

Self-Consistency Benefits Large Language Models

Shuyue Jia

Ph.D. Student

Boston University

September 12, 2023

Dependable Computing Laboratory,
Department of Electrical and Computer Engineering,
Boston University

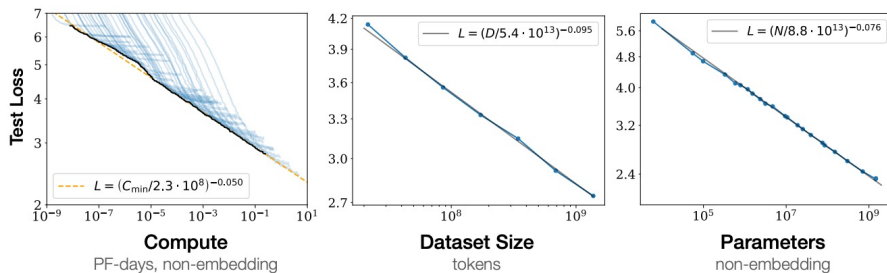


Outline

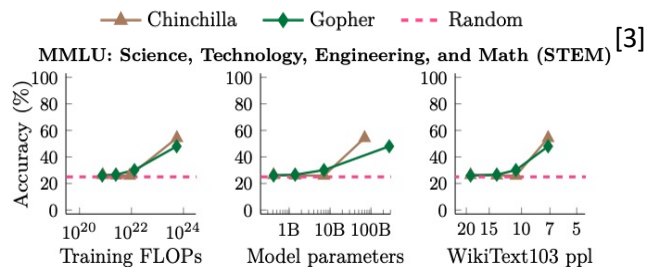
- Scaling Laws and Pre-training
- Supervised Fine-tuning (**SFT**)
- Reinforcement Learning with Human Feedback (**RLHF**)
- Chain of Thought (**CoT**) Prompting
- Self-consistency Benefits Reasoning of LLMs

Part 1 – Scaling Laws and Pre-training

- **Neural Scaling Laws:** **Power-law** with **model size N** , **dataset size D** , and **computation power C** . [1]
- An ability is **emergent** if it is not present in smaller models but is present in **larger models**. [2]



$$L = L_0 + \left(\frac{x_0}{x}\right)^\alpha [1]$$



Phase Transition [2]

Credits:

[1] Kaplan *et al.*, Scaling Laws for Neural Language Models, In arXiv'20.

[2] Wei *et al.*, Emergent Abilities of Large Language Models, In TMLR'22.

[3] MMLU: **M**assive **M**ultitask Language **U**nderstanding - 57 subjects across STEM, humanities, social sciences, and more.

Part 1 – Scaling Laws and Pre-training

- **Vocabulary:** Sub-words Tokenization ([Byte-Pair Encoding](#), e.g., “Biden” → tokens “bi” and “den”)
- **[Input] Prompt:** A text string description with instructions, goals, or examples
- **Word Embedding:** Linear Layer matrix \mathbf{W} and Layer Normalization
- **Positional Embedding:** [AliBi](#) [2]
- **Self-attention Blocks:** Multi-head Self-attention + Feedforward NN
- **[Output] Token:** \mathbf{W}^T and Softmax
- **Large-scale Dataset:** multilingual data + **codes** (*Code-davinci-002*)
- **Learning Objective:**

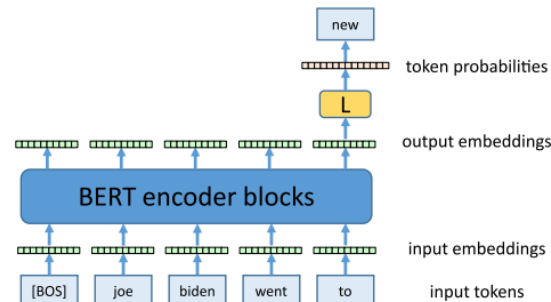
$$p(x_l | x_{<l}; \theta) = \text{softmax}(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}),$$

$$L_{\text{LM}}(p) := -\mathbb{E}_{x \sim D} \left[\sum_{l=1}^K \log p(x_l | x_{<l}; \theta) \right].$$

Credits:

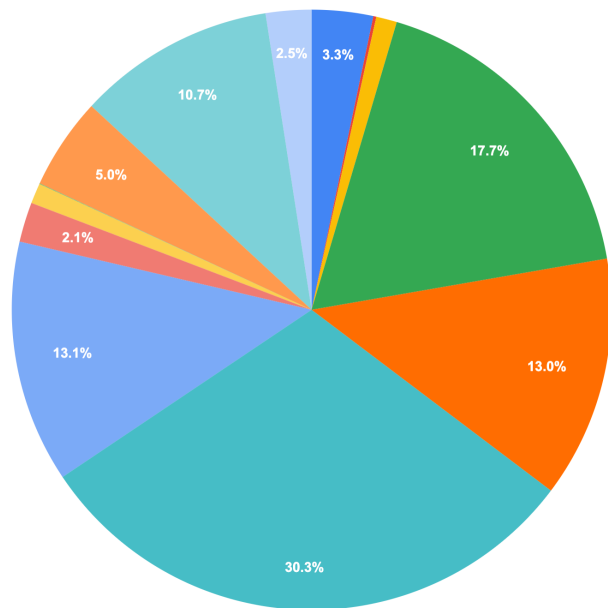
[1] Paaß *et al.*, Foundation Models for Natural Language Processing: Pre-trained Language Models Integrating Media, In Springer Nature’23.

[2] [The Technology Behind BLOOM Training](#).



A Framework of
Autoregressive LM [1]

Part 1 – Scaling Laws and Pre-training



- Arabic (3,3%)
- Basque (0,2%)
- Catalan (1,1%)
- Chinese (17,7%)
- Code (13%)
- English (30,3%)
- French (13,1%)
- Indic (2,1%)
- Indonesian (1,1%)
- Niger Congo (0,03%)
- Portuguese (5%)
- Spanish (10,7%)
- Vietnamese (2,5%)

- **46** Languages
- **350 Billion** tokens
- 1.5 TB text data
- 13% is **Codes**
(Reasoning and long-range modeling)

High Quantity, Low Quality

Part 1 – Scaling Laws and Pre-training

🤖 Open LLM Leaderboard

📄 The 🤖 Open LLM Leaderboard aims to track, rank and evaluate open LLMs and chatbots.

📄 Submit a model for automated evaluation on the 🤖 GPU cluster on the "Submit" page! The leaderboard's backend runs the great [Eleuther AI Language Model Evaluation Harness](#) - read more details in the "About" page!

LLM Benchmark About Submit here!

Select columns to show

Average ARC HellaSwag MMLU TruthfulQA Type

Precision Hub License #Params (B) Hub Model sha

Show gated/private/deleted models

Search for your model and press ENTER...

Model types

pretrained fine-tuned instruction-tuned RL-tuned

Model sizes

< 1.5B ~3B ~7B ~13B ~35B 60B+

T	Model	Average	ARC	HellaSwag	MMLU	TruthfulQA
📄	uni.tianyan/Uni-TianYan	73.81	72.1	87.4	69.91	65.81
📄	fangleveskazi/ORCA_LLaMA_70B_QLoRA	73.4	72.27	87.74	70.23	63.37
📄	garage-b2Ind/Platypus2-70B-instruct	73.13	71.84	87.94	70.48	62.26
📄	upstage/Llama-2-70b-instruct-v2	72.95	71.08	87.89	70.58	62.25
📄	fangleveskazi/Platypus_QLoRA_LLaMA_70b	72.94	72.1	87.46	71.02	61.18
📄	ysontaek/llama-2-70b-ensemble-v5	72.86	71.16	87.24	69.6	63.45
🚫	TheBloke/Genz-70b-GPTQ	72.82	71.08	87.64	70.26	62.28
📄	TheBloke/Platypus2-70B-Instruct-GPTQ	72.81	71.25	87.55	69.89	62.54
📄	psmathur/model_007	72.72	71.08	87.65	69.04	63.12
📄	ysontaek/llama-2-70b-ensemble-v4	72.64	70.9	87.34	69.71	62.6

- Fluent text generation
- In-context few-shot learning
- World knowledge and commonsense
- Code understanding and generation
- Complex reasoning

A hand is shown from the bottom left, reaching up to hold a glowing, wireframe brain. The brain is rendered in bright blue and cyan lines, with a red and purple glow emanating from it. The background is a dark, starry space with a gradient of blue and purple light. The text 'Foundation Models' is written in a bold, white, sans-serif font across the middle of the image, partially overlapping the hand and the brain.

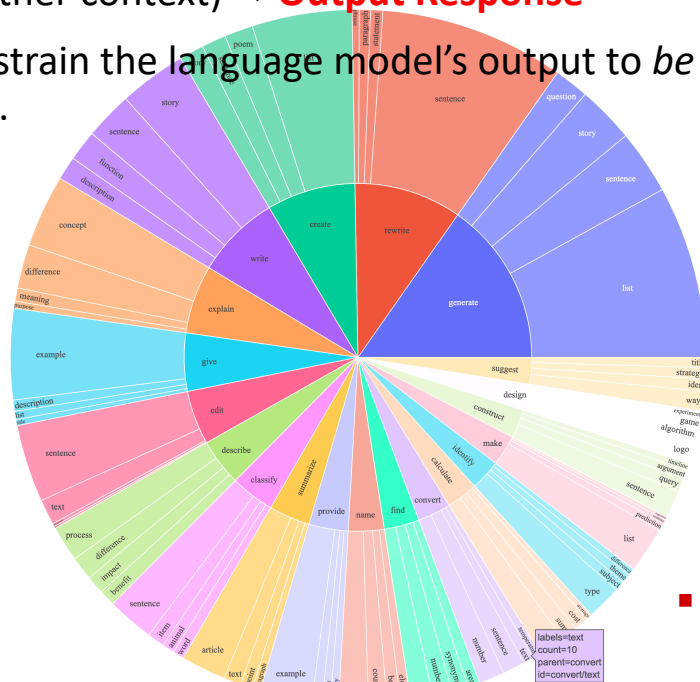
Foundation Models

Part 2 – Supervised Fine-tuning (SFT)

- **Instruction Tuning:** Fine-tuning an LLM **on a collection of tasks** described via **Instructions** [1]
- **Instruction** (Describe a task) + **Input** (Provide further context) → **Output Response**
- Allow humans to steer the conversation and constrain the language model's output to *be more natural, useful, and aligned with the users' goals.*
- **High Quality, High Diversity, Low quantity**

Brainstorming	Provide a diverse set of creative ideas for new flavors of ice cream.
Classification	Categorize these movies as either comedy, drama, or horror based on the plot summary.
Closed QA	Answer the question "What is the capital of France?" with a single word.
Generation	Write a poem in the style of Robert Frost about nature and the changing seasons.
Information Extraction	Extract the names of the main characters from this short story.
Open QA	Why do leaves change color in autumn? Explain the scientific reasons.
Summarization	Summarize this article on recent advancements in renewable energy in 2-3 sentences.

Instruction-following Demonstrations

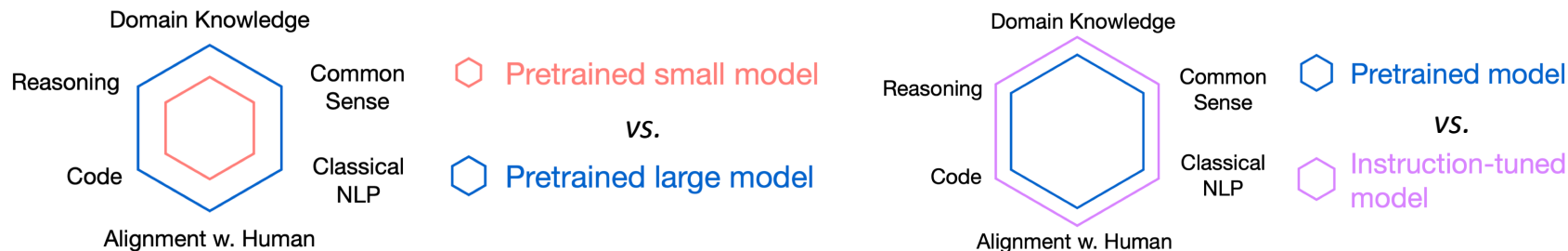


▪ 52k data

Image Credit: [Stanford Alpaca: An Instruction-following LLaMA Model.](#)

Part 2 – Supervised Fine-tuning (SFT)

- **Instruction Tuning:** Highly relies on **Data Engineering**
- Could fine-tuned by **Low-Rank Adaptation (LoRA)** [1] or its variants, *e.g.*, **QLoRA** [2]
- **Unlock**, not learn
- **Elicit**, not inject



Credits: Image: [Dr. Yao Fu](#), University of Edinburgh.

[1] Hu *et al.*, LoRA: Low-Rank Adaptation of Large Language Models, In ICLR'22.

[2] Dettmers *et al.*, QLoRA: Efficient Finetuning of Quantized LLMs, In arXiv'23.

Part 3 – Reinforcement Learning with Human Feedback

- **Alignment**-focused Fine-tuning
- Align model behavior with **human preferences and values** [1]
- Align LLMs to **follow instructions**, become more **helpful, honest, and harmless** (HHH), and generate **truthful, fair, and safe** responses
- **Overrefusing Problem**
 - “Tend to become overly **cautious in certain ways**, refusing innocuous requests and excessively hedging or ‘**overrefusing**’” [2]

Credits:

[1] Ouyang *et al.*, Training language models to follow instructions with human feedback, In arXiv'22.

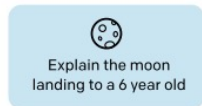
[2] OpenAI, GPT-4 System Card, In <https://cdn.openai.com/papers/gpt-4-system-card.pdf>.

Self-Consistency Benefits Large Language Model

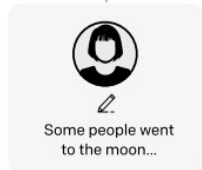
Step 1

Collect demonstration data, and train a supervised policy.

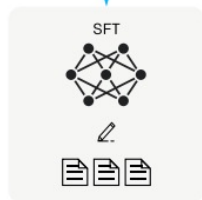
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.

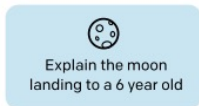


**Supervised Fine-tuning
a pre-trained LLM**

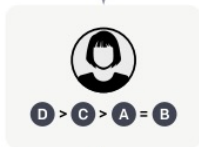
Step 2

Collect comparison data, and train a reward model.

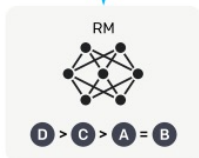
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



**Supervised Learning
a Reward Model**

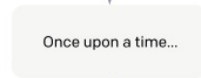
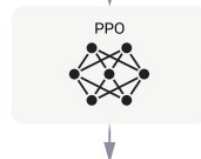
Step 3

Optimize a policy against the reward model using reinforcement learning.

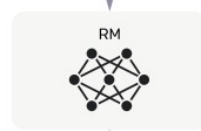
A new prompt is sampled from the dataset.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

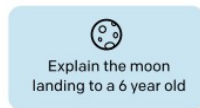


**Reinforcement Learning
the fine-tuned LLM**

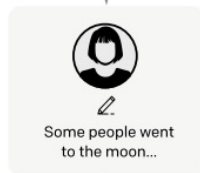
Step 1

**Collect demonstration data,
and train a supervised policy.**

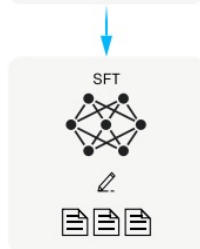
A prompt is
sampled from our
prompt dataset.



A labeler
demonstrates the
desired output
behavior.



This data is used
to fine-tune GPT-3
with supervised
learning.



STEP 1

**Supervised Fine-tuning (SFT)
a pre-trained LLM**

$$L_{\text{LM}}(p) := -\mathbb{E}_{x \sim D} \left[\sum_{l=1}^K \log p(x_l | x_{<l}; \theta) \right].$$

- Given **prompts** and **desired behavior (demonstrations)**
- K is the number of tokens in the output response.
- Model initialized by the *supervised fine-tuned model*

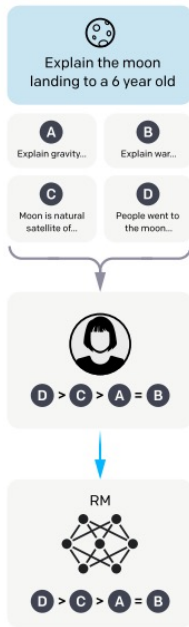
STEP 2

Preference Sampling and Reward Learning

Step 2

Collect comparison data,
and train a reward model.

A prompt and
several model
outputs are
sampled.



A labeler ranks
the outputs from
best to worst.

This data is used
to train our
reward model.

Bradley-Terry model:

$$p^*(y_c > y_r | x) = \sigma(r_\theta(x, y_c) - r_\theta(x, y_r)).$$

- p^* is the human preference distribution

Learning Objective: Binary Classification Task

$$L_R(r_\theta, \mathcal{D}) = -\mathbb{E}_{(x, y_c, y_r) \sim \mathcal{D}_r} \left[\log \left(\sigma(r_\theta(x, y_c) - r_\theta(x, y_r)) \right) \right].$$

- x is the prompt
- y_c and y_r are preferred and dispreferred responses
- r_θ is the reward model, initialized by the SFT LLMs
- σ is a logistic function
- Add an **extra linear layer** on top of final transformer layer

Step 3

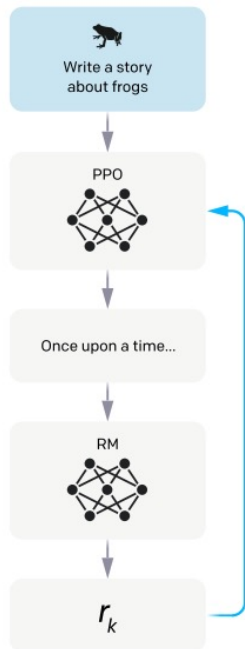
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



STEP 3 Reinforcement Learning Fine-tuning ^[1]

$$\max_{\pi_{\gamma}} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\gamma}(y|x)} [r_{\theta}(x, y)] - \beta \mathbb{D}_{\text{KL}}[\pi_{\gamma}(y|x) || \pi_{\text{ref}}(y|x)].$$

- β controlling the deviation from the base reference policy π_{ref}
- π_{ref} is the initial SFT model
- π_{γ} is the language model policy, initialized by the initial SFT model

$$r(x, y) = r_{\theta}(x, y) - \beta (\log \pi_{\gamma}(y|x) - \log \pi_{\text{ref}}(y|x)).$$

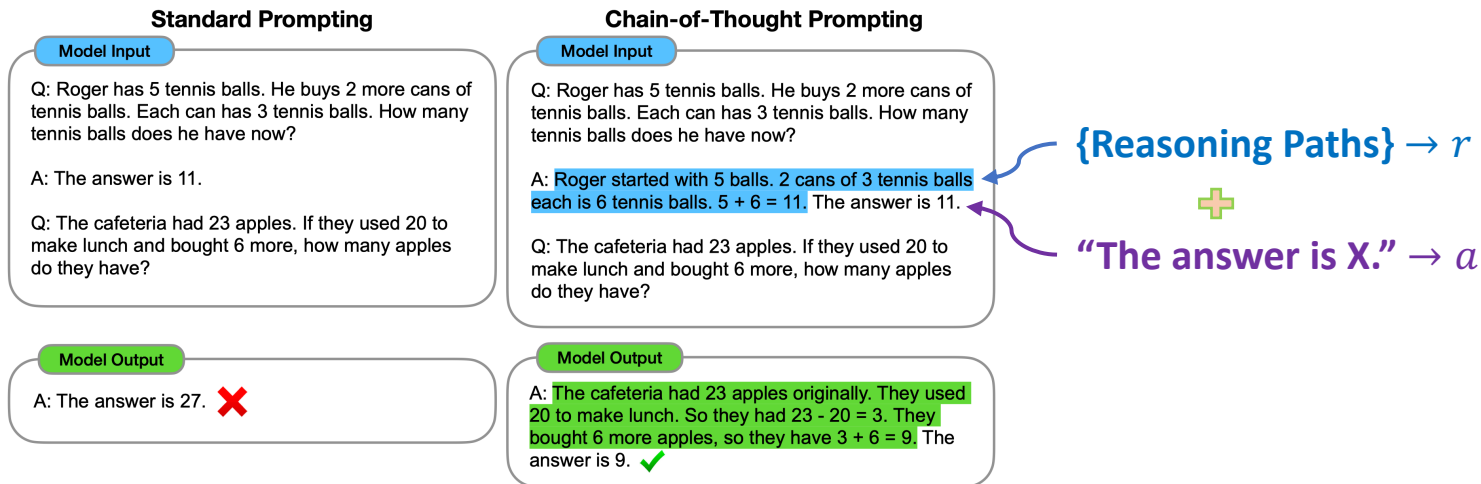
- Proximal Policy Optimization (PPO) is applied to maximize $r(x, y)$ ^[2]

Credits: [1] Ouyang *et al.*, Training language models to follow instructions with human feedback, In arXiv'22.

[2] Schulman *et al.*, Proximal Policy Optimization Algorithms, In arXiv'17.

Part 4 – Chain of Thought (CoT-prompting)

- **Chain of Thought (CoT):** a series of **intermediate natural language reasoning steps** that lead to the final output
- Mimic an intuitive thought process, and decompose multi-step problems into intermediate steps
- **In context few-shot learning** with **8 few-shot exemplars**



Part 4 – Chain of Thought (CoT-prompting)

- Reasoning Tasks: *Arithmetic*, *Commonsense*, and *Symbolic* Reasoning
- Applied Models: **UL2** 20B, **GPT-3** (text-ada-001 350M, text-babbage-001 1.3B, text-curie-001 6.7B, and text-davinci-002 175B), **Codex** (code-davinci-002 14.8B), **LaMDA** (422M, 2B, 8B, 68B, and 137B), and **PaLM** (8B, 62B, and 540B)

```
base ~ (5.111s)
llm -m gpt-3.5-turbo-16k "Ducks need to eat 3.5 pounds of insects each week to survive. If there is a flock of ten ducks, how many pounds of insects do they need per day?"

To find out how many pounds of insects they need per day, divide the total pounds they need per week by the number of days in a week: 3.5 pounds/week / 7 days/week = <<3.5/7=>.5>>.5 pounds/day
Since there are ten ducks in the flock, they need 10 * .5 pounds/day = <<10*.5=>5>> 5 pounds of insects per day. Answer: \boxed{5}.
```

```
base ~ (3.658s)
llm -m gpt-3.5-turbo-16k "Ducks need to eat 3.5 pounds of insects each week to survive. If there is a flock of ten ducks, how many pounds of insects do they need per day?"

To determine the pounds of insects the flock of ten ducks needs per day, we need to divide the weekly requirement by 7 days in a week: 3.5 pounds / 7 days = <<3.5/7=0.5>> 0.5 pound. Answer: \boxed{0.5}.
```

```
base ~ (4.054s)
llm -m gpt-3.5-turbo-16k "Ducks need to eat 3.5 pounds of insects each week to survive. If there is a flock of ten ducks, how many pounds of insects do they need per day?"

To determine the amount of insects needed per day, we need to divide the weekly amount by 7 days. Therefore, each day, the flock of ten ducks needs to eat 3.5 pounds / 7 days = <<3.5/7=0.5>>0.5 pounds of insects. Answer: \boxed{0.5}.
```

base ~ Arithmetic reasoning, GSM8K

```
base ~ (0.047s)
cl
```

```
base ~ (1.934s)
llm -m gpt-3.5-turbo-16k "Take the last letters of the words in 'Elon Musk' and concatenate them."

The last letters of the words in 'Elon Musk' are 'n' and 'k'. Concatenate them to get 'nk'.
```

```
base ~ (2.529s)
llm -m gpt-3.5-turbo-16k "Take the last letters of the words in 'Elon Musk' and concatenate them."

The last letters of the words in 'Elon Musk' are 'n' and 'k'. Concatenate them to get 'nk'.
```

```
base ~ (0.621s)
llm -m gpt-3.5-turbo-16k "Take the last letters of the words in 'Elon Musk' and concatenate them."

Error: Rate limit reached for default-gpt-3.5-turbo-16k in organization gpt-3.5-turbo-16k on requests per min. limit: 3 / min. Please try again later.
```

base ~ Symbolic reasoning

```
base ~ (1.041s)
llm -m gpt-3.5-turbo-16k "Yes or no: Could a llama birth twice during War in Vietnam (1945-46)?"

No.
```

```
base ~ (2.978s)
llm -m gpt-3.5-turbo-16k "Yes or no: Could a llama birth twice during War in Vietnam (1945-46)?"

No, it is not possible for a llama to give birth twice within a specific time frame such as the Vietnam War (1945-46). The gestation period for llamas is approximately 11 months, so a llama cannot give birth multiple times within such a short duration.
```

```
base ~ (1.046s)
llm -m gpt-3.5-turbo-16k "Yes or no: Could a llama birth twice during War in Vietnam (1945-46)?"

No.
```

base ~ Commonsense reasoning

Part 4 – Chain of Thought (CoT-prompting)

- Significantly enhance the performance of **>100B** model, worsen the performance of **<10B** model

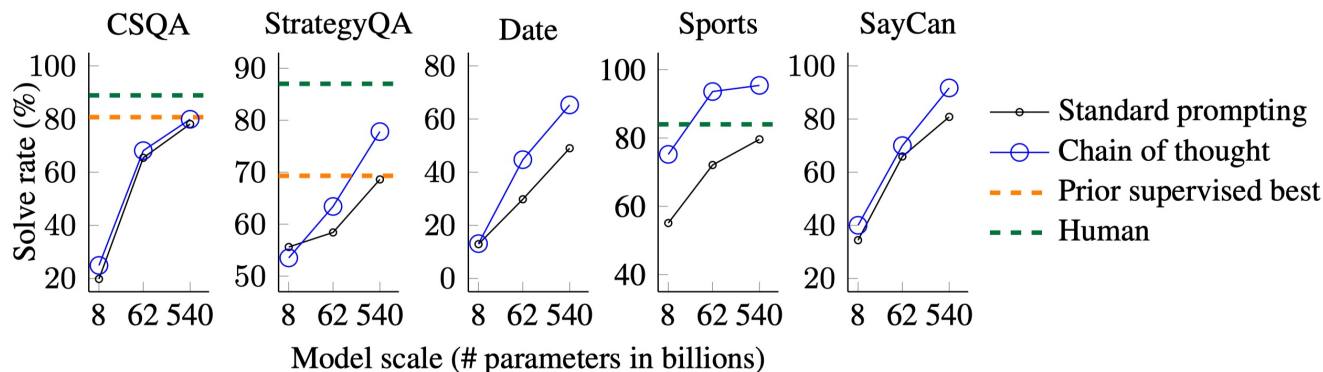


Figure 7: Chain-of-thought prompting also improves the commonsense reasoning abilities of language models. The language model shown here is PaLM. Prior best numbers are from the leaderboards of CSQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021) (single-model only, as of May 5, 2022). Additional results using various sizes of LaMDA, GPT-3, and PaLM are shown in Table 4.

Part 4 – Chain of Thought (CoT-prompting)

- Significantly enhance the performance of **>100B** model, worsen the performance of **<10B** model

Types of errors made by a 62B language model:

Semantic understanding

(62B made 20 errors of this type,
540B fixes 6 of them)

One step missing

(62B made 18 errors of this type,
540B fixes 12 of them)

Other

(62B made 7 errors of this type,
540B fixes 4 of them)

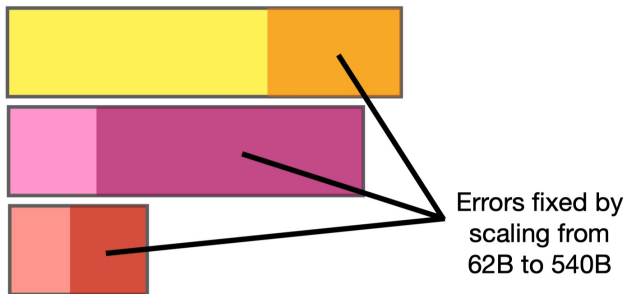
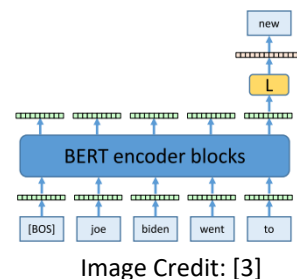


Figure 9: Error analysis of 45 problems that PaLM 62B got incorrect. These errors were categorized that semantic understanding, one step missing, and other. The other category includes hallucinations, repetitive outputs, and symbol mapping errors. Scaling PaLM to 540B fixed a substantial portion of errors in all categories.

Part 5 – Self-consistency Benefits Reasoning

- **Problem inside Chain of Thought Prompting:** naïve greedy decoding strategy [1, 2]
- Chain of Thought → **Chain of Errors**
- **Token Sampling Strategy** [3]:
 1. **Random** Sampling (**Temperature Sampling**, $T \in [0, 1]$): **randomly** select token
 2. **Top- k** Sampling: **select k tokens** with the highest probability. **Hard to choose k**
 3. **Top- p** Sampling (**Nucleus Sampling**): **smallest set of top candidates** with the cumulative probability above a threshold, *e.g.*, $p=0.95$
 4. **Greedy Decoding**: simply pick the **most likely token with the highest possibility**
 5. **Beam Search**: keep track the k most probable partial translations and pick the transaction with the highest probability (normalized by the number of target words) from the list.
- **Select the most consistent answer** by marginalizing out (**Voting**) the sampled reasoning paths [1]
- complex reasoning problem typically **admits multiple different ways**



Credits: [1] Wang *et al.*, Self-Consistency Improves Chain of Thought Reasoning in Language Models, In ICLR'23.

[2] Wei *et al.*, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, In NeurIPS'22

[3] Paaß *et al.*, Foundation Models for Natural Language Processing: Pre-trained Language Models Integrating Media, In Springer Nature'23.

Part 5 – Self-consistency Benefits Reasoning

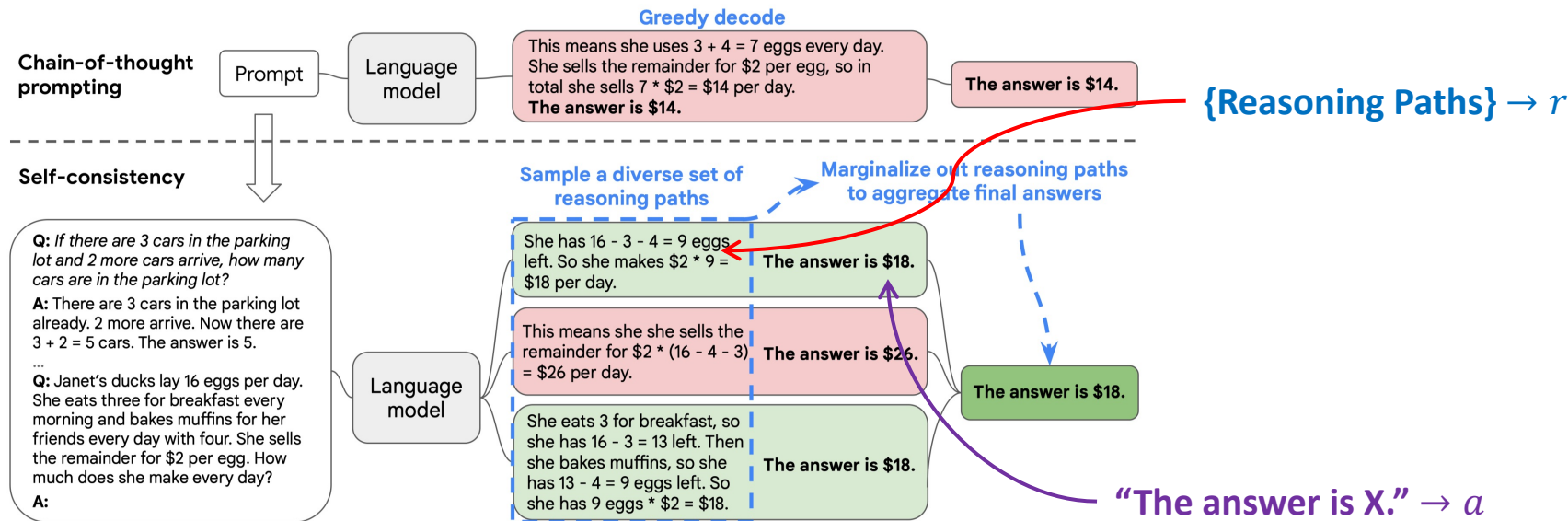
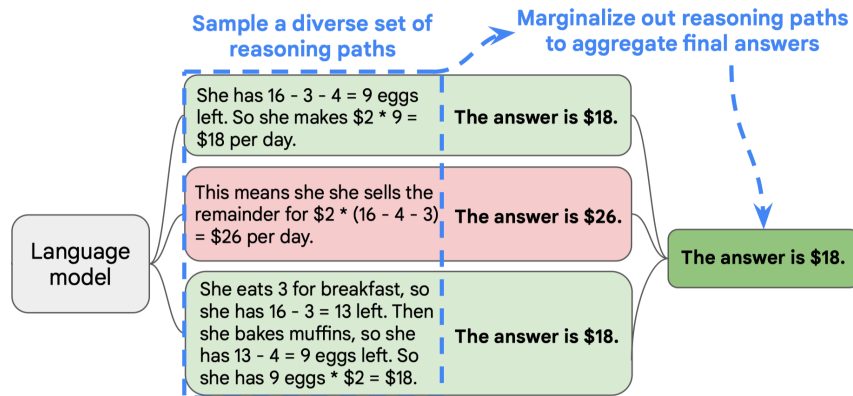


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the “greedy decode” in CoT prompting by sampling from the language model’s decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

Part 5 – Self-consistency Benefits Reasoning

- **Response Template:** “{Reasoning Paths}. The answer is X.”
- **Hypothesis:** correct reasoning processes, even if they are diverse, **tend to have greater agreement in their final answer** than incorrect processes.
- **Method:** Sample-and-Marginalize Decoding
- Sampling diverse reasoning paths → Aggregate the answers

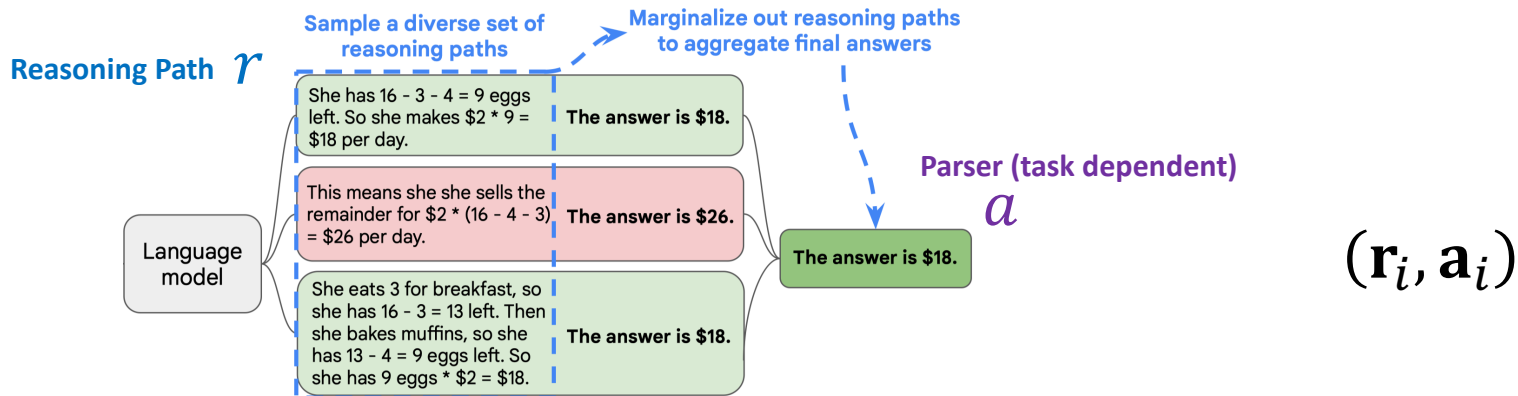


Part 5 – Self-consistency Benefits Reasoning

- **Response Template:** “{Reasoning Paths}. The answer is X.”
- **Majority Vote (Unweighted Sum):**

$$\operatorname{argmax}_a \sum_{i=1}^m \mathbb{1}(\mathbf{a}_i = a).$$

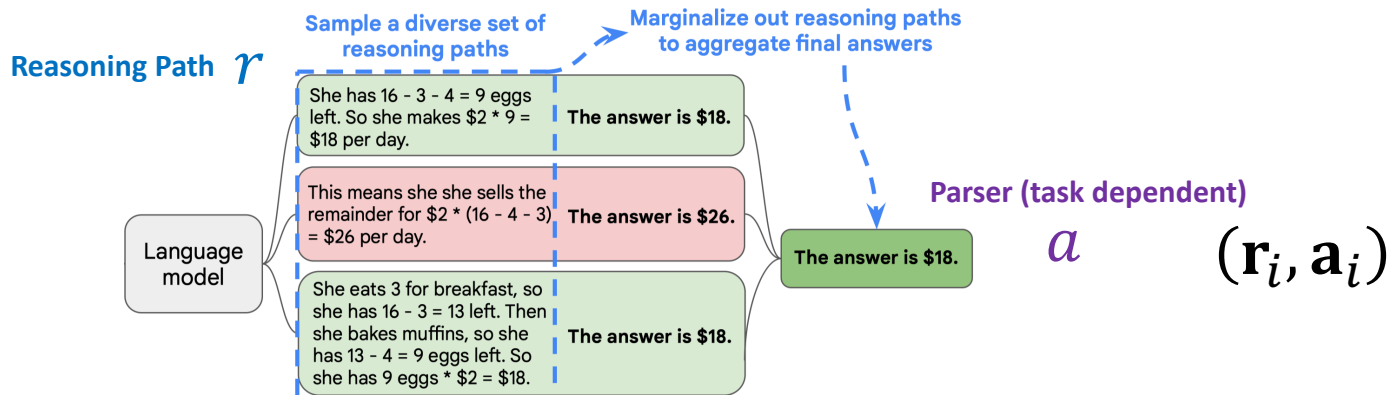
- By assuming the weight $\mathbb{1} = 1$



Part 5 – Self-consistency Benefits Reasoning

- **Weighted Vote (Unnormalized):** $p(\mathbf{r}_i, \mathbf{a}_i \mid \text{Prompt, Question})$

$$\operatorname{argmax}_a \sum_{i=1}^m \mathbb{1}(\mathbf{a}_i = a) \times p(\mathbf{r}_i, \mathbf{a}_i \mid \text{Prompt, Question}).$$

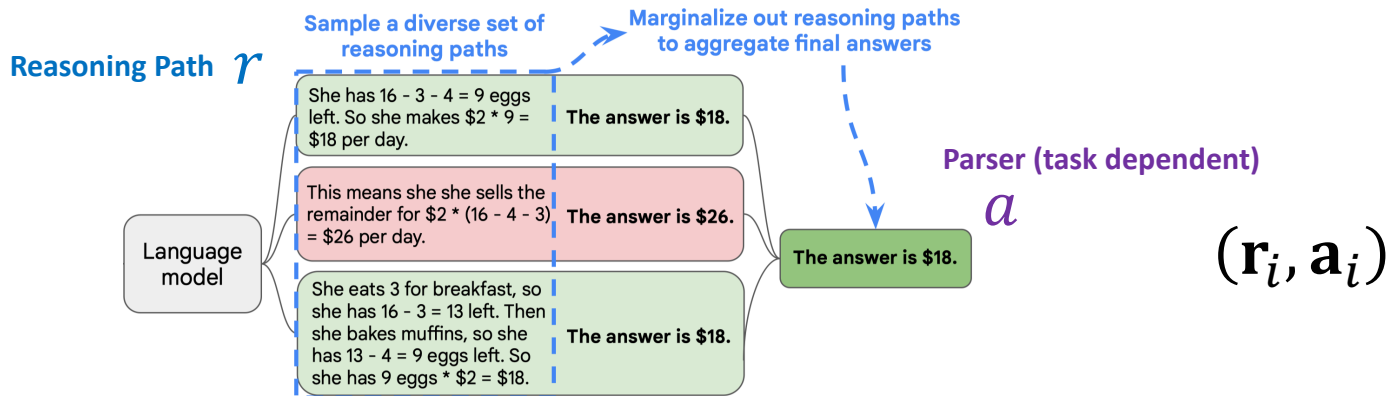


Part 5 – Self-consistency Benefits Reasoning

- **Weighted Vote (Normalized):** $p(\mathbf{r}_i, \mathbf{a}_i \mid \text{Prompt, Question})$

$$\operatorname{argmax}_a \sum_{i=1}^m \mathbb{1}(\mathbf{a}_i = a) \times \exp \frac{1}{K} \sum_{k=1}^K \log p(t_k \mid \text{Prompt, Question, } t_1, \dots, t_{k-1})$$

- K is the total number of tokens in $(\mathbf{r}_i, \mathbf{a}_i)$
- Normalize the conditional probability by the output sum



Part 5 – Self-consistency Benefits Reasoning

	GSM8K	MultiArith	AQuA	SVAMP	CSQA	ARC-c
Greedy decode	56.5	94.7	35.8	79.0	79.0	85.2
Weighted avg (unnormalized)	56.3 ± 0.0	90.5 ± 0.0	35.8 ± 0.0	73.0 ± 0.0	74.8 ± 0.0	82.3 ± 0.0
Weighted avg (normalized)	22.1 ± 0.0	59.7 ± 0.0	15.7 ± 0.0	40.5 ± 0.0	52.1 ± 0.0	51.7 ± 0.0
Weighted sum (unnormalized)	59.9 ± 0.0	92.2 ± 0.0	38.2 ± 0.0	76.2 ± 0.0	76.2 ± 0.0	83.5 ± 0.0
Weighted sum (normalized)	74.1 ± 0.0	99.3 ± 0.0	48.0 ± 0.0	86.8 ± 0.0	80.7 ± 0.0	88.7 ± 0.0
Unweighted sum (majority vote)	74.4 ± 0.1	99.3 ± 0.0	48.3 ± 0.5	86.6 ± 0.1	80.7 ± 0.1	88.7 ± 0.1

Table 1: Accuracy comparison of different answer aggregation strategies on PaLM-540B.

$p(\mathbf{r}_i, \mathbf{a}_i \mid \text{Prompt, Question})$ are quite close to each other

→ Potential Improvement: **Self-weighted self-consistency**

Part 5 – Self-consistency Benefits Reasoning

- Reasoning Tasks:**
Arithmetic (8 manually written exemplars),
Commonsense (4-7 exemplars), and *Symbolic* Reasoning
- Applied Models:** UL2 20B, GPT-3 (code-davinci001 and code-davinci-002 from the Codex series 175B), Codex (code-davinci-002 14.8B), LaMDA 137B, and PaLM 540B

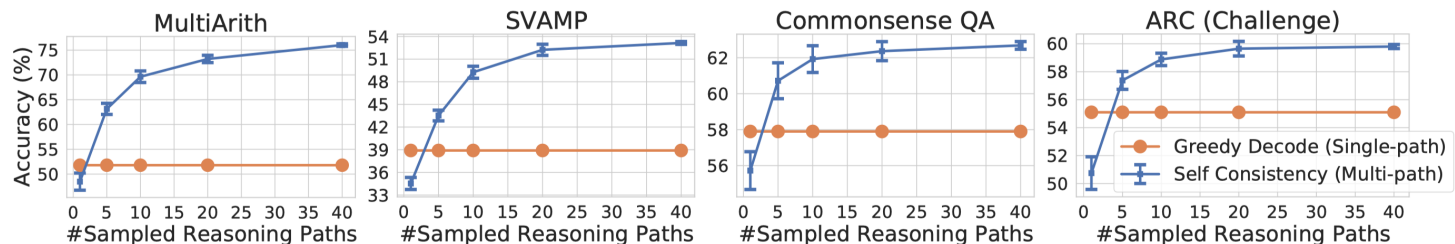


Figure 2: Self-consistency (blue) significantly improves accuracy over CoT-prompting with greedy decoding (orange) across arithmetic and commonsense reasoning tasks, over LaMDA-137B. Sampling a higher number of diverse reasoning paths consistently improves reasoning accuracy.

Part 5 – Self-consistency Benefits Reasoning

■ Sampling Strategies Impact:

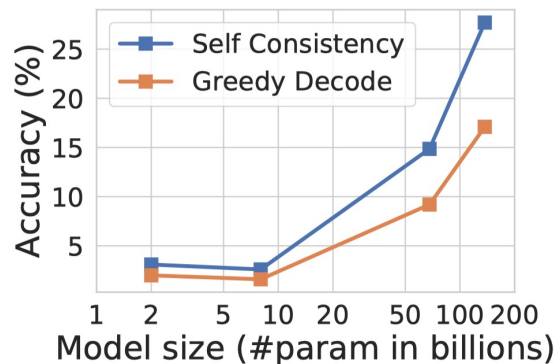
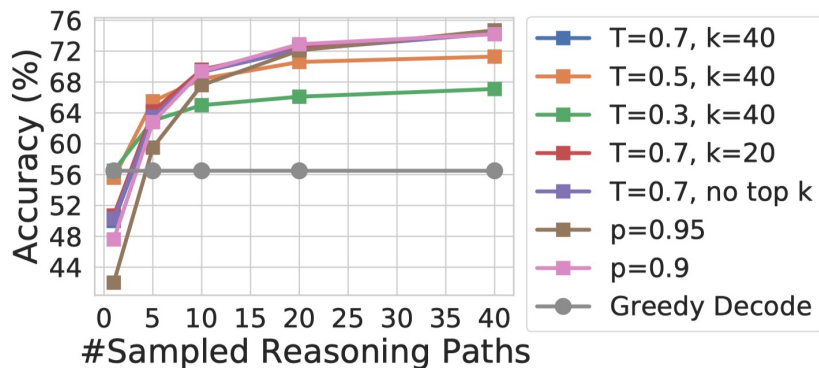


Figure 4: GSM8K accuracy. (Left) Self-consistency is robust to various sampling strategies and parameters. (Right) Self-consistency improves performance across language model scales.

- $T \in [0, 1]$: *Temperature* in **Temperature Sampling**
- k : **Top-k Sampling**
- $p \in [0, 1]$: Top- p Sampling (**Nucleus Sampling**)

$$p(x_l | x_{<l}) = \frac{\exp(x_l/T)}{\sum_K \exp(x_i/T)}$$

Part 5 – Self-consistency Benefits Reasoning

- Self-consistency improves **Robustness to Imperfect Prompts**

	Prompt with correct chain-of-thought	17.1
LaMDA-137B	Prompt with imperfect chain-of-thought	14.9
	+ Self-consistency (40 paths)	23.4
	Prompt with equations	5.0
	+ Self-consistency (40 paths)	6.5
PaLM-540B	Zero-shot CoT (Kojima et al., 2022)	43.0
	+ Self-consistency (40 paths)	69.2

Table 8: Self-consistency works under imperfect prompts, equation prompts and zero-shot chain-of-thought for GSM8K.

Robustness
to Imperfect Prompts

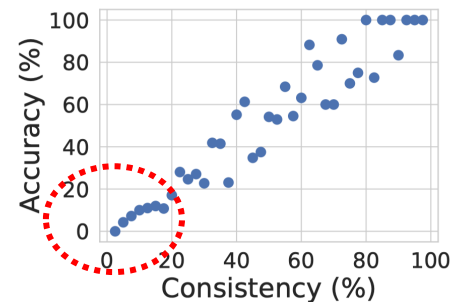


Figure 5: The consistency is correlated with model's accuracy.

Uncertainty Estimation

Low consistency → know when it doesn't know"

Thank you very much for your attention!