7] S. K. Revkin, M. Piazza, V. Izard, L. Cohen, and S. Dehaene, “Does subitizing reflect numerical estimation?” *Psychological science*, vol. 19, no. 6, pp. 607–614, 2008.
- [8] V. Izard and S. Dehaene, “Calibrating the mental number line,” *Cognition*, vol. 106, no. 3, pp. 1221–1247, 2008.

- [9] J. L. McClelland, M. M. Botvinick, D. C. Noelle, D. C. Plaut, T. T. Rogers, M. S. Seidenberg, and L. B. Smith, "Letting structure emerge: connectionist and dynamical systems approaches to cognition," *Trends in cognitive sciences*, vol. 14, no. 8, pp. 348–356, 2010.
- [10] A. Testolin and J. L. McClelland, "Do estimates of numerosity really adhere to weber's law? a reexamination of two case studies," *Psychonomic Bulletin & Review*, vol. 28, pp. 158–168, 2021.
- [11] S. Dehaene and J.-P. Changeux, "Development of elementary numerical abilities: A neuronal model," *Journal of cognitive neuroscience*, vol. 5, no. 4, pp. 390–407, 1993.
- [12] T. Verguts and W. Fias, "Representation of number in animals and humans: A neural model," *Journal of cognitive neuroscience*, vol. 16, no. 9, pp. 1493–1504, 2004.
- [13] D. E. Rumelhart, J. L. McClelland, P. R. Group *et al.*, *Parallel distributed processing, volume 1: Explorations in the microstructure of cognition: Foundations*. The MIT press, 1986.
- [14] M. Cappelletti, D. Didino, I. Stoianov, and M. Zorzi, "Number skills are maintained in healthy ageing," *Cognitive Psychology*, vol. 69, pp. 25–45, 2014.
- [15] S. Y. Chen, Z. Zhou, M. Fang, and J. L. McClelland, "Can generic neural networks estimate numerosity like humans?" in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 40, 2018.
- [16] I. Stoianov and M. Zorzi, "Emergence of a 'visual number sense' in hierarchical generative models," *Nature neuroscience*, vol. 15, no. 2, pp. 194–196, 2012.
- [17] M. Zorzi and A. Testolin, "An emergentist perspective on the origin of number sense," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 373, no. 1740, p. 20170043, 2018.

- [18] D. Odic, M. E. Libertus, L. Feigenson, and J. Halberda, “Developmental change in the acuity of approximate number and area representations.” *Developmental psychology*, vol. 49, no. 6, p. 1103, 2013.
- [19] M. Piazza, A. Facoetti, A. N. Trussardi, I. Berteletti, S. Conte, D. Lucangeli, S. Dehaene, and M. Zorzi, “Developmental trajectory of number acuity reveals a severe impairment in developmental dyscalculia,” *Cognition*, vol. 116, no. 1, pp. 33–41, 2010.
- [20] F. Zhu, D. Dai, and Z. Sui, “Language models encode the value of numbers linearly,” in *Proceedings of the 31st International Conference on Computational Linguistics*, 2025, pp. 693–709.
- [21] M. Binz and E. Schulz, “Using cognitive psychology to understand gpt-3,” *Proceedings of the National Academy of Sciences*, vol. 120, no. 6, p. e2218523120, 2023.
- [22] M. Le Corre and S. Carey, “One, two, three, four, nothing more: An investigation of the conceptual sources of the verbal counting principles,” *Cognition*, vol. 105, no. 2, pp. 395–438, 2007.
- [23] M. D. Lee and B. W. Sarnecka, “Number-knower levels in young children: Insights from bayesian modeling,” *Cognition*, vol. 120, no. 3, pp. 391–402, 2011.
- [24] Y. Zhou, Z. Liu, J. Jin, J.-Y. Nie, and Z. Dou, “Metacognitive retrieval-augmented large language models,” in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 1453–1463.
- [25] B. Plank, “Sliced at semeval-2022 task 11: Bigger, better? massively multilingual lms for multilingual complex ner on an academic gpu budget,” in *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, 2022, pp. 1494–1500.
- [26] W. O’Grady and M. Lee, “Natural syntax, artificial intelligence and language acquisition,” *Information*, vol. 14, no. 7, p. 418, 2023.

- [27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need.(nips), 2017," vol. 10, 2017, p. So140525XI6001837.
- [28] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, "Phishing email detection using natural language processing techniques: a literature survey," *Procedia Computer Science*, vol. 189, pp. 19–28, 2021.
- [29] A. Stolfo, Y. Belinkov, and M. Sachan, "A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis," *arXiv preprint arXiv:2305.15054*, 2023.
- [30] S. Venkatesh, D. Moffat, and E. R. Miranda, "Word embeddings for automatic equalization in audio mixing," *arXiv preprint arXiv:2202.08898*, 2022.
- [31] W. Khan, A. Daud, F. Alotaibi, N. Aljohani, and S. Arafat, "Deep recurrent neural networks with word embeddings for urdu named entity recognition," *ETRI Journal*, vol. 42, no. 1, pp. 90–100, 2020.
- [32] A. Rajasekharan, Y. Zeng, P. Padalkar, and G. Gupta, "Reliable natural language understanding with large language models and answer set programming," *arXiv preprint arXiv:2302.03780*, 2023.
- [33] S. Sadeghi, A. Bui, A. Forooghi, J. Lu, and A. Ngom, "Can large language models understand molecules?" *BMC bioinformatics*, vol. 25, no. 1, p. 225, 2024.
- [34] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili *et al.*, "A survey on large language models: Applications, challenges, limitations, and practical usage," *Authorea Preprints*, 2023.
- [35] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

- [36] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing,” *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.
- [37] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” *arXiv preprint*, vol. 2005.14165, 2020. [Online]. Available: <http://arxiv.org/abs/2005.14165>
- [38] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen, “What makes good in-context examples for gpt-3?” *arXiv preprint arXiv:2101.06804*, 2021.
- [39] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large language models are zero-shot reasoners,” *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.
- [40] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”, 2009.
- [41] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, “Gemini: a family of highly capable multimodal models,” *arXiv preprint arXiv:2312.11805*, 2023.
- [42] A. Anthropic, “Claude 3.5 sonnet model card addendum,” p. 6, 2024.
- [43] OpenAI, “Gpt-4o overview,” 2024, available online: <https://openai.com/index/hello-gpt-4o/> (accessed on 26 Jan 2025).
- [44] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. D. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M. Lachaux, P. Stock,

- T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, “Mistral 7b,” *arXiv preprint*, 2023. [Online]. Available: <https://arxiv.org/abs/2310.06825>
- [45] AI@Meta, “Llama 3 model card,” 2024, available at: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.
- [46] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.
- [47] W. Talukdar and A. Biswas, “Improving large language model (llm) fidelity through context-aware grounding: A systematic approach to reliability and veracity,” *arXiv preprint arXiv:2408.04023*, 2024.
- [48] M. Ringnér, “What is principal component analysis?” *Nature biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.
- [49] T. Kurita, “Principal component analysis (pca),” in *Computer vision: a reference guide*. Springer, 2021, pp. 1013–1016.
- [50] K. P. Sinaga and M.-S. Yang, “Unsupervised k-means clustering algorithm,” *IEEE access*, vol. 8, pp. 80 716–80 727, 2020.
- [51] A. Testolin, K. Hou, and M. Zorzi, “Visual enumeration is challenging for largescale generative ai,” 2024.



Appendices

In this section, the appropriate responses used to calculate the MSE scores but are not shown in the graphs due to the scaling limits will be exemplified for the categories and models that have them. Numerosity naming task outputs were all represented in the graphs, therefore the following examples will be from Numerosity Generation Task.

LLAMA 8B

Generate Letter A, range 10:

Target number: 10

Trial: 12

Generated num:22

AAAAAA AAAA AAAA AAAA AAAA

Generate Letter A, range 100:

Target number: 50

Trial: 1

Generated Number: 431

AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA

Generating Random Letters, range 100:

Target number: 100

Trial: 15

Generated number: 715

jgkmlhgfedcbaqwertyuiohpqsd fghjklmnpoiuqwertyuiopasdfghjklzxcvbnmuiopasdfghjklpoiuytrewqazxcv
bnmnbvcxzasdfghjklpoiuytrtrewqazxcvbnmnbvcxzasdfghjklpoiuytrtrewqazxcvbnuioytrtyuowertyuiopa
sdfghjklm hgfedcbaqwertyuiohpqsd fghjklmnbvcxzasaioytrtryioyqwertyuiopasdffghjklpoiuytrewqazxcv
bnmnbvcxzasdfghjklpoiuytrtrewqazxcvbnuioytrtyuowertyuiopasdfghjklm hgfedcbaqwertyuiopasdfghjkl
poi usdffghjklpoiuytrewqazxcvbnmnbvcxzasdfghjklpoiuytrtrewqazxcvbnui oysdffghjklpoiuytrewqazxc
vbnmnbvcxzasdfghjklpoiuytrtrewqazxcvbnui osdfghjklpoiuytrewqazxcvbnmnbvcxzasdfghjklpoiuytrtrew
qazxcvbnui oysdffghjklpoiuytrewqazxcvbnmnbvcxzasdfghjklpoiuytrtrewqazxcvbnui osdfghjklpoiuytrew
qazxcvbnmnbvcxzasdfghjklpoiuytrtrewqazxcvbnui oysdf

LLAMA 70B

Generating random letters, range 100:

Target number: 70

Trial: 0

Generated num: 577

jhgfd sakljhgfdl kjhgfd sakljhgfdl kjhgfd sakljhgfdl k

jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlk
jhgfdsakljhgfdlkjhgfdsakljhgfdlkjhgfdsakljhgfdlkj

MISTRAL

Generating Letter A, range 10:

Target number: 10

Trial: 22

Generated num:37

AA AAA AAAA AAAAA AAAAAA AA AAAA AAAA AAAA AAA

Generating Random Letters, range 10:

Target number: 1

Trial: 2

Generated num:27

fghjklmnopqrstuvwxyzabcdefg

Generating random words, range 10:

Target number: 7

Trial: 2

Generated num:25

Seven Eleven Twenty-three One hundred twenty-seven Three thousand four hundred seventy-nine Nine million five hundred eighty-three thousand seven hundred forty-six

B

Coefficient of Variation Results

Table B.1: MWE Scores for Numerosity Naming Task

Model	Range 10				Range 100			
	Uniform Letter	Random Letter	Uniform Word	Random Words	Uniform Letter	Random Letter	Uniform Word	Random Words
Mistral	0.017	0.099	0.364	0.121	0.173	0.157	0.855	0.139
Llama8b	0.007	0.067	0.003	0.077	0.153	0.228	0.003	0.150
Llama70b	0.000	0.016	0.000	0.048	0.019	0.051	0.051	0.068
Gemini	0.000	0.013	0.000	0.035	0.016	0.049	0.012	0.077
Claude	0.000	0.006	0.000	0.006	0.008	0.042	0.002	0.068
GPT ₄	0.000	0.002	0.000	0.070	0.020	0.046	0.061	0.087

Table B.2: MWE Scores for Numerosity Production Task

Model	Range 10				Range 100			
	Uniform Letter	Random Letter	Uniform Word	Random Words	Uniform Letter	Random Letter	Uniform Word	Random Words
Mistral	0.491	0.977	0.684	1.694	1.916	3.737	0.555	2.440
Llama8b	0.062	0.056	0.000	0.334	0.755	0.507	0.117	0.932
Llama70b	0.006	0.072	0.000	0.017	0.061	0.453	0.070	0.163
Gemini	0.002	0.000	0.000	0.000	0.134	0.096	0.019	0.122
Claude	0.000	0.000	0.000	0.043	0.001	0.181	0.161	0.054
GPT ₄	0.000	0.290	0.000	0.253	0.020	0.107	0.014	0.028

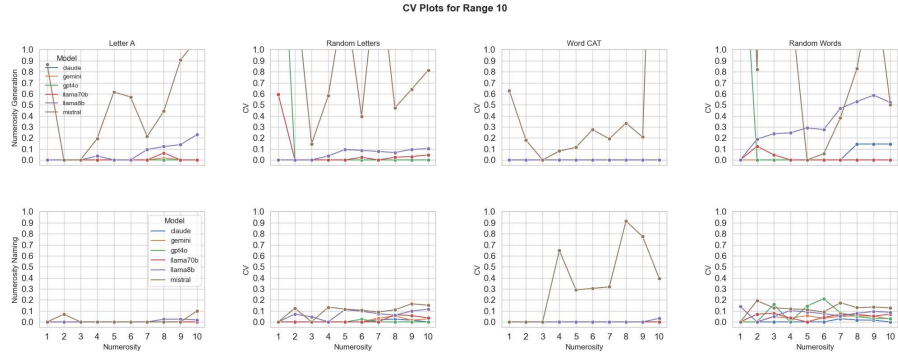


Figure B.1: Coefficient of Variance scores for each Numerosity, range 10

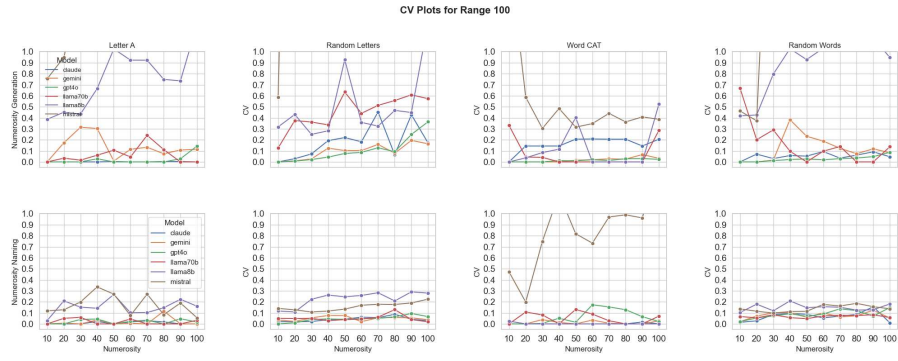


Figure B.2: Coefficient of Variance scores for each Numerosity, range 100