INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Great Learning

POWER AHEAD

CAPSTONE PROJECT

Project name:

NEWS ARTICLES CLASSIFIER

INDEX

1. PROJECT OBJECTIVE ____ Page 1
2. PROBLEM STATEMENT ____ Page 1
3. PROJECT SETUP ____ Page 2
4. INFRASTRUCTURE & DEPLOYMENT ____ Page 4
5. CAPSTONE MILESTONE ____ Page 5

1. OBJECTIVE

1. Design complete solution to demonstrate end-to-end pipeline development and manage a machine/ deep learning project
2. Develop a understanding of all stages of a machine learning project lifecycle
3. Demonstrate understanding of challenges encountered during the project development and provide ways to tackle them
4. Showcase understanding of software engineering best practices while developing the project

2. PROBLEM STATEMENT

Classify News Articles into categories - With information overload today users are inundated with news articles of all topics, even the ones which may not be relevant to users. Design a system which can classify incoming news articles and appropriately tag the corresponding category. Develop a data pipeline which includes the all the following stages of Machine Learning Project Life Cycle –

1. Data Ingestion
2. Data Preparation
3. Data segregation & Model Training
4. Model Deployment
5. Model Prediction

3. PROJECT SETUP

Stages are elaborated as follows –

1. DATA INGESTION

Project Name - data-ingestion-service

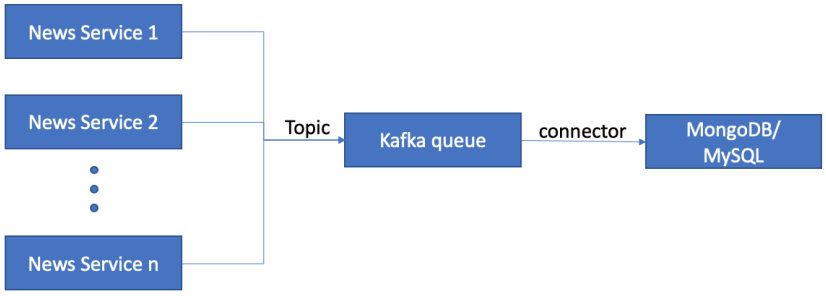
The objective of this project is to source new data to re-train the model.
This project should connect to at least 2 different news sources. Ingestion could be done either as follows:

- Via REST APIs (search news APIs available on rapidAPI, for example https://rapidapi.com/newscatcher-api-newscatcher-api-default/api/free-news/
- Via RSS feed (For e.g. by using BeautifulSoup to scrap RSS feed)
- Create your own mock news generator service

Please ensure that the collected news has "category" or "topic" as a part of raw data irrespective of the source. Data Cleaning - Since different news sources could have different response formats, transform the API response for all news sources to same raw format, which should include following fields –

- title
- date/ time
- summary
- topic/ category
- source

Scheduling - Schedule all news source services to collect news using a process scheduler (every few mins/ hours), or with custom thread management code.
Data Storage – Use a Publisher – Subscriber model to collect data and push it to the store. Create a Kafka topic and use it to queue cleaned responses from all sources. The sink for the stream should be MySQL/ MongoDB.



2. DATA PREPARATION (PRE-PROCESSING), SEGREGATION AND MODEL TRAINING

Project Name – model-training-service

The objective of this project is to trigger model re-training and deployment on-demand. Over a period, the performance of models degrades, and it becomes important to retrain them.

Before using the raw data can be used for model training/ retraining it needs to be preprocessed to relevant structure.

- Load the data from "raw_data" source (MySQL/ MongoDB) into Spark by using relevant connector for PySpark
- Perform data cleaning and preprocessing, followed by segregation to train and test datasets. (Necessary pre-training steps to be shared with participants)
- Perform model re-training (a pre-trained model along with code to retrain the model could be provided to participants).
- Highlight the explainability of the AIML model by white boxing which patterns learnt by the training data arrives to a particular class [News topic].
- Serialise the model and save it to a location (or push the model to a model registry like MLFlow), which can later be used for model retrieval model deployment

3. MODEL PREDICTION

Project Name – model-prediction-service

A separate classifier project picks up the trained model either from a location or from the model registry, and exposes it for prediction in following modes –

- Real Time REST (flask) API – allows users to initiate a classification request in JSON or any other format using a REST client and returns the classified category in real time. This API should be a POST request and should accept JSON in request and return JSON as response. Request body should contain 2 fields - the title and description/summary of a single news article
  Response should contain the predicted category as a field
- Batch Mode (IF TIME PERMITS) – Allow user to pass bulk news articles, and project returns a bulk classification response. It could also be an API which accepts multipart requests containing excel in a predetermined format, and simply submits the request to the bulk process. Response would be a unique id. Another API retrieves the processed excel with classified responses, once the processing is complete, or returns "WIP" if the processing is still in progress.
- User Interface - A simple Streamlit app or a basic HTML page containing a form can be exposed in this project.
  1. It takes the title and the description/ summary of the article, calls the flask API for prediction, and shows the predicted classification.
  2. It provides an option to upload a batch file and response can be processed and stored at a folder location.

4. INFRASTRUCTURE & DEPLOYMENT -

Additional guidelines

1. Docker images may be created for all the 3 projects. These images can then be used for deployment as containers.
2. Deployment can be orchestrated by using docker-compose (optional)
3. Flask APIs (wherever required) should use Gunicorn/ Bjoern/ CherryPy as a WSGI server.
4. Use PEP guidelines for python code standards

ML Tools

Following tools to be used for project setup –

1. PyCharm as Python IDE
2. Virtual Environment – use venv or virtualenvwrapper to setup separate environments for all projects described.
3. MySQL/ MongoDB as datastore.
4. PySpark for stream processing.
5. POSTMAN for testing Flask APIs.
6. Apache Zookeeper + Kafka for message queue/ streams.
7. Tensorboard for monitoring the progress of model retraining.
8. MLFlow for model versioning + hyper-parameters versioning.
9. Python cookiecutter templates may be used for setting up the project.

5. CAPSTONE MILESTONE

There would be 4 milestones –

1. Week 1 –
   - Participants are expected to complete documentation and architectural design.
   - Prepare the local environment with the right tools and installations.
   - Ingest data source and prepare data-ingestion-service which populates "raw_data".

2. Week 2 –
   - Setup the model-training-service project
   - Setup appropriate connectors to load data from MongoDB/ MySQL into PySpark RDD.
   - Understand cleaning + preprocessing steps necessary to transform the raw data and complete data preparation step.

3. Week 3 –
   - Segregate the preprocessed data into training and test
   - complete model training and deployment (and model registry)
   - Convert the project to On-Demand service, to be able to retrain the model as needed. This could be a flask API which could trigger the entire pipeline in model-training-service (loading data into PySpark, preprocessing, model training)
   - Serialize the re-trained model.
   - Push this model to model-registry (MLFlow) along with hyperparameters

4. Week 4 –
   - Expose model via model-prediction-service in the form of flask API.
   - Dockerize all the projects by adding appropriate Dockerfile
   - Prepare a simple HTML page containing a form to take an article as input and print the predicted category.