Introduction to Large Language Models (LLMs)

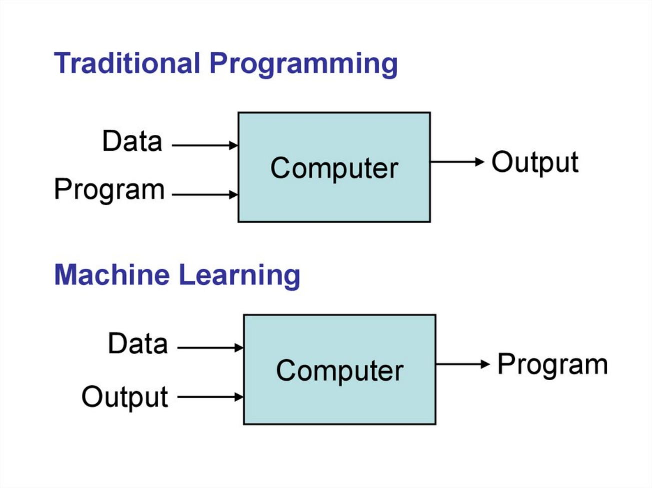
Yogesh Haribhau Kulkarni

Outline

YHK

Background

Traditional vs. Machine Learning?



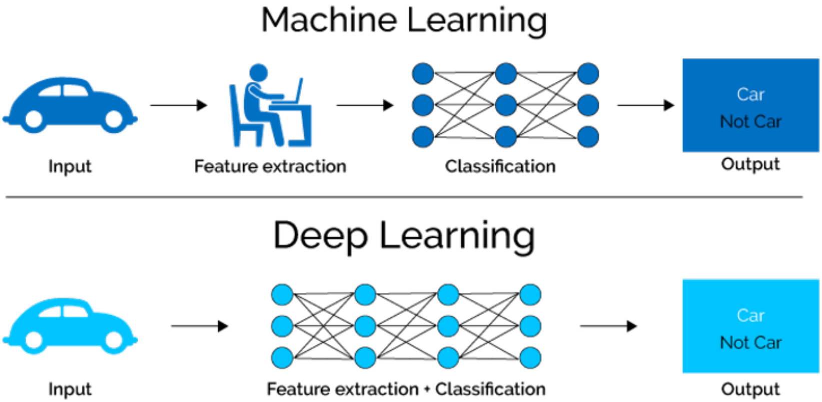
Why Machine Learning?

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

YHK

ML vs DL: What's the difference?

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



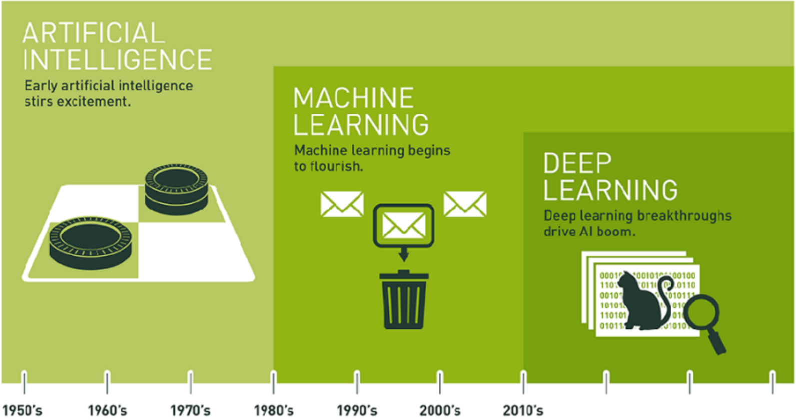
(Reference: https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png)

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.

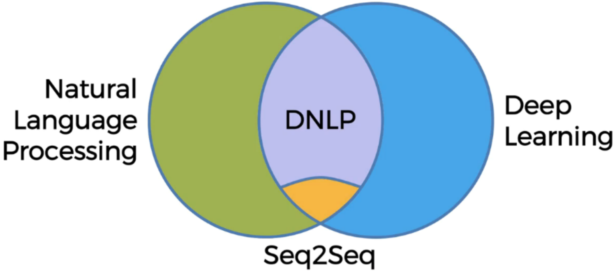
(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Relationship between AI, ML, DL



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

What is Deep NLP



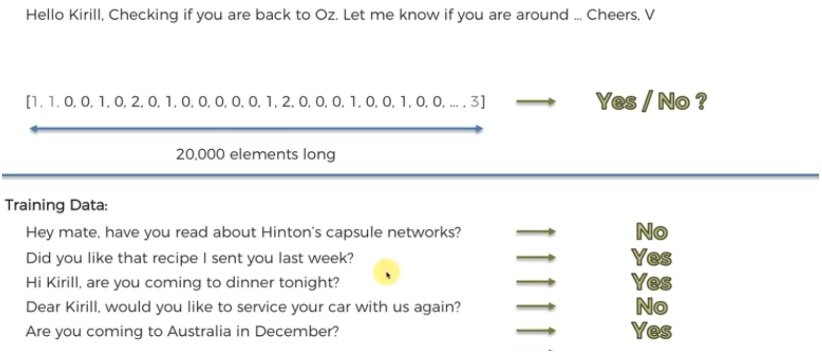
(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)
(Note: Size is not indicative of importance)

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)

Overview of Large Language Models

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.



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

YHK

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

- <u>Distributional semantics</u>: **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)

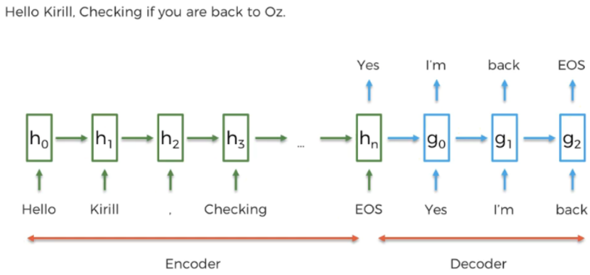
Word vectors

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

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

YHK

Seq2Seq architecture

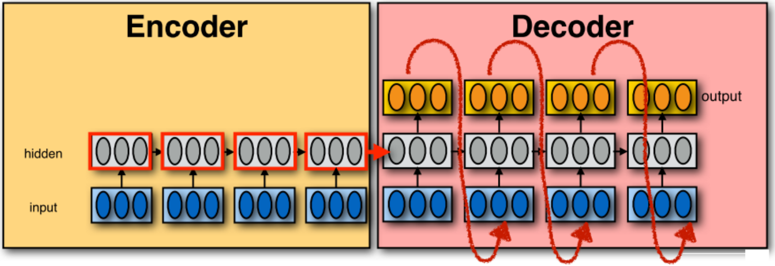


(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)

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)

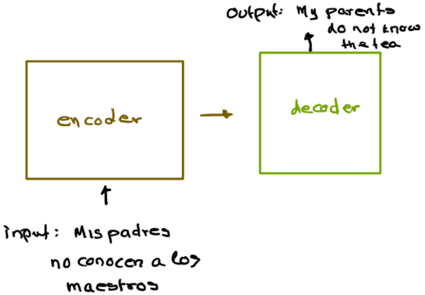
Transformers use Self-Attention

- 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

YHK

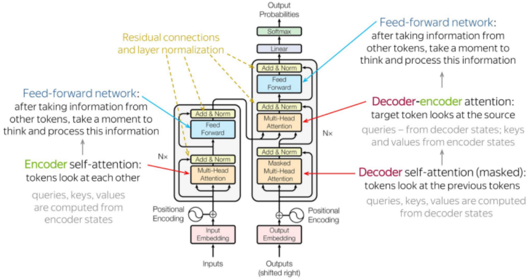
Transformers

- 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?



YHK

Transformer Models



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

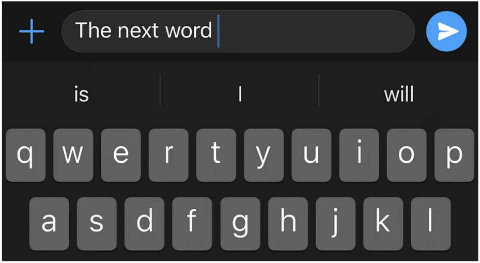
Transformers are basis of (the most) Large Language Models

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

YHK

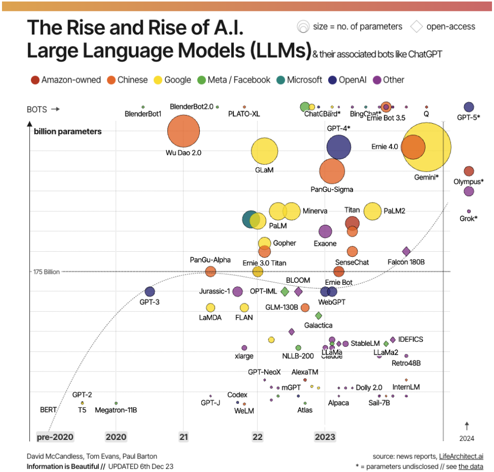
What is a Language Models?

- 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)

LLM - Information is beautiful



(Ref: https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-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.

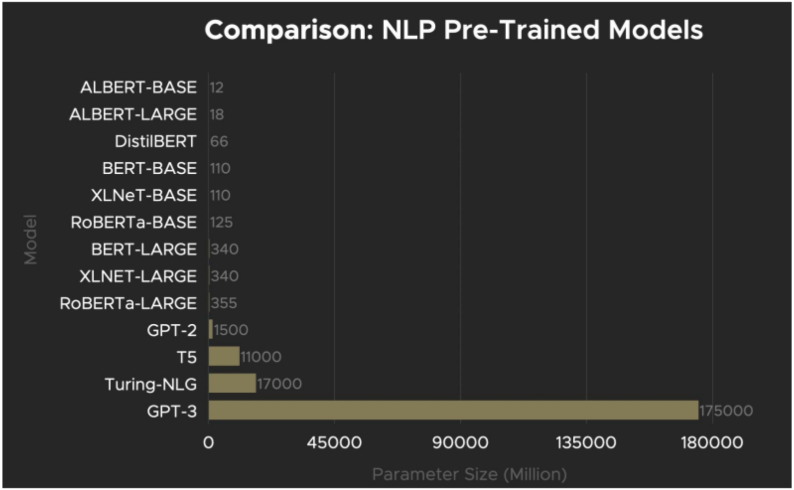
YHK

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.

YHK

Large Language Models - Comparison



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

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

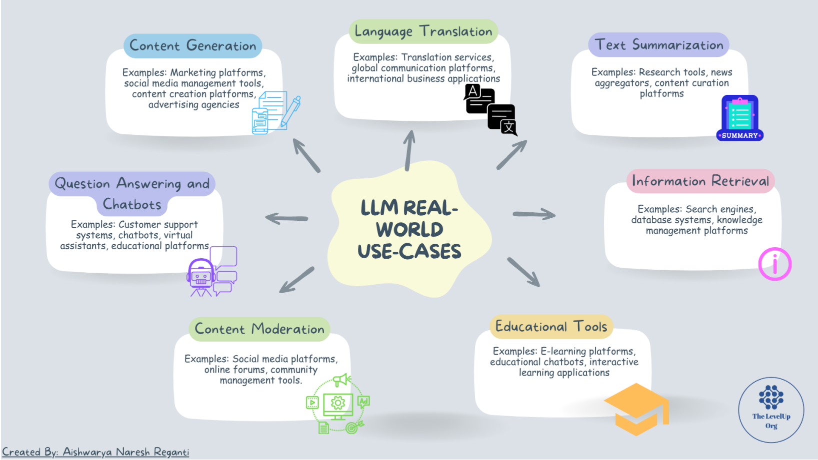
- **Model Architecture:** LLM performance is influenced by the design and intricacy of its architecture.
- **Dataset Quality:** The quality and diversity of the training dataset shape the model's language understanding.
- Training a private LLM demands substantial computational resources and expertise.
- 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.

YHK

LLM Real World Use Cases



(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.

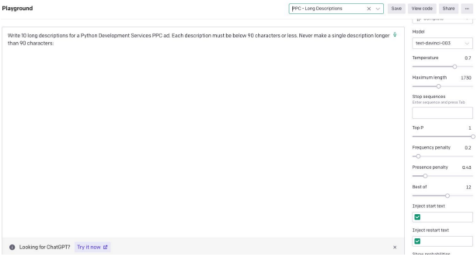
(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

YHK

LLM Challenges

(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

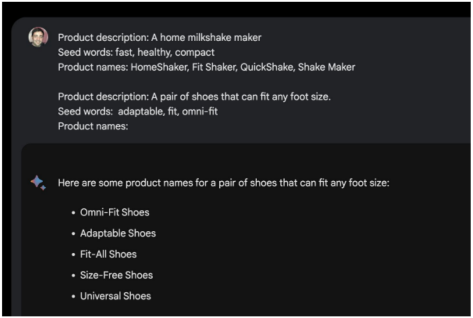
ChatGPT — GPT3.5/GPT4



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

Created by OpenAI Access it with code or without (Playground https://platform.openai.com/playgro

Bard — Palm 2/Gemini



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

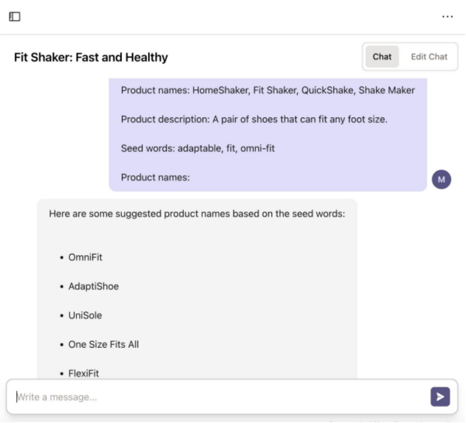
Created by Google Access it via chat https://bard.google.com/ or encounter it in search results

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))

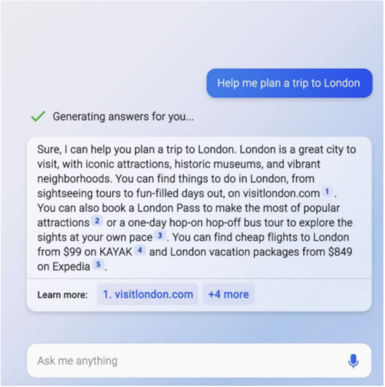
Anthropic Claude



(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

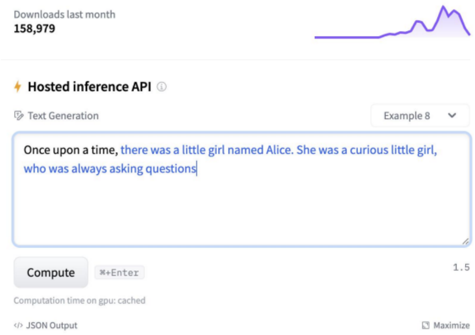
Microsoft Bing — GPT 4



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

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

Falcon



(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/)

Want to give it a try? - Hugging Face APIs

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

YHK

Sentence Completion

```python
import requests
from pprint import pprint

API_URL = 'https://api-inference.huggingface.co/models/bigscience/bloomz'
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': 'Sherlock Holmes is a',
    'parameters': params,
})

print(output)

[{'generated_text': 'Sherlock Holmes is a private investigator whose cases '
                    'have inspired several film productions'}]
```

YHK

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

```
API_URL = "https://api-inference.huggingface.co/models/facebook/bart-large-cnn"
headers = {'Authorization': 'Entertheaccesskeyhere'}

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

params = {'do_sample': False}

full_text = '''AI applications are summarizing articles, writing stories and
engaging in long conversations and large language models are doing
the heavy lifting.

:
'''

output = query({
    'inputs': full_text,
    'parameters': params
})
print(output)

[{'summary_text': 'Large language models - most successful '
                  'applications of transformer models. ...'}]
```

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.

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."

YHK

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.

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)

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 . . .

- Search **"Yogesh Haribhau Kulkarni"** on Google and follow me on LinkedIn and Medium
- Office Hours: Saturdays, 2 to 3 pm (IST); Free-Open to all; email for appointment.
- Email: yogeshkulkarni at yahoo dot com

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

(https://medium.com/@yogeshharibhaukulkarni)

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