Demystifying Large Language Models (LLM)

How do we use them in reliability?

RAM Summit

LOCKHEED MARTIN

Nov 1, 2023

Nathan Rigoni AI/ML Engineer Staff (NLP Tech Lead)

Over view

Types of LLM

- Transformers
  - Encoder

    BERT - Compressing context to be analyzed
  - Decoder

    GPT - Generating expected output based on context

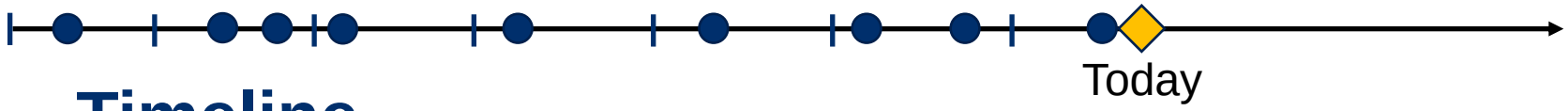
    Diffusion - Generating novel content based on context

How do we use them in Reliability?

- *Remaining Useful Life (UBL)*
- *Improved MTTR (ADaRA)*
- *Improved Maintainability (Sensor Rig)*

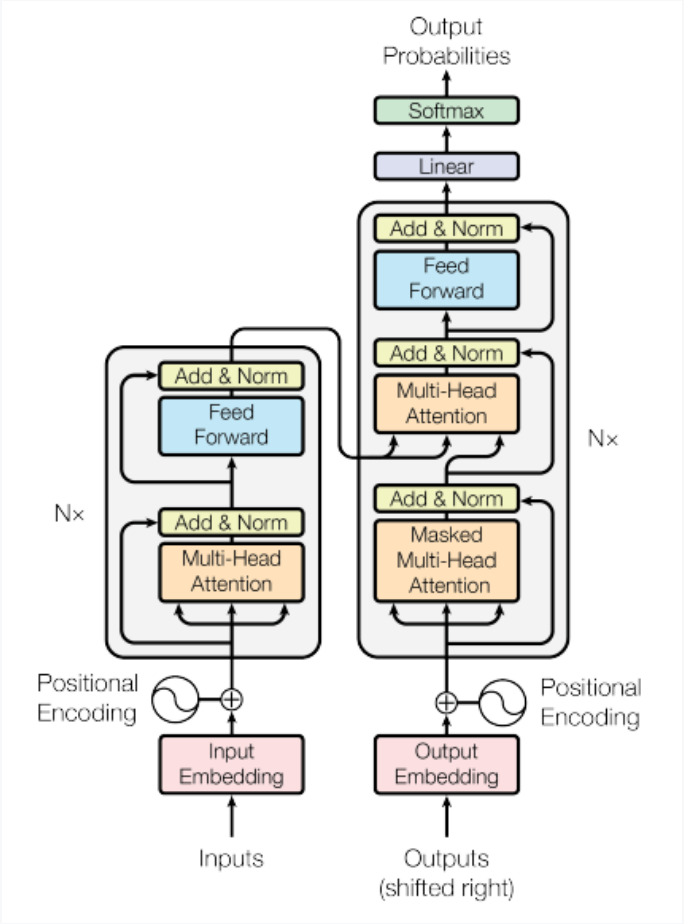
Things to consider:

Experimental Example



Timeline

- (Birth) June 12, 2017: ***Attention Is All You Need*** (Vaswani et al)
- June 2018: ***GPT*** (OpenAI)
- Oct 11, 2018: ***BERT*** (Devlin et al)
- Feb 14, 2019: **GPT2** (OpenAI)
  - Wrote a story about a unicorn
  - Deemed too dangerous to share with the public
- May 2020: **GPT3** (OpenAI)
  - 175B parameters
  - Can write code *
- May 2021: **Diffusion**
- Apr 2022: **DALL-E 2** (OpenAI)
- Nov 2022: **chatGPT** (OpenAI)
- July 2023: **GPT4** (OpenAI)



What are Large Language Models?

Attention based network divided into encoder and decoder blocks

Encoder and decoders are configured depending on the task

Block networks are configured for optimization and processing

Currently 3 Main architectures:

- BERT (embeddings)
- GPT (generative)
- Diffusion (generative)

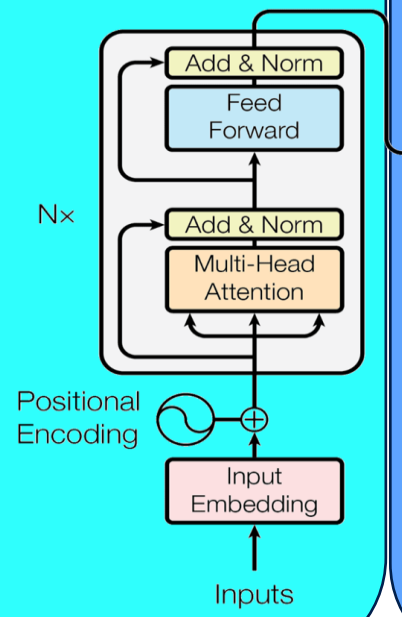
They are not just chatbots, *they are early versions of AGI that have divergent behavior that provide insight across domains of the training data.*

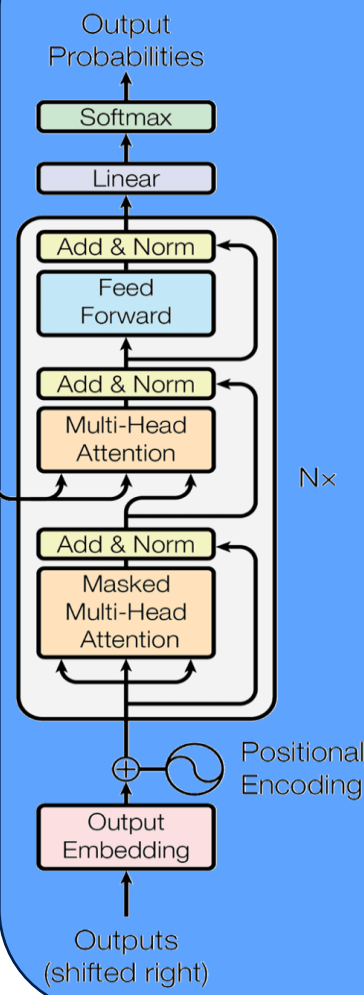
"Theory of Mind" https://arxiv.org/abs/2302.02083

BERT

Google

Embedding model intended explicitly for transfer learning using its massive learned embedding space based on sub-word tokens





GPT

OpenAI

Generative model that decodes text from a massive learned embedding space for sub-word tokens

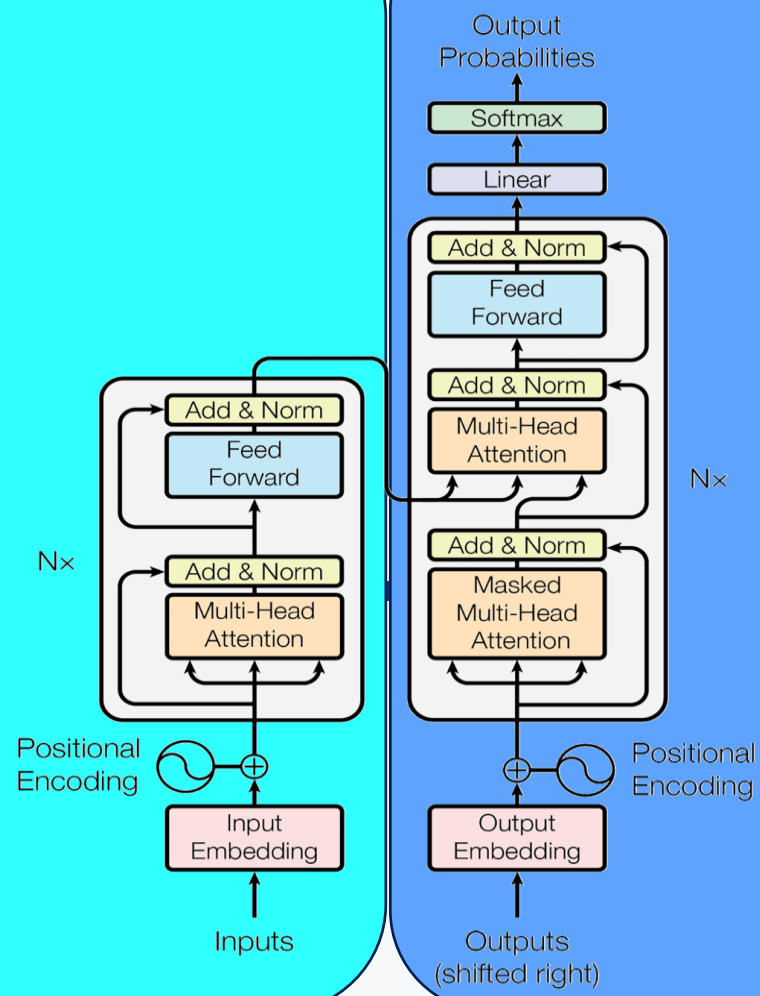
What Large Language Models

- Encoder Decoder Transformer:
  - Utilizing BERT for reading comprehension of text prompts from users.
  - Using GPT for generating text responses
  - Trained on conversational interaction
- Encoder and Decoder can be pretrained in a disconnected format.
- They are connected by a cross attention layer
- This full transformer can then post trained in conversations with engineers or for other tasks

The human brain exhibits relationships to information similar to encoding.

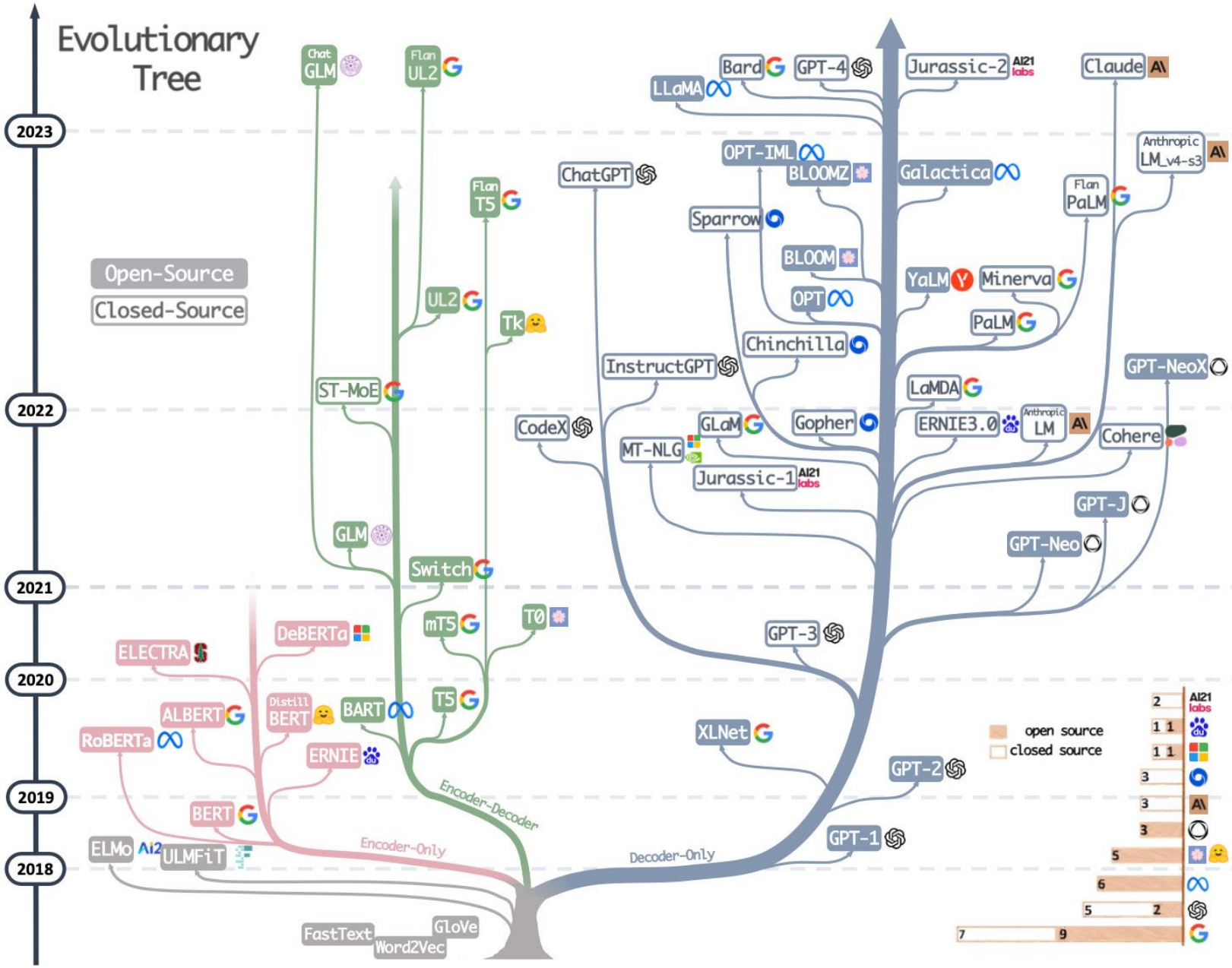
BERT
Google

Embedding model intended explicitly for transfer learning using its massive learned embedding space based on sub-word tokens



GPT
OpenAI

Generative model that decodes text from a massive learned embedding space for sub-word tokens



https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fb1f675dc-d643-405d-8111-eab067b7012f_1538x1188.png

BERT Problem

The setup for BERT involves solving 2 separate problems:

- What are the Masked words?
  - A random 10% to 15% of tokens (words) are masked from the sentences
- Which sentence comes first?
  - 2 sentences are sampled at random from a dataset and ordered randomly

These problems teach BERT to compress information within sentences and preserve the meaning of the sentence and words in the compression.

Because of the efficient compression of context in BERT it is typically used as the model that takes in text and feeds it to other models for inference.

GPT Problem

The setup for GPT involves solving 1 problem

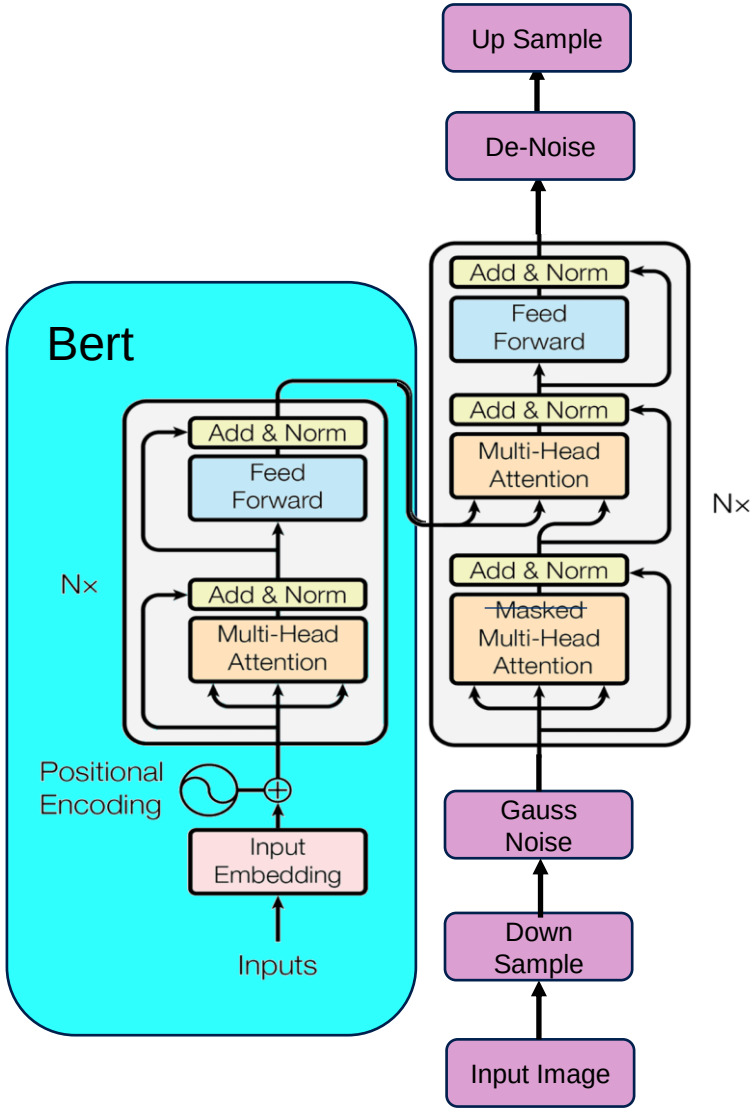
- What word comes next?
  - A block of text is randomly sampled from a dataset
  - Starting from the first word in the sequence GPT learns to predict the next word over and over until the entire block is complete

ChatGPT takes this model a step further by adding a BERT model to encode a context for prompt and response pairing along with RLHF to tune responses.

This problem teaches GPT how to construct sentences and paragraphs in a coherent fashion in differing formats.

Diffusion

- Novel part of the architecture and use is the addition of gaussian noise
- Training is done using incremental steps of noise
- Training goal is to predict the amount of noise in the image
- Final product allows for an input of random gaussian noise that the model uses as a seed in order to generate a novel image
- Bert is used to provide context to the image in order to allow the model to be guided in its generation



Diffusion Problem

The setup for Diffusion involves solving 2 problems

- How much noise has been added to the image?
  - An image has randomly generated Gaussian noise added to it in constant increments.
  - Since the increments are constant the model should be able to evaluate the image with noise and try to say how many increments have been added to the original image.
  - Subtracting this many increments of noise from the image should reproduce the original image.
- Given this smaller image what does the larger image look like?
  - Images are resized (down sized) in order to fit into memory for training
  - Once noise training is complete the image must be resized to its original perspective.

These problems teach diffusion how to arrange pixels in an image to generate coherent images.

The sum of two independent normally distributed random variables is normal, with its mean being the sum of the two means, and its variance being the sum of the two variances

How are these models useful?

Generative capabilities

Images

Surrogate Data

Text

Reports

Templates

Messaging

Code

Examining the digital fingerprint of how files are stored and how digital systems communicate, everything works off of text.

Relationship with Reliability

Reliability

Two main ideas in how LLMs are related to reliability

Generative applications in monitoring reliability and maintainability of systems

Remaining Useful Life (UBL)

Improved MTTR (ADaRA)

Improved CBM (Sensor Rig)

Reliability of generative applications

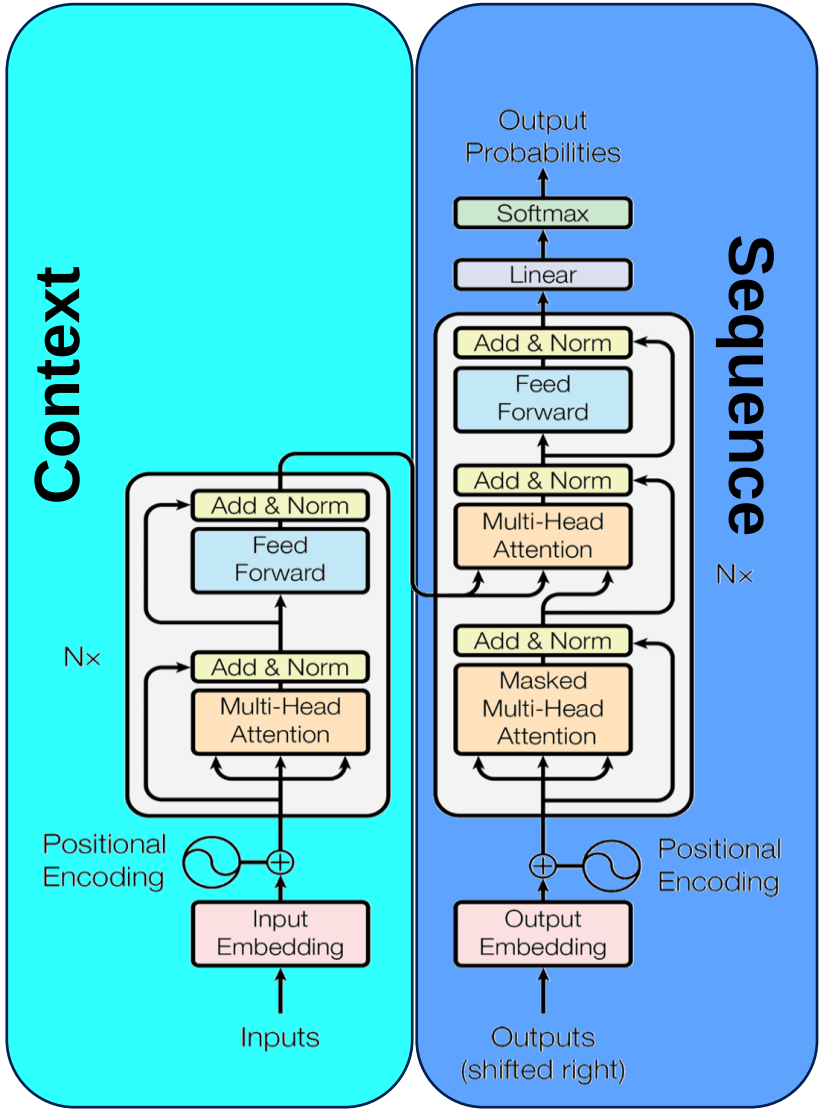
Failure rate of predictions

Transformers

The tech behind LLMs is the transformer, and its not limited to language.

- Left side is input context. Important information that informs the sequential pattern being generated

- Right side is the sequential pattern of interest. This could be a time series, sensor values, or an ordered sequence of words (like GPT)



Remaining Useful Life

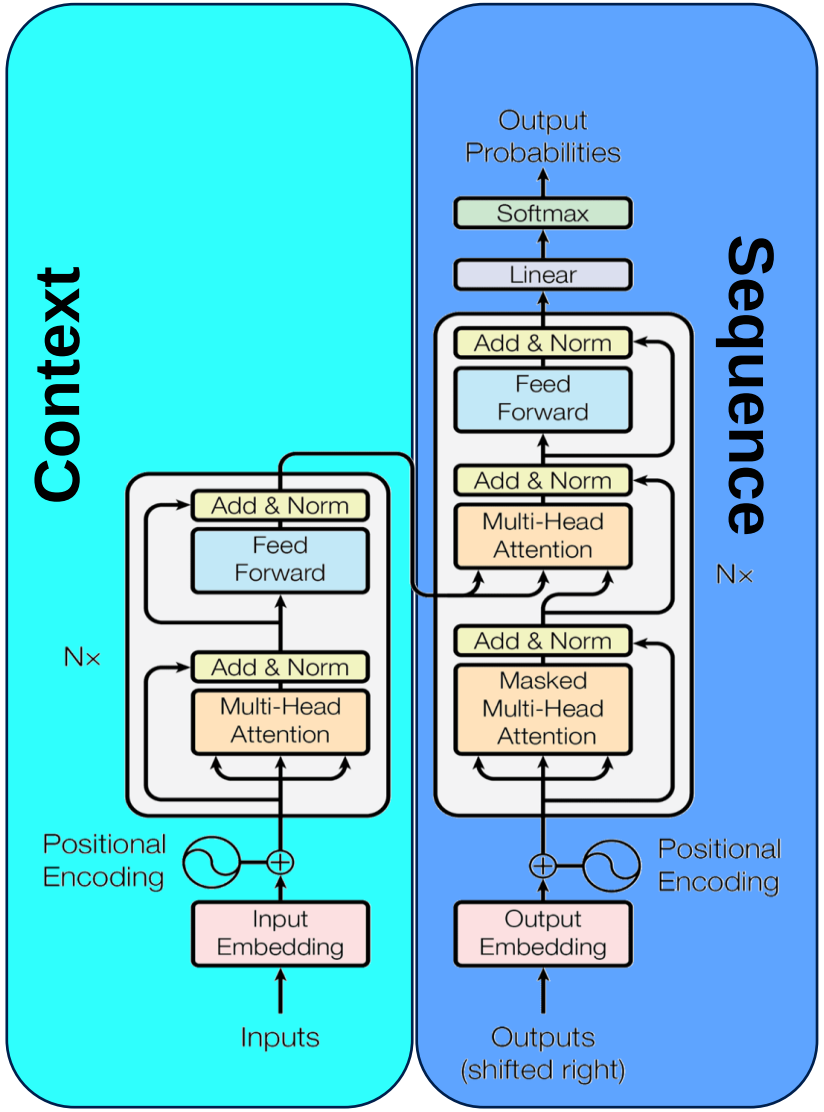
Left Side:

- Part relevant maintenance events since install
- Previous flight sensor values
- Previous flight records
- Current planned mission flight record

Right Side:

- Potential maintenance events after flight (predictive maintenance)
- Probability of failure during mission (fleet readiness)
- Sensor profile for planned flight (advanced diagnostics)

Almost anything can be turned into a sequence.
Sequence = ordered target information



Aircraft Damage Resolution Assistant (ADaRA)

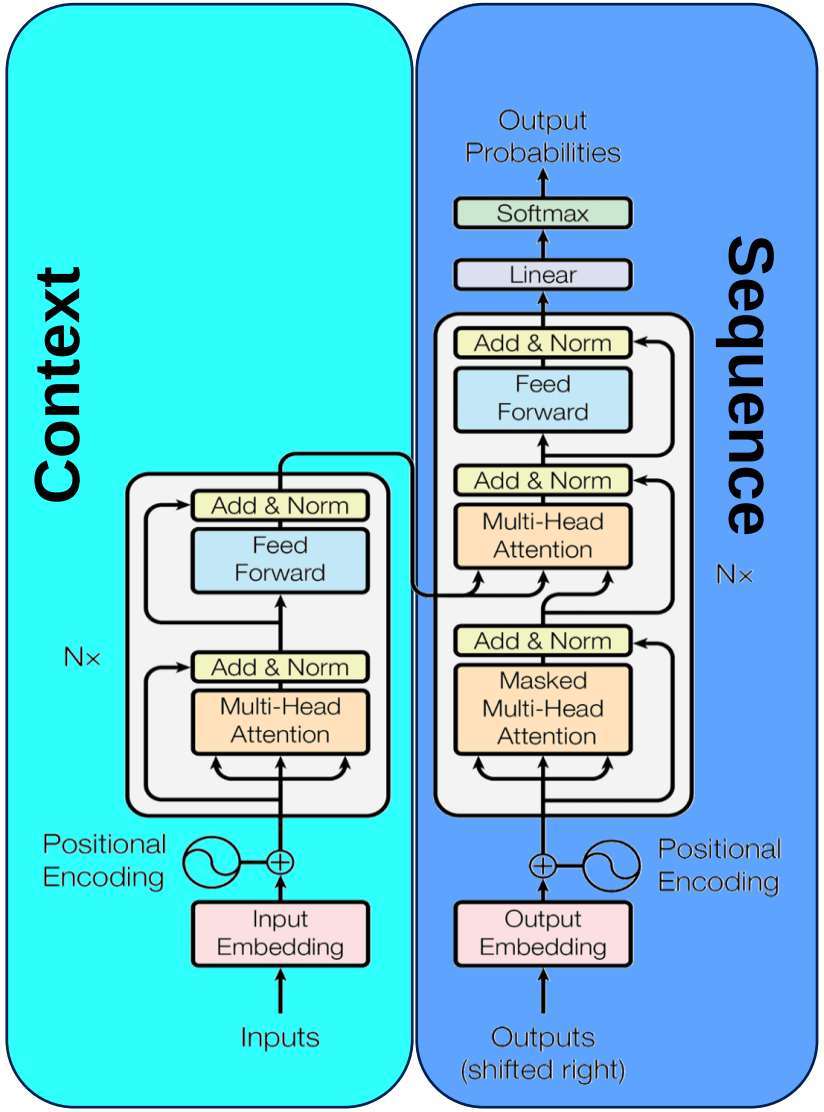
Left Side:

- Maintenance records
- Tear down analysis

Right Side:

- Images of failures
- Technical drawings

This tool is used to provide multi-modal search to maintainers. They can search for information in the TM, maintenance records, or tear downs, using text or image search. Not every fix is routine.



Sometimes the sequence isn't the goal

Sensor Rig (Behavioral Analysis)

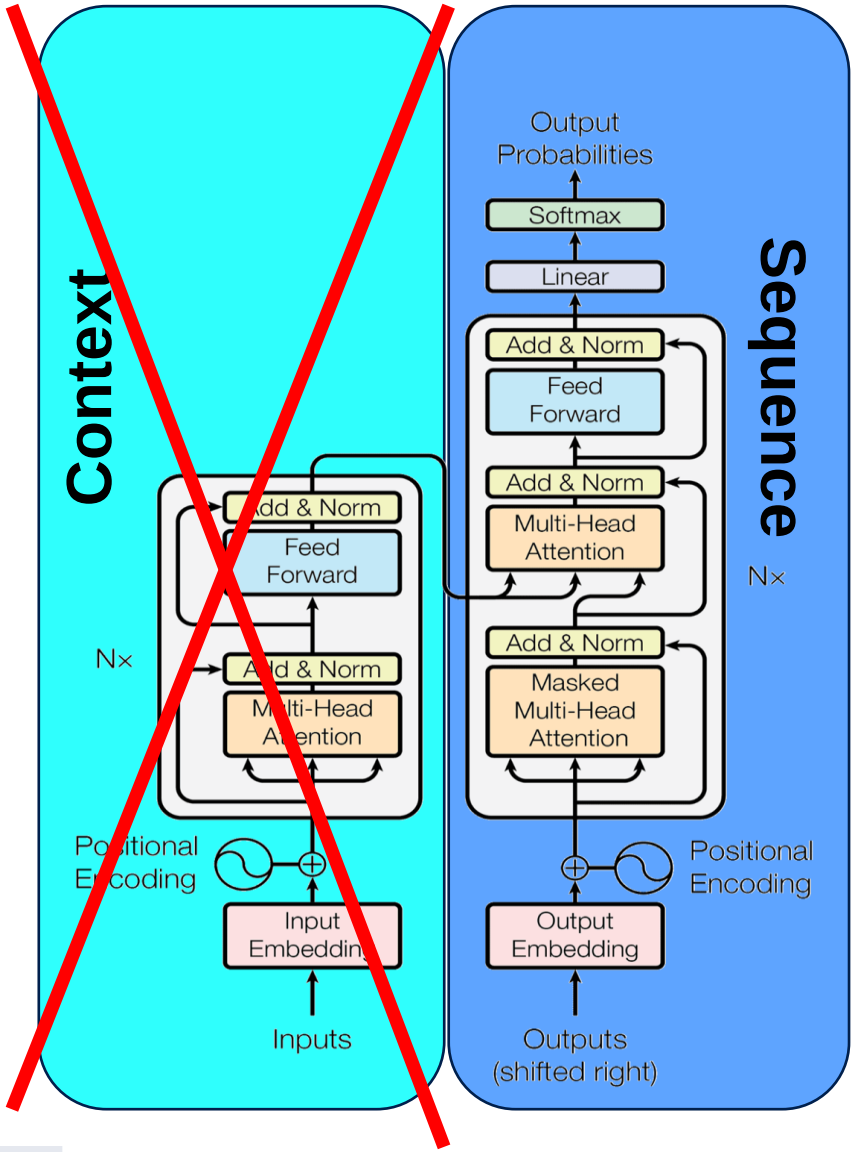
Left Side:

- Nothing

Right Side:

- Temporal sensor readings

This tool provides an analysis for the patterns of reading over time. Bucketing the patterns into groups helps to identify behaviors associated with different operating conditions and pinpoints behaviors associated with wear out or failure.



Sometimes both sides aren't needed

Experiment

Limitation of AGI pre-training

Setup

An inference attack is a method of probing a model to try to get it to generate or confirm a sample from its training data. In order to explore the limitation of general intelligence training in large AGI models we are going to attempt an inference attack and analyze the results.

Subject: DALL-E 2

Prompt: '*Mona Lisa as painted by Leonardo DaVinci*'

Target: →

Comparing results of generated samples from this prompt vs the original will give us insight into how and why things are generated the way they are.

Setup

An inference attack is a method of probing a model to try to get it to generate or confirm a sample from its training data. In order to explore the limitation of general intelligence training in large AGI models we are going to attempt an inference attack and analyze the results.

Subject: DALL-E 2

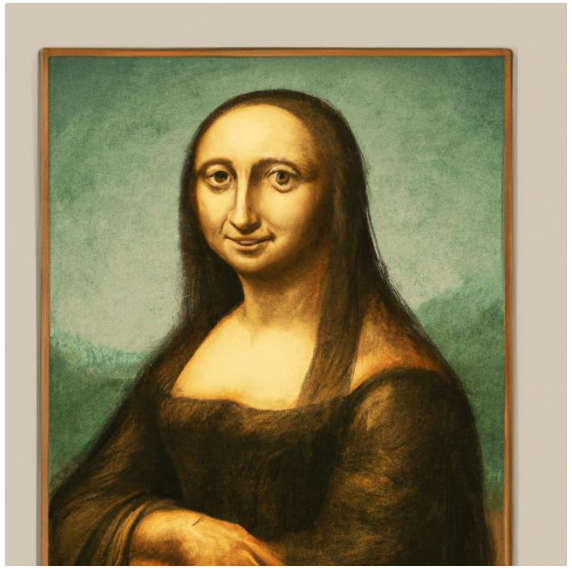
Prompt: '*Mona Lisa as painted by Leonardo DaVinci*'

Target: →

Comparing results of generated samples from this prompt vs the original will give us insight into how and why things are generated the way they are.

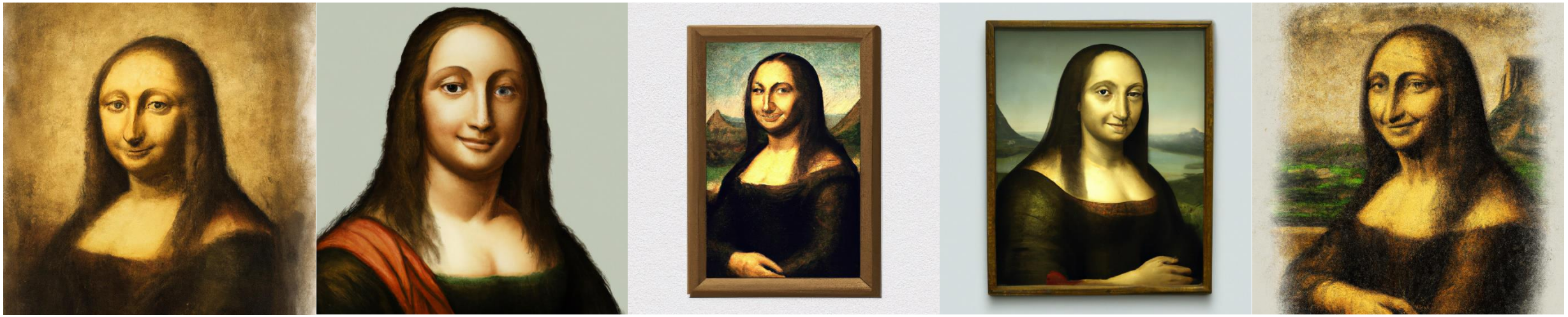
First Image

- Image is cropped differently (training data was cropped to be square for DALL-E)
- Image includes the frame
- Pose is similar
- Background color is similar but lacks detail
- Not enough detail in clothing, hairline, hair, etc.



We can infer that there is a “Mona Lisa” style that is trying to be copied

A sample of generations

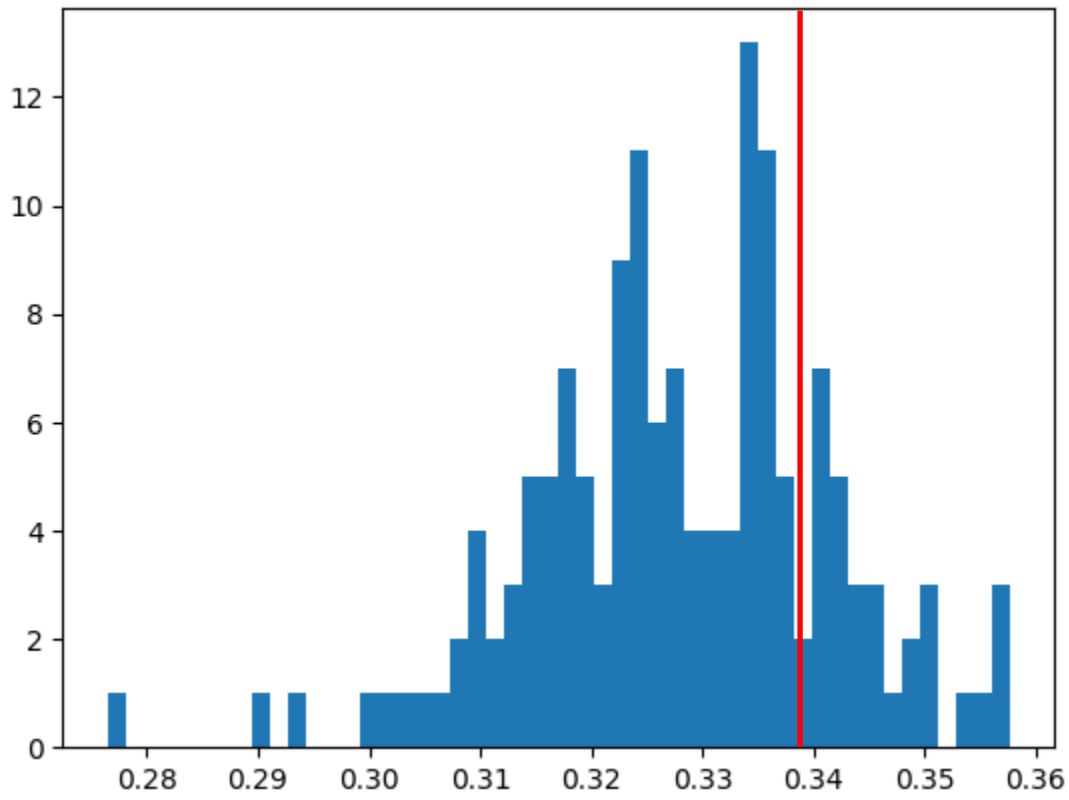


Biases guide the generation even if its following a prompt

Similarity of the image vectors vs prompt

Plot shows the normalized cosine similarity score for all 200 generated images vs the text vector of the prompt.

- Similarity is low due to major difference in information between types of data (text vs image)
- Images contain so much more information, prompts must describe in painful detail what an image contains.
- This distribution of similarity reflects a flaw in the underlying multimodal problem. Images contain more bits of information than text.



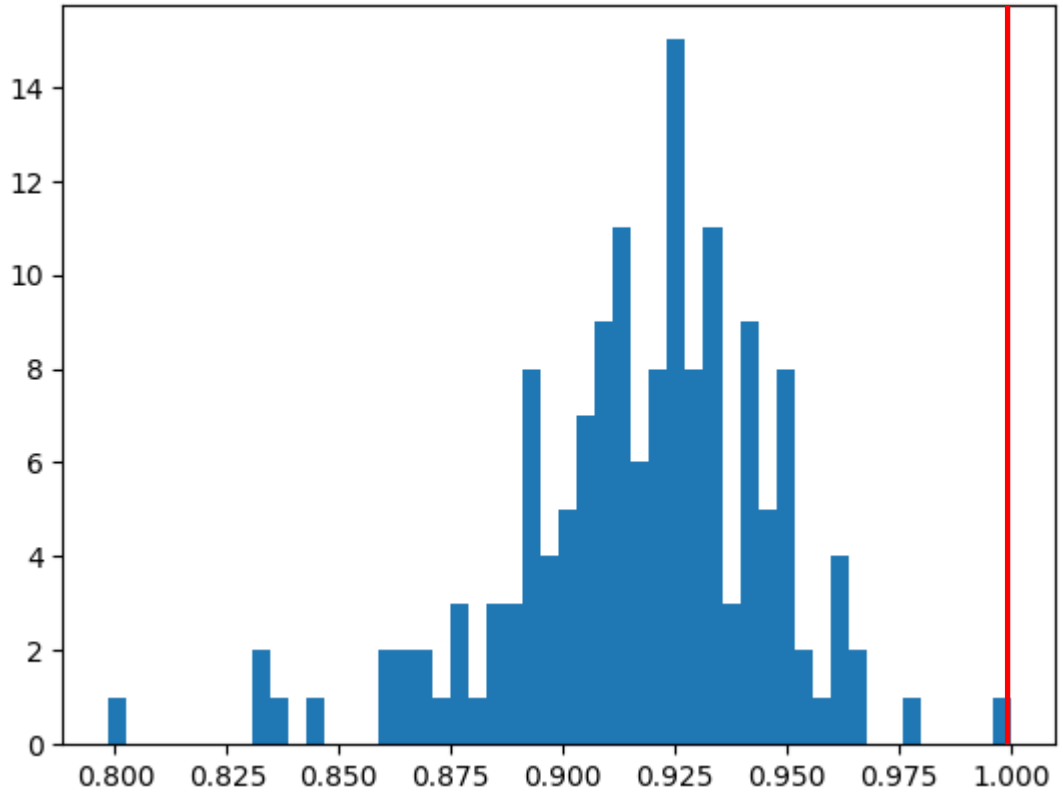
Making educated guesses leads DALL-E to use biases to inform image generation, even when its not necessary.

Similarity of the image vectors vs original image

Original Painting

Plot shows the normalized cosine similarity score for all 200 generated images vs the original vector of the prompt.

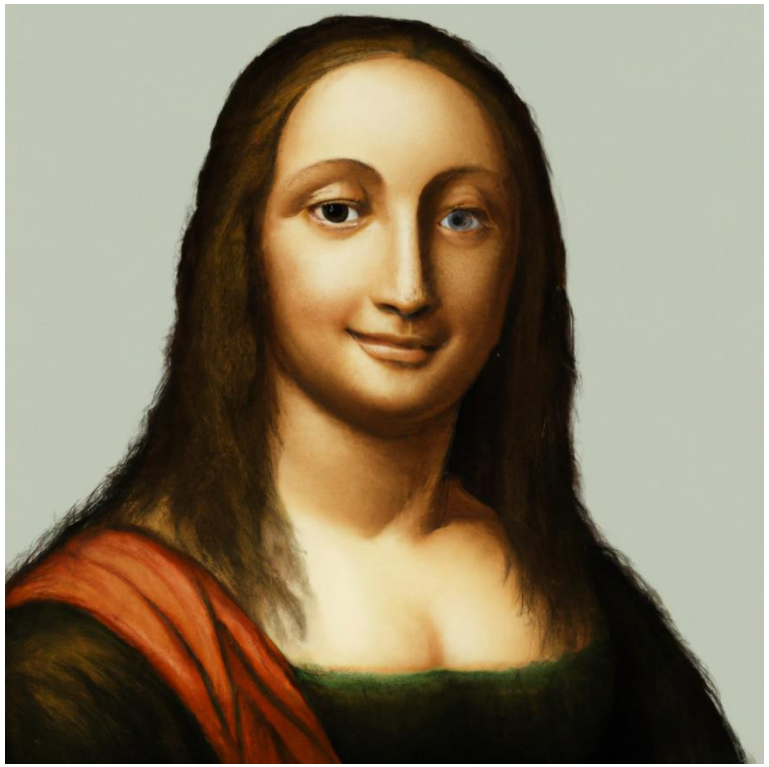
- There is a defined gap between the original image and the generated images.
- One expectation is that generated images would contain more of the original image information, with slight alterations
- The gap represents the weight that training data bias has shifted the expectation away from the original image.



Using prompts to overcome bias in generation may not even be possible with this model.

Smile Bias

- Smile bias has been shown to be present in the model (https://medium.com/@socialcreature/ai-and-the-american-smile-76d23a0fbfaf)
- Smile bias is even present in these generations.
- The smile shown in the image does not occur in any of Leonardo's work



Many other types of training data bias are unknown but are reflected in the generated output.

Gender Bias

- These images tend to have a masculine feature bias
- The masculine feature bias may explain the hair line error in the generations (which is present in all 200 generations)
- Most paintings of this type were of men

Chat GPT

Chat GPT is a bit more difficult to test and find evidence systematically.

Evidence is typically shown in some generations and is considered a “hallucination”

One Example:

When asking GPT to generate a press release about a wild fire it will blame humans 50% of the time in the generated press release. This is because humans are to blame for the majority of wild fires and this information is represented in the training data that it saw.

Its important to note that this problem of bias is not specific to GPT or DALL-E but is related to the setup of the training data, the length of the training time, the size of the model, and the influence of the RLHF.

Some biases are less noticeable but would have unpredictable impacts if deployed in an automated pipeline

So how should we use them?

LLMs are very good at doing a few tasks:

- Generating creative inspiration content (i.e. templates or starting points)

- Summarizing information in a human readable format

- Turning data into a human readable text report

Use things like LangChain to place them in data pipelines as a piece of the user experience.

Don't expect them to solve the problem, only verbalize it.

Treat them like the language center of the brain, they are a part of a larger system

Final thoughts

How do we control generative output from AGI?

How do we harness AGI and still provide specific use cases?

How do we accurately measure and predict the reliability of autonomous and AI systems?

How can we measure the reliability growth of these systems?

LOCKHEED MARTIN