Summary:

- Development approaches with Databricks include Git setup, project setup, and CI-CD flow.
- Orchestration involves Databricks workflows, Airflow integration, and AWS account configurations.
- Workspace configurations focus on security measures like S3 encryption and workspace login.
- Data storage is organized by catalogs, with separate physical storage per catalog.
- ML model deployment involves training, evaluation, and deployment in staging and production environments.
- Unity Catalog provides centralized metadata management and access controls across Databricks workspaces.
- Cluster policies enforce configurations for instance types, DBR versions, and pool usage.

Can you provide me more elaborate summarization on how I can crack an interview on asking me a question?

Summary:

To excel in an interview when asked about cracking a question, focus on showcasing problem-solving skills, communication abilities, and technical knowledge. Be prepared to discuss your approach to breaking down complex problems, demonstrating logical thinking and creativity. Emphasize your experience in handling challenging situations, collaborating with team members, and effectively communicating solutions. Highlight your technical expertise, practical skills, and willingness to learn and adapt. Practice problem-solving exercises, coding challenges, and mock interviews to enhance your problem-solving abilities and boost your confidence. Remember to stay calm, articulate your thought process clearly, and showcase your passion for problem-solving during the interview.

I am being asked in an interview on whether how did you imply and deploy and train models in tabcorp? give me the answer based on the input that I have provided in PDF?

Summary:

In Tabcorp, model training is conducted in the development environment using Databricks. The trained model artifact is then moved to the staging environment for validation checks before deployment to the production environment. Two deployment patterns are used: "Deploy Model" and "Deploy Code". The "Deploy Model" pattern involves training models in the development environment on production data, logging experiments in the MLFlow tracking server, and registering the final model in UC under the appropriate schema in the production catalog. The "Deploy Code" pattern requires a separate path for deploying ancillary code, with pipelines tested in staging before deployment to production. The MLOps pipeline at Tabcorp caters to both deployment patterns, with 70% of models fitting the "Deploy Model" pattern and 30% expected to fit the "Deploy Code" pattern.

Also, Provide me the MLOPS framework involved?

MLOps Framework Involved:

The MLOps framework involved in Tabcorp includes the following key components and processes:

1. **Model Training and Deployment:** Models are trained in the development environment using Databricks. The trained models are then moved to the staging environment for validation before deployment to the production environment.
2. **Deployment Patterns:** Two deployment patterns are utilized - "Deploy Model" and "Deploy Code". The "Deploy Model" pattern involves training models on production data, logging experiments in the MLFlow tracking server, and registering the final model in UC under the appropriate schema in the production catalog. The "Deploy Code" pattern requires separate testing of ancillary code in staging before deployment to production.
3. **Unity Catalog (UC):** Centralized metadata management system used for registering ML models and organizing features. Models are registered with a three-level namespace in UC.
4. **MLFlow Tracking Server:** Used for logging model training experiments, parameters, hyperparameters, and artifacts during the model development process.
5. **Workspace Design:** Follows a three-workspace design pattern initially, which can be expanded to dedicated dev/stage/prod workspaces per Line of Business (LOB) as needed.
6. **Monitoring and Evaluation:** Lakehouse monitoring pipelines are turned on in production to monitor model metrics. Performance evaluation of models on production data is conducted before transitioning to production.
7. **Compliance Checks:** Conducted on models in the staging environment to ensure adherence to regulatory requirements and quality standards.
8. **Feature Engineering:** Utilizes SQL and Python UDFs registered within UC for computing on-demand features used in model training, batch, and real-time inference.
9. **Data Access:** Data scientists have read-only access to the production catalog in the development workspace for building EDA and models on production data.
10. **Model Transition:** Models are transitioned from challenger to champion status upon deployment in the production environment.

This MLOps framework at Tabcorp ensures a structured and automated process for model deployment, governance, and monitoring across different environments.

based on the above PDF, provide me all the interview question and answers that can be created and asked in an interview?

Interview Questions and Answers:

1. **Question:** What is the Solution Architecture proposed for setting up a lake house architecture at TabCorp? **Answer:** The Solution Architecture involves storing data in Delta format for high performance querying, time travel capabilities, fault recovery, and ACID properties. Databricks Unity Catalog is used for workspace and metadata management, with UC integration for user provisioning and group privileges.
2. **Question:** How is the Workspace Design structured at TabCorp for data engineering and analytics users? **Answer:** The Workspace Design follows a three-workspace approach initially, with the flexibility to migrate to a more LOB-centric architecture in the future. It minimizes management overhead while supporting a proper SDLC.
3. **Question:** What are the different Cluster Policies enforced in the workspace? **Answer:** Cluster Policies are JSON documents used to enforce configurations like instance types, DBR versions, standard tags, and pool usage across the workspace. They ensure consistency and compliance with enterprise standards.

4. **Question:** Explain the Data Ingestion Patterns followed at TabCorp. **Answer:** Data ingestion at TabCorp follows the medallion architecture with Bronze, Silver, and Gold layers. Bronze represents inbound data with minimal transformations, Silver includes validated data, and Gold is the final processed data layer.
5. **Question:** How are ML models trained and deployed in TabCorp's MLOps framework? **Answer:** Models are trained in the development environment using Databricks, validated in staging, and deployed to production. Two deployment patterns, "Deploy Model" and "Deploy Code," are used based on the model type and requirements.
6. **Question:** What is the role of Unity Catalog in managing ML models at TabCorp? **Answer:** Unity Catalog serves as a centralized metadata management system for registering ML models, organizing features, and ensuring data isolation. Models are registered under appropriate schemas in the production catalog.
7. **Question:** How does TabCorp handle workspace provisioning and management? **Answer:** Workspace provisioning involves creating workspaces for different user groups, managing access controls, and ensuring secure configurations. Workspace artifacts are provisioned based on user roles and responsibilities.
8. **Question:** What are the key components of the CI/CD pipeline design at TabCorp? **Answer:** The CI/CD pipeline design includes Databricks integration, development approaches, Git setup, and project setup. It focuses on automating workflows, version control, and deployment processes for efficient development.
9. **Question:** How does TabCorp ensure security in its architecture? **Answer:** Security measures include serverless architecture, audit logs, bucket and gateway endpoint policies, customer-managed keys, and workspace configurations. Private Link and Security Groups are used to enhance network security.
10. **Question:** Describe the process of model monitoring and evaluation in TabCorp's MLOps framework. **Answer:** Models are monitored in production using lakehouse monitoring pipelines to track performance metrics. Evaluation is conducted in staging to ensure model quality before transitioning to production.

provide me code based examples or whiteboard based examples and explanations for the same? Also provide me more question and answers please

Code-Based Example:

Whiteboard-Based Example:

Explanation: The whiteboard example illustrates the data flow from the Bronze layer to the Silver layer in TabCorp's data architecture. Raw data is ingested into the Bronze layer, processed, and stored in the Silver layer for further analysis and modeling.

More Interview Questions and Answers:

1. **Question:** How does TabCorp handle data lineage and metadata management? **Answer:** Data lineage is tracked using Databricks Delta, capturing the flow of data transformations. Metadata management is centralized through the Unity Catalog, ensuring data provenance and governance.
2. **Question:** Can you explain the process of model versioning and tracking in TabCorp's MLOps framework? **Answer:** Models are versioned using MLFlow, tracking parameters,

hyperparameters, and artifacts for each model iteration. Model versions are registered in the Unity Catalog for traceability.

3. **Question:** What are the key considerations for scaling ML models in a production environment at TabCorp? **Answer:** Scaling considerations include resource allocation, cluster configurations, data partitioning, and monitoring for performance optimization and cost efficiency.
4. **Question:** How does TabCorp ensure data security and compliance in its data architecture? **Answer:** Data security measures include encryption at rest and in transit, role-based access controls, data masking, and compliance checks to adhere to regulatory requirements.
5. **Question:** Explain the process of model retraining and reevaluation in TabCorp's MLOps pipeline. **Answer:** Models are retrained periodically using updated data, with reevaluation conducted to assess model performance against benchmarks. MLFlow experiments track retraining iterations for comparison.
6. **Question:** What role does CI/CD play in the deployment of ML models at TabCorp? **Answer:** CI/CD automates the deployment process, ensuring consistency and reliability in model deployment. It facilitates version control, testing, and deployment of models across environments.
7. **Question:** How does TabCorp handle feature engineering and preprocessing in its ML workflows? **Answer:** Feature engineering involves creating new features from raw data, while preprocessing includes data cleaning, normalization, and transformation to prepare data for model training.
8. **Question:** Can you explain the process of model evaluation and performance monitoring in TabCorp's MLOps framework? **Answer:** Model evaluation involves assessing model accuracy, precision, recall, and other metrics using validation data. Performance monitoring tracks model metrics in production for continuous improvement.
9. **Question:** What are the best practices for model deployment and rollback strategies at TabCorp? **Answer:** Best practices include A/B testing for model comparison, canary deployments for gradual rollout, and rollback strategies to revert to previous model versions in case of issues.
10. **Question:** How does TabCorp ensure collaboration and knowledge sharing among data science teams? **Answer:** Collaboration is fostered through shared workspaces, version control systems, code reviews, and knowledge sharing sessions to promote transparency and learning across teams.

Provide me more technical question and answers that we can get from PDF< so that one can answer in interview, may be upto 25 question and answers

More Technical Interview Questions and Answers:

1. **Question:** What are the advantages of storing data in Delta format for querying in a lake house architecture? **Answer:** Storing data in Delta format provides high performance querying, time travel capabilities, fault recovery, and ACID properties for data consistency and reliability.
2. **Question:** How does Databricks Unity Catalog facilitate metadata management and governance in TabCorp's architecture? **Answer:** Databricks Unity Catalog centralizes metadata management for workspaces, catalogs, schemas, and tables, ensuring data lineage, provenance, and governance.
3. **Question:** What is the significance of using Cluster Policies in enforcing configurations across the workspace? **Answer:** Cluster Policies enforce consistent configurations for instance types,

DBR versions, standard tags, and pool usage, ensuring compliance with enterprise standards and best practices.

4. **Question:** How does TabCorp handle data level access controls within the Unity Catalog to restrict access to production data? **Answer:** Data level access controls within the Unity Catalog restrict access to production data at lower environments, ensuring data security and compliance with access policies.
5. **Question:** Can you explain the process of workspace provisioning and management using Terraform and the Databricks Terraform provider? **Answer:** Workspace provisioning involves using Terraform with the Databricks Terraform provider to manage and govern Databricks workspaces, ensuring consistent and automated workspace configurations.
6. **Question:** What are the key considerations for scaling workloads in Databricks clusters at TabCorp? **Answer:** Key considerations for scaling workloads include resource allocation, instance types, cluster configurations, and monitoring for optimal performance and cost efficiency.
7. **Question:** How does TabCorp ensure data integrity and consistency in the Delta Lake architecture? **Answer:** Data integrity and consistency are maintained through ACID properties in Delta Lake, ensuring atomicity, consistency, isolation, and durability for reliable data operations.
8. **Question:** What role does MLFlow Tracking Server play in tracking model training experiments and artifacts in TabCorp's MLOps framework? **Answer:** MLFlow Tracking Server logs model training experiments, parameters, hyperparameters, and artifacts, providing visibility and traceability for model development and versioning.
9. **Question:** How does TabCorp handle model transition from challenger to champion status in the production environment? **Answer:** Models transition from challenger to champion status upon successful deployment in the production environment, undergoing validation and performance evaluation before promotion.
10. **Question:** Explain the process of workspace design for Data Engineering and Data Science & Analytics users at TabCorp. **Answer:** Workspace design includes separate prod and non-prod workspaces, with data level access controls applied at the catalog level within UC to restrict access to production data.
11. **Question:** What are the benefits of using SQL Warehouses for business users and data analysts in TabCorp's architecture? **Answer:** SQL Warehouses provide a compute resource for running SQL commands on data objects, supporting queries, visualization widgets, dashboards, and backend compute for reporting engines like PowerBI or Tableau.
12. **Question:** How does TabCorp handle workspace limits and considerations to ensure efficient job execution and resource management? **Answer:** TabCorp monitors workspace limits for jobs, simultaneous jobs, users, groups, and clusters to optimize resource usage, prevent bottlenecks, and ensure smooth job execution.
13. **Question:** What are the best practices for managing dormant jobs and preventing workspace limit issues in Databricks workspaces? **Answer:** Best practices include conducting regular inventory of actively used jobs, cleaning up unused jobs, and optimizing job configurations to prevent hitting workspace limits due to dormant jobs.
14. **Question:** How does TabCorp ensure efficient trigger mechanisms for workflow jobs based on cron schedule, continuous mode, or file arrival triggers? **Answer:** TabCorp leverages efficient trigger mechanisms for workflow jobs, including cron schedule, continuous mode, and file arrival triggers monitoring cloud storage paths for new files, ensuring timely job execution and automation.

15. **Question:** Can you explain the process of workspace artifact provisioning based on user roles and responsibilities in TabCorp's architecture? **Answer:** Workspace artifact provisioning involves assigning appropriate privileges and access controls based on user roles and responsibilities, ensuring secure and controlled access to workspace resources and data.
16. **Question:** What are the key components of the CI/CD pipeline design with Databricks at TabCorp? **Answer:** The CI/CD pipeline design includes Databricks integration, development approaches, Git setup, and project setup, focusing on automating workflows, version control, and deployment processes for efficient development and collaboration.
17. **Question:** How does TabCorp handle infrastructure architecture, including AWS accounts, IAM roles, and network security? **Answer:** TabCorp manages infrastructure architecture with dedicated AWS accounts, IAM roles for access control, private link for secure communication, security groups for network isolation, and compliance checks for data security and governance.
18. **Question:** What are the security measures implemented in TabCorp's architecture to ensure data protection and compliance? **Answer:** Security measures include serverless architecture for scalability and cost efficiency, audit logs for tracking user activities, bucket and gateway endpoint policies for secure data access, customer-managed keys for encryption, and workspace configurations for access control and compliance.
19. **Question:** How does TabCorp handle CI/CD with Databricks to streamline development, testing, and deployment processes? **Answer:** TabCorp integrates CI/CD with Databricks to automate workflows, version control, and deployment, ensuring consistency, reliability, and efficiency in the development lifecycle.
20. **Question:** What are the development approaches followed with Databricks at TabCorp for building and deploying data solutions? **Answer:** Development approaches include iterative development, testing in staging environments, collaboration with team members, and continuous integration and deployment to ensure quality and reliability in data solutions.

These technical interview questions and answers cover various aspects of TabCorp's architecture, MLOps framework, workspace design, security measures, CI/CD pipeline, and infrastructure management, providing insights into the company's data engineering and analytics practices.

Sure, here is an example of how you can fine-tune GPT-2 using PyTorch and the Hugging Face Transformers library.

First, you need to import the necessary modules:

python

EditFull ScreenCopy code

1import torch

2from transformers import GPT2Tokenizer, GPT2LMHeadModel

3import numpy as np

4import logging

5import warnings

```

device = 'cpu'
if torch.cuda.is_available():
    device = 'cuda'
```

Next, you can load the pre-trained GPT-2 model and tokenizer:

python

EditFull ScreenCopy code

```
tokenizer = GPT2Tokenizer.from_pretrained('gpt2-medium')
model = GPT2LMHeadModel.from_pretrained('gpt2-medium')
model = model.to(device)
```

Then, you can define a function to sample from the top k most probable next tokens:

python

EditFull ScreenCopy code

```
def choose_from_top(probs, n=5):
    ind = np.argpartition(probs, -n)[-n:]
    top_prob = probs[ind]
    top_prob = top_prob / np.sum(top_prob) # Normalize
    choice = np.random.choice(n, 1, p = top_prob)
    token_id = ind[choice][0]
    return int(token_id)
```

Now, you can create a PyTorch Dataset for your fine-tuning data:

python

EditFull ScreenCopy code

```
from torch.utils.data import Dataset, DataLoader
import os
```

```
import json
import csv

class JokesDataset(Dataset):
    def __init__(self, jokes_dataset_path = 'jokes_data/'):
        super().__init__()

        short_jokes_path = os.path.join(jokes_dataset_path, 'shortjokes.csv')

        self.joke_list = []
        self.end_of_text_token = "<|endoftext|>"

        with open(short_jokes_path) as csv_file:
            csv_reader = csv.reader(csv_file, delimiter=',')
            x = 0
            for row in csv_reader:
                joke_str = f"JOKE:{row[1]}{self.end_of_text_token}"
                self.joke_list.append(joke_str)

    def __len__(self):
        return len(self.joke_list)

    def __getitem__(self, item):
        return self.joke_list[item]
```

And here's an example of how you can train the model:

python

EditFull ScreenCopy code

```
BATCH_SIZE = 16
EPOCHS = 5
LEARNING_RATE = 3e-5
WARMUP_STEPS = 5000
MAX_SEQ_LEN = 400

from transformers import AdamW, WarmupLinearSchedule

optimizer = AdamW(model.parameters(), lr=LEARNING_RATE)
scheduler = WarmupLinearSchedule(optimizer, warmup_steps=WARMUP_STEPS, t_total = -1)

proc_seq_count = 0
sum_loss = 0.0
batch_count = 0

model.train()
for epoch in range(EPOCHS):
    for idx,joke in enumerate(joke_loader):
        joke_tens = torch.tensor(tokenizer.encode(joke[0])).unsqueeze(0).to(device)

        if joke_tens.size()[1] > MAX_SEQ_LEN:
            continue

        if not torch.is_tensor(tmp_jokes_tens):
            tmp_jokes_tens = joke_tens
```

```
        continue
    else:
        if (tmp_jokes_tens + joke_tens).size()[1] > MAX_SEQ_LEN:
            output = model(tmp_jokes_tens)[0]
            loss_fct = torch.nn.CrossEntropyLoss()
            loss = loss_fct(output.reshape(-1, output.size(-1)), tmp_targets.reshape(-1))
            sum_loss += loss.item()
            proc_seq
```

Is this conversation helpful so far?

Can you provide me the UML diagram steps for the same?

Edit

Sources: discuss.huggingface.co (1) towardsdatascience.com (2) lucidbeaming.com (3) kaggle.com (4)

I'm sorry for any confusion, but creating an UML diagram for the process of fine-tuning GPT-2 is not common or standardized, as it's typically represented in code and not in a flowchart format. However, I can provide you with a high-level overview of the steps involved in fine-tuning GPT-2 using PyTorch and the Hugging Face Transformers library:

1. **Import necessary modules**: Import required libraries and modules, such as PyTorch, the GPT2LMHeadModel, GPT2Tokenizer, and the dataset for fine-tuning.
2. **Load the pre-trained GPT-2 model and tokenizer**: Instantiate the GPT2LMHeadModel and GPT2Tokenizer using the 'gpt2-medium' or other pre-trained model configurations.
3. **Prepare the fine-tuning dataset**: Load the dataset and preprocess it by encoding the text inputs using the tokenizer and splitting the data into input and target sequences.
4. **Create a PyTorch DataLoader**: Wrap your dataset with a PyTorch DataLoader for easier batch processing.
5. **Define a training function**: Implement a function to train the model on the fine-tuning dataset. This function should include the following steps:
   - Loop through the epochs
   - Loop through the batches in the DataLoader
   - Forward pass the input batch through the model
   - Compute the loss

- Backward pass and update model parameters
- Log metrics and save checkpoints

6. **Initialize the optimizer and scheduler**: Instantiate the optimizer (e.g., AdamW) and learning rate scheduler (e.g., WarmupLinearSchedule) for training.
7. **Train the model**: Call the training function with the appropriate parameters, such as the number of training steps, learning rate, and other hyperparameters.

These steps should help you understand the high-level process of fine-tuning GPT-2 using PyTorch and Hugging Face Transformers. However, creating an UML diagram for this process is not a conventional or standard approach.

Is this conversation helpful so far?

Can you provide the previous two results in markdown?

Edit

Sources: community.openai.com (1) community.openai.com (2) towardsdatascience.com (3) modeldifferently.com (4)

Here are the previous two results in markdown format:

Question: Can you show me (with code) how to fine-tune GPT-2?

<details> <summary>Click to expand the answer</summary>

python

EditFull ScreenCopy code

```
import torch
from transformers import GPT2Tokenizer, GPT2LMHeadModel

tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2LMHeadModel.from_pretrained('gpt2')

# Prepare your fine-tuning dataset
# ...

```

```
# Move the model to the device (GPU if available)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)

# Fine-tune the model
# ...
```

</details>

Question: Can you provide me the UML diagram steps for the same?

<details> <summary>Click to expand the answer</summary>

As I mentioned earlier, creating an UML diagram for this process is not common or standardized. However, I can provide you with a high-level overview of the steps involved in fine-tuning GPT-2 using PyTorch and the Hugging Face Transformers library:

1. **Import necessary modules**
2. **Load the pre-trained GPT-2 model and tokenizer**
3. **Prepare the fine-tuning dataset**
4. **Create a PyTorch DataLoader**
5. **Define a training function**
6. **Initialize the optimizer and scheduler**
7. **Train the model**

These steps should help you understand the high-level process of fine-tuning GPT-2 using PyTorch and Hugging Face Transformers. However, creating an UML diagram for this process is not a conventional or standard approach.

</details>

The above answers are provided in markdown format, with each question and answer separated by a horizontal rule (<hr/>). The code snippets within the answers are displayed using the markdown <details> and <summary> tags, allowing the user to expand and view the code as needed.

Is this conversation helpful so far?