[A comprehensive network and content-based security Solution to Protect Consumers against Mobile Abuse]

Task

A comprehensive network and content-based security solution enabling operators to effectively monetize on their A2P traffic and protect their consumer against the growing threat of mobile abuse.

Client

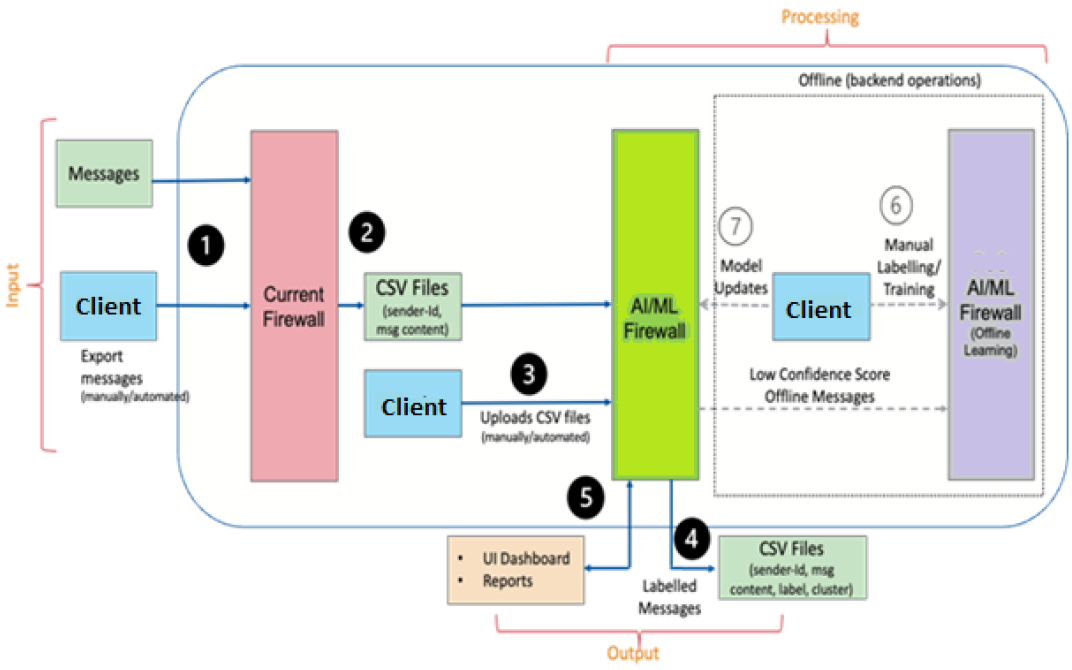
Leaders of mobility solutions providers catering to over two billion platform users globally.

Business Challenge

The rise of SMS as a communication channel has led to a corresponding increase in fraudulent and spam messages being sent to mobile users. These messages can cause harm by stealing personal information, infecting devices with malware, or tricking users into making financial transactions. The current rule-based SMS firewall relying on static rules are now having limited effectiveness in stopping such SMS, as fraudsters/spammers are adapting to these rules and continuously changing their tactics.

So, there is a need for more advanced solution that can adapt and learn over time.

Disruption Delivered



- A NLP based model that aligns with the current rule-based firewall to enhance the effectiveness of blocking fraudulent and spam messages.
- A server based pipeline with an endpoint for efficient training and updating of the NLP model and another endpoint for performing effective inference on large files of SMS data.
- A serverless pipeline for real time inference achieving processing capabilities of nearly 10,000 SMS per second.

Solution Overview

Server Based Pipeline

Herein we provide the facility for training a ML Firewall model on large volumes of data and perform inference on similar scale of data. We wrap this as an API and provide two endpoints namely `/train` and `/predict`

The `/train` endpoint accepts the files for training, make use of Sentence Transformer to encode the SMS data and perform feature engineering using other information like Sender ID, Receiver ID etc., to add more context to our data. Then we train a Neural Network while performing hyper parameter search and tracking via MLFlow.

The `/predict` endpoints accepts the files for inference, makes use of the latest trained model and performs inference. We further perform clustering on these inferred data to identify keywords related to fraudulent and spam messages. We also provide a callback url to notify the client once the inference is done.

Serverless Pipeline

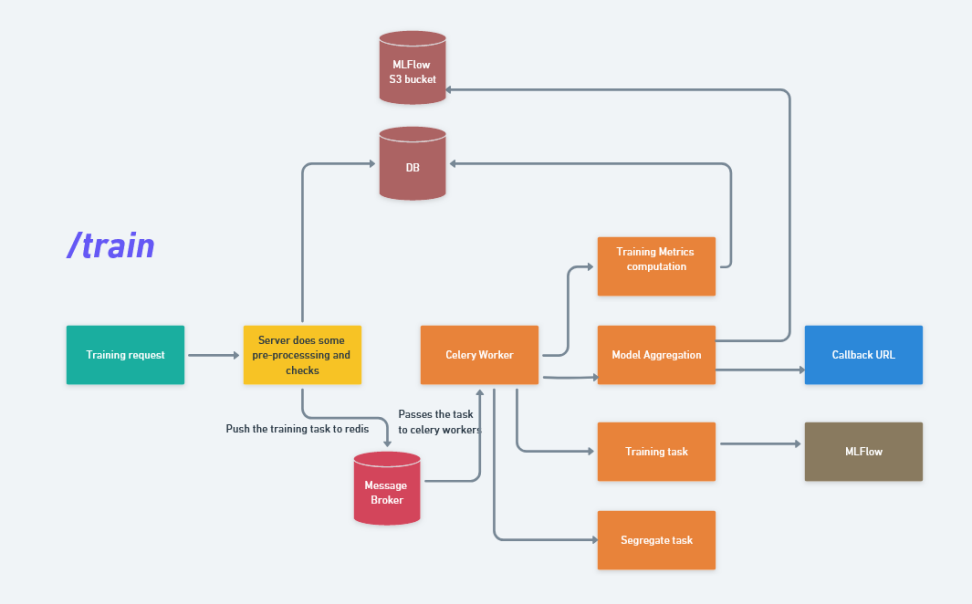
Herein we provide facility for real time inference on messages. This eliminates the need for storing the messages in a file and ping via callback url.

Architecture Overview

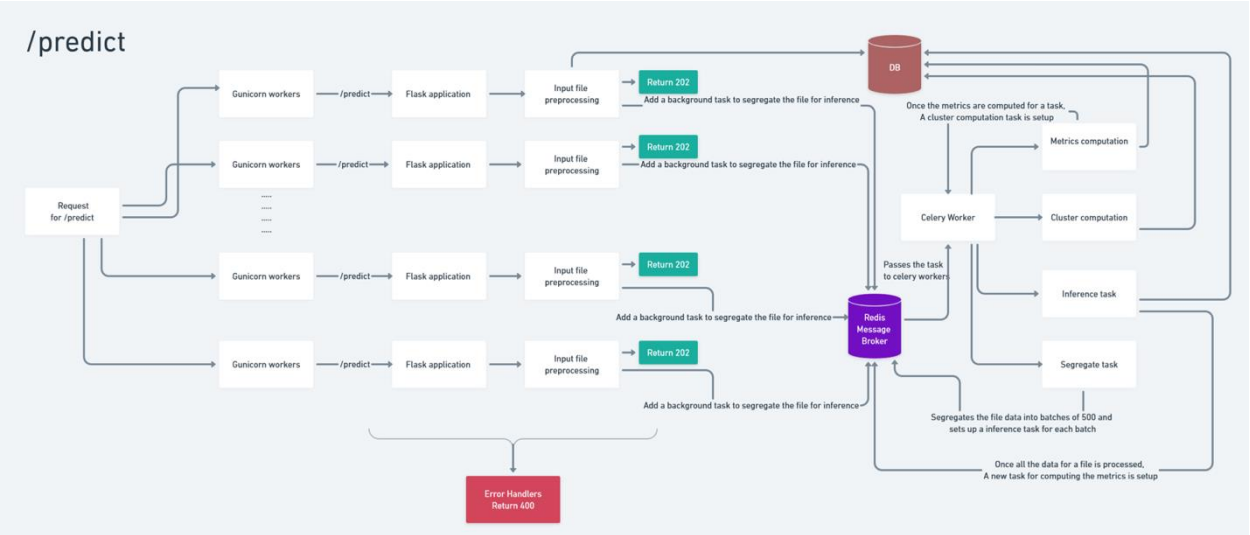
Server Based Pipeline

Herein we make use of flask server with Gunicorn handling parallel requests.

- `/train`: A training request is made to this endpoint. The server processes this request, performing necessary pre-processing and checks. The task is then pushed to a Redis message broker. A Celery worker picks up the task from the message broker to perform the actual training. The Celery worker conducts the training task, computes training metrics, and possibly performs model aggregation. Upon completion, the model is stored and tracked via MLflow server. And the training information are sent back via a callback URL.



- `/predict`: An inference request is made to this endpoint, managed by Gunicorn workers. The application preprocesses the input file and queues a background task for data segregation in Redis. A 202 HTTP status is returned to acknowledge the request reception. Celery workers then retrieve and execute tasks for metrics computation, cluster computation, and inference on segregated data batches. Once the processing is complete, metrics for the file are computed. If an error occurs, a 400 status is returned to indicate a bad request. Throughout this process, data is stored and managed within a database.



Serverless Pipeline

Herein we use a publisher-subscriber model in a serverless architecture, where a publisher sends out messages in JSON format to a Redis Queue/Message Broker. The message broker acts as an intermediary, managing message distribution to various consumers. Each consumer, in turn, allocates a dedicated thread to process the messages. Within these threads, tasks are carried out that involve model prediction and the generation of inference analysis (cluster computation) to interpret the prediction outcomes. This architecture is highly scalable, allowing for the dynamic

addition of consumers to handle increased message loads and facilitating parallel processing for efficiency.