MLOps – NLP Case Study

Prepared By,

Sanjeev Surendran

Contents

Problem Statement 4

Business Goal 4

Q1. System design: Based on the above information, describe the KPI that the business should track. 5

Q2. System Design: Your company has decided to build an MLOps system. What advantages would you get by opting to build an MLOps system? 5

Q3. System design: You must create an ML system that has the features of a complete production stack, from experiment tracking to automated model deployment and monitoring. For the given problem, create an ML system design (diagram). 6

Q4. System design: After creating the architecture, please specify your reasons for choosing the specific tools you chose for the use case. 8

Q5. Workflow of the solution: 10

Q5.1 What component/pipeline will be triggered if there is any drift detected? What if the drift detected is beyond an acceptable threshold? 11

Q5.2 What component/pipeline will be triggered if you have additional annotated data? 11

Q5.3 How will you ensure the new data you are getting is in the correct format that the inference pipeline takes? 12

Table of Figures

Figure 1: ML System Design 7

Problem Statement

You are working as a data scientist for a Health Tech company called BeHealthy. It is a late-stage startup.

Be Healthy provides a web-based platform for doctors to list their services and manage patient interactions. It provides various services for patients, such as booking interactions with doctors and ordering medicines online. (Similar to 1mg, PharmEasy)

Here, doctors can easily manage appointments, track past medical records and reports and give e-prescriptions.

We can assume that BeHealthy has enrolled thousands of doctors across the major cities in India. You can also assume that millions of patients are using BeHealthy's services; hence, a lot of free-text clinical notes would be present in the e-prescriptions.

BeHealthy would want to convert these free-text clinical notes to structured information for analytics purposes. You have seen such a problem in your previous courses on NLP. You had created a CRF model to recognise the Named Entity in the medical data set.

Clinical Note: "The patient was a 62-year-old man with squamous cell lung cancer, which was first successfully treated by a combination of radiation therapy and chemotherapy."

Disease: Lung Cancer

Treatment: Radiation Therapy and Chemotherapy

Before solving any business problem, we must always keep in mind that why a business needs to resolve the problem. In many cases, data scientists directly jump to the solution, using data sets like Kaggle competitions. But in the real world, you should be able to justify the business need, define KPIs and then design an optimal solution.

Business Goal

Currently, if you need to extract diseases and treatments from free text, a trained person with clinical knowledge must manually look at the clinical notes and then pull this information.

A data entry team would manually look at the clinical text and extract information about diseases and their treatment data. A data validation team would validate the extracted information. This process is prone to errors, and as the data increases, the data-entry team's size would need to be scaled up.

Automating this process would result in reduced man-hours. The data-entry team would not be required. The data validation team would still need to perform validation on extracted data. It would also significantly reduce manual errors.

Q1. System design: Based on the above information, describe the KPI that the business should track.

From given problem statement and business goal, the business should track the following Key performance indicators (KPIs). The KPIs are designed to align with the business goals and help the business to measure the success of the solution.

1. Reduction in manual errors: This KPI would help the business ensure that the Machine Learning (ML) model reduces the manual errors previously made by the data entry team.
2. Reduction in man hours: This KPI would help the business ensure that the model reduces the time and resources required to extract information from clinical notes.
3. ML model Accuracy: This KPI would help the business to ensure that the model is performing well and is extracting accurate information (disease vs treatment given) from the clinical notes.
4. ML model Execution in Real-Time: This KPI would help the business to ensure that the model is processing the data in real-time and providing the information promptly by taking less time in general.

Q2. System Design: Your company has decided to build an MLOps system. What advantages would you get by opting to build an MLOps system?

Huge volume of Clinical Data is generated everyday by Doctors by treating millions of patients subscribed to BeHealthy. Machine Learning Operations (MLOps) increases productivity and allows organizations to remove many of the obstacles they face in creating production-level Machine Learning (ML) solutions by providing a technical backbone for managing the ML life cycle. MLOps stands for automating the entire workflow of the ML model which handles repetitive tasks, saves time and avoids human-induced errors.

Following are the benefits of applying MLOps system:

1. Less Time taken for model development and deployment:
    a. Reduction of complexity and deletion of unused features. Keeping dataset as simple as possible is a benefit and it includes cleaning up data, preprocessing and managing huge amount of data.
    b. Setting up tracking and versioning for experiments and model training runs.
    c. Setting up the deployment and monitoring pipelines for the models that goes to production.
2. Automation and Testing – Testing is necessary for data, models and code used for integrating all the elements. Automating as much as possible, such as automated testing, automated training, automated pipelines, etc. This avoids manual error in triggering pipelines while training or testing.
3. Code, artifact and experiment tracking – Helps in bridging the gap between model building and model deployment tasks. We can build a Repeatable process and focus on collecting, organizing and tracking model training information like hyperparameters, best model using

different classifiers, model size, different NLP process to perform Named Entity Recognition (NER), preprocessing scripts etc. Using experiment tracking tools for benchmarking different models available in market, in-house developed etc.

4. Continuous integration and deployment – We can provide end-to-end traceability so that we can achieve Improvement in time to reach the market. It would be seamless to upgrade model and stage them for production.
5. Continuous training and model monitoring:
    a. Continuous training is an aspect of machine learning operations that automatically and continuously retrains machine learning models to adapt to changes in the data before it is redeployed. Example of New Clinical Data – New diseases (Covid 19 and its versions), New effective treatments to known diseases (Cancer). Extracting medical NER is important to improve out models to avoid model decay over time.
    b. Model monitoring enables your us to identify and eliminate a variety of issues, including bad quality predictions and poor technical performance. This will help to check the drift level of the input data. If the level of drift is significant, the model will not perform well and therefore, necessary action should be taken to control it.
6. Real-time analytics – Providing APIs for running models whenever clinical data arrives, easy and user-friendly monitoring alerts etc is an advantage.
7. Risk Probability and Rollbacks – Before pushing ML models into production, spend some time doing risk assessments so that your model is robust and does not fail in production. Automated rollbacks are necessary in case of any failure.

Q3. System design: You must create an ML system that has the features of a complete production stack, from experiment tracking to automated model deployment and monitoring. For the given problem, create an ML system design (diagram).

Each and every stage is critical in ML Design system, as it helps us create a reliable, scalable, maintainable and adaptable model. With respect to given use case, following components are required and their roles are explained in detail:

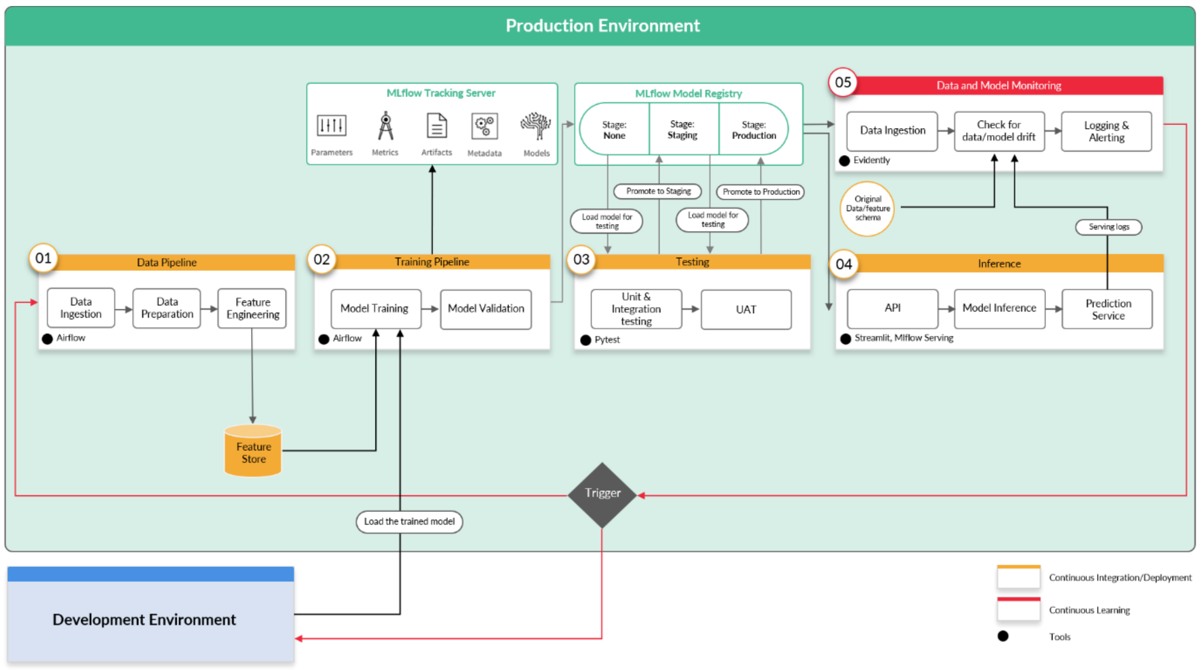

Figure 1: ML System Design

1. Data pipeline: The data pipeline is responsible for acquiring, processing, and transforming raw data into a suitable format for training the model. The role of the data pipeline include:
    a. Extracting clinical notes from the BeHealthy platform's database and storing it in a data storage system like AWS S3 bucket or GCP Cloud Storage.
    b. Preprocessing the raw data by cleaning and normalizing it, removing stop words, and identifying the Named Entities.
    c. Creating training, validation, and test datasets and storing them in a data storage system.

2. Training pipeline: The training pipeline trains the machine learning model using the pre-processed data. The role of the training pipeline include:
    a. Developing a model architecture and training algorithm.
    b. Defining hyperparameters, like learning rate, batch size, etc.
    c. Training the model using the pre-processed data, and saving the best model checkpoints.

3. Testing pipeline: The testing pipeline tests the machine learning model using a separate dataset. The role of the testing pipeline include:
    a. Loading the best model checkpoints from the training pipeline.
    b. Evaluating the performance of the model using the test dataset.
    c. Logging the evaluation metrics, like precision, recall, and F1 score.

4. Model inference pipeline: The model inference pipeline is responsible for deploying the trained machine learning model to production and predicting new data. The role of the model inference pipeline include:
    a. Building an API endpoint using Flask or FastAPI that can receive clinical notes as input and return the identified Named Entities as output.
    b. Deploying the API on a serverless computing platform, like AWS Lambda or GCP

Cloud Functions.

c. Scaling the API according to the incoming traffic.

5. Model monitoring pipeline: The model monitoring pipeline monitors the deployed machine learning model's performance and detects drift or degradation. The role of the model monitoring pipeline include:
    a. Collecting real-time logs of the API endpoint, including input data and predicted output.
    b. Comparing the predicted output with the ground truth to detect any drift or degradation.
    c. Alerting the data science team if drift or degradation is detected, so they can take corrective measures.

Q4. System design: After creating the architecture, please specify your reasons for choosing the specific tools you chose for the use case.

BeHealthy focuses on providing solution to help patients and the company's primary focus is not developing in-house ML solutions or sell such tools to other companies. Hence, we can go with readily available Cloud based platform tools for designing our solution. The specific tools were chosen based on ease of use, setting up, and scalability for the given use case.

1. Data pipeline:
    a. Data storage: We will use AWS S3 bucket. It is a public cloud storage resource available in Amazon Web Services (AWS) Simple Storage Service (S3) platform. It provides object-based storage, where data is stored inside S3 buckets in distinct units called objects instead of files. It provides industry-leading scalability, data availability, security, and performance.
    b. Data Processing: We will use Apache Spark. It is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming clusters with implicit data parallelism and fault tolerance. It can handle large volumes of real-time data and is suitable for processing continuous data streams.
    c. Data versioning: We will use Data Version Control (DVC). It is as an open-source version control system for data science and machine learning projects. It enables reproducibility and collaboration by tracking changes to data, models, and code. With DVC, we can version our datasets and track changes to our data pipeline, ensuring reproducibility and traceability.
    d. Data cleaning and pre-processing:
        i. Pandas and Numpy to handle data
        ii. Spacy to load English dictionary, tokenizer etc.

2. Training pipeline:
    a. Machine Learning Framework: scikit-learn for building various classifier, sklearn_crfsuite for building a Conditional Random Fields (CRF) model.

b. Model Versioning: We will use DVC for versioning machine learning models and data sets. It provides a way to track changes to machine learning models and data sets over time, making it easier to reproduce experiments and collaborate with team members.
c. Model Tuning: We will use Optuna. It is a software framework for automating the optimization process of these hyperparameters. It automatically finds optimal hyperparameter values by making use of different samplers such as grid search, random, Bayesian, and evolutionary algorithms.
d. Model Training: Docker is a set of platforms as a service product that use OS-level virtualization to deliver software in packages called containers. We will use Docker containers with GPU support to train models. AWS Sagemaker studio can help in tracking experiments while training and deploying final model for production.
e. Model packaging: We will use Docker container to pack models.

3. Testing pipeline:
    a. Continuous Integration: We will use Jenkins or Travis CI, an open-source tool that provides continuous integration and continuous delivery (CI/CD) capabilities. It enables automated builds, testing, and deployment of software applications, which is crucial in ensuring that the ML model performs well in a production environment. Additionally, it can be integrated with other tools like GitHub for version control and AWS for cloud deployment.
    b. Unit Testing: For unit testing, we will use Pytest. It is a Python testing framework that originated from the PyPy project. It can be used to write various types of software tests, including unit tests, integration tests, end-to-end tests, and functional tests. Its features include parametrized testing, fixtures, and assert re-writing.
    c. Code Quality and Linting: We will use Pylint, a static code analysis tool for the Python programming language. And, we will use Flake8 for catching more errors.

4. Model inference pipeline:
    a. Web Framework: We will use Flask, as it is a lightweight web framework that provides a RESTful API for serving ML models. It can be easily integrated with AWS for deploying the trained model, and can handle multiple requests simultaneously. Additionally, it provides flexibility in choosing the programming language for writing the API code.
    b. Model Serving: Docker containers with GPU support.
    c. Gateway: We will use AWS API Gateway. It is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.
    d. Load Balancing: We will use AWS Elastic Load Balancing. Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple targets and virtual appliances in one or more Availability Zones (AZs).
    e. Auto-scaling: We will use AWS Auto Scaling. It monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

5. Model monitoring pipeline:
    a. Monitoring Framework: We will use Prometheus, as it provides a time-series database for storing metrics data. It can be integrated with other systems like Kubernetes and Jenkins for automated monitoring and alerting.
    b. Dashboard: We will use Grafana, as it provides a user-friendly interface for visualizing the data. It can be integrated with other systems like Kubernetes and Jenkins for automated monitoring and alerting.
    c. Alerting: Prometheus Alertmanager - The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as email, PagerDuty, or OpsGenie. It also takes care of silencing and inhibition of alerts. AWS CloudWatch Alarms can also be used to send an Amazon SNS message or performs an action when the alarm changes state.
    d. Log Management: Elastic Stack offers centralized logging capabilities to aggregate server logs from complex cloud environments into a single searchable index. AWS CloudWatch Logs can be used to monitor and troubleshoot systems and applications using your existing system, application and custom log files. With CloudWatch Logs, you can monitor your logs, in near real time, for specific phrases, values or patterns.

Q5. Workflow of the solution:

Following are the steps to build an end-to-end ML system for the given use case:

1. Data and Model Experimentation:
    a. Get and pre-process the data: The initial phase would be to collect clinical notes data from the BeHealthy platform. The data should be cleaned and pre-processed for model training. Apache Spark can be used to preprocess the data.
    b. Develop a CRF model: Next, a CRF model can be developed using sklearn_crfsuite. The model can recognise the Named Entity in the medical dataset, such as diseases and treatments. The model will be versioned using DVC.
    c. Experiment Tracking: AWS Sagemaker can track the CRF model's performance metrics during experimentation.

2. Automation of Data Pipeline:
    a. Data Version Control: DVC can be used to version the input data, preprocess scripts, and processed data.
    b. Data Preprocessing: Apache Spark can preprocess the data in parallel and distribute it across multiple machines.
    c. Data Storage: The processed data can be stored in a cloud-based service like AWS S3.

3. Automation of Training Pipeline:
    a. Model Version Control: DVC can version the CRF model, its dependencies, and configuration files.
    b. Hyperparameter Tuning: Optuna can be used to tune the hyperparameters of the CRF model.

    c. Model Training: Docker containers with GPU support can be used to train the model in parallel.
    d. Model Packaging: The trained model can be packaged into a Docker container.

4. Automation of Inference Pipeline:
    a. Model Deployment: Kubernetes can deploy the model in a containerised environment.
    b. API Development: Flask or FastAPI can be used to develop the API that takes the input data, preprocesses it, and runs the trained model to predict the diseases and treatments.

5. Continuous Monitoring Pipeline:
    a. Log Monitoring: Elasticsearch and Kibana can monitor the application logs generated by the inference pipeline.
    b. Model Performance Monitoring: Prometheus and Grafana can monitor the model's performance metrics, such as inference latency and accuracy.
    c. Alerting: Alertmanager can alert the relevant stakeholders if performance metrics breach the defined thresholds.

Q5.1 What component/pipeline will be triggered if there is any drift detected? What if the drift detected is beyond an acceptable threshold?

We can define data drift deviation spectrum to trigger necessary pipeline

a. 1-2% deviation (Acceptable / Normal): No action is needed as the deviation is not significant.
b. 5-8% deviation: The data and model training pipelines are triggered for retraining the model with the new data.
c. Greater than 10% deviation (Beyond acceptable): This can occur when the user behaviour changes drastically. In such scenarios, you must go back to the development environment to perform data and model experimentation.

Q5.2 What component/pipeline will be triggered if you have additional annotated data?

a. We can follow 5-8% deviation trigger. This will allow us to retrain our model on new data, which contains some more useful info for our model to learn.
b. Training pipeline can be triggered and followed by deploying the model to production if accuracy, recall etc scores looks OK.
c. We can then use this new model to infer incoming data and monitor for deviation again.
d. In case it drastically worsens the performance, we should immediately rollback to previous version and once more retrain model with different hyperparameters.

Q5.3 How will you ensure the new data you are getting is in the correct format that the inference pipeline takes?

a. In the inference stage, a sample of the request-response (user-model) interaction during prediction serving is captured as serving logs.
b. A comparison is made between the original data/ feature schema or distribution and the generated schema or distribution from the serving logs.
c. If there is any deviation between the two schemas/ distribution, an alert is raised to the respective stakeholders to detect data drift.
d. Based on data drift, deviation is calculated for taking further actions.