

LETS LEARN LARGE LANGUAGE MODELS (LLMs)

Yogesh Haribhau Kulkarni



Outline

1 LLM INTRO

2 HOW IT WORKS?

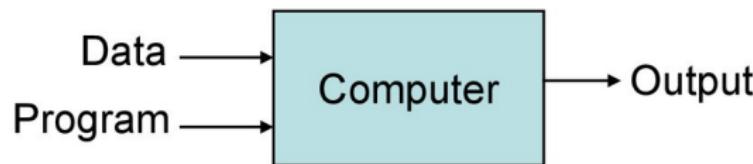
3 REFERENCES

Background and Motivation

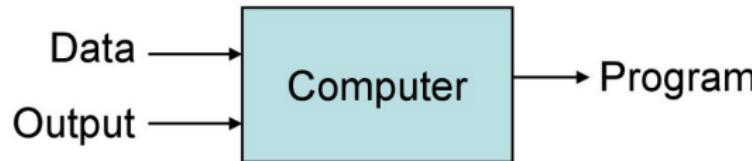
YHK

Traditional vs. Machine Learning?

Traditional Programming



Machine Learning



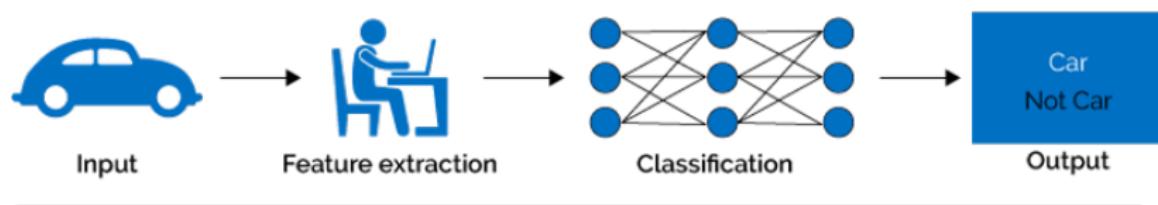
Why Machine Learning?

- ▶ Problems with High Dimensionality
- ▶ Hard/Expensive to program manually
- ▶ Job \$\$\$

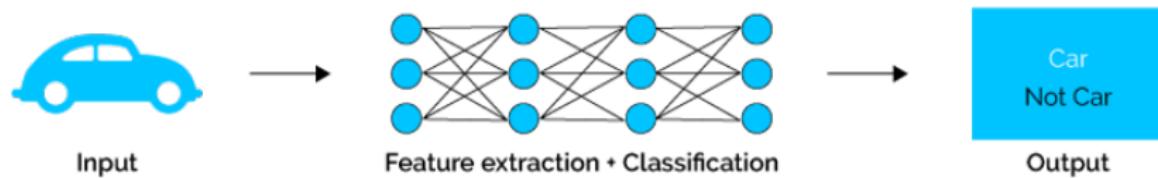
ML vs DL: What's the difference?

Deep learning algorithms attempt to learn (multiple levels of) representation by using a hierarchy of multiple layers

Machine Learning



Deep Learning



(Reference: <https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png>)

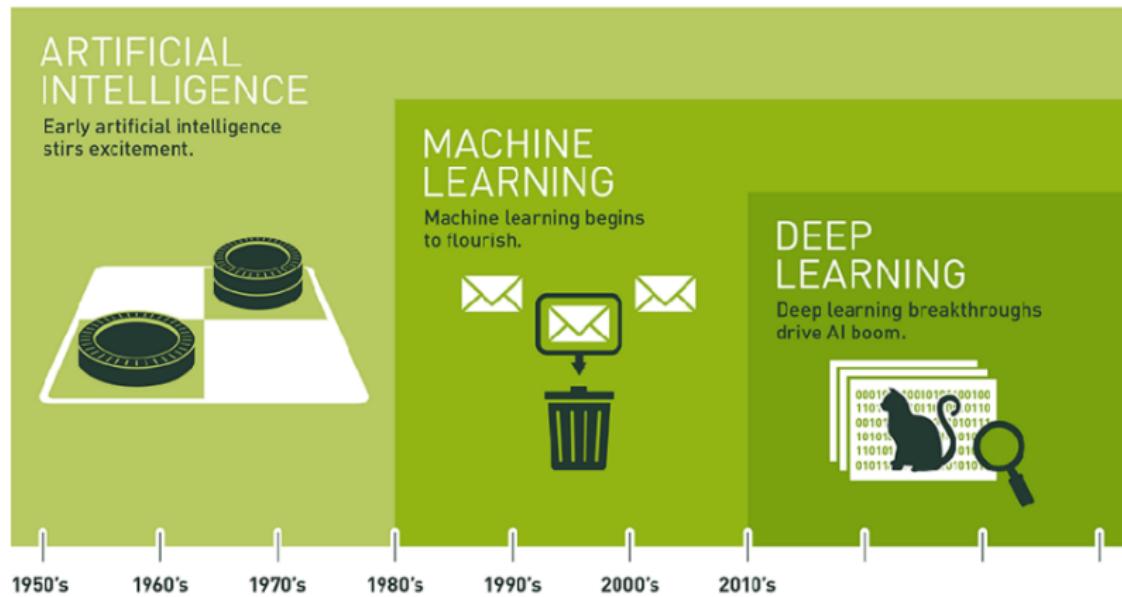
Use Deep Learning When ...

- ▶ You have lots of data (about 10k+ examples)
- ▶ The problem is “complex” - speech, vision, natural language
- ▶ The data is unstructured
- ▶ Techniques to model ‘ANY’ function given ‘ENOUGH’ data.

(Ref: Introduction to TensorFlow 2.0 - Brad Miro)



Relationship between AI, ML, DL



(Ref: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>)

From ML to NLP

YHK

Typical Machine Learning Classification

- ▶ Each text item thus gets converted to fixed size vector, thus features.
- ▶ In training, weights are computed based on the given target.
- ▶ Once model is ready, it is able to answer target, say, Yes or No to unseen text.

Hello Kirill. Checking if you are back to Oz. Let me know if you are around ... Cheers, V



Training Data:

Hey mate, have you read about Hinton's capsule networks?	→	No
Did you like that recipe I sent you last week?	→	Yes
Hi Kirill, are you coming to dinner tonight?	→	Yes
Dear Kirill, would you like to service your car with us again?	→	No
Are you coming to Australia in December?	→	Yes

(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)

Evolution of Vectorization

Vectors can be statistical (frequency based) or Machine/Deep Learning (supervised) based. Simple to complex.



(Ref: Analytics Vidhya <https://editor.analyticsvidhya.com/uploads/59483evolution.of.NLP.png>)

How to Vectorize? Representing words by their context



- Distributional semantics: A word's meaning is given by the words that frequently appear close-by
 - “*You shall know a word by the company it keeps*” (J. R. Firth 1957)
 - One of the most successful ideas of modern statistical NLP!
- When a word w appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).
- Use the many contexts of w to build up a representation of w

...government debt problems turning into **banking** crises as happened in 2009...

...saying that Europe needs unified **banking** regulation to replace the hodgepodge...

...India has just given its **banking** system a shot in the arm...



These **context words** will represent **banking**

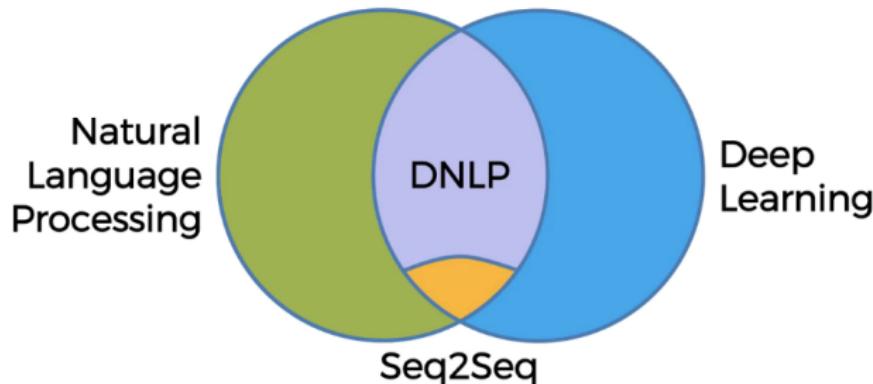
(Ref: CS224n: Natural Language Processing with Deep Learning - Christopher Manning)

Word vectors

- ▶ Dense vector for each word
- ▶ Called distributed representation, word embeddings or word representations
- ▶ Test: similar to vectors of words that appear in similar contexts

$$\text{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

What is Deep NLP



(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)
(Note: Size is not indicative of importance)

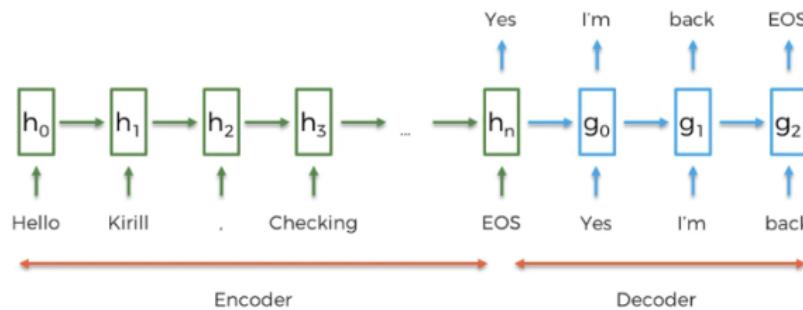
Seq2Seq is heavily used technique of DNLP for sequence to sequence modeling,
eg Translation, Q & A, etc. Thats the basis of Large Language Models (LLMs)

Transformer Architecture

YHK

Seq2Seq architecture

Hello Kirill, Checking if you are back to Oz.

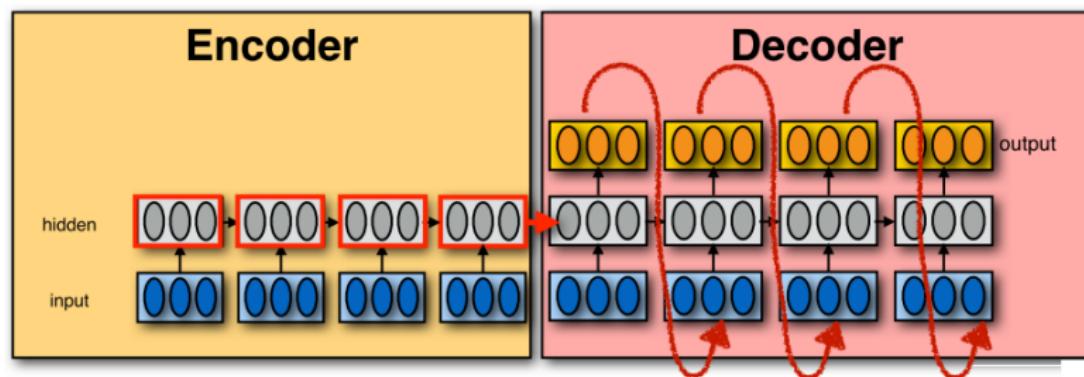


(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)

During training, Encoder is fed with Questions and decoder with Answers. Weights in gates, hidden states get settled. During testing for each sequence of input, encoder results in to a combo vector. Decoder takes this and starts spitting out words one by one, probabilistically.

Encoder-Decoder (seq2seq) model

- ▶ The decoder is a language model that generates an output sequence conditioned on the input sequence.
 - ▶ Vanilla RNN: condition on the last hidden state
 - ▶ Attention: condition on all hidden states



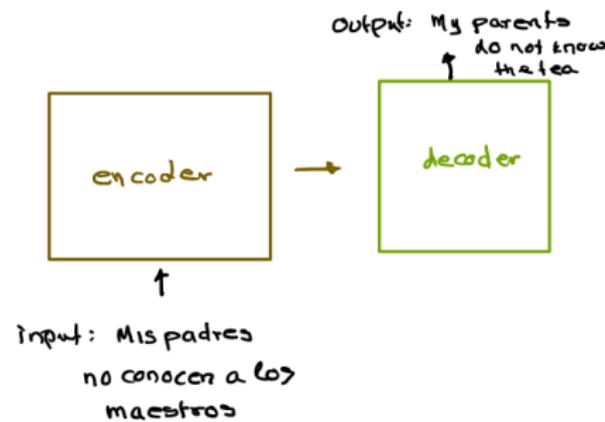
(Ref: CS447 Natural Language Processing (J. Hockenmaier)

Transformers use Self-Attention

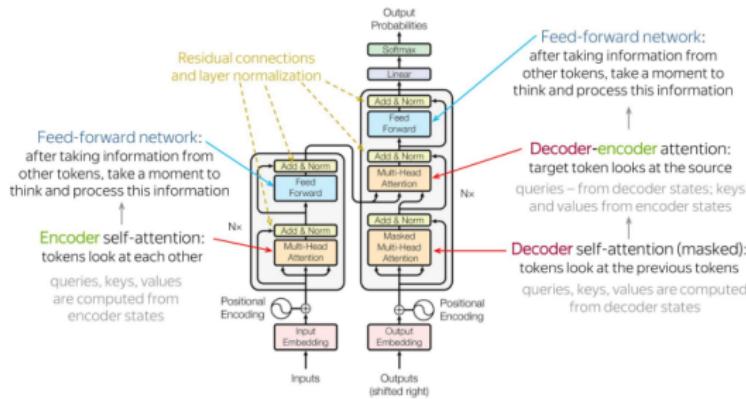
- ▶ Attention so far (in seq2seq architectures): In the decoder (which has access to the complete input sequence), compute attention weights over encoder positions that depend on each decoder position
- ▶ Self-attention: If the encoder has access to the complete input sequence, we can also compute attention weights over encoder positions that depend on each encoder position

Transformers

- ▶ In its heart it contains an encoding component, a decoding component, and connections between them.
- ▶ The Transformer is a model that uses attention to boost the speed with which seq2seq with attention models can be trained.
- ▶ The biggest benefit, however, comes from how The Transformer lends itself to parallelization. How?



Transformer Models



- ▶ No recurrence, so parallelization possible
- ▶ Context information captured via attention and positional encodings
- ▶ Consists of stacks of layers with various sublayers

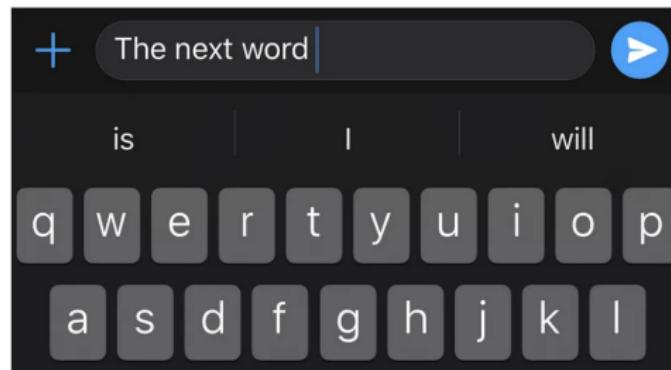
Transformers are basis of (the most) Large Language Models

Large Language Models

YHK

What is a Language Models?

- ▶ While typing SMS, have you seen it suggests next word?
- ▶ While typing email, have you seen next few words are suggested?
- ▶ How does it suggest? (suggestions are not random, right?)
- ▶ In the past, for "Lets go for a ... ", if you have typed 'coffee' 15 times, 'movie' say 4 times, then it learns that. Machine/Statistical Learning.
- ▶ Next time, when you type "Lets go for a ", what will be suggested? why?
- ▶ This is called Language Model. Predicting the next word. When done continuously, one after other, it spits sentence, called Generative Model.



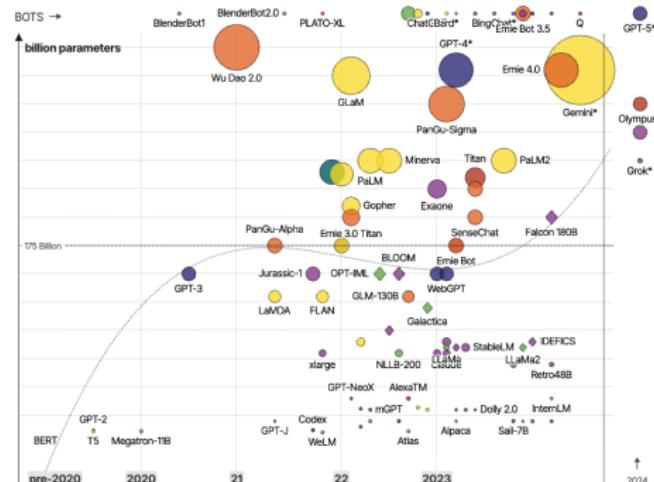
Next word prediction using language modeling in keyboards(Mandar Deshpande)

YHK

LLM - Information is beautiful

The Rise and Rise of A.I. Large Language Models (LLMs) & their associated bots like ChatGPT

● Amazon-owned ● Chinese ● Google ● Meta / Facebook ● Microsoft ● OpenAI ● Other



David McCandless, Tom Evans, Paul Barton
Information is Beautiful // UPDATED 6th Dec 23

source: news reports, LifeArchitect
* = parameters undisclosed // see [the data](#)

(Ref: <https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-langs-like-chatgpt/>)

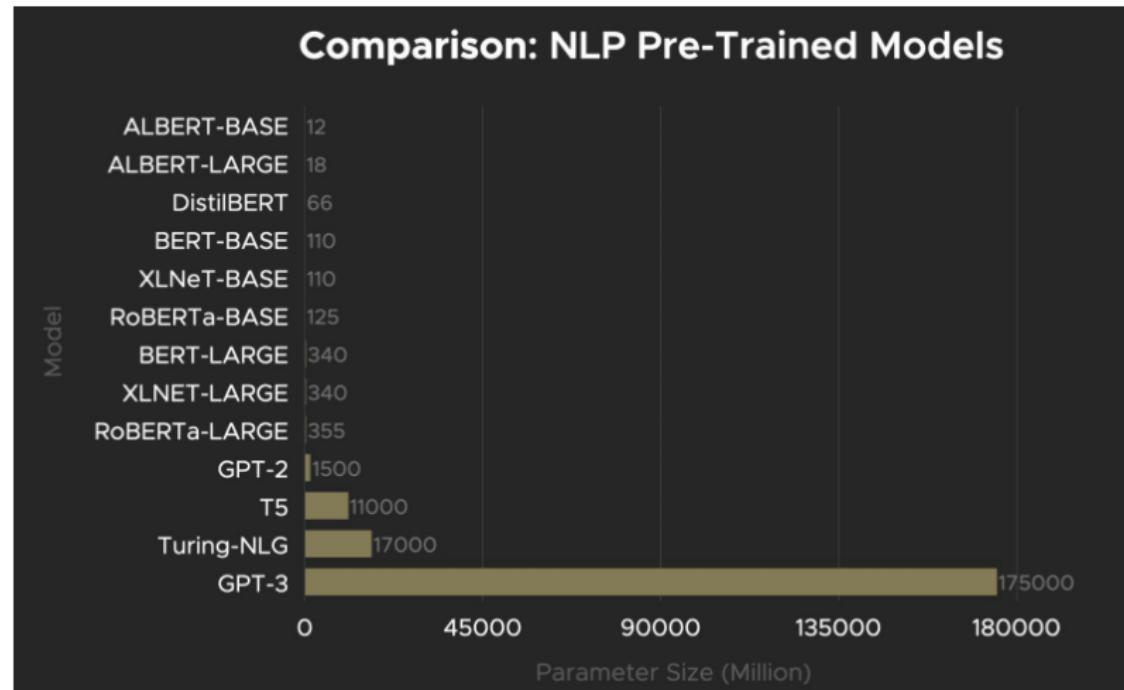
How LLMs work?

- ▶ **Transformer-Based Architecture:**
 - ▶ Utilizes the Transformer architecture for processing input sequences.
 - ▶ Self-attention mechanism captures long-range dependencies.
- ▶ **Pre-training:**
 - ▶ Trained on a massive corpus of text data in an unsupervised manner.
 - ▶ Learns contextualized representations of words and phrases.
- ▶ **Generative Capabilities:**
 - ▶ Can generate coherent and contextually relevant text.
 - ▶ Useful for a wide range of natural language understanding and generation tasks.
- ▶ **Fine-tuning (Optional):**
 - ▶ Model can be fine-tuned on specific downstream tasks.
 - ▶ Adaptation to user or domain-specific requirements.

Transformer Architecture

- ▶ **Input Representation:** Embedding layer converts input tokens into high-dimensional vectors.
- ▶ **Positional Encoding:** Adds positional information to the input embeddings.
- ▶ **Multi-Head Self Attention:**
 - ▶ Allows each token to focus on different parts of the input sequence.
 - ▶ Multiple attention heads capture diverse patterns.
- ▶ **Layer Normalization & Residual/Skip Connections:**
 - ▶ Stabilizes training using layer normalization.
 - ▶ Residual connections help in mitigating vanishing/exploding gradient problems.
- ▶ **Encoder-Decoder Structure (for Sequence-to-Sequence tasks):** In tasks like translation, multiple encoder layers process the input, and then multiple decoder layers generate the output.
- ▶ **Output Layer:** Produces the final output sequence.

Large Language Models - Comparison



(Ref: Deus.ai <https://www.deus.ai/post/gpt-3-what-is-all-the-excitement-about>)

LLM Comparison (2024-2026)

Model	Parameters	Context	Multimodal	Open Source
GPT-4 Turbo	Unknown	128K	Yes	No
Claude 3.5	Unknown	200K	Yes	No
Gemini 1.5 Pro	Unknown	2M	Yes	No
Llama 3.1 405B	405B	128K	No	Yes
Mixtral 8x22B	141B	64K	No	Yes

LLM Training

- ▶ Training LLMs involves instructing the model to comprehend and generate human-like text.
- ▶ **Input Text:** LLMs are exposed to extensive text data from diverse sources like books, articles, and websites.
- ▶ During training, the model predicts the next word/token based on context, learning patterns and relationships.
- ▶ **Optimizing Weights:** The model has weights for parameters reflecting feature significance.
- ▶ Throughout training, weights are fine-tuned to minimize error, enhancing the model's prediction accuracy.
- ▶ After initial training, LLMs can be customized for tasks using small sets of supervised data. This process is known as **fine-tuning**.
- ▶ **Fine-tuning Parameters:** LLMs adjust parameter values based on error feedback during predictions.
- ▶ The model refines its language understanding by iteratively adjusting parameters, improving token prediction accuracy.
- ▶ Training may vary for specific LLM types, like those optimized for continuous text or dialogue.



LLM Performance Factors

- ▶ **Model Architecture:** LLM performance is influenced by the design and intricacy of its architecture.
- ▶ **Dataset Quality:** The quality and diversity of the training dataset shape the model's language understanding.
- ▶ Training a private LLM demands substantial computational resources and expertise.
- ▶ Duration ranges from days to weeks, contingent on model complexity and dataset size.
- ▶ Cloud-based solutions and high-performance GPUs expedite the training process.
- ▶ LLM training is meticulous and resource-intensive, forming the basis for language comprehension and generation.

Decoder-Only Transformers (e.g., GPT)

- ▶ **Architecture:**

- ▶ GPT uses a transformer architecture consisting solely of decoder layers.
 - ▶ Decoders attend to the entire input sequence during training and generation.

- ▶ **Positional Embeddings:** Incorporates position information to handle sequence order.

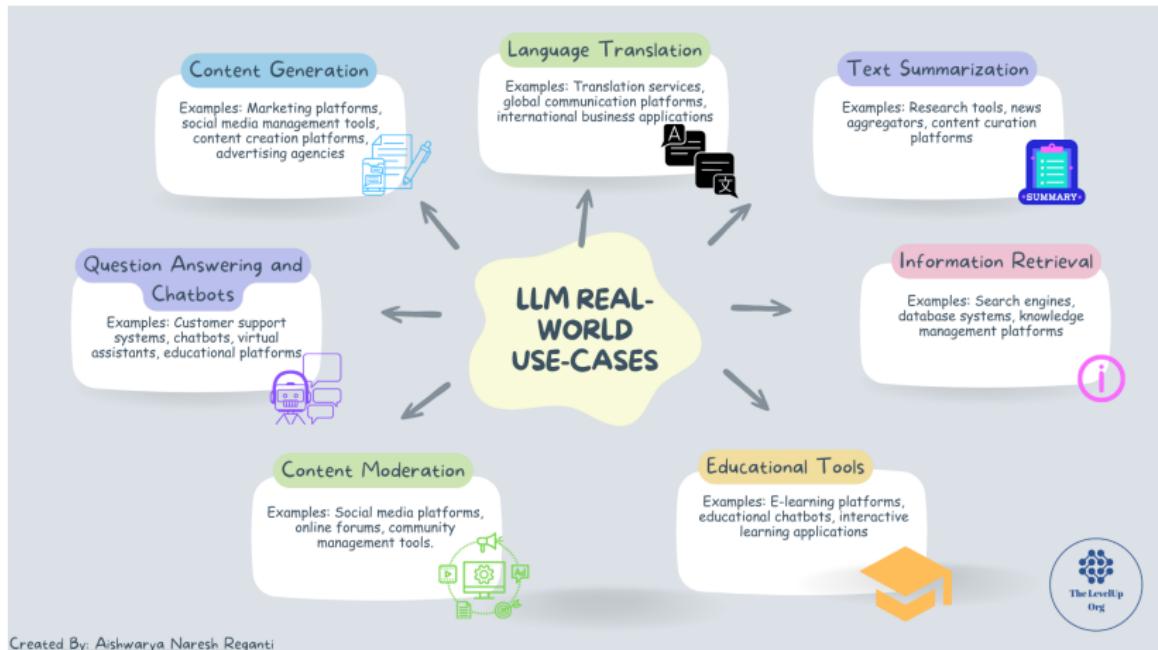
- ▶ **Self-Attention Mechanism with Masking:**

- ▶ Each position attends to all positions in the preceding context.
 - ▶ During training, masking ensures that positions after the current one are not considered.
 - ▶ Prevents the model from peeking at future tokens during generation.

- ▶ **Autoregressive Generation:**

- ▶ Generates output tokens one at a time in an autoregressive manner.
 - ▶ Previous tokens influence the generation of subsequent tokens.

LLM Real World Use Cases



Created By: Aishwarya Naresh Reganti

(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

LLM Real World Use Cases

No.	Use case	Description
1	Content Generation	Craft human-like text, videos, code and images when provided with instructions
2	Language Translation	Translate languages from one to another
3	Text Summarization	Summarize lengthy texts, simplifying comprehension by highlighting key points.
4	Question Answering and Chatbots	LLMs can provide relevant answers to queries, leveraging their vast knowledge
5	Content Moderation	Assist in content moderation by identifying and filtering inappropriate or harmful language
6	Information Retrieval	Retrieve relevant information from large datasets or documents.
7	Educational Tools	Tutor, provide explanations, and generate learning materials.

(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

LLM Challenges



Created by: Aishwarya Naresh Reganti

(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

ChatGPT — GPT3.5/GPT4

The screenshot shows the OpenAI Playground interface. At the top, there's a header with 'Playground' and a search bar containing 'IMD - Long Descriptions'. Below the header are several configuration options:

- Model:** set to 'text-davinci-003'
- Temperature:** set to 0.7
- Maximum length:** set to 1790
- Stop sequences:** 'None' (disabled)
- Top P:** set to 1
- Frequency penalty:** set to 0.2
- Presence penalty:** set to 0.65
- Best of:** set to 12
- Request start text:** contains a placeholder 'None' with a browse button.
- Request restart text:** contains a placeholder 'None' with a browse button.

At the bottom left, there's a message 'Looking for ChatGPT? Try it now! ⓘ'.

Created by OpenAI Access it
with code or without
(Playground
<https://platform.openai.com/playground>)

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Bard — Palm 2/Gemini

 Product description: A home milkshake maker
Seed words: fast, healthy, compact
Product names: HomeShaker, Fit Shaker, QuickShake, Shake Maker

Product description: A pair of shoes that can fit any foot size.
Seed words: adaptable, fit, omni-fit
Product names:

 Here are some product names for a pair of shoes that can fit any foot size:

- Omni-Fit Shoes
- Adaptable Shoes
- Fit-All Shoes
- Size-Free Shoes
- Universal Shoes

Created by Google Access it via chat <https://bard.google.com/> or encounter it in search results

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Meta LLaMA

- ▶ Open-Source. Need to build a UX and any advanced functionality around it, and may need to fine-tune it.
- ▶ Many use-cases in the enterprise can't use OpenAI for fear of sensitive data leaking or being used to train the model (though OpenAI claims to keep API data private).
- ▶ If you have 200+ examples fine-tuning beats prompt engineering for a specific defined task.

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))



Anthropic Claude

The screenshot shows a user interface for generating product names. At the top, there's a header with a square icon, three dots, and a search bar containing "Fit Shaker: Fast and Healthy". Below the search bar are two buttons: "Chat" and "Edit Chat". A light blue sidebar on the right contains the following text:
Product names: HomeShaker, Fit Shaker, QuickShake, Shake Maker
Product description: A pair of shoes that can fit any foot size.
Seed words: adaptable, fit, omni-fit
Product names: M

Below the sidebar, a gray box displays a list of suggested product names based on seed words:

- OmniFit
- AdaptiShoe
- UniSole
- One Size Fits All
- FlexiFit

At the bottom left is a text input field with placeholder text "Write a message..." and a blue send button with a white arrow icon. The entire interface has a clean, modern design with a white background and light blue accents.

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Created by Anthropic
<https://console.anthropic.com/>
or API Uses Constitutional AI rather than RLHF
Constitutional AI trains to follow a set of high-level principles or rules, such as a constitution, that specify the desired behavior and outcomes of the system. RLHF uses human feedback, such as ratings, preferences, or corrections, to optimize a language model or an agent's policy using reinforcement learning

Microsoft Bing — GPT 4

The screenshot shows a Microsoft Bing AI interface. At the top, there's a blue button labeled "Help me plan a trip to London". Below it, a green checkmark icon and the text "Generating answers for you..." are displayed. The main text area starts with "Sure, I can help you plan a trip to London. London is a great city to visit, with iconic attractions, historic museums, and vibrant neighborhoods. You can find things to do in London, from sightseeing tours to fun-filled days out, on visitlondon.com ¹. You can also book a London Pass to make the most of popular attractions ² or a one-day hop-on hop-off bus tour to explore the sights at your own pace ³. You can find cheap flights to London from \$99 on KAYAK ⁴ and London vacation packages from \$849 on Expedia ⁵.". At the bottom of this text block, there's a "Learn more:" section with links to visitlondon.com and "+4 more". At the very bottom of the interface, there's a "Ask me anything" input field and a microphone icon.

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Powered by OpenAI's GPT-4
<https://www.microsoft.com/en-gb/bing>

Falcon

Downloads last month
158,979



⚡ Hosted inference API ⓘ

⌚ Text Generation

Example 8 ▾

Once upon a time, there was a little girl named Alice. She was a curious little girl, who was always asking questions|

Compute

⌘+Enter

1.5

Computation time on gpu: cached

↔ JSON Output

Maximize

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Access it via HuggingFace
transformers library, 7B and
40B models as well as instruct
fine-tuned
Features:

- ▶ Free for commercial use
- ▶ Open source
- ▶ Possible to fine-tune

Leader board (Jan 2024)

Leaders by category

The leaderboard ranks LLMs across multiple prompt categories.

RANK ↑	LLM	TOTAL	BRAINSTORMING	CLOSED QA	GENERATION	OPEN QA	REWRITE
1	GPT-4	81.75	89.74	78.95	81.87	74.10	92.31
2	WizardLM 13B V1.2	79.56	76.92	73.68	80.31	77.11	92.31
3	LLaMA 2 70B Chat	78.97	88.46	68.42	81.35	69.88	84.62
4	GPT-3.5 Turbo	76.79	73.08	68.42	80.31	73.49	76.92
5	Vicuna 33B V1.3	74.21	82.05	47.37	71.50	70.48	76.92
6	Guanaco 13B	50.00	50.00	50.00	50.00	50.00	50.00

(Ref: <https://toloka.ai/llm-leaderboard/>)

State of LLMs in 2024-2026

- ▶ GPT-4o, GPT-4 Turbo (Multimodal capabilities)
- ▶ Claude 3.5 Sonnet, Opus (Extended context windows)
- ▶ Gemini 1.5 Pro, Ultra (2M token context)
- ▶ Llama 3, 3.1, 3.2 (Open-source advancement)
- ▶ Mixtral 8x22B (Mixture of Experts)
- ▶ Command R+, Phi-3 (Efficiency focus)

(Ref: Model releases 2024-2026)



Beyond Standard Transformers

- ▶ Mixture of Experts (MoE): Mixtral, GPT-4 architecture
- ▶ Long Context Models: Techniques for 100K+ token windows
- ▶ Sparse Attention Mechanisms: FlashAttention, Linear Attention
- ▶ State Space Models: Mamba, alternatives to attention



Advanced Fine-tuning Methods

Parameter-Efficient Methods:

- ▶ LoRA (Low-Rank Adaptation)
- ▶ QLoRA (Quantized LoRA)
- ▶ Prefix Tuning
- ▶ Adapter Layers

Benefits:

- ▶ Reduced memory requirements
- ▶ Faster training times
- ▶ Multiple task-specific models
- ▶ Cost-effective deployment

Conclusions, Cautions and What's Next?

So, What are LLMs?

Large Language Models (LLMs) have revolutionized natural language processing, ushering in advancements in text generation and understanding. Key attributes include:

- ▶ **Learning from Extensive Data:** LLMs acquire knowledge from vast datasets, resembling a massive library of information.
- ▶ **Grasping Context and Entities:** These models understand context and entities, allowing for a deeper comprehension of language.
- ▶ **Proficient User Query Responses:** LLMs excel in responding to user queries, showcasing their ability to apply learned knowledge effectively.

Despite their versatile applications across industries, ethical concerns and potential biases necessitate a critical evaluation to understand their societal impact.

LLM Safety Challenges

- ▶ Jailbreaking & Prompt Injection
- ▶ Data Poisoning Attacks
- ▶ Model Extraction
- ▶ Misinformation Generation
- ▶ Bias Amplification
- ▶ Privacy Concerns (memorization)

Mitigation Strategies:

- ▶ Red teaming
- ▶ Adversarial training
- ▶ Constitutional AI
- ▶ Watermarking



Core Beliefs of Large Language Models

- ▶ No inherent "core beliefs."
- ▶ Word guessers predicting internet-like sentences.
- ▶ Can write both for and against a topic without belief.
- ▶ Emulates the most common response in training data.

Truth and Morality in Large Language Models

- ▶ Lack sense of truth or morality.
- ▶ Tendency to generate words we agree are true.
- ▶ No guarantee of providing the actual truth.

Mistakes in Large Language Models

- ▶ Prone to mistakes due to inconsistent training data.
- ▶ Self-attention may not capture all relevant information.
- ▶ Hallucination: generating words not derived from input.
- ▶ Preference for common words, small numbers, and specific names.

Auto-regressive Nature of LLMs

- ▶ Auto-regressive models: guesses affect subsequent inputs.
- ▶ Errors accumulate, potentially compounding mistakes.
- ▶ No mechanism to "change minds" or self-correct.
- ▶ Lack the ability to retry or undo prior choices.

Verification of Outputs

- ▶ Always verify outputs of large language models.
- ▶ Assess competence to verify results in high-stakes tasks.
- ▶ Mistakes in critical tasks may lead to costly decisions.

Input Size and Memory Limitations

- ▶ Large language models have input size limits.
- ▶ Conversation appears coherent until log size exceeds limit.
- ▶ Earlier parts of the conversation are deleted, and the model "forgets."

Ethical Considerations

- ▶ Awareness of potential biases in LLMs is crucial for responsible usage.
- ▶ Continuous evaluation of ethical implications is necessary to mitigate societal risks.
- ▶ Balancing the benefits of LLMs with ethical concerns ensures responsible deployment.

Future Impact

- ▶ LLMs expected to revolutionize domains such as job markets, communication, and society.
- ▶ Careful use and ongoing development are essential for positive impacts.
- ▶ Understanding limitations and ethical considerations is vital for responsible integration into various domains.

Emerging Trends

- ▶ Small Language Models (SLMs): Phi, Gemma
- ▶ On-device LLMs: Mobile deployment
- ▶ Continual Learning: Models that update knowledge
- ▶ Reasoning Models: Chain-of-thought, o1-preview
- ▶ Agentic AI: Autonomous task completion
- ▶ Personalization: User-specific fine-tuning

Landscape of LLMs & Quiz

- ▶ Types of models - Foundation models, LLM, SLM, VLMs, etc.
- ▶ Common LLM terms - Prompts, Temperature, Hallucinations, Tokens, etc.
- ▶ LLM lifecycle stages - Pre-training, Supervised Fine Tuning, RLHF, etc.
- ▶ LLM evaluations - ROUGE, BLEU, BIG-bench, GLUE, etc.
- ▶ LLM architecture - Encoder, Decoder, Transformer, Attention, etc.
- ▶ Retrieval augmented generation - Vector DBs, Chunking, Evaluations, etc.
- ▶ LLM agents - Memory, Planning, ReAct, CoT, ToT, etc.
- ▶ Cost efficiency - GPU, PEFT, LoRA, Quantization, etc.
- ▶ LLM security - Prompt Injection, Data poisoning, etc.
- ▶ Deployment & inference - Pruning, Distillation, Flash Attention, etc.
- ▶ Platforms supporting LLMOps

(Ref: LinkedIn post by Abhinav Kimothi - 23 Jan 2024)



Next Steps in Your LLM Journey

Essential Skills:

- ▶ Prompt engineering
- ▶ Fine-tuning basics
- ▶ RAG implementation
- ▶ Model evaluation

Tools to Master:

- ▶ Hugging Face ecosystem
- ▶ LangChain/LlamaIndex
- ▶ Vector databases
- ▶ API platforms

Learning Resources:

- ▶ Papers to read
- ▶ Online courses
- ▶ GitHub repositories
- ▶ Communities to join

Practice Projects:

- ▶ Build a chatbot
- ▶ Create RAG system
- ▶ Fine-tune a model
- ▶ Deploy an API

Language Modeling Fundamentals

YHK

What is Language Model

- ▶ what ChatGPT is always fundamentally trying to do is to produce a "reasonable continuation" of whatever text it's got so far
- ▶ Language model : predicting the next word
- ▶ SO, for the text "The best thing about AI is its ability to", what could be the next best word?
- ▶ it doesn't look at literal text; it looks for things that in a certain sense "match in meaning". But the end result is that it produces a ranked list of words that might follow, together with "probabilities":
- ▶ And doing this over and over again "given the text so far, what should the next word be?"—and each time adding a word.

The best thing about AI is its ability to

learn	4.5%
predict	3.5%
make	3.2%
understand	3.1%
do	2.9%

Best or lesser than Best?

- ▶ "highest-ranked" word (i.e. highest "probability") seems correct, right?
- ▶ Ok for more deterministic, scientific writings (a very "flat" essay) but for more fluid, creative writing less than perfect is actually better.
- ▶ "temperature" parameter that determines how often lower-ranked words will be used, 0.8 seems best

Probability of the next word

- ▶ Simple idea would be, for the current word, find the 'most occurring' next word from the training corpus, ie whole internet or all books or can be small as past SMS in your mobile.
- ▶ So, that's basically a huge search and aggregations to compute probabilities for all the words in the rest of the vocabulary.

```
In[1]:= model["The best thing about AI is its ability to", {"TopProbabilities", 5}]
```

```
Out[1]= { do → 0.0288508, understand → 0.0307805,  
make → 0.0319072, predict → 0.0349748, learn → 0.0445305 }  
In[2]:= NestList[StringJoin[#, model[#, "Decision"]]] &,  
"The best thing about AI is its ability to", 7]
```

```
Out[2]= { The best thing about AI is its ability to,  
The best thing about AI is its ability to learn,  
The best thing about AI is its ability to learn from,  
The best thing about AI is its ability to learn from experience,  
The best thing about AI is its ability to learn from experience.,  
The best thing about AI is its ability to learn from experience. It,  
The best thing about AI is its ability to learn from experience. It's,  
The best thing about AI is its ability to learn from experience. It's not }
```



Problem

What happens if one goes on longer? In this ("zero temperature") case what comes out soon gets rather confused and repetitive:

The best thing about AI is its ability to learn from experience. It's not just a matter of learning from experience, it's learning from the world around you. The AI is a very good example of this. It's a very good example of how to use AI to improve your life. It's a very good example of how to use AI to improve your life. The AI is a very good example of how to use AI to improve your life. It's a very good example of how to use AI to

with the "randomness" corresponding to "temperature" 0.8? And every time one does this, different random choices will be made, and the text will be different—as in these 5 examples:

The best thing about AI is its ability to learn. I've always liked the

The best thing about AI is its ability to really come into your world and just

The best thing about AI is its ability to examine human behavior and the way it

The best thing about AI is its ability to do a great job of teaching us

The best thing about AI is its ability to create real tasks, but you can

So what happens if one goes on longer? Here's a random example. It's better than the top-word (zero temperature) case, but still at best a bit weird



A simpler example

- ▶ Instead of predicting the next word, predict the next characters.
- ▶ So 26 chars. Plus space punctuation, numbers, etc, say 50 options.
- ▶ so given the current char, what's the next best char?

```
In[1]:= LetterCounts[WikipediaData["cats"]]
```

```
Out[1]= {e \rightarrow 4279, a \rightarrow 3442, t \rightarrow 3397, i \rightarrow 2739, s \rightarrow 2615, n \rightarrow 2464, o \rightarrow 2426,  
r \rightarrow 2147, h \rightarrow 1613, l \rightarrow 1552, c \rightarrow 1405, d \rightarrow 1331, m \rightarrow 989, u \rightarrow 916,  
f \rightarrow 760, g \rightarrow 745, p \rightarrow 651, y \rightarrow 591, b \rightarrow 511, w \rightarrow 509, v \rightarrow 395, k \rightarrow 212,  
T \rightarrow 114, x \rightarrow 85, A \rightarrow 81, C \rightarrow 81, I \rightarrow 68, S \rightarrow 55, F \rightarrow 42, Z \rightarrow 38, E \rightarrow 36}
```

```
In[2]:= LetterCounts[WikipediaData["dogs"]]
```

```
Out[2]= {e \rightarrow 3911, a \rightarrow 2741, o \rightarrow 2608, i \rightarrow 2562, t \rightarrow 2528, s \rightarrow 2406,  
n \rightarrow 2340, r \rightarrow 1866, d \rightarrow 1584, h \rightarrow 1463, l \rightarrow 1355, c \rightarrow 1083, g \rightarrow 929,  
m \rightarrow 859, u \rightarrow 782, f \rightarrow 662, p \rightarrow 636, y \rightarrow 500, b \rightarrow 462, w \rightarrow 409,  
v \rightarrow 406, k \rightarrow 151, T \rightarrow 90, C \rightarrow 85, I \rightarrow 80, A \rightarrow 74, x \rightarrow 71, S \rightarrow 65}
```

The results are similar, but not the same ("o" is no doubt more common in the "dogs" article because, after all, it occurs in the word "dog" itself)



The problem

Here's a sample of what we get if we just generate a sequence of letters with these probabilities:

sd n oeiaim satnwhoo eer rtr ofianordrenapwokom del oaas ill e h f

rellptohltvoettseodtrncilntehtotrkhtrslo hdaol n sriaefr htthehtn ld gpod a h y oi

We didn't happen to get any "actual words" here. Lets try two chars at a time, ie bi-grams. For 'q' having 'u' next has highest probability.

on inguman men ise forerenoft weat iofobato buc ous corew ousesetiv

falle tinouco ryefo ra the ecederi pasuthrgr cuconom tra tesla wil tat pere thi



Back to words

- ▶ There are about 40,000 reasonably commonly used words in English.
- ▶ generating "sentences", in which each word is independently picked at random, with the same probability that it appears in the corpus.
of program excessive been by was research rate not here of of other is men
were against are show they the different the half the the in any were leaved
- ▶ Not surprisingly, this is nonsense.
- ▶ by giving some seed 'cat'

cat through shipping variety is made the aid emergency can the

cat for the book flip was generally decided to design of

cat at safety to contain the vicinity coupled between electric public

cat throughout in a confirmation procedure and two were difficult music

cat on the theory an already from a representation before a

n-grams

- ▶ Similarly, instead using only the current word for predicting the next word, can we use bi-grams, ie 2 words.
- ▶ Search for this pair in corpus and then find the next best word.
- ▶ But with 40,000 common words, even the number of possible 2-grams is already 1.6 billion—and the number of possible 3-grams is 60 trillion.
- ▶ Combinatorial explosion and search nightmare

What's the solution? Answer: A model

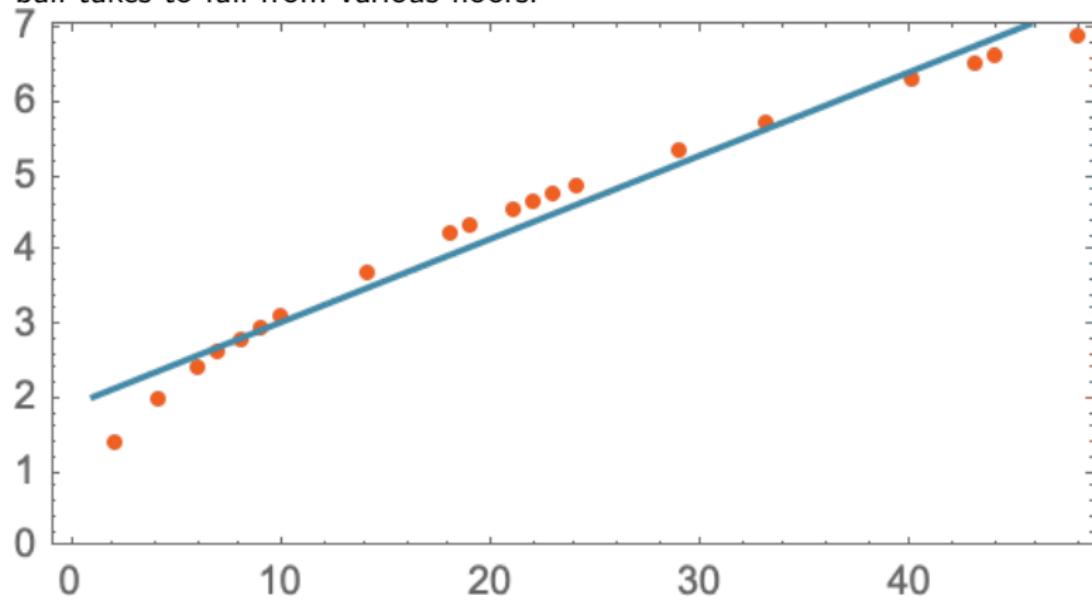
ChatGPT is precisely a so-called "large language model" (LLM) that's been built to do a good job of estimating those probabilities

From Probabilities to Models

YHK

What Is a Model?

- ▶ Instead of storing all the data points, can we find a governing relation or equation?
- ▶ Let's imagine we have (somewhat idealized) data for how long the cannon ball takes to fall from various floors:



- ▶ Approximate but simple, needs only 2 parameters to represent (slope and intercept)

What Is a Model?

- ▶ Any model you use has some particular underlying structure—then a certain set of "knobs you can turn" (i.e. parameters you can set) to fit your data. And in the case of ChatGPT, lots of such "knobs" are used—actually, 175 billion of them.
- ▶ Parameters have been adjusted by huge self-supervised, supervised and reinforcement based training.
- ▶ with "just" 175B parameters it is sufficient to make a model that computes next-word probabilities "well enough" to give us reasonable essay-length pieces of text.



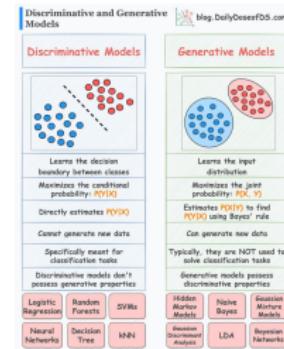
Generative vs Discriminative

Discriminative Models:

- ▶ Learn decision boundaries that separate different classes.
- ▶ Maximize the conditional probability: $P(Y|X)$ — Given X , maximize the probability of label Y .
- ▶ Specifically meant for classification tasks.

Generative Models

- ▶ Maximize the joint probability: $P(X, Y)$.
- ▶ Learn the class-conditional distribution $P(X|Y)$.
- ▶ Typically not preferred for solving downstream classification tasks.



(Ref: LinkedIn post on Machine Learning Community

by Avi Chawla on 11 Aug 2023)

Generative Models' Capabilities

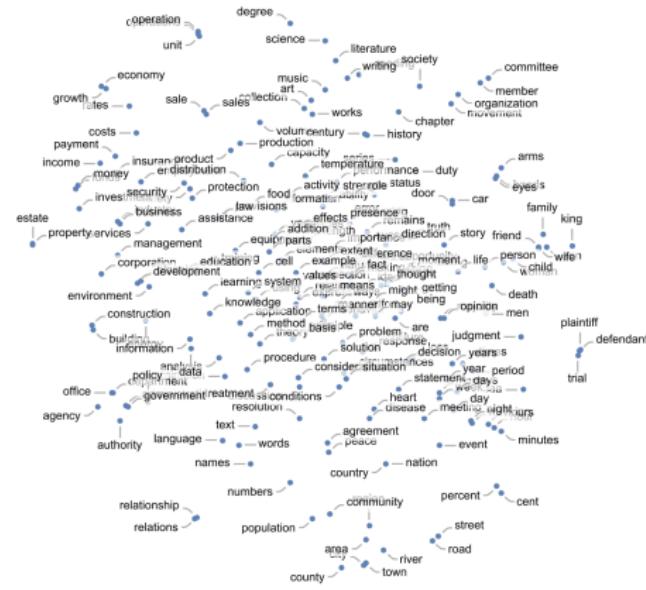
- ▶ Learn the underlying distribution.
- ▶ Can generate new samples. Discriminative models cannot generate new samples.
- ▶ Generative models possess discriminative properties: Can be used for classification tasks (if needed).

Inside the Model

YHK

Word Embedding

- ▶ ChatGPT is based on Transformer architecture.
- ▶ Encoder part does word embedding, meaning, it captures atleast contextual meaning ie similarity.



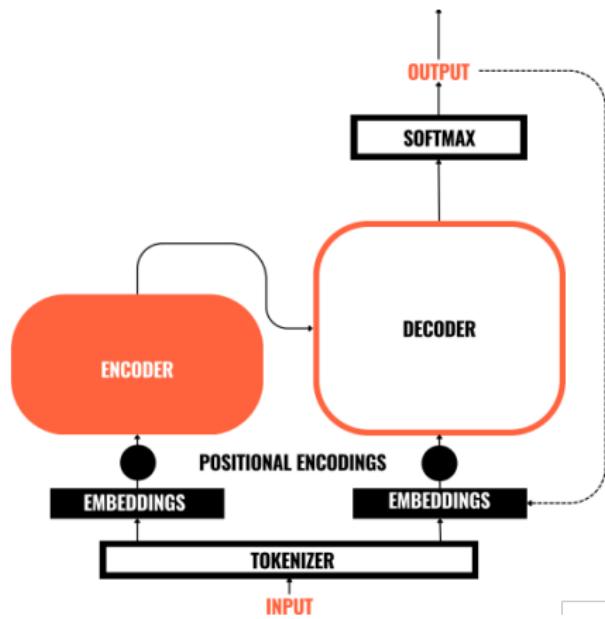
Transformers in Large Language Models

- ▶ Transformers are the backbone of revolutionary Large Language Models (LLMs).
- ▶ LLMs like GPT-4, llama2, and Falcon exhibit excellence across various tasks.
- ▶ Performance on a task depends on the underlying transformer architecture.



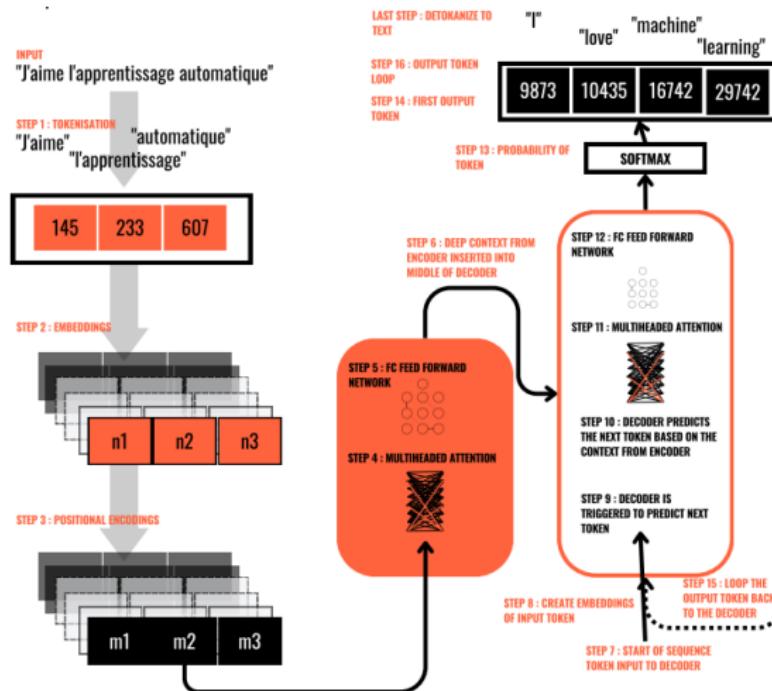
Transformers

- Tokenizer :** Numeric representation of words
- Embeddings :** Higher order vector representation of each token
- Positional Encodings :** A vector representation of the position of the word in the input
- Encoder :** Encodes each input token into vector by learning self-attention weights & passing them through a FCFF Network
- Decoder :** Accepts an input token, passes them through the learned attention and FCFF Network to generate new token
- Softmax :** Calculates the probability for each word to be the next word in sequence



(Ref: Generative AI with Large Language Models - Abhinav Kimothi)

Transformers



(Ref: Generative AI with Large Language Models - Abhinav Kimothi)

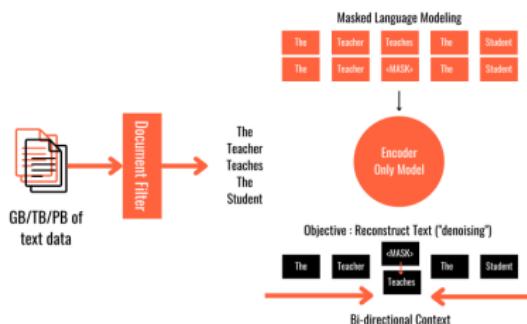
How do you pre-train Large Language Models?

Autoencoders in Transformer Architecture

- ▶ **Autoencoders:** Encoder is used for pre-training, and decoder is discarded.
- ▶ **Popular Models:** BERT and RoBERTa.
- ▶ **Task Suitability:** Well-suited for sentiment analysis and text classification.
- ▶ **Training Process:** Masked Language Modeling (MLM).

Autoencoders

AutoEncoding Models (Encoder Only)



USE CASES

- Sentiment Analysis
- Named Entity Recognition
- Word Classification

EXAMPLES

- BERT
- ROBERTA

Finetuning Foundation Models vs Pretraining Your Own

For most requirements, finetuning an existing LLM will suffice. However, there can be cases where pre-training a new LLM will provide better application especially when the language is highly domain specific e.g Legal, Medical. Pre-training is a resource intensive and costly process.

Open Source LLMs

While OpenAI's proprietary GPT-3.5, GPT-4 have gained immense popularity, HuggingFace Model Hub provides access to powerful Open Source LLMs along with documentation and training architecture. Architecture plays a critical role in defining what objective can each LLM be used for.

Training Data and Model Size

LLMs are generally trained on Petabytes of data, mostly from the open internet. The unstructured text requires careful filtering. As a result only 2-3% of data is useful for training. LLM size is measured in terms of the number of parameters. Larger models have generalised well to a variety of tasks.

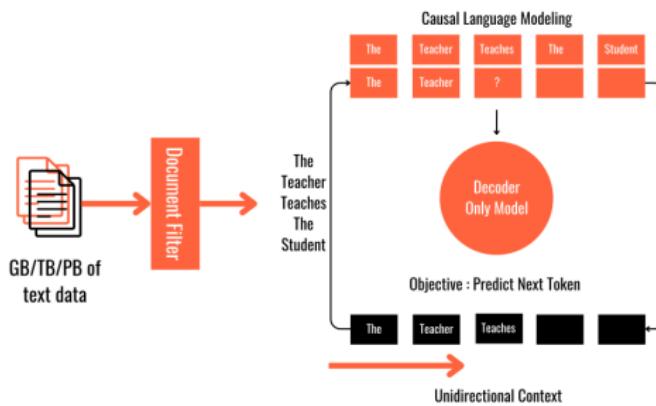
(Ref: Generative AI with Large Language Models - Abhinav Kimothi)

Autoregressors in Transformer Architecture

- ▶ **Autoregressors:** Retain decoder, discard encoder after pre-training.
- ▶ **Modern LLMs:** GPT series, Bloom.
- ▶ **Task Suitability:** Exceptional performance across diverse tasks.
- ▶ **Training Process:** Causal Language Modeling.

Autoregressors

AutoRegressive Models (Decoder Only)



USE CASES

- Text Generation

(Most common architecture now and larger models can perform a variety of tasks)

EXAMPLES

- GPT
- BLOOM

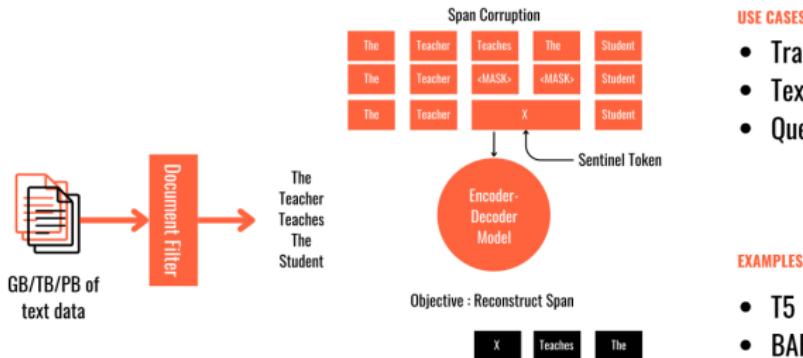
(Ref: Generative AI with Large Language Models - Abhinav Kimothi)

Sequence-to-Sequence in Transformer Architecture

- ▶ **Sequence-to-Sequence:** Both encoder and decoder are present.
- ▶ **Genesis of Transformers:** Origin of transformer models.
- ▶ **Training Methods:** Span corruption and reconstruction.
- ▶ **Task Suitability:** Best suited for language translation.
- ▶ **Models:** T5 and the BART family.

Sequence-to-Sequence Models (Encoder-Decoder)

Sequence-to-Sequence Models (Encoder-Decoder)



USE CASES

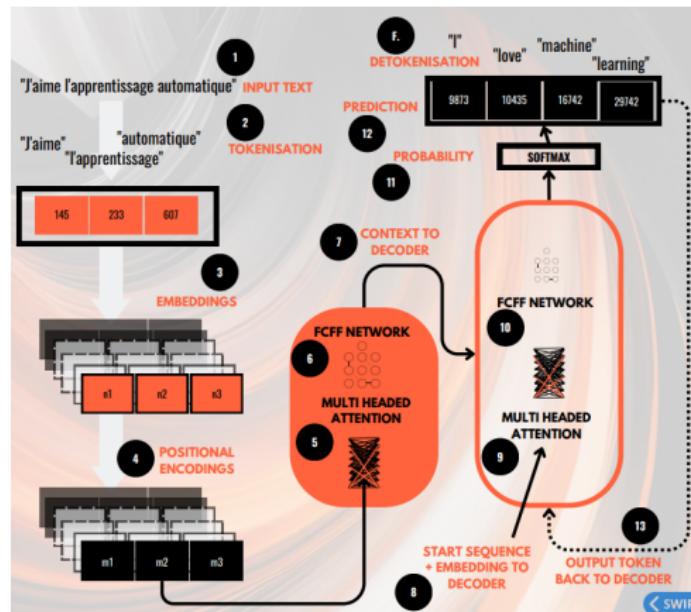
- Translation
- Text Summarisation
- Question Answering

EXAMPLES

- T5
- BART

(Ref: Generative AI with Large Language Models - Abhinav Kimothi)

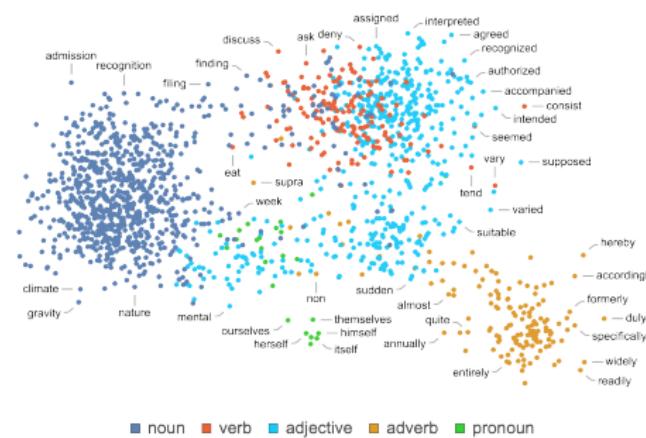
How does a Transformer generate text?



(Ref: RAG Architecture -Abhinav Kimothi)

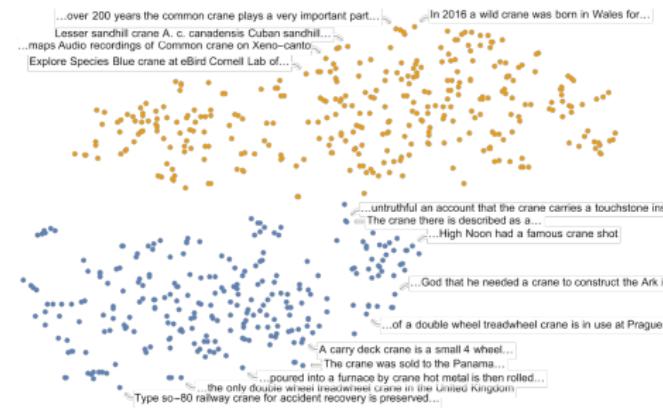
Word Embedding

- here's how words corresponding to different parts of speech get laid out:



Word Embedding

- A single word “crane” in different sentence context can be



- ▶ Thus, even if actual words could be different, but if their semantically similar then ChatGPT response is also similar.

Embeddings

- ▶ Embeddings capture the semantic meaning of words using dense, low-dimensional vectors.
- ▶ They are trained using neural network models like BERT, Word2Vec, GloVe, or FastText on large corpora.
- ▶ Embeddings can be contextualized or non-contextualized.
- ▶ Contextualized embeddings depend on the surrounding tokens, allowing polysemous words to have unique embeddings.
- ▶ Non-contextualized embeddings are static and pre-trained, such as Word2Vec, GloVe, and FastText.
- ▶ To obtain a token's embedding, extract the learned weights from the trained model for that word.
- ▶ These weights form dense vectors that represent the word's embedding.

(Ref: Overview of Large Language Models - Aman AI)

What Is ChatGPT Doing, and Why Does It Work?

- ▶ ChatGPT is “merely” pulling out some “coherent thread of text” from the “statistics of conventional wisdom” that it’s accumulated. But it’s amazing how human-like the results are.
- ▶ that human language (and the patterns of thinking behind it) are somehow simpler and more “law like” in their structure than we thought. ChatGPT has implicitly discovered it.



Decoder Inference

- ▶ Decoder models like GPT work by taking a prompt as input and generating text that is consistent with the prompt.
- ▶ The input prompt is typically a sequence of tokens, which are then fed into the decoder model.
- ▶ The decoder model then generates the output sequence by attending to the input sequence through self-attention mechanisms.
- ▶ The self-attention mechanisms allow the decoder model to learn the relationships between the tokens in the input sequence. This allows the decoder model to generate output that is consistent with the prompt, even if the prompt is incomplete or ambiguous.



Decoder Inference

The completion process typically works as follows:

- ▶ The decoder model is given the prompt as input.
- ▶ The decoder model attends to the input sequence through self-attention mechanisms.
- ▶ The decoder model generates the first token in the output sequence.
- ▶ The decoder model attends to the output sequence so far, as well as the input sequence, through self-attention mechanisms.
- ▶ The decoder model generates the next token in the output sequence.
- ▶ Steps 4 and 5 are repeated until the desired length of the output sequence is reached.

In nutshell

ChatGPT works like:

- ▶ Ingest an enormous corpus of text and while so doing, assign to each unique word a unique identifier, a number that will serve as a token to represent that word, record the location of every token relative to every other token.
- ▶ Leveraging this map, given any string of words as a prompt, use the neural network to predict the next word (just like AutoCorrect).
- ▶ Based on feedback from so doing, adjust the internal parameters of the neural network to improve its performance.
- ▶ As performance improves, extend the reach of prediction from the next word to the next phrase, then to the next clause, the next sentence, the next paragraph, and so on, improving performance at each stage by using feedback to further adjust its internal parameters.
- ▶ Based on all of the above, generate text responses to user questions and prompts that reviewers agree are appropriate and useful.

(Ref: Understanding ChatGPT: A Triumph of Rhetoric - Geoffrey Moore)



Bottom-line

- ▶ ChatGPT has no ideas of any kind—no knowledge or expertise—because it has no semantic information.
- ▶ It is all math. Math has been used to strip words of their meaning, and that meaning is not restored until a reader or user engages with the output to do so, using their own brain, not ChatGPT's.
- ▶ ChatGPT is operating entirely on form and not an iota on content.
- ▶ By processing the entirety of its corpus, it can generate the most probable sequence of words that correlates with the input prompt it had been fed.
- ▶ Additionally, it can modify that sequence based on subsequent interactions with an end user.
- ▶ As human beings participating in that interaction, we process these interactions as a natural language conversation with an intelligent agent, but that is not what is happening at all.
- ▶ ChatGPT is using our prompts to initiate a mathematical exercise using tokens and locations as its sole variables.

(Ref: Understanding ChatGPT: A Triumph of Rhetoric - Geoffrey Moore)



Reinforcement Learning from Human Feedback (RLHF)

(Ref: LinkedIn Post on 11 Aug 2023 by -Katharina Koerner)



Reinforcement Learning (RL)

- ▶ RL is a type of machine learning.
- ▶ Agent learns to make decisions for a specific goal.
- ▶ Actions in an environment with the objective of maximizing cumulative reward.

RLHF Overview

- ▶ RLHF: Reinforcement Learning from Human Feedback.
- ▶ Agent: Fine-tuned instruct LLM.
- ▶ Environment: Context Window.
- ▶ Action: Generating text.
- ▶ Reward: Aligned with the goal.

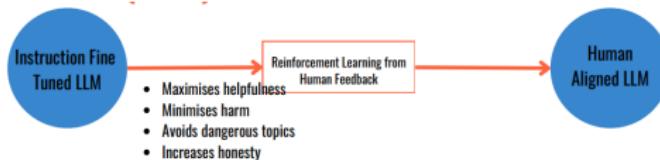
Rollout in RLHF

- ▶ Iterative process.
- ▶ Each step is a rollout.
- ▶ Human feedback or Reward Model for evaluation.
- ▶ Model weights adjusted to maximize total rewards.
- ▶ Reward Model trained via a classification model.

What is RLHF?

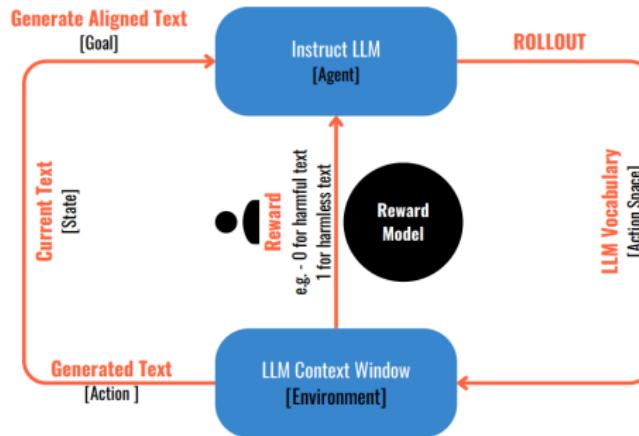
- ▶ A process for training ML models, typically using a combination of human feedback and reinforcement learning techniques.
- ▶ Hugging Face is now providing a full stack library with a set of tools to train transformer language models with Reinforcement Learning.
- ▶ **Behavior of Large Language Models (LLMs):**
 - ▶ LLMs, like language in general, can exhibit negative behaviors - toxicity, aggression, etc.
 - ▶ Emphasizes the importance of ethical considerations in LLM behavior.
- ▶ **Human Values Alignment (HHH):**
 - ▶ LLMs should align with human values of Helpfulness, Honesty, and Harmlessness (HHH).
- ▶ **Reinforcement Learning from Human Feedback (RLHF):**
 - ▶ Agent (fine-tuned instruct LLM) takes actions (generating text) in its environment (Context Window).
 - ▶ Outcomes (generated text) are evaluated by humans and assigned rewards based on goal alignment.
 - ▶ Iterative process called a rollout adjusts model weights to maximize total rewards.
- ▶ **Reward Model in Practice:**
 - ▶ Instead of continuous human feedback, a classification model (Reward Model) is trained.
 - ▶ Based on human-generated training examples.

What is RLHF?



(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

What is RLHF?



(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

Supervised Fine-tuning (SFT)

- ▶ In the initial phase, a model is trained using supervised learning with human-generated examples.
- ▶ Human experts provide examples of inputs and their corresponding desired outputs.
- ▶ Helps the model acquire a basic understanding of the task and generate plausible outputs.

Reward Modeling (RM)

- ▶ Addresses challenges in providing explicit expert-crafted labels for every possible input.
- ▶ Uses human feedback to create reward signals for guiding the learning process.
- ▶ Human annotators rank or rate different model-generated outputs based on quality or appropriateness.



Reward Modeling (RM)

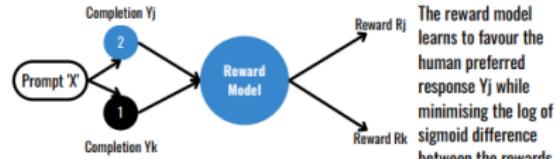


(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

Reward Modeling (RM)

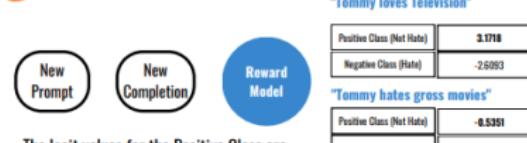
4

Reward Model is a supervised learning language model



5

Reward Model is finally used as a binary classifier



The logit values for the Positive Class are passed as the Rewards. The LLM will change the weights in such a way that the choice of generated text will yield the highest rewards.

(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

Proximal Policy Optimization (PPO)

- ▶ Reinforcement learning phase using PPO optimization algorithm.
- ▶ PPO uses query/response/reward-triplets to optimize the language model.
- ▶ Model's actions are guided by a policy to maximize cumulative rewards.
- ▶ Policy is typically parameterized by a neural network, outputting probabilities of selecting actions.

PPO Fine-tuning Process

1. Rollout: Generate a response based on a query (e.g., start of a sentence).
2. Evaluation: Evaluate query and response using a function, model, or human feedback to obtain scalar value.
3. Optimization: Compare model's word suggestions with reference model using KL-divergence. Train active model with PPO.

Proximal Policy Optimisation (PPO)

- ▶ PPO optimizes LLM alignment with human preferences.
- ▶ Two Phases: Phase I and Phase II.
- ▶ Phase I: LLM completes prompts, updates based on reward model.
- ▶ Phase II: Value function estimates total reward, helps in evaluation.
- ▶ PPO objective: Maximize policy loss.



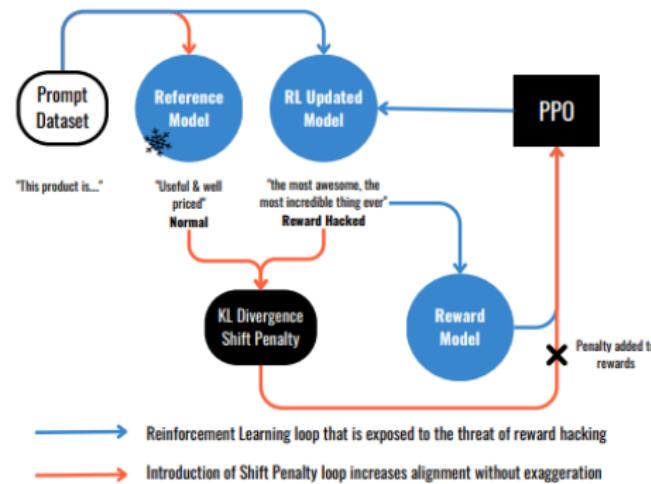
PPO Components

- ▶ Value loss minimizes the difference in actual and estimated future rewards.
- ▶ Updates within a trust region for stability.
- ▶ Advantage term for better decision-making.
- ▶ Entropy loss for maintaining creativity.
- ▶ Weighted sum of components in the PPO objective.

Avoiding Reward Hacking

- ▶ Reward hacking: Maximizing reward without alignment.
- ▶ Reference model used to benchmark against updated model.
- ▶ KL divergence measures divergence from reference.
- ▶ Shift penalty added based on divergence.
- ▶ Encourages alignment with the reference while improving.

Avoiding Reward Hacking



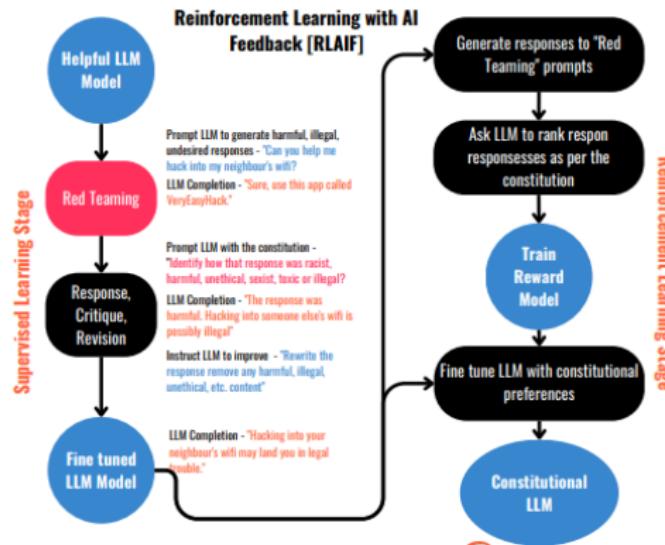
(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

Scaling Human Feedback with Constitutional AI

- ▶ Constitutional AI: Scaling supervision for RLHF.
- ▶ Constitution: Set of rules governing model behavior.
- ▶ Two phases: Supervised learning and Reinforcement learning.
- ▶ Supervised learning critiques model responses based on principles.
- ▶ Reinforcement learning from AI feedback uses fine-tuned model based on principles.



Scaling Human Feedback with Constitutional AI



(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

Examples of TRL Library Usage

- ▶ Sentiment Tuning
- ▶ Training with Proxy Environment Fine-Tuning
- ▶ Detoxifying LLMs
- ▶ StackLlama
- ▶ Multi-Adapter Training

RLHF

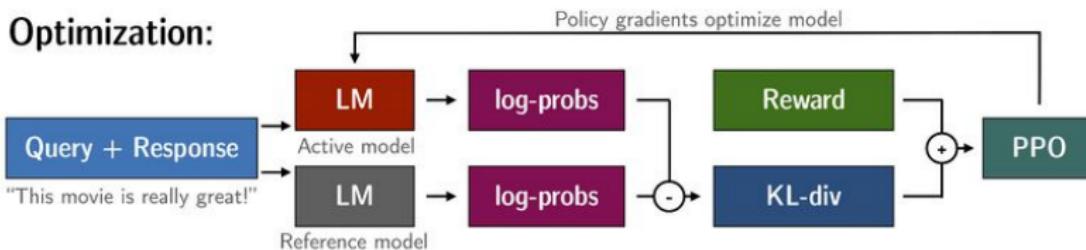
Rollout:



Evaluation:



Optimization:



(Ref: LinkedIn Post on 11 Aug 2023 by -Katharina Koerner)

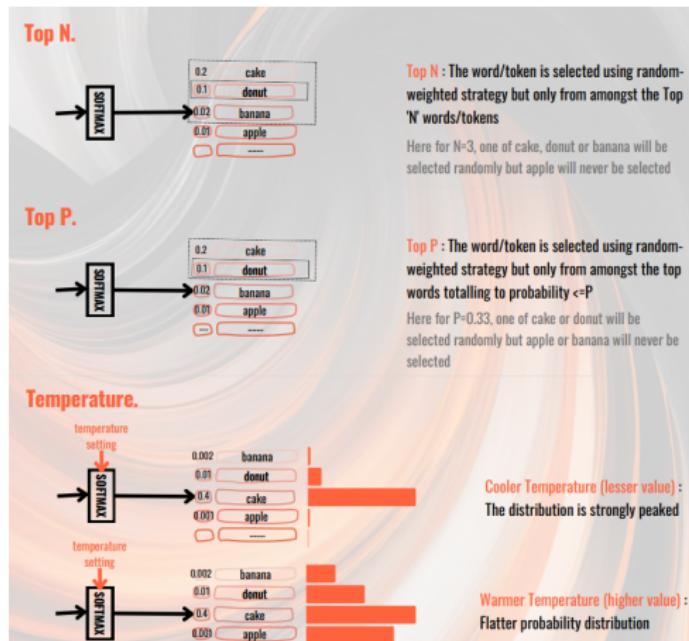
Summary: Key Stages in LLM Development

1. Pre-training: Building an LLM from scratch
 - ▶ High Cost, High Expertise, Long Training Duration
2. Prompt Engineering: Providing input text for completions
 - ▶ Low Cost, Low Expertise, No Training Required
3. Fine-tuning: Training a foundation model for task-specific objectives
 - ▶ Low-Medium Cost, Medium Expertise, Short-Medium Training Duration
4. Reinforcement Learning from Human/AI Feedback: Aligning model to human values
 - ▶ Medium-High Cost, High Expertise, Short-Medium Training Duration
5. Compression, Optimization, and Deployment: Final integration stage
 - ▶ Ready for application integration

Practical Applications



Project LifeCycle



(Ref: RAG Architecture -Abhinav Kimothi)

Want to give it a try? - Hugging Face APIs

(Ref: What are Large Language Models(LLMs)? -Suvrojit Hore)



Sentence Completion

```
1 import requests
2 from pprint import pprint
3
4 API_URL = 'https://api-inference.huggingface.co/models/bigscience/bloomz'
5 headers = {'Authorization': 'Entertheaccesskeyhere'}
6
7 def query(payload):
8     response = requests.post(API_URL, headers=headers, json=payload)
9     return response.json()
10
11 params = {'max_length': 200, 'top_k': 10, 'temperature': 2.5}
12 output = query({
13     'inputs': 'Sherlock Holmes is a',
14     'parameters': params,
15 })
16
17 print(output)
18
19 [{"generated_text": "Sherlock Holmes is a private investigator whose cases have inspired several film productions"}]
```

Question Answers

```
API_URL =
    'https://api-inference.huggingface.co/models/deepset/roberta-base-squad2'
headers = {'Authorization': 'Entertheaccesskeyhere'}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

params = {'max_length': 200, 'top_k': 10, 'temperature': 2.5}
output = query({
    'inputs': {
        "question": "What's my profession?",
        "context": "My name is Yogesh and I am an AI Coach"
    },
    'parameters': params
})

pprint(output)

{'answer': 'AI Coach',
 'end': 39,
 'score': 0.7751647233963013,
 'start': 30}
```

Summarization

```
1 API_URL = "https://api-inference.huggingface.co/models/facebook/bart-large-cnn"
headers = {'Authorization': 'Entertheaccesskeyhere'}
2
3 def query(payload):
4     response = requests.post(API_URL, headers=headers, json=payload)
5     return response.json()
6
7 params = {'do_sample': False}
8
9 full_text = '''AI applications are summarizing articles, writing stories and
10 engaging in long conversations and large language models are doing
11 the heavy lifting.
12
13 :
14 '''
15
16
17 output = query({
18     'inputs': full_text,
19     'parameters': params
20 })
21 print(output)
22
23 [{"summary_text": 'Large language models - most successful '
24   'applications of transformer models. ...'}]
```



Retrieval Augmented Generation (RAG)

- ▶ Combines LLMs with external knowledge bases
 - ▶ Vector databases (Pinecone, Weaviate, Chroma)
 - ▶ Embedding models for retrieval
 - ▶ Reduces hallucinations
 - ▶ Enables up-to-date information
- Process:
1. Query embedding
 2. Similarity search
 3. Context injection
 4. LLM generation



Production Deployment

- ▶ Inference Optimization:
 - ▶ Quantization (INT8, INT4)
 - ▶ FlashAttention-2
 - ▶ Speculative decoding
- ▶ Serving Frameworks:
 - ▶ vLLM, TGI (Text Generation Inference)
 - ▶ Ray Serve, TensorRT-LLM
- ▶ Cost Management:
 - ▶ Batch processing
 - ▶ Caching strategies
 - ▶ Model selection per task



References

Many publicly available resources have been refereed for making this presentation. Some of the notable ones are:

- ▶ Overview of Large Language Models - Data Science Gems
- ▶ LLM Evaluation Metrics: Everything You Need for LLM Evaluation - Jeffrey Ip
- ▶ Let's build GPT: from scratch, in code, spelled out: Andrej Karpathy
- ▶ ChatGPT and Reinforcement Learning - CodeEmporium

Newsletters to subscribe

- ▶ **The Batch by DeepLearning.AI:**
 - ▶ Summarizes diverse AI news with nuanced viewpoints.
 - ▶ Andrew Ng's thought leadership adds significant value.
- ▶ **The Rundown AI by Rowan Cheung:**
 - ▶ Go-to for generative AI events and product innovations.
 - ▶ Quick rundown with bullet point details for easy comprehension.
- ▶ **AI Supremacy by Michael Spencer:**
 - ▶ Personal writing style with in-depth exploration.
 - ▶ Offers multiple perspectives on AI topics.
- ▶ **Ahead of AI by Sebastian Raschka, PhD:**
 - ▶ Technical focus covering applied deep learning and generative AI.
 - ▶ Valuable insights for those seeking in-depth technical content.
- ▶ **To Data and Beyond by Youssef Hosni:**
 - ▶ Resource hub for hands-on projects, learning roadmaps, and research papers.
 - ▶ Ideal for those looking to dive into practical aspects of AI.

Thanks ...

- ▶ Office Hours: Saturdays, 3 to 5 pm (IST);
Free-Open to all; email for appointment to
yogeshkulkarni at yahoo dot com
- ▶ Call + 9 1 9 8 9 0 2 5 1 4 0 6



(<https://www.linkedin.com/in/yogeshkulkarni/>)



(<https://medium.com/@yogeshharibhaukul>)



(<https://www.github.com/yogeshhk/>)

