Transformer&Meta Factory

목표

- Transformer의 데이터베이스 이해
- Attention mechanism의 성공적인 활용
- Trouble Shooting

▪ 1. 프로젝트 계획 개요

▫ 1. 목표설정

> 공공 제조 데이터를 활용한 데이터베이스 구축과 Transformer AI 모델을 사용한 메타팩토리 구축

- **데이터베이스 구축**: 공공 제조 데이터를 효율적으로 저장 및 관리할 수 있는 데이터베이스 구축.
- **AI 모델 적용**: Transformer 모델을 활용해 메타팩토리 내에서 기계 간 관계를 분석하고, 병렬 처리를 통해 생산 공정 최적화 및 문제 해결 기능 추가.

▫ 2. 기술 스택 정의

- **개발언어**: Python, SQL
- **데이터베이스**: Oracle RDBMS 또는 MySQL
- **AI 모델**: Hugging Face Transformers 라이브러리 (PyTorch)
- **개발 도구**: Visual Studio Code

▪2. 단계별 진행 계획

▫ 1. 데이터 수집 및 전처리

- **공공 제조 데이터 수집**: 필요로 하는 데이터(예: 생산 공정, 품질 지표, 기계 동작 데이터)를 정부 포털 또는 관련 API를 통해 수집.
- **데이터 정제 및 변환**: 데이터의 정합성 및 일관성을 위해 결측치 처리, 이상치 제거, 데이터 타입 변환 등을 진행.
- **데이터 저장**: 전처리된 데이터를 CSV 파일이나 초기에는 SQLite에 임시 저장 후 데이터베이스 구조 검토.

▫ 2. 데이터베이스 설계 및 구축

- **RD 설계**: 기계, 생산 공정, 품질 지표 등 엔티티와 관계성 정의.
- **데이터베이스 스키마 정의**: ERD에 따라 테이블 생성, 주요 필드 및 외래 키 지정.
- **인덱싱 및 최적화**: 데이터 조회 성능 향상을 위해 인덱스 추가 및 정규화 진행.
- **데이터베이스 구축 및 데이터 로드**: Python의 `SQLAlchemy`와 같은 ORM을 사용해 데이터 삽입 자동화.

▫ 3. Transformer 모델 설계 및 학습

- **Transformer 구조 이해 및 학습**: 메타팩토리 내 각 기계의 데이터를 학습해 관계성을 파악하는 Attention 메커니즘 활용.
- **병렬처리 설정**: 기계들의 실시간 관계성을 파악하기 위한 병렬 연산 최적화.
- **모델 평가 및 최적화**: 모델의 정확성과 효율성 평가, 하이퍼파라미터 튜닝 등 성능 개선 진행.

▫ 4. 생산 공정 시뮬레이션 및 트러블 슈팅 기능 구현

- **메타팩토리 내 데이터 흐름 시뮬레이션**: 가상의 시나리오 설정 후, 기계 간 데이터 흐름 테스트.
- **트러블 슈팅 알고리즘 구현**: 기계 간 관계성 데이터와 과거 오류 데이터를 바탕으로 Transformer가 자동으로 문제 원인 추론.
- **문제 해결책 제공**: Transformer 모델이 파악한 문제에 대한 해결책 제시 및 관련 데이터를 관리자에게 시각화.

▫ 5. 평가 및 성능 개선

- **결과 분석**: 구축한 시스템의 정확도, 효율성, 안정성을 검토해 평가 지표 수립.

- **고도화 및 확장 가능성 검토**: 필요 시 SQL 데이터베이스로의 전환 및 데이터셋 확장을 고려.

▪ 3. 진행

> 목표를 이루기 위해 공공 제조 데이터를 바탕으로 생산 관리 시스템을 구축하고 트러블 슈팅 자동화를 구현하는 프로젝트. 각 단계에서 사용할 데이터 주제와 트러블 유형, 대처 방식 등을 구체적으로 정리해서 진행할 것.
>
> 여기서는 **제조 기계의 작동 상태 및 온도 데이터**를 주요 데이터로 사용하고, **과열**, **고장** 등의 트러블이 발생했을 때 이를 탐지하고 해결 방안을 찾는 시스템을 구축할 것임. Python과 SQL로 MySQL에 데이터베이스를 구축하고, Pytorch를 사용해 Transformer 모델을 활용하여 데이터 패턴을 학습하고 분석할 것.

▫ 1. 프로젝트 기획: 데이터 주제 및 트러블 유형

1. **주제**: 제조업 공정에서 기계 작동 상태 및 온도 데이터를 수집하고, 기계 과열 또는 고장 시 자동으로 탐지해 대처하는 시스템 구축.
2. **트러블 유형 및 대처**:
    - **과열(Overheat)**: 기계의 온도가 80도 이상일 경우, 과열로 판별하고 경고 메시지를 출력함.
    - **고장(Malfunction)**: 기계 상태가 “Stopped”로 일정 시간 유지되면 고장으로 간주하고 경고 메시지 및 유지보수 요청을 출력함.

▫ 2. 데이터 준비 및 전처리

2.1 공공 제조 데이터 수집 및 파일 준비

1. **가상 데이터 생성**:
    - 기계 ID, 작동 상태, 온도, 타임스탬프를 포함한 CSV 파일(`manufacturing_data.csv`)을 생성함.
        - 여기서 사용할 주요 컬럼:

- `machine_id`: 기계 고유 ID
- `status`: 작동 상태 (`Running`, `Stopped`, `Maintenance`)
- `temperature`: 기계 온도 (°C)
- `timestamp`: 데이터 수집 시간

2. **CSV 파일 샘플 생성 코드**

```
import pandas as pd
import random
from datetime import datetime, timedelta

# 샘플 데이터 생성
data = []
machine_ids = ['A001', 'A002', 'B001', 'B002', 'C001']

for _ in range(500):
    machine_id = random.choice(machine_ids)
    status = random.choice(['Running', 'Stopped', 'Maintenanc
    temperature = round(random.uniform(50, 100), 2)
    timestamp = datetime.now() - timedelta(minutes=random.ran
    data.append([machine_id, status, temperature, timestamp])

# DataFrame 생성 및 CSV 파일로 저장
df = pd.DataFrame(data, columns=['machine_id', 'status', 'tem
df.to_csv('manufacturing_data.csv', index=False)
```

2.2 데이터 전처리

- Python의 Pandas 라이브러리를 사용하여 데이터 전처리 작업을 수행하겠음.
- 결측치 처리, 데이터 타입 변환, 이상치 제거 작업 진행함.

```
# CSV 파일 로드
df = pd.read_csv('manufacturing_data.csv')

# 결측치 처리 (온도가 없으면 평균값으로 대체)
df['temperature'].fillna(df['temperature'].mean(), inplace=Tr
```

```
# 데이터 타입 변환
df['timestamp'] = pd.to_datetime(df['timestamp'])

# 이상치 처리 (온도 범위 제한)
df = df[(df['temperature'] >= 0) & (df['temperature'] <= 150)
```

▫ 3. MySQL 데이터베이스 구축 및 데이터 삽입

3.1 MySQL 데이터베이스 생성 및 테이블 구성

1. **데이터베이스 생성**: MySQL에서 새로운 데이터베이스 생성

```
CREATE DATABASE manufacturing_db;
USE manufacturing_db;
```

2. **테이블 생성**

- `Machines`: 기계 정보 저장
- `Operations`: 기계 작동 상태 및 온도 정보 저장
- `Failures`: 트러블 발생 시 로그 기록

```
CREATE TABLE Machines (
    machine_id VARCHAR(10) PRIMARY KEY,
    name VARCHAR(50),
    type VARCHAR(50),
    manufacturer VARCHAR(50)
);

CREATE TABLE Operations (
    operation_id INT PRIMARY KEY AUTO_INCREMENT,
    machine_id VARCHAR(10),
    timestamp DATETIME,
    status VARCHAR(50),
    temperature DECIMAL(5,2),
    FOREIGN KEY (machine_id) REFERENCES Machines(machine_i
);

CREATE TABLE Failures (
```

```
    failure_id INT PRIMARY KEY AUTO_INCREMENT,
    machine_id VARCHAR(10),
    timestamp DATETIME,
    error_code VARCHAR(20),
    description TEXT,
    FOREIGN KEY (machine_id) REFERENCES Machines(machine_i
);
```

3.2 Python으로 MySQL에 데이터 삽입

1. **Python의** `pymysql` **사용**하여 전처리된 데이터를 `Operations` 테이블에 삽입.

```
import pymysql
import pandas as pd

# 데이터 로드
df = pd.read_csv('manufacturing_data.csv')

# MySQL 연결
conn = pymysql.connect(
    host='localhost',
    user='username',
    password='password',
    database='manufacturing_db'
)
cursor = conn.cursor()

# Operations 테이블에 데이터 삽입
for _, row in df.iterrows():
    sql = '''
    INSERT INTO Operations (machine_id, timestamp, status,
    VALUES (%s, %s, %s, %s)
    '''
    cursor.execute(sql, (row['machine_id'], row['timestamp

conn.commit()
cursor.close()
conn.close()
```

▫ 4. Transformer 모델 학습 및 트러블 슈팅 기능 구현

4.1 Pytorch를 사용한 Transformer 모델 학습 준비

1. **Transformer 모델 정의**:
   - 제조 데이터의 작동 패턴 학습에 BERT 모델을 활용하겠음.
   - 상태 및 온도 데이터에서 패턴을 학습하여 트러블 상황을 예측하게 함.

```
from transformers import BertModel, BertTokenizer
import torch

tokenizer = BertTokenizer.from_pretrained('bert-base-uncas
model = BertModel.from_pretrained('bert-base-uncased')

# 예제 텍스트 인코딩
inputs = tokenizer("Machine A001 is overheating.", return_
outputs = model(**inputs)
attention = outputs.attentions
```

4.2 트러블 슈팅 함수 구현

1. **과열 탐지 및 경고**

```
def troubleshoot(machine_id, status, temperature):
    if temperature > 80:
        print(f"Warning: {machine_id}가 과열 상태임. 즉시 점검
    elif status == 'Stopped':
        print(f"Warning: {machine_id}가 정지 상태임. 고장 가능성
    else:
        print(f"{machine_id} 상태 양호.")
```

2. **트러블 로그 기록**
   - 과열 및 정지 상태를 `Failures` 테이블에 기록함.

```
for _, row in df.iterrows():
    troubleshoot(row['machine_id'], row['status'], row['te
    if row['temperature'] > 80 or row['status'] == 'Stoppe
        sql = '''
```

```
        INSERT INTO Failures (machine_id, timestamp, error
        VALUES (%s, %s, %s, %s)
        '''
        error_code = "OVERHEAT" if row['temperature'] > 80
        description = "Overheat detected" if row['temperat
        cursor.execute(sql, (row['machine_id'], row['times

conn.commit()
```

▪5. 성능 평가 및 개선

1. **모델 평가**: 온도 상승 패턴 학습 및 고장 예측 성능을 평가하기 위해 예측 정확도를 확인함.
2. **데이터 시각화 및 분석**: 시간에 따른 온도 변화 및 고장 발생 빈도 시각화.

```
import matplotlib.pyplot as plt

df.plot(x='timestamp', y='temperature', title='Temperature
plt.xlabel('Time')
plt.ylabel('Temperature')
plt.show()
```