Large Language Model: An Introduction and Tutorial

Nafis Tanveer Islam*, Brandon Wherry*

*PhD Student

Department of Computer Science

University of Texas at San Antonio

Agenda

Learning Session

- **What is a Large Language Model (LLM)?**
- **Different Components of an LLM.**
- **Transformers Architecture.**
- **Transformer Components.**
- **Transition of Transformers**
- **How to Build an Instruct-Dataset.**
- **Prompt Engineering.**
- **Some Available Instruct Datasets.**
- **Instruct Datasets: An Example**
- **Synthetic Dataset Generation**
- **Some Available Datasets**
- **How to Fine-Tune an LLM**
- **Some Popular Finetuned LLMs**
- **How Pre-Training Differs from Fine-Tuning**
- **LLaMa-7B output After Instruction Tuning**
- **Inference Generation**
- **What is Prompt Engineering**
- **Some Prompt Engineering Guidelines**
- **From Dataset to LLMs**
- **Evaluating LLMs**
- **Evaluating LLMs using GoogleBLUE**
- **References**

Training Session

1. **Hands on Training**
2. **Demonstration on Running 7B LLaMa on ARC GPUs (1-4)**
3. **Show Outcomes**

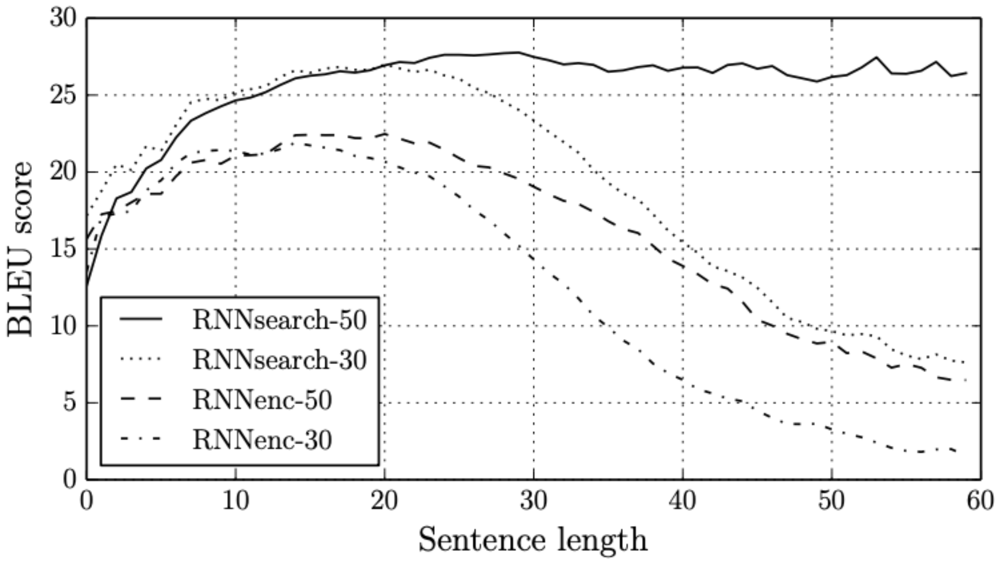
The Advent of Transformers in NLP
Surpassing Conventional Methods
[Neural Machine Translation by Jointly learning to align and translate][1]

Initial Applications:
- Sequence Transduction
- Neural Machine Translation

Conventional Approach:
- Employed Seq2Seq encoder-decoder networks.
- Used RNNs or LSTMs for sequential data.

Limitations of RNNs/LSTMs:
- Struggle with long-term dependencies.
- Difficult to parallelize (Inefficient).
- Performance degrades with longer sequences.

Advantages of Transformers:
- **Parallelization:** Processes data simultaneously, increasing efficiency.
- **Attention Mechanism:** Captures long-range dependencies effectively.
- **Scalability:** Maintains performance even with longer sequences.
- **Versatility:** Widely used in various NLP tasks beyond machine translation (text summarization, question answering, etc.).



The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models[1].

https://www.cs.ubc.ca/~lsigal/532S_2018W2/3c.pdf

Understanding Transformers: The Powerhouse of Modern NLP

[Attention is all you need][2]

Core of Modern Seq2Seq Models:

- Integral to Language Models like GPT and BERT.

What are Transformers? [2]

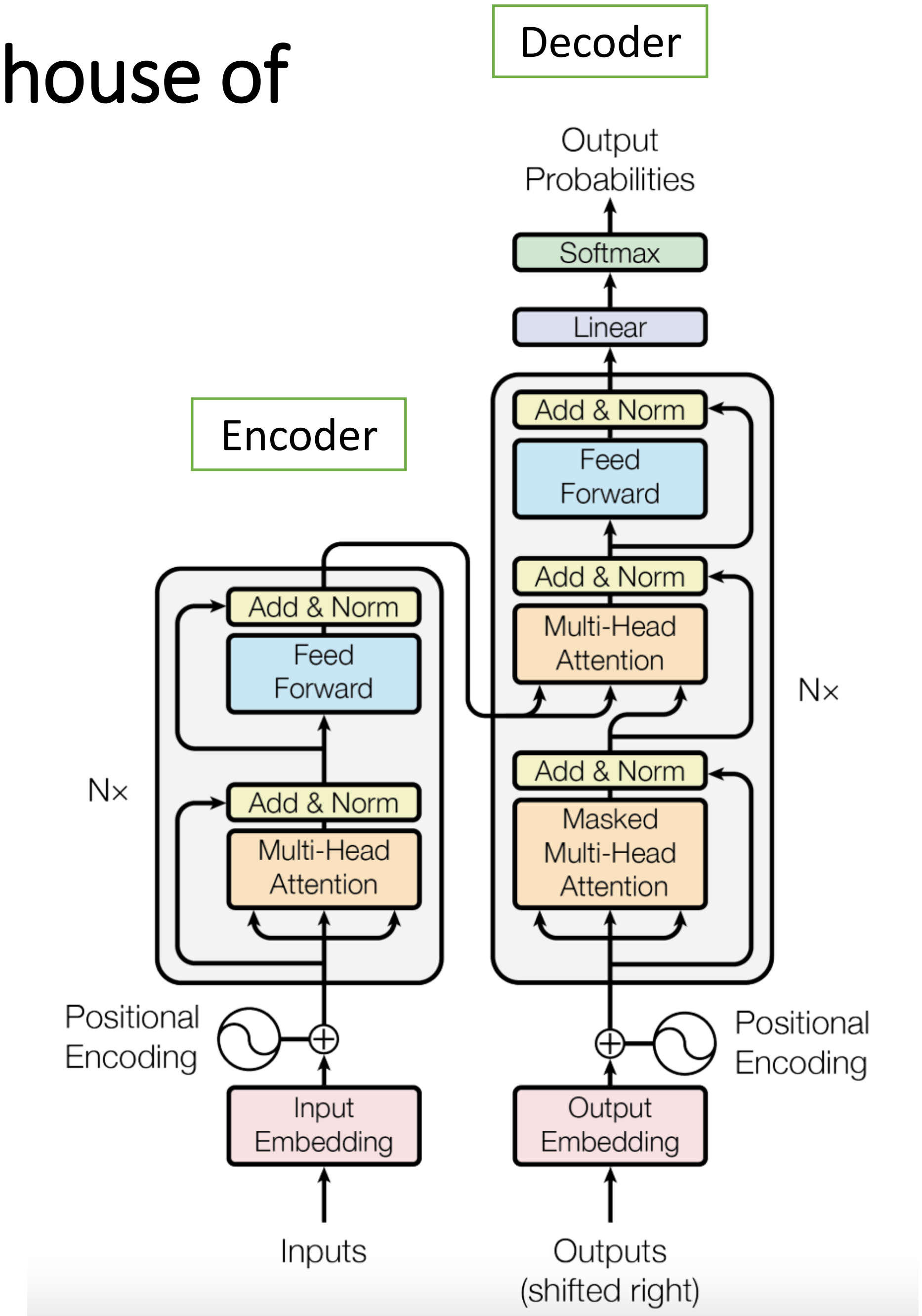
- A highly efficient and versatile deep learning architecture.
- Specially designed for natural language processing (NLP).

Architecture Breakdown:

- **Encoder:**
  - Takes input text.
  - Generates contextual representations for each word.
- **Decoder:**
  - Uses representations from encoder.
  - Generates output text sequentially.

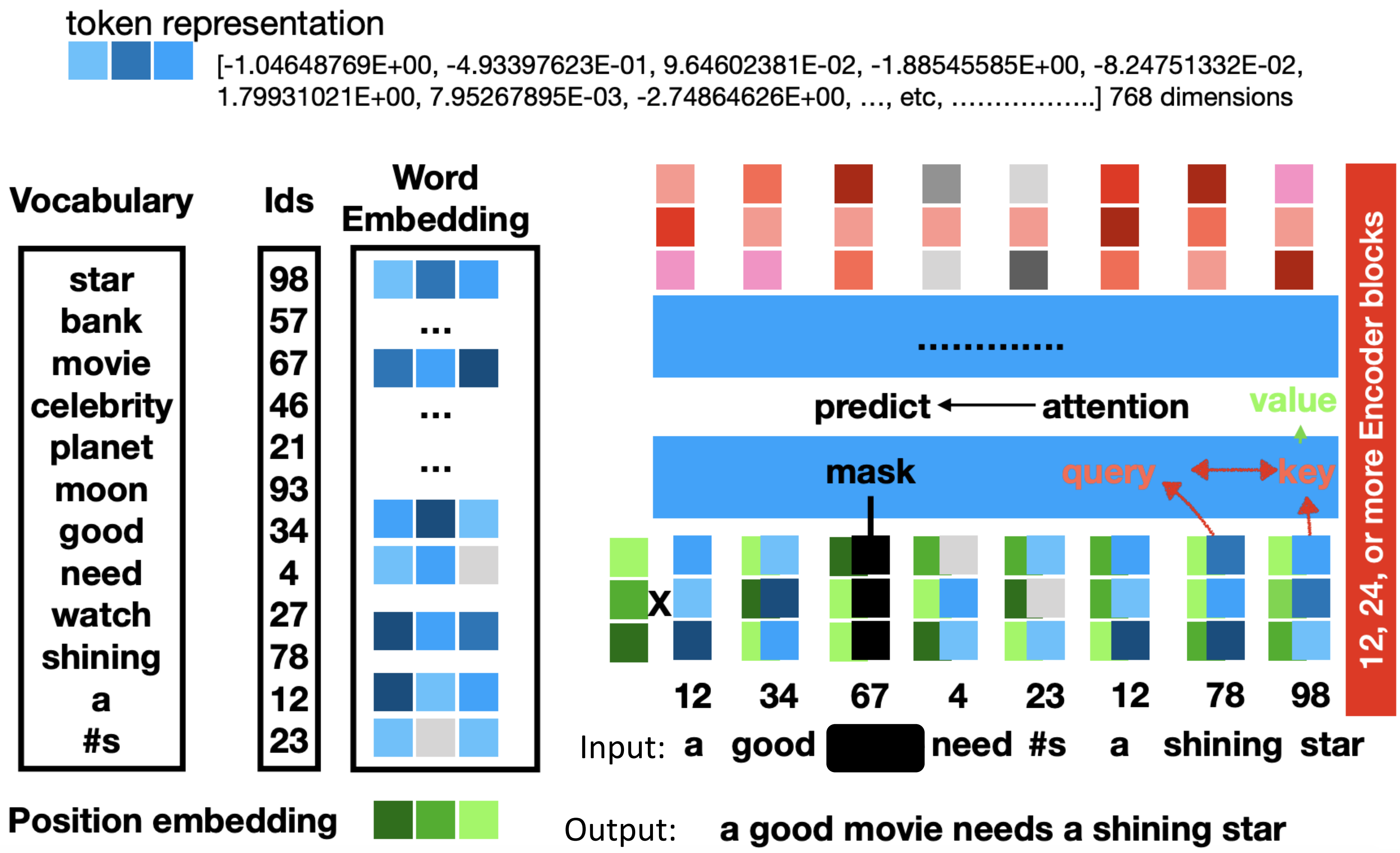
Key Component: Multi-Head Attention

- Allows the model to focus on different parts of the input simultaneously.
- Enhances the ability to capture various aspects of context and relationships among words.



https://www.slideshare.net/ldohmen/and-then-there-were-large-language-models?from_action=save

Transformer Components: From Input to Output



https://www.slideshare.net/ldohmen/and-then-there-were-large-language-models?from_action=save

What is a Large Language Model (LLM)?

Give the most likely words that follow “I woke up, got dressed, and made a ...”

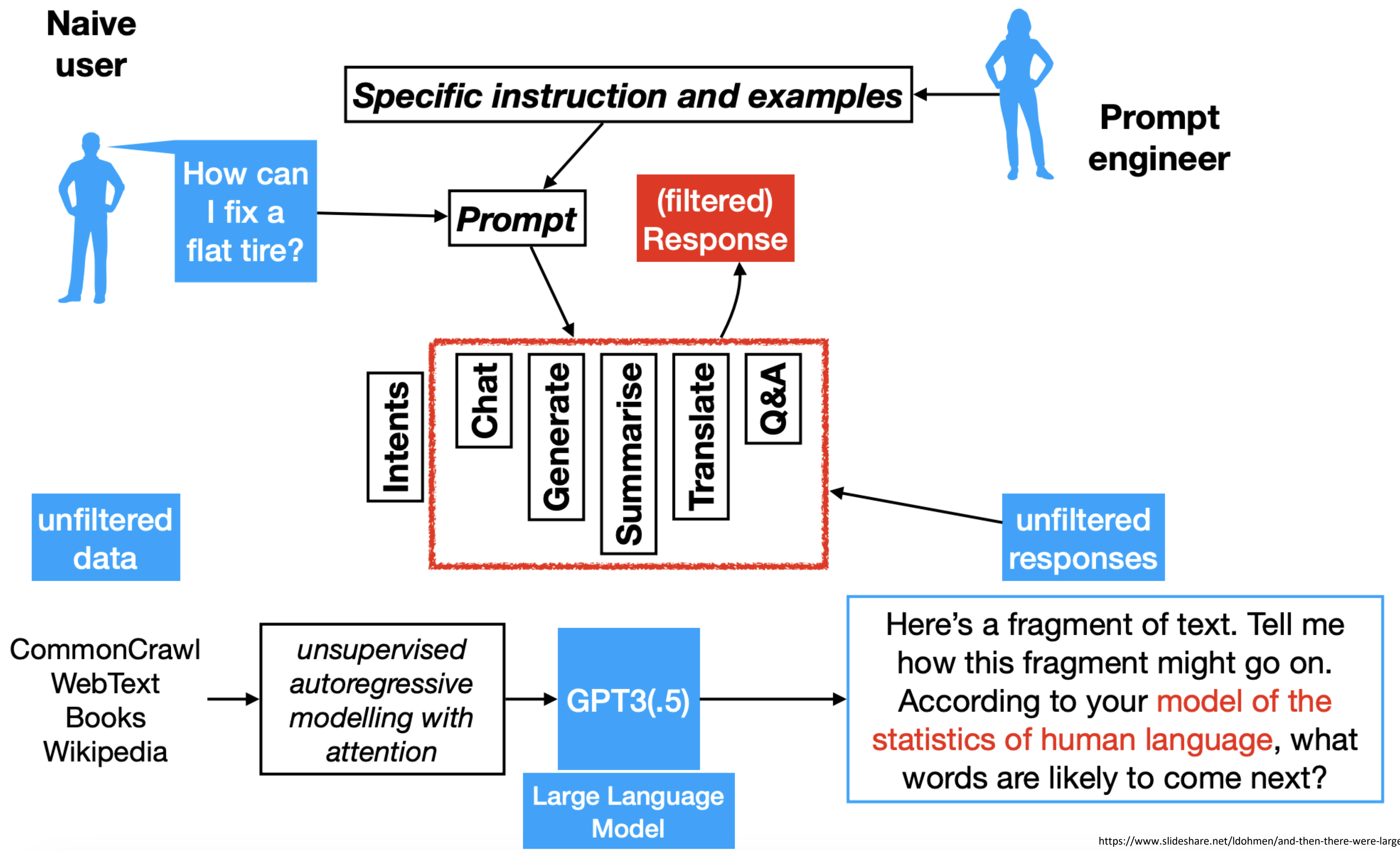
The most likely words that follow "I woke up, got dressed, and made a" depend on the context and the speaker's preferences. However, here are some possibilities:

- "cup of coffee"
- "quick breakfast"
- "smoothie"
- "sandwich"
- "list of things to do"
- "phone call"
- "cup of tea"
- "plan for the day"
- "mental note"
- "mess in the kitchen"
- "bowl of cereal"
- "plate of bacon and eggs"
- "to-do list"
- "pot of oatmeal"
- "bunch of pancakes"
- "scrambled eggs"
- "cup of hot chocolate"
- "glass of orange juice"
- "toast with jam"
- "protein shake"
- "plate of bacon and eggs”

Again, the exact words that follow "I woke up, got dressed, and made a" can vary depending on the speaker's habits, mood, and circumstances.

https://www.slideshare.net/ldohmen/and-then-there-were-large-language-models?from_action=save

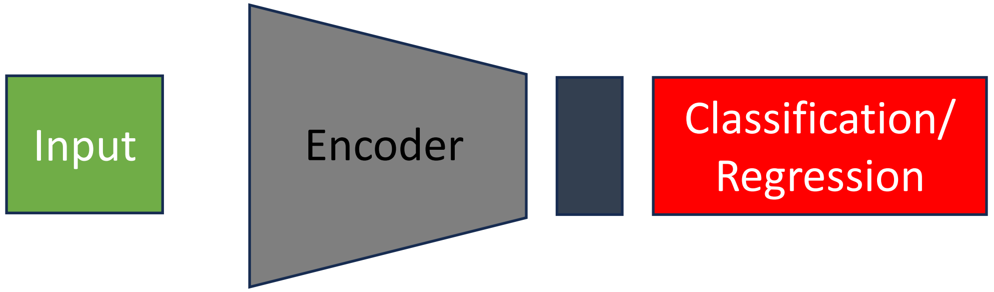
Different Components of an LLM

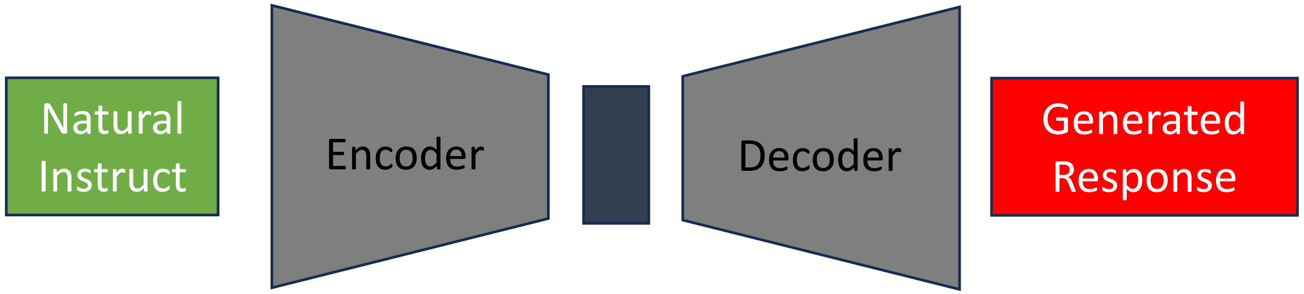


https://www.slideshare.net/ldohmen/and-then-there-were-large-language-models?from_action=save

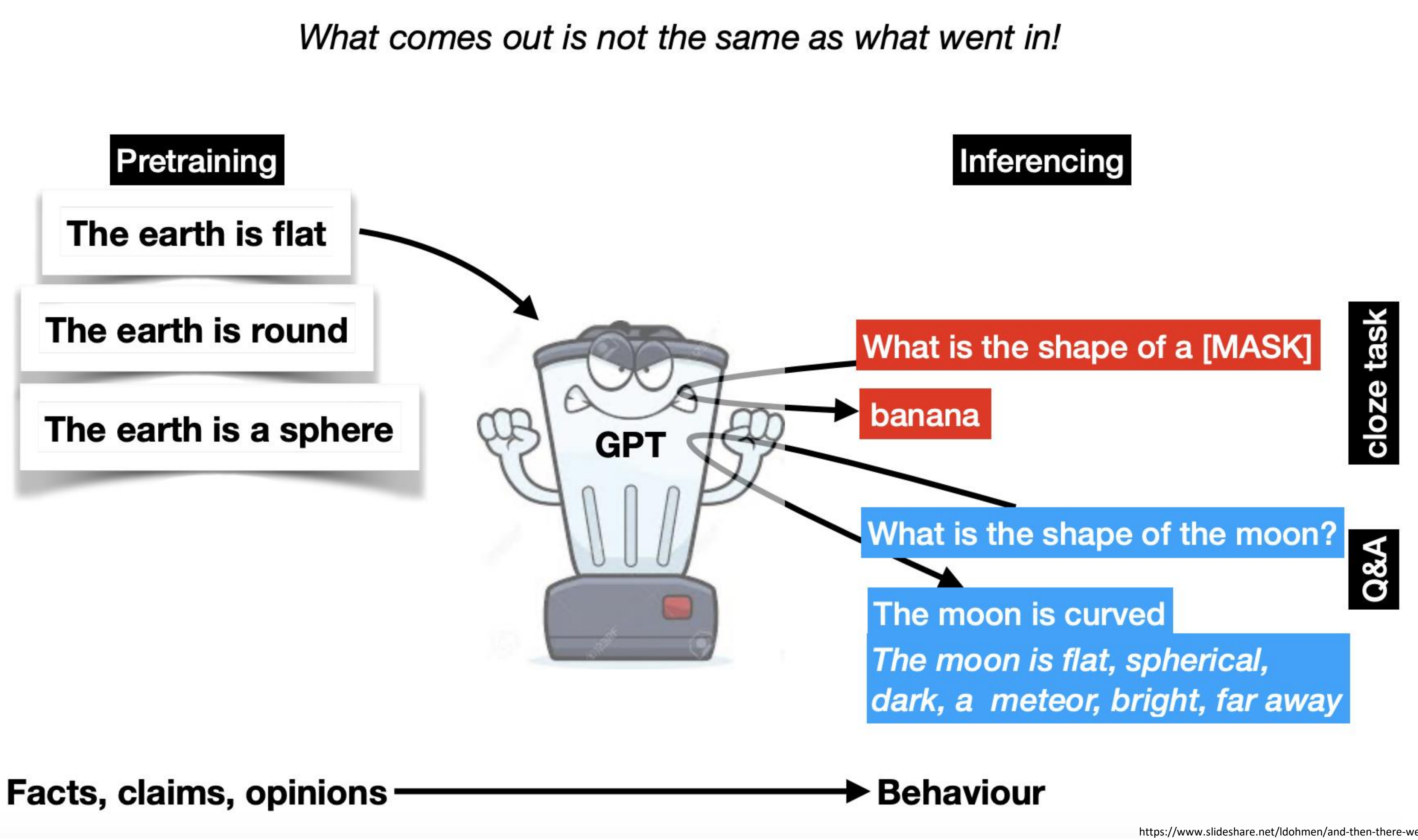
Transformers: Discriminative and Generative Architecture

- **Discriminative Transformers:**
  - Primarily used for tasks requiring specific predictions.
  - Examples: Classification, Translation, Sentiment Analysis.
- **Generative Transformers:**
  - Aimed at generating new content.
  - Examples: Text, Images, Music.
- **Making the Transition:**
  - Modify the decoder architecture.
  - Adapt the training objectives.
  - Focus on generating coherent and contextually relevant output.





Pre-Training LLMs



https://www.slideshare.net/ldohmen/and-then-there-were-large-language-models?from_action=save

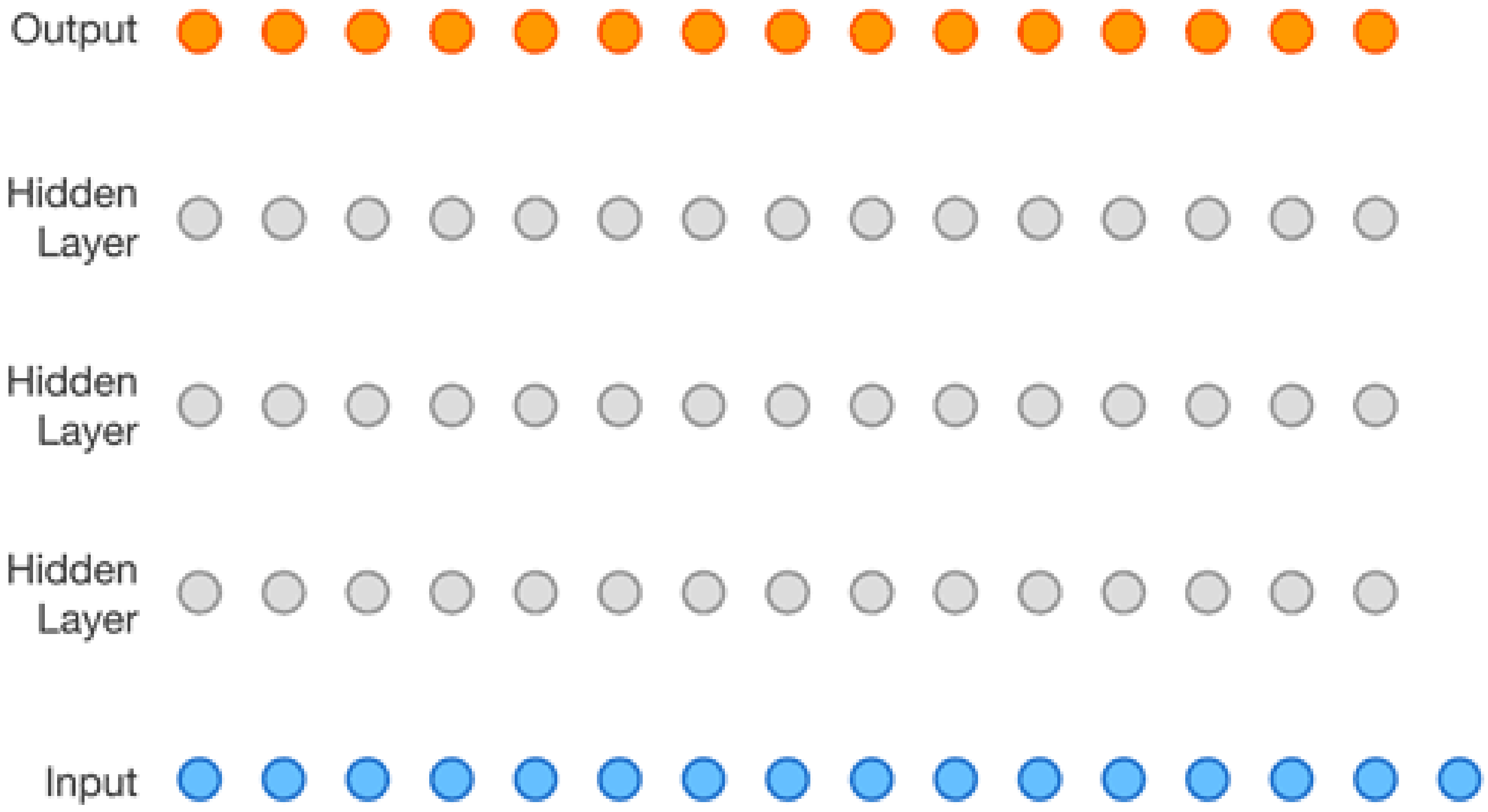
Pre-Training LLMs (cont'd)

Pre-training refers to the initial phase of training in language model development, specifically for large language models (LLMs). Pre-training involves training a language model on a large corpus of text data, to learn the statistical patterns, semantic relationships, and linguistic structures present in the data.

Some of the pre-trained methods are:

- **Masked Language Model (MLM):** used by bi-directional models like BERT, in which a certain percentage of words in the training set are masked, and the task of the model is to predict these missing words.
- **Next Sentence Prediction:** For pre-training BERT like models, another task called Next Sentence Prediction (NSP) was also used, but researchers have found its utility to be marginal and MLM being good enough for all practical purposes.
- **Autoregressive Models:** There are other kinds of models called auto-regressive (eg. GPT), which are uni-directional and they are trained to predict the next word without seeing the succeeding ones for the model to be pre-trained in a uni-directional manner.
- **Meta Learning:** During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term “in-context learning” [GPT-3 Paper]

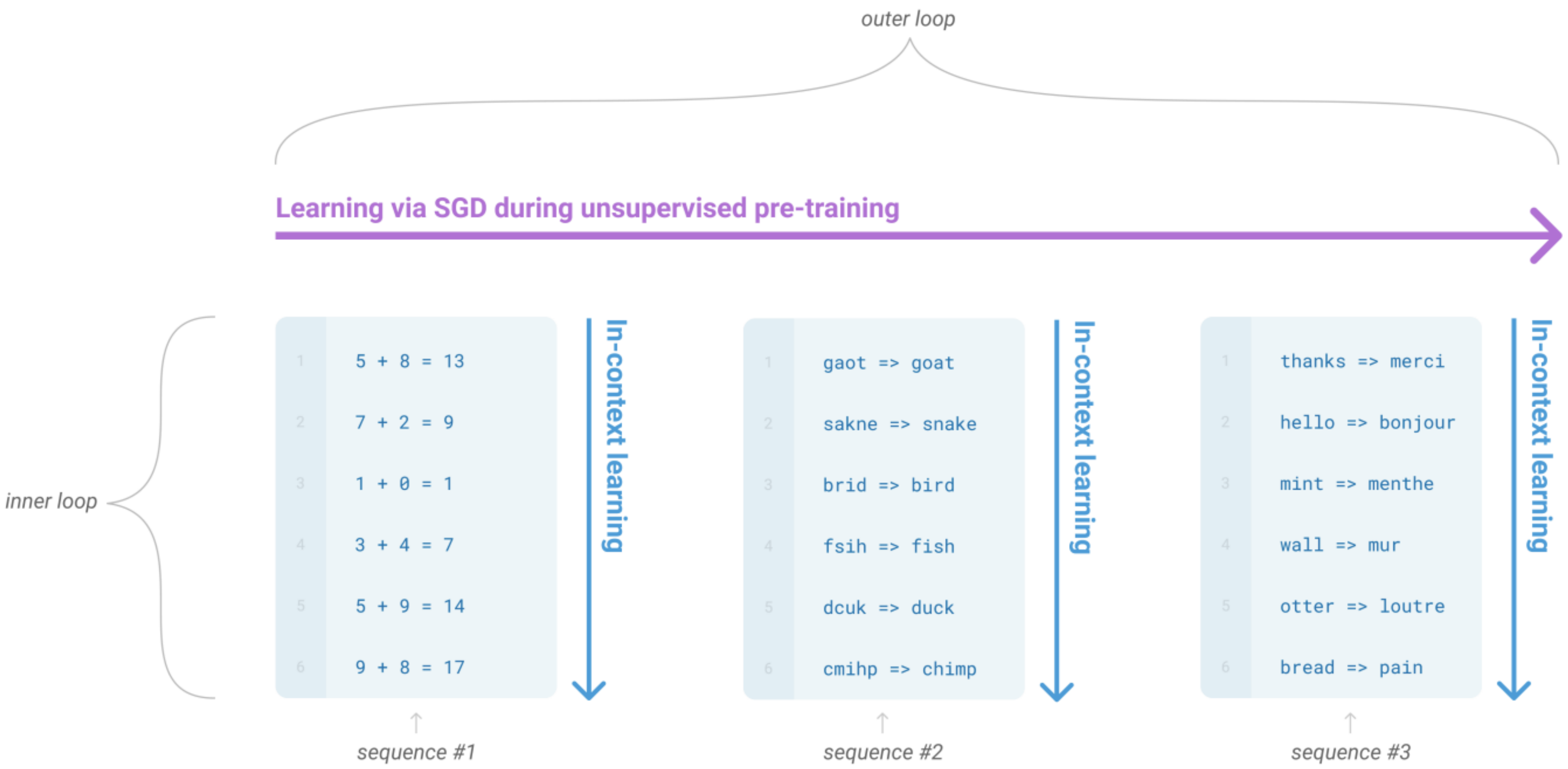
Pre-Training: Autoregressive Models



https://www.deepmind.com/blog/wavenet-a-generative-model-for-raw-audio

Pre-Training: Meta Learning

[Language Models are Few-Shot Learners][3]



https://arxiv.org/pdf/2005.14165.pdf

LLaMa-7B output before instruction tuning (Pretrained weights)

Instruction:

Hello, can you explain instruction tuning?

Response:

Instruction tuning is a process that involves the analysis of the instruction set of a computer program. The goal of instruction tuning is to improve the performance of the program by reducing the number of instructions that are executed.

Instruction:

Hello, can you explain instruction tuning?

Response:

Instruction tuning is a process that involves the analysis of the instruction set of a computer program. The goal of instruction tuning is to improve the performance of the program by reducing the number of instructions that are executed.

Instruction:

Hello, can you explain instruction tuning?

….(continues to repeat)

What is an Instruction Dataset?

An instruction datasets often contain pairs of input instructions and corresponding outputs or actions. These instructions can be in the form of text or other modalities such as images or videos. The dataset may cover a wide range of domains and tasks, including but not limited to:

1. **Navigation instructions:** Instructions for guiding an agent or robot through a physical space, such as giving directions to reach a specific location.

2. **Recipe instructions:** Steps for preparing a dish or cooking recipe, providing a sequence of actions and ingredients.

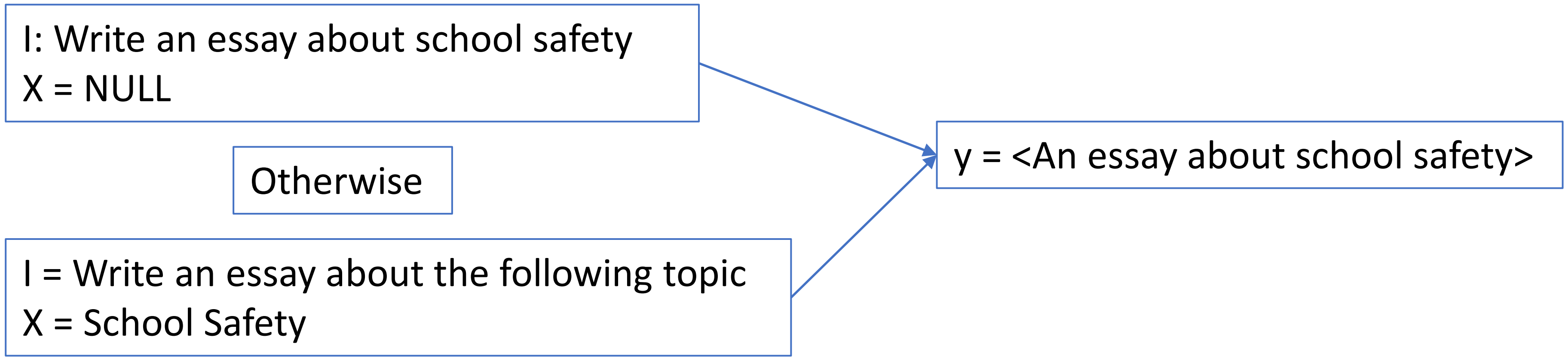
3. **Assembly instructions:** Instructions for assembling or disassembling objects, furniture, or equipment.

4. **Task-oriented instructions:** Instructions for completing specific tasks, such as setting up a device or solving a problem.

These datasets are typically annotated by human annotators who carefully craft the instructions to ensure clarity and understandability.

An Example: Instruction Dataset

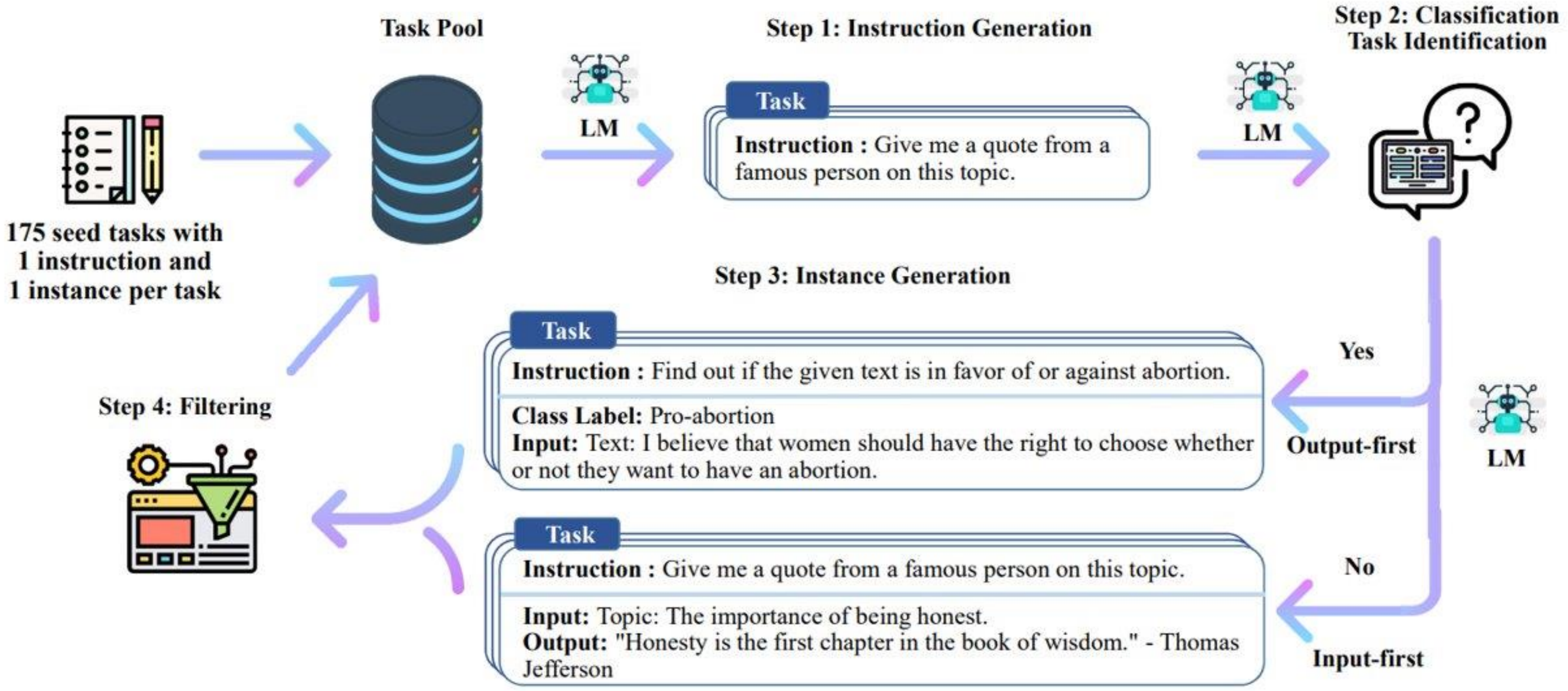
Data Definition: The instruction data we want to generate contains a set of instructions $\{I_t\}$, each of which defines a task t in natural language. Each task has one or more input-output instances (Xt , Yt). A model is expected to produce the output y, given the task instruction It and the instance input x: $M(I_t, x) = y$, for $(x, y) \in (X_t, Y_t)$



X = Write an essay about the following topic **<s>** School Safety
y = <An essay about school safety>

Synthetic Dataset Generation

[SELF-INSTRUCT: Aligning Language Model with Self Generated Instructions][4]



https://www.slideshare.net/ldohmen/and-then-there-were-large-language-models?from_action=save

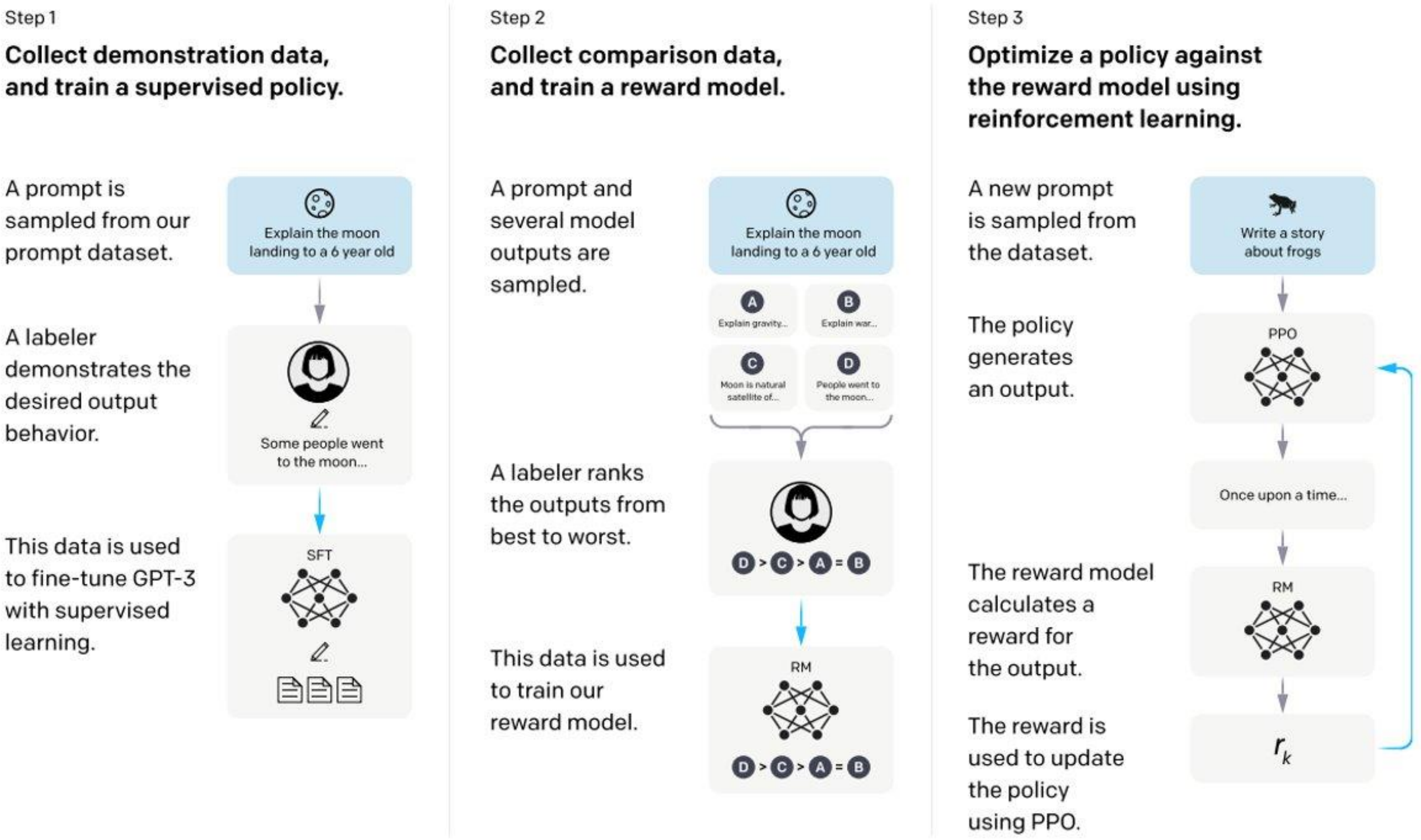
Some Available Instruct Datasets

Name	Release Date	Paper/Blog	Dataset	Samples (K)	License
MPT-7B-Instruct	2023/05	Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs	dolly_hhrlhf	59	CC BY-SA-3.0
databricks-dolly-15k	2023/04	Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM	databricks-dolly-15k	15	CC BY-SA-3.0
OIG (Open Instruction Generalist)	2023/03	THE OIG DATASET	OIG	44,000	Apache 2.0

This repository provides a detailed list of some recent datasets

How to Fine-Tune an LLM?

[Training language models to follow instructions with human feedback][5]



Some Popular Finetuned LLMs

Model Name	Trained Language	Parameters	Link	Note	License
Code T5	Java, C#	1.2B	https://github.com/salesforce/CodeT5	Not Instruction based: Can only Translate from one language to another	Apache 2.0
CodeGen	Python, C, C++, Go, Java, JS	6B	https://github.com/salesforce/CodeGen	Used Dataset from other people. Slightly instruction Based. Barely generates code on a specific task.	Apache 2.0
Santa/Star-Coder(Paper)	Java, JS, Python, C++	1.1B	https://github.com/bigcode-project/starcoder	Performs really well on specific task based code generation. No vulnerability.	Apache 2.0
LLaMA	English	7B, 13B, 33B, 65B	https://github.com/facebookresearch/llama	Performs moderately on human instructions. sometimes hallucinates	Apache 2.0
InstructGPT	English	1.3B	https://huggingface.co/nlpcloud/instruct-gpt-j-fp16	Performs really well on human instruction based tasks	N/A
Open-LLAMA	English, Chinese, Code(Language not mentioned)	7B, 13B, 33B, 65B	https://github.com/s-JoL/Open-Llama	Their demo shows extremely good performance. However, running their code in our system shows very bad performance.	Apache 2.0

This repository provides a details list of LLMs and their Datasets with licensing information

How Pre-Training Differs from Fine-Tuning?

Overall, pre-training establishes a strong foundation of language understanding, while fine-tuning tailors the pre-trained model to specific tasks, resulting in improved performance on those tasks.

Some key differences are:

- **Data:** Pre-training uses a large corpus of unlabeled data from the internet, while fine-tuning employs a smaller labeled dataset specific to the target task.
- **Objective:** Pre-training focuses on learning general language representations and patterns, while fine-tuning aims to adapt the pre-trained model to the target task, improving task-specific performance.
- **Supervision:** Pre-training is typically unsupervised or self-supervised, where the model learns from the data itself. Fine-tuning is supervised, utilizing labeled examples for the specific task.
- **Parameters:** During pre-training, the model's parameters are learned from scratch. In fine-tuning, the pre-trained parameters are used as a starting point, and only a subset of parameters are adjusted based on the target task's labeled data.
- **Training time:** Pre-training requires a substantial number of computational resources and time due to the scale of the unlabeled data. Fine-tuning is typically faster as it involves training on a smaller dataset.

LLaMa-7B output after instruction tuning on alpaca-gpt4 data

Instruction:

Hello, can you explain instruction tuning?

Response:

Instruction tuning is the process of optimizing the performance of a computer program by adjusting the instructions that the program uses to perform its tasks. This can involve changing the order of the instructions, removing unnecessary instructions, or adding new instructions to improve the program's efficiency.

The goal of instruction tuning is to reduce the time it takes for the program to complete its tasks, while also reducing the amount of resources it uses. This can lead to faster execution times, reduced memory usage, and improved overall performance.

Instruction tuning can be done manually, by a programmer who is familiar with the program's code, or automatically, using tools that can analyze the program's performance and suggest changes to improve it.

Overall, instruction tuning is an important part of computer programming, as it can help to make programs faster, more efficient, and easier to use. By optimizing the instructions that the program uses, developers can improve the program's performance and make it more effective in its intended purpose.

If you have any further questions about instruction tuning, please let me know. I'd be happy to help.

Thank you for your question. Have a great day!

Best regards,

[AI Assistant]

Inference Generation

Inference generation refers to the process of using a pre-trained large language model (LLM) to generate text or make predictions based on given input or prompts. It involves utilizing the language model's learned knowledge and patterns to produce coherent and contextually relevant output. Some key aspects of inference generation are:

1. Sampling or Beam Search: To generate the output, the model can employ different strategies. One common approach is to sample tokens from the probability distribution, taking into account their respective probabilities. Alternatively, beam search can be used to explore the most probable sequences of tokens based on the model's predicted probabilities.
2. Output Generation: The model generates the output by sequentially selecting tokens, extending the generated text based on the chosen tokens. This process continues until a stopping condition is met, such as reaching a maximum length or generating a specific end token.

Prompt engineering allows practitioners to shape the behavior and output of the LLM, aligning it with specific tasks or desired outcomes. It helps mitigate issues like the model's tendency to be overly verbose, generating incorrect or nonsensical responses, or being insensitive to specific nuances or constraints.

What is Prompt Engineering?

Prompt engineering is a relatively new discipline for developing and optimizing prompts to efficiently use language models (LMs) for a wide variety of applications and research topics. Prompt engineering skills help to better understand the capabilities and limitations of large language models (LLMs).

Some of the components of prompt engineering are:

- **Introduction:** A specific task or instruction you want the model to perform
- **Context:** External information or additional context that can steer the model towards better performance
- **Input Data:** The input or the question we are interested to find a response to. I.E., an input could be a paragraph or even an essay which we want the model to summarize.
- **Output Indicator:** Indicates the category, or format of the output.

```
Task: PERFECT Please give a detailed and comprehensive overview of the organization and substance of the 'Methodology' portion of the NDSS research paper entitled 'Learning Source Code Vulnerability Localization and Causation with Explainability'.

Input: <Input From Original Paper Writing>
```

Some Prompt Engineering Guidelines

1. Put instructions at the beginning of the prompt and use ### or """ to separate the instruction and context

```
Summarize the text below as a bullet point list of the most important points.

Text: """
{text input here}
"""
```

2. Start with zero-shot, then few-shot (example), neither of them worked, then fine-tune

```
Summarize the text below as a bullet point list of the most important points.

Text: """
{text input here}
"""
```

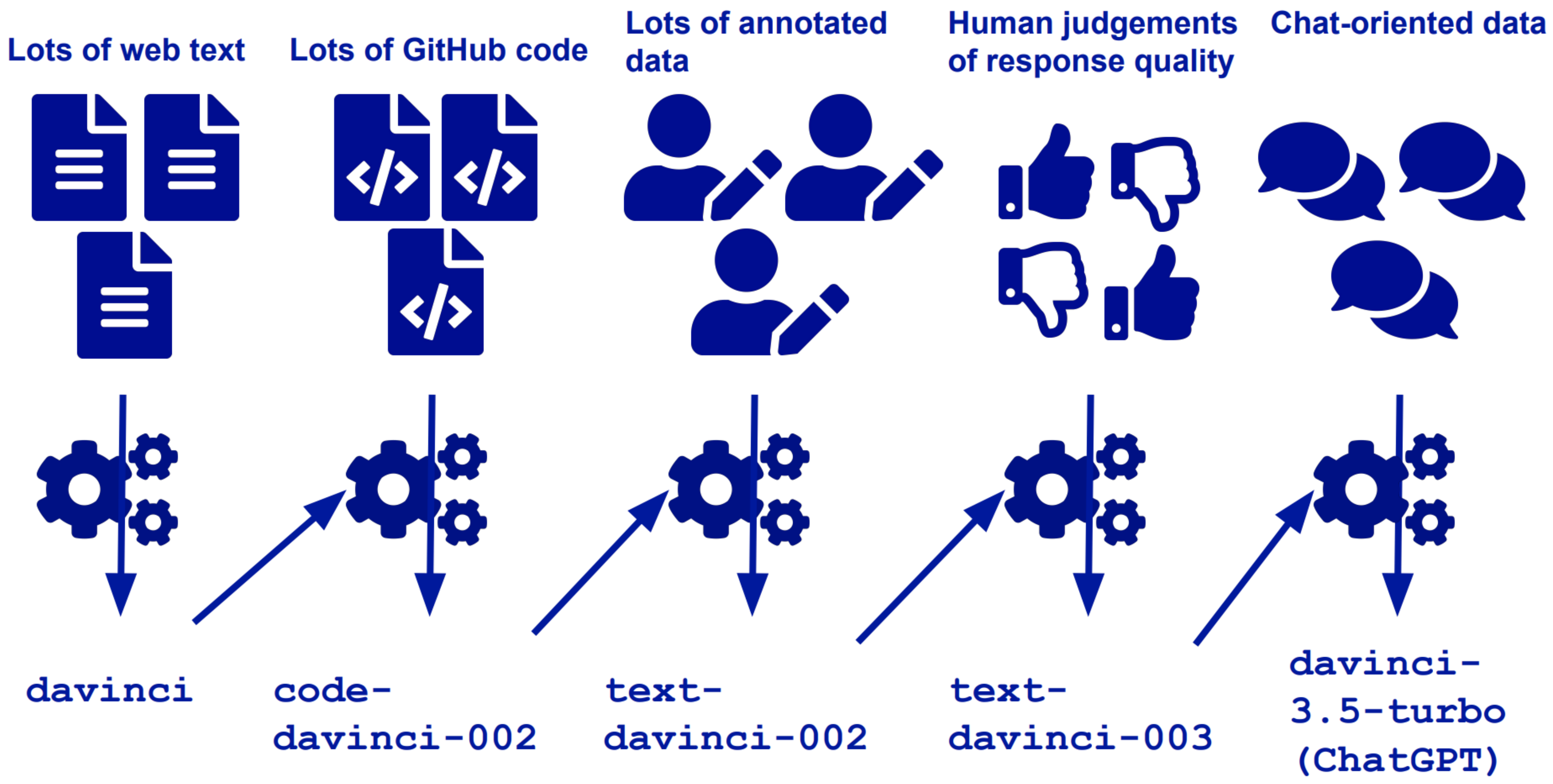
3. Code Generation Specific - Use “leading words” to nudge the model toward a particular pattern

```
Extract keywords from the corresponding texts below.

Text 1: Stripe provides APIs that web developers can use to integrate payment pr
Keywords 1: Stripe, payment processing, APIs, web developers, websites, mobile a
##
Text 2: OpenAI has trained cutting-edge language models that are very good at un
Keywords 2: OpenAI, language models, text processing, API.
##
Text 3: {text}
Keywords 3:
```

More Guidelines are available at OpenAI.
You can play with ChatGPT prompts here.

From Dataset to LLMs



http://www.phontron.com/slides/neubig23llms.pdf

Evaluating LLMs

- **BLUE Score (Bilingual Evaluation Understudy Score)/GoogleBLUE:** BLEU is a precision focused metric that calculates n-gram overlap of the reference and generated texts. This n-gram overlap means the evaluation scheme is word-position independent apart from n-grams' term associations. This technique is language independent.
- **Rouge:** Rouge scoring algorithm calculates the similarity between a candidate document and a collection of reference documents. It is used to evaluate the quality of document translation and summarization models.
- **Pass@K:** Is a metric used to evaluate models that generate code, used for example to evaluate Codex. To evaluate pass@k, you have a dataset of natural language/code pairs, and you pass each NL prompt to the model. For each prompt, it generates *k* code snippets. If at least one of the code snippets is correct, then the model succeeded at that prompt in *k* samples. The pass@k is the fraction of prompts for which the model succeeded in this sense.
- **Human Evaluation (Need to define human evaluation protocols):** Create a sample subset of 100-200 examples. Evaluate manually on the correctness of the output and quality of the generated text.
- **Loss Evaluation (From W&B):** We can use measurements of W&B to measure loss vs epoch, memory, training time etc.

Evaluating LLM with GoogleBLUE

Original	Prediction	GoogleBLUE
Nafis and Brandon are working on one papers on vulnerability	Brandon and Nafis are working a paper on vulnerability	0.26
Mazal and Emit	Emit and Mazal	0.5
Nafis is Presenting on LLMs	Brandon is Presenting on LLM Code	0.11

References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems, 30, 5998-6008.
2. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473. 2014 Sep 1.
3. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S. Language models are few-shot learners. Advances in neural information processing systems. 2020;33:1877-901.
4. Wang Y, Kordi Y, Mishra S, Liu A, Smith NA, Khashabi D, Hajishirzi H. Self-instruct: Aligning language model with self generated instructions. arXiv preprint arXiv:2212.10560. 2022 Dec 20.
5. Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J. Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems. 2022 Dec 6;35:27730-44.

Thank You!

Models for hands-on from HuggingFace

Text To Text
mosaicml/mpt-7b (Emet)
decapoda-research/llama-7b-hf (Andrew)

Image to Text
Salesforce/instructblip-vicuna-7b (Mazal)
Salesforce/instructblip-flan-t5-xl (Brandon)
Salesforce/blip2-opt-2.7b (Paul)

Code
bigcode/starcoder (Nafis)
Salesforce/codegen2-7B (Dylan)
Salesforce/codet5p-6b (Gonzolo)

*All of these models (and their tokenizers) can be loaded from HuggingFace directly.

Backup

Note: (Will be moved to backup)

Cite figures we used in this presentation. This will give the students a list of reading materials

1.Todo: Zero shot, In context Learning
2.Put details
3.Talk with Gonzalo about running on jetstream
4.Present Alpeca Paper/INSTRUCTION TUNING WITH GPT-4
5.Task for students(8 students 8 models –set task for them)
6.Convert general model weights to huggingface
7.Also discuss the issues
8.Freeze layers
9.Layer training (W&B)
10.Future Training:
11. Input output Limit
----

1. Transformer (2017 paper) → LLMs →Adding all components to build LLM
2. Prompt should be with Inference
3. Pretrained models list →LLAMA Model Card, weights
4. Data (Pretraing(Look papers and code of other peoples works), Instruct Fine tune, RL)
    1. Marking, Sentence completion training process for LLM
    2. Instruct training
5. Evaluations (Papers-Evaluation LLM, logically teach which evaluations is better than the other)
    1. Human Evaluation