

Leveraging Large Language Models for Advanced AI Applications

Conf42 Observability 2024



Satya Kale, Sr. SDE Amazon



Gayathri Shivaraj, Sr. PM Amazon

Agenda

- Introduction to Large Language Models
- Understanding LLM Architecture
- Methods for Leveraging LLMs
- Limitations of LLMs
- Real world success stories

Understanding Large Language Models:

1. What are Large Language Models (LLMs)?

Large Language Models (LLMs) are advanced AI models trained on extensive datasets to understand and generate human-like language.

2. Key Components of LLMs:

- ❑ **Transformer Architecture**
- ❑ **Pre-trained Parameters**
- ❑ **Fine-tuning**

3. Capabilities of LLMs:

- ❑ **Content Generation and Comprehension:** Text Generation, Question Answering
- ❑ **Language Processing:** Language Translation, Summarization
- ❑ **Analysis and Recognition:** Sentiment Analysis, Text Classification, Named Entity Recognition (NER)

4. Applications Across Industries:

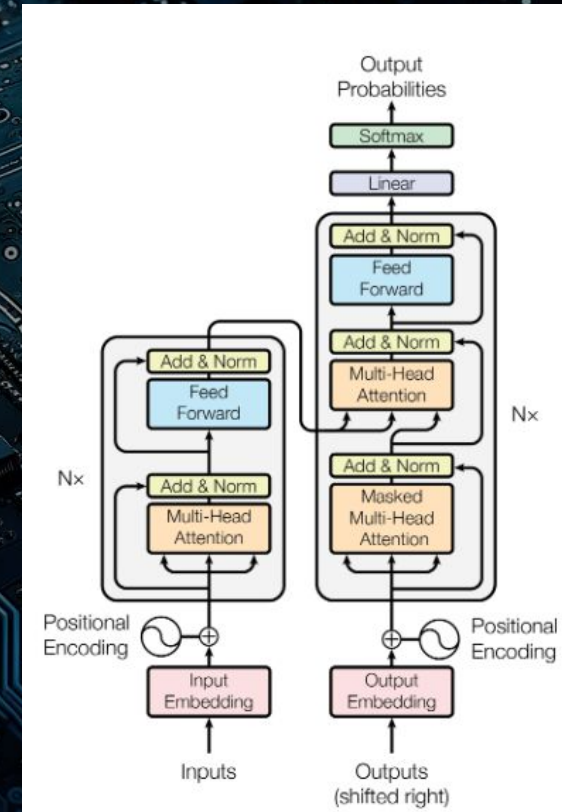
Software Development: LLMs facilitate code summarization, natural language code search, and automated documentation generation, enhancing developer productivity and code understanding.

Learning: LLMs can serve as educational tools for learning programming languages, providing personalized feedback and tutoring to aspiring developers. They can also support the creation of interactive coding exercises and adaptive learning platforms.

Transformer Architecture

- The Transformers architecture, introduced in the paper "Attention is All You Need" by Vaswani et al. (2017), revolutionized natural language processing (NLP)

- Self-Attention Mechanism
- Positional Encoding
- Feedforward Neural Networks
- Encoder and Decoder
- Multi-Head Attention
- Layer Normalization and Residual Connections



Pre-Trained Parameters

- ▶ The model is trained in a self-supervised manner on a large corpus to predict the next tokens given the input
- ▶ Learned weights and biases in the model obtained during pre-training on large datasets. Serve as the initial knowledge base for downstream tasks
- ▶ Components of Pre-trained parameters:
 - ▶ **Word Embeddings**
 - ▶ **Transformer Layers**
 - ▶ **Output Layer Parameters**

Fine Tuning

- ▶ Hyper parameterization
- ▶ One-shot/Few-shot learning
- ▶ Domain adaptation

Applications that benefit from fine-tuning are Sentiment analysis, Chatbots, Summarization

How Can I Leverage Large Language Models (LLMs)?

□ For Developers

▪ Integration in Applications

- API and SDK based offerings
- Pre-trained foundational model hosting
- Custom/Fine tuned model hosting

▪ Quick Playgrounds

- AWS Bedrock
- Quick UIs with Gradio

□ For Non Developers

No-Code/Low-Code Platforms— AWS Bedrock, Chat GPT, Claude AI

Using Pre-Trained Models via API Integration

- leverage pre-trained models through APIs provided by relevant parties such as AWS Bedrock, Google AI Platform etc

```
import boto3
import json
brt = boto3.client(service_name='bedrock-runtime')

body = json.dumps({
    "prompt": "\n\nHuman: explain black holes to 8th graders\n\nAssistant:",
    "max_tokens_to_sample": 300,
    "temperature": 0.1,
    "top_p": 0.9,
})

modelId = 'anthropic.claude-v2'
accept = 'application/json'
contentType = 'application/json'

response = brt.invoke_model(body=body, modelId=modelId, accept=accept, contentType=contentType)

response_body = json.loads(response.get('body').read())


# text
print(response_body.get('completion'))
```


Model Type	Model Name
Text	<ul style="list-style-type: none">Jurassic-2 from AI21 LabsTitan Text from AmazonClaude v2 from AnthropicCommand R, R+ from CohereLlama 2 & 3 from MetaMistral 7B, Large from Mistral AI
Image Generation	<ul style="list-style-type: none">Titan Image Generator from Amazon.Stable Diffusion from Stability AI
Text & Vision	<ul style="list-style-type: none">Claude 3 Haiku, Sonnet from Anthropic
Embedding	<ul style="list-style-type: none">Embed by CohereTitan by Amazon

Using Pre-Trained Models via Playgrounds

Chat playground [Info](#)

AI Claude [Change](#)

v2 | ODT 

 Human: I'm going to give you a document. Then I'm going to ask you a question about it. I'd like you to first write down exact quotes of parts of the document that would help answer the question, and then I'd like you to answer the question using facts from the quoted content. Here is the document:

<document>
Anthropic: Challenges in evaluating AI systems

Introduction
Most conversations around the societal impacts of artificial intelligence (AI) come down to discussing some quality of an AI system, such as its truthfulness, fairness, potential for misuse, and so on. We are able to talk about these characteristics because we can technically evaluate models for their performance in these areas. But what many people working inside and outside of AI don't fully appreciate is how difficult it is to build robust and reliable model evaluations. Many of today's existing evaluation suites are limited in their ability to serve as accurate indicators of model capabilities or safety.

At Anthropic, we spend a lot of time building evaluations to better understand our AI systems. We also use evaluations to improve our safety as an organization, as illustrated by our Responsible Scaling Policy. In doing so, we have grown to appreciate some of the ways in which developing and running evaluations can be challenging.

Here, we outline challenges that we have encountered while evaluating our own models to give readers a sense of what developing, implementing, and interpreting model evaluations looks like in practice. We hope that this post is useful to people who are developing AI governance initiatives that rely on evaluations, as well as people who are launching or scaling up organizations focused on evaluating AI systems. We want readers of this post to have two main takeaways: robust evaluations are extremely difficult to develop and implement, and effective AI governance depends on our ability to meaningfully evaluate AI systems.

In this post, we discuss some of the challenges that we have encountered in developing AI evaluations. In order of less-challenging to more-challenging, we discuss:

Multiple choice evaluations
Third-party evaluation frameworks like BIG-bench and HELM
Using crowdworkers to measure how helpful or harmful our models are
Using domain experts to red team for national security-relevant threats
Using generative AI to develop evaluations for generative AI
Working with a non-profit organization to audit our models for dangerous capabilities
We conclude with a few policy recommendations that can help address these challenges.



Challenges
The supposedly simple multiple-choice evaluation
Multiple-choice evaluations, similar to standardized tests, quantify model performance on a variety of tasks, typically with a simple metric—accuracy. Here we discuss some challenges we have identified with two popular multiple-choice evaluations for language models: Measuring Multitask Language Understanding (MMLU) and Bias Benchmark for Question Answering (BBQ).

MMLU: Are we measuring what we think we are?


The Massive Multitask Language Understanding (MMLU) benchmark measures accuracy on 57 tasks ranging from mathematics to history to law. MMLU is widely used because a single accuracy score represents performance on a diverse set of tasks that require technical knowledge. A higher accuracy score means a more capable model.

We have found four minor but important challenges with MMLU that are relevant to other multiple-choice evaluations:

Write a prompt... (Shift + ENTER to start a new line, and ENTER to generate a response)


  Run

Load examples ☐ Compare mode


Configurations 


Reset


▼ System prompts [Info](#)

Add system prompts 


▼ Randomness and diversity [Info](#)

Temperature



Top P


Top K


▼ Length [Info](#)

Maximum length


Stop sequences

Human: 

Examples

► Hugging Face(Serverless Inference For Prototyping)

The screenshot shows the Hugging Face model card for `facebook/detr-resnet-50-panoptic`. The card includes a title, a description of the model, a disclaimer, and a detailed model description. The model is a DETR (End-to-End Object Detection) model trained end-to-end on COCO 2017 panoptic (118k annotated images). It was introduced in the paper [End-to-End Object Detection with Transformers](#) by Carion et al. and first released in [this repository](#).

Model description

The DETR model is an encoder-decoder transformer with a convolutional backbone. Two heads are added on top of the decoder outputs in order to perform object detection: a linear layer for the class labels and a MLP (multi-layer perceptron) for the bounding boxes. The model uses so-called object queries to detect objects in an image. Each object query looks for a particular object in the image. For COCO, the number of object queries is set to 100.

The model is trained using a "bipartite matching loss": one compares the predicted classes + bounding boxes of each of the $N = 100$ object queries to the ground truth annotations, padded up to the same length N (so if an image only contains 4 objects, 96 annotations will just have a "no object" as class and "no bounding box" as bounding box). The Hungarian matching algorithm is used to create an optimal one-to-one mapping between each of the N queries and each of the N annotations. Next, standard cross-entropy (for the classes) and a linear combination of the L1 and generalized IoU loss (for the bounding boxes) are used to optimize the parameters of the model.

DETR can be naturally extended to perform panoptic segmentation, by adding a mask head on top of the decoder outputs.

The card also features a section for the Inference API, showing a sample image of a soccer game and a table of performance metrics.

Class	Score
person	0.999
person	0.947
playingfield	0.987
person	1.000
person	0.951
sports ball	1.000

The screenshot shows the Hugging Face Inference API interface. It displays the model `facebook/detr-resnet-50-panoptic` and provides a code snippet for using the model with the Inference API (serverless).

Use this model with the **Inference API (serverless)**

Python JavaScript cURL

Manage tokens

```
import requests

API_URL = "https://api-inference.huggingface.co/models/facebook/detr-resnet-50-panoptic"
headers = {"Authorization": "Bearer hf_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"}

def query(filename):
    with open(filename, "rb") as f:
        data = f.read()
    response = requests.post(API_URL, headers=headers, data=data)
    return response.json()

output = query("cats.jpg")
```


Hosting Models - Hugging Face Model Hub

- Deploying Pre-Trained Models with Hugging Face Model Hub
- Easy integrations with AWS SageMaker
- Vast selection of Models on Hugging Face Model Hub

```
import sagemaker
import boto3
from sagemaker.huggingface import HuggingFaceModel

try:
    role = sagemaker.get_execution_role()
except ValueError:
    iam = boto3.client('iam')
    role = iam.get_role(RoleName='sagemaker_execution_role')['Role']['Arn']

# Hub Model configuration. https://huggingface.co/models
hub = {
    'HF_MODEL_ID': 'Salesforce/instructblip-flan-t5-xxl',
    'HF_TASK': 'image-to-text'
}

# create Hugging Face Model Class
huggingface_model = HuggingFaceModel(
    transformers_version='4.37.0',
    pytorch_version='2.1.0',
    py_version='py310',
    env=hub,
    role=role,
)

# deploy model to SageMaker Inference
predictor = huggingface_model.deploy(
    initial_instance_count=1, # number of instances
    instance_type='ml.m5.xlarge' # ec2 instance type
)
```

The screenshot displays the Hugging Face Model Card for **mistralai/Mistral-7B-Instruct-v0.3**. The card includes a header with the model name, a 'like' button (498), and a list of tags: Text Generation, Transformers, Safetensors, mistral, conversational, Inference Endpoints, and text-generation-inference. The license is specified as apache-2.0. Below the header, there are tabs for Model card, Files, and Community (27). A 'Gated model' notice states: 'You have been granted access to this model'. The main section, titled 'Model Card for Mistral-7B-Instruct-v0.3', describes the model as a Large Language Model (LLM) and an instruct fine-tuned version of the Mistral-7B-v0.3. It lists changes compared to the previous version: Extended vocabulary to 32768, Supports v3 Tokenizer, and Supports function calling. On the right side, there is a 'Deploy' dropdown menu with options: Inference API (serverless), Inference Endpoints (dedicated), Amazon SageMaker, Azure ML, Google Cloud, and Spaces. A 'Use this model' button is also present. At the bottom right, there is a chatbot interface with the text 'Clara! How are you?'.

Continued...

Create a new Dedicated Endpoint

Model Repository

Import a model from the Hugging Face hub

microsoft/Phi-3-mini-4k-instruct

Endpoint Name

Input a name for your new endpoint

phi-3-mini-4k-instruct-lcj

Instance Configuration

[Contact us](#) if you need a custom solution or if the instance type cannot be selected.



Amazon Web Services



Microsoft Azure



Google Cloud Platform



N. Virginia us-east-1



CPU



GPU



INF2

Intel Ice Lake

1 vCPU · 2 GB

\$ 0.032/h

Intel Ice Lake

2 vCPU · 4 GB

\$ 0.064/h

Intel Ice Lake

4 vCPU · 8 GB

\$ 0.128/h

Intel Ice Lake

8 vCPU · 16 GB

\$ 0.256/h

Automatic Scale-to-Zero

Endpoints scaled to 0 replicas are not billed. They may take some time to scale back up once they start receiving requests again.

Never automatically scale to zero

Endpoint security level

Choose your endpoint's level of privacy.



Protected



Public



Private

A Protected Endpoint is available from the Internet, secured with TLS/SSL and requires a valid Hugging Face Token for Authentication.

► Advanced configuration

Click to expand and configure replica autoscaling, framework, container type...

Hosting Models – AWS SageMaker Studio

SageMaker Studio > Jumpstart > Huggingface > Huggingface Llm Mistral 7b

Applications (5)

JupyterLab

RStudio

Canvas

Code Edi...

Studio CL...

Home

Running Instances

Data

Auto ML

Experiments

Jobs

Training

Model evaluation

Pipelines

Models

JumpStart

Deployments

Mistral 7B

by HuggingFace

AboutNotebooks

Description

Mistral 7B

License

This model has Apache 2.0 License. Please read the terms carefully.

Mistral 7B is the first foundation model from Mistral AI, supporting English text generation tasks and showing natural coding abilities. It is a powerful model with a variety of use cases such as text summarization, classification, text completion, code completion. To demonstrate the easy customizability of the model, Mistral AI has also released a Mistral 7B-instruct model for chat use cases, fine-tuned using a variety of publicly available conversation datasets.

Mistral 7B is a transformer model and uses Grouped-query attention and Sliding-window attention to achieve faster inference (low latency) and handle longer sequences. Mistral 7B has a 8k context length and is an optimal model for low latency with low memory requirement and high throughput for its size. The model is made available under the permissive Apache 2.0 license, for use without restrictions.

Note that we utilize a single GPU to load the model.

Use the Deployed Model for Inference

The deployed model can be used for running inference on any input text. Example python code for how to run inference on the deployed model is given in a notebook you can open by clicking 'Open Notebook' on the model endpoint page that is created when deploying the model.

For any given input text, the model outputs predicted next words in the sequence. Below are some example inputs.

Input Text: What is the recipe for a delicious lemon cheesecake?

Generated Text:

Ingredients

- 100g digestive biscuits

- 50g butter

- 250g cream cheese

- 100g caster sugar

- 1 lemon

- 1 egg

TrainDeployEvaluate

Preview notebooks >

1.9M3.1K

Tags

HuggingfaceText Generation

Papers

Arxiv:2310.06825

AuthorHuggingFace

ProviderHuggingFace

TaskText Generation

LicenseApache-2.0

Deploy Custom/Fine tuned Models

- ▶ For the foundational models, AWS Bedrock allows customizing the model for some popular LLMs like Llama, Titan and Command
- ▶ Import a model with Custom Model Import using Bedrock – Supported for Mistral, Flan and Llama architectures
- ▶ Build Sync/Async endpoint on SageMaker via SageMaker SDK by creating the Tar ball of the model artifacts

```
import sagemaker
from sagemaker import Model

# Specify the location of model artifacts in Amazon S3
model_data = "s3://your-bucket/your-model/model.tar.gz"

# Create a SageMaker model
sagemaker_session = sagemaker.Session()
model = Model(model_data=model_data,
              image_uri="your-custom-container-image", # Specify the Docker
              image URI
                  role="your-sagemaker-role",
                  sagemaker_session=sagemaker_session)

# Deploy the model as an endpoint
predictor = model.deploy(initial_instance_count=1,
                        instance_type="ml.g4dn.12xlarge")
```

```
from djl_python import Input, Output
import os
import torch
from transformers import pipeline, AutoModelForCausalLM, AutoTokenizer

predictor = None

def get_model(properties):
    model_name = properties['model_id']
    local_rank = int(os.getenv('LOCAL_RANK', '0'))
    dtype = torch.float16
    model = AutoModelForCausalLM.from_pretrained(model_name,
        low_cpu_mem_usage=True, torch_dtype=dtype)
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    generator = pipeline(task='text-generation', model=model,
        tokenizer=tokenizer, device=local_rank)
    return generator

def handle(inputs: Input) -> None:
    global predictor
    if not predictor:
        predictor = get_model(inputs.get_properties())

    if inputs.is_empty():
        # Model server makes an empty call to warmup the model on startup
        return None

    data = inputs.get_as_json()[0]['prompt']
    result = predictor(data, do_sample=True)
    return Output().add(result)
```


Limitations of Standalone LLMs

- Potential inaccuracies and hallucinations in generated content.
- The challenge of providing up-to-date information due to training data cut-off.
- Difficulty in handling domain-specific queries with general-purpose LLMs.
- Limited Contextual Understanding
- Ethical and Bias Issues
- Fine-tuning large LLMs require significant computational resources
- Handling of potentially sensitive data underscores the importance of stringent data governance

Introducing Retrieval-Augmented Generation (RAG)

□ What is RAG?

- Retrieval-Augmented Generation (RAG) is an advanced AI approach that combines the strengths of retrieval systems with generative models
- It aims to enhance the capabilities of LLMs by grounding generated responses in factual information retrieved from knowledge bases

□ How RAG Works?

- Retrieval Component
- Generative Component

□ Benefits of RAG

- Improved Accuracy
- Enhanced Contextual Relevance
- Overcoming Knowledge Cut-off

Implementation Steps for RAG

❑ Select a Knowledge Base

- ❑ Internal company database
- ❑ Ensure the knowledge base is comprehensive, up-to-date, and well-maintained

❑ Data Preparation

- ❑ Clean and preprocess the data in the knowledge base to ensure consistency and quality.
- ❑ Use scalable and efficient storage solutions such as **AWS OpenSearch**
- ❑ Index the data using appropriate techniques to facilitate efficient retrieval (**Vector database with FAISS(Facebook AI Similarity Search) Engine**)

❑ Develop the Retrieval System

- ❑ Implement a retrieval system that can efficiently search and retrieve relevant documents or data from the knowledge base

❑ Combine Retrieval with LLM

- ❑ Concatenate or format the retrieved data with the original query to provide context to the LLM

Cost Concerns and Reduction Strategies for LLMs

► High Computational Requirements

Model Name	Model ID	Max Total Tokens	Default instance type
Meta-Llama-3-8B	meta-textgeneration-llama-3-8b	8192	ml.g5.12xlarge
Meta-Llama-3-8B-Instruct	meta-textgeneration-llama-3-8b-instruct	8192	ml.g5.12xlarge
Meta-Llama-3-70B	meta-textgeneration-llama-3-70b	8192	ml.p4d.24xlarge
Meta-Llama-3-70B-Instruct	meta-textgeneration-llama-3-70b-instruct	8192	ml.p4d.24xlarge

Instance Type	GPUs	Cost per hour
ml.g5.12xlarge	4 NVIDIA A10G	7\$
ml.p4d.24xlarge	8 NVIDIA A100	37\$

► Storage and Data Management

► Knowledge base costs

► Operational and Maintenance Costs

► Maintenance of LLMs

► Availability

► Low latency

Cost Reduction Strategies

- ❑ **Use Pre-Trained Models by APIs and Services**
 - ❑ AWS Bedrock, OpenAI etc.
- ❑ **Leverage Foundation Models offered by cloud service providers**
 - ❑ AWS SageMaker, AWS Bedrock
- ❑ **Optimize Model Size**
 - ❑ Model Distillation
 - ❑ Quantization and Pruning
- ❑ **Efficient Resource Utilization**
 - ❑ Auto scaling, Asynchronous invocation, Spot instances, Reserved instances, Caching, Input Batching.
- ❑ **Data Management and Storage Solutions**
 - ❑ IVF-Flat/IVF PQ Indexing techniques, Compression
- ❑ **Model cascading**
 - ❑ Deploying a series of models with increasing complexity and computational cost

Improvisation of LLMs

Prompt engineering is about crafting inputs that guide the model towards the desired output

A good example of prompt construction

The following is text from a restaurant review:

Contextual information about the task.

"I finally got to check out Alessandro's Brilliant Pizza and it is now one of my favorite restaurants in Seattle. The dining room has a beautiful view over the Puget Sound but it was surprisingly not crowded. I ordered the fried Castelvetrano olives, a spicy Neapolitan-style pizza and a gnocchi dish. The olives were absolutely decadent, and the pizza came with a smoked mozzarella, which was delicious. The gnocchi was fresh and wonderful. The waitstaff were attentive, and overall the experience was lovely. I hope to return soon."

Reference text for the task.

Summarize the above restaurant review in one sentence.

Simple, clear and complete instructions.

Instructions placed at the end of the prompt.

The form of output is specifically described.

Text Classification Prompt

Prompt template for Anthropic Claude:

```
Human: {{classification task description}}  
<text>  
{{input text content to be classified}}  
</text>
```

```
Categories are:  
{{category name 1}}  
{{category name 2}}  
{{category name 3}}
```

```
Assistant: ""
```

(Source: Written by AWS)

User prompt:

Human: *Classify the given product description into given categories. Please output the category label in <output></output> tags.*

Here is the product description.

```
<text>  
Safe, made from child-friendly materials with smooth edges.  
Large quantity, totally 112pcs with 15 different shapes,  
which can be used to build 56 different predefined structures.  
Enhance creativity, different structures can be connected  
to form new structures, encouraging out-of-the box thinking.  
Enhance child-parent bonding, parents can play with their  
children together to foster social skills.  
</text>
```

```
Categories are:  
(1) Toys  
(2) Beauty and Health  
(3) Electronics
```

Assistant:

```
Output:  
<output>Toys</output>
```


Question-Answer prompt

Prompt template for Anthropic Claude:

"""

Human: {{Instruction}}
<text>
{{Text}}
<text>
{{Question}}

Assistant: """

User prompt:

Human: Read the following text inside <text></text>
XML tags, and then answer the question:

<text>
On November 12, 2020, the selection of the Weeknd to headline the show was announced; marking the first time a Canadian solo artist headlined the Super Bowl halftime show. When asked about preparations for the show, the Weeknd stated, "We've been really focusing on dialing in on the fans at home and making performances a cinematic experience, and we want to do that with the Super Bowl."

The performance featured a choir whose members were dressed in white and wore masks over their faces with glowing red eyes, and were standing within a backdrop of a neon cityscape. The performance opened with a white figure dressed the same as the choir being lowered into the backdrop where the choir was standing while singing "Call Out My Name". At this time, the Weeknd sat in a convertible against a skyline backdrop designed to resemble the Las Vegas Strip. For the next part of the performance, the backdrop then split open to reveal the Weeknd, who then performed "Starboy", followed by "The Hills".

Next, performing the song "Can't Feel My Face", the Weeknd traveled through a labyrinth constructed behind the stage, joined by dancers dressed in red blazers and black neckties similar to his, but with their faces covered with bandages, in keeping with the aesthetic of his fourth studio album After Hours (2020). The dancers would wear these bandages throughout the performance. In the labyrinth section of the performance, camerawork was visually unsteady. The next songs performed were "I Feel It Coming", "Save Your Tears", and "Earned It". For the "Earned It" performance, the Weeknd was accompanied by violinists. For the finale of the show, the Weeknd took to the field of the stadium with his dancers to perform "Blinding Lights". He and the dancers entered the field by performing "House of Balloons / Glass Table Girls". The performance ended with an array of fireworks.

</text>
Based on the text above, what songs did the Weeknd play at the Super Bowl halftime show?

Assistant:

Output:

Based on the text, the songs the Weeknd played at the Super Bowl halftime show were:

- Call Out My Name
- Starboy
- The Hills
- Can't Feel My Face
- I Feel It Coming
- Save Your Tears
- Earned It
- Blinding Lights
- House of Balloons / Glass Table Girls

Text Summarization Prompt

Prompt template for Anthropic Claude:
"""

Human: Please read the text:

*<text>
{{text context}}
</text>*

Summarize the text in {{length of summary, e.g., "one sentence" or "one paragraph"}}

Assistant: """

User prompt:

Human: Please read the text:

*<text>
In game theory, the Nash equilibrium, named after the mathematician John Nash, is the most common way to define the solution of a non-cooperative game involving two or more players. In a Nash equilibrium, each player is assumed to know the equilibrium strategies of the other players, and no one has anything to gain by changing only one's own strategy. The principle of Nash equilibrium dates back to the time of Cournot, who in 1838 applied it to competing firms choosing outputs.
</text>*

Summarize the text in one sentence.

Assistant:

Output:

The Nash equilibrium is a concept in game theory where each player has adopted a strategy that they have no incentive to deviate from given the other players' strategies.

Code generation Prompt

Prompt template for Anthropic Claude

"""

*Human: Write a function in {{programming language}} to
{{task or purpose}}*

Assistant: """

User prompt:

*Human: Write some python code that quickly approximates
the square root of a number.*

Assistant:

Output:

*Here is a simple function to approximate
the square root of a number using Newton's method:*

```
```python
def sqrt(n, iterations=10):
 x = n
 for i in range(iterations):
 x = (x + n/x) / 2
 return x
```
```

To use it:

```
```python
print(sqrt(25))
5.0

print(sqrt(15, iterations=20))
3.872983346207417
```
```


Leveraging LLMs as a Software Engineer and Tech Professional

- **Automated Code Generation:**
 - Use LLMs to generate boilerplate code, automate repetitive coding tasks, and suggest code completions.
 - **Example:** GitHub Copilot, powered by OpenAI Codex, Amazon Q can help write code snippets and entire functions based on comments and context within your codebase.
- **Code Review and Debugging:**
 - LLMs can assist in identifying bugs, suggesting fixes, and performing initial code reviews to enhance code quality.
 - **Example:** Tools like DeepCode and CodeGuru use AI to analyze code, provide recommendations, and detect potential vulnerabilities.
- **Documentation Generation:**
 - Automate the creation of comprehensive documentation for your codebase, including function descriptions, usage examples, and API documentation.
 - **Example:** LLMs can generate docstrings, README files, and API documentation from code comments and structure.
- **Natural Language Interfaces:**
 - Develop applications that allow users to interact with software using natural language commands and queries.
 - **Example:** Integrating LLMs into chatbots or voice assistants to perform tasks such as scheduling, querying databases, and controlling software applications.
- **Technical Support and Troubleshooting:**
 - Implement AI-driven chatbots and virtual assistants to provide first-level support, answer technical questions, and guide users through troubleshooting steps.
 - **Example:** AI-powered help desks can resolve common issues, reducing the workload on human support teams and improving response times.
- **Data Analysis and Insights:**
 - Use LLMs to analyze large datasets, generate reports, and extract meaningful insights from textual data such as logs and customer feedback.
 - **Example:** Leveraging LLMs for sentiment analysis, trend identification, and summarizing complex datasets into actionable insights.

Leveraging LLM In Infringement Detection

❑ Trademark and Copyright Violations

- ❑ Identifying unauthorized use of brand names, logos, and other intellectual properties

❑ Counterfeit Detection

- ❑ Recognizing subtle differences in genuine and fake product listings

❑ Obfuscation detection

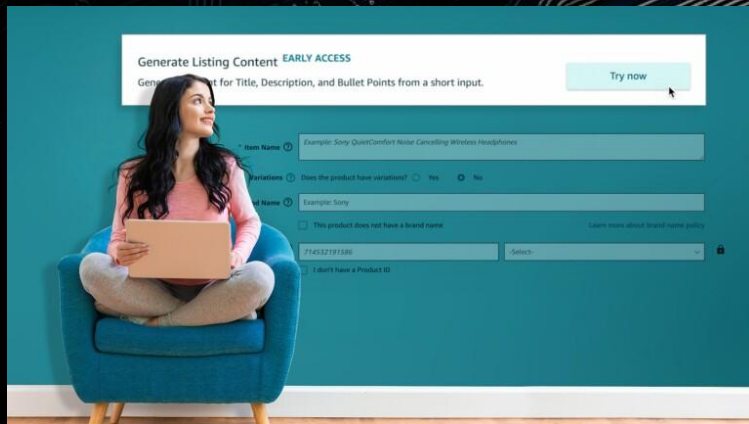
- ❑ Spotting subtle variations in brand names intended to bypass detection (e.g., "N1ke" instead of "Nike")

❑ Behavioral Analysis

- ❑ Analyzing seller behavior and history to identify potential infringer

Real World Examples

- ▶ Review highlights on product listings by Amazon
- ▶ Creation of compelling product titles and descriptions for Product data on Amazon websites
- ▶ Amazon Pharmacy staff answer questions more quickly because it scours internal wikis and other sources of information, and then summarizes the findings
- ▶ Reduced the human audits for detecting infringements by 80% for famous brands like Apple etc





**THANK
YOU!**