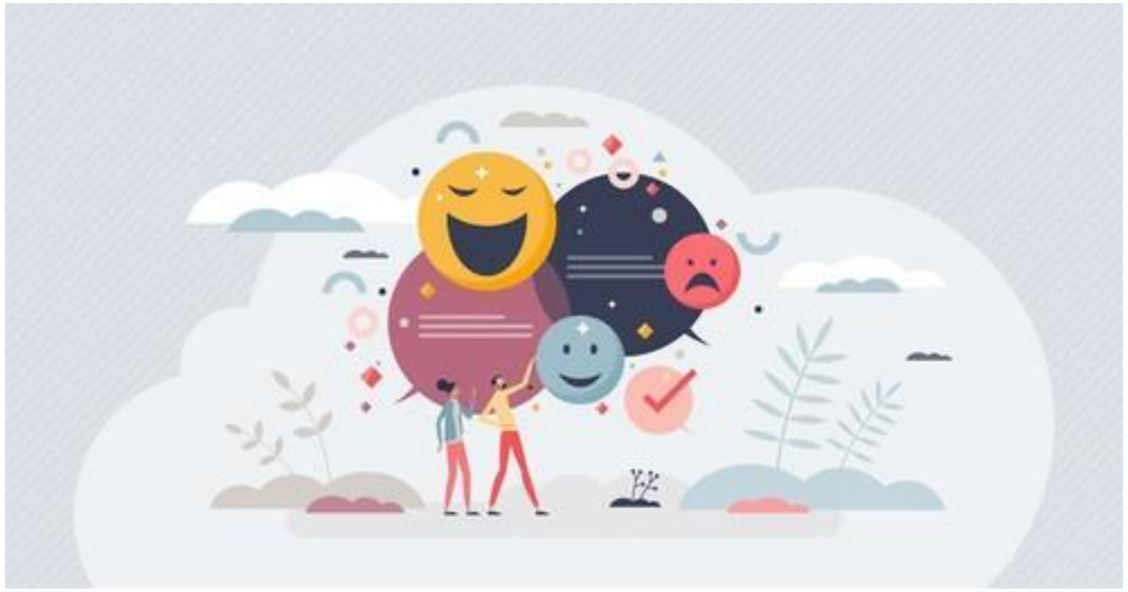
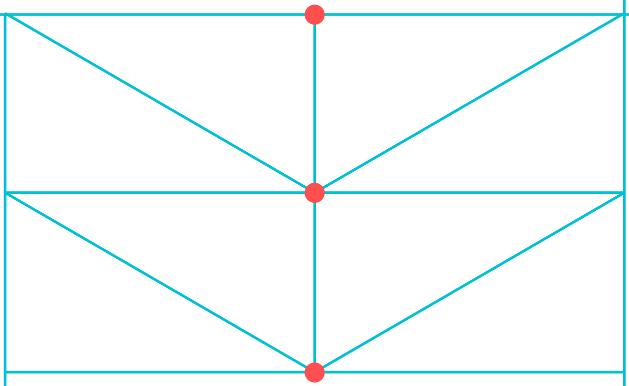
Federated Sentiment Analysis

Large-Scale Sentiment Analysis on Reddit Comments

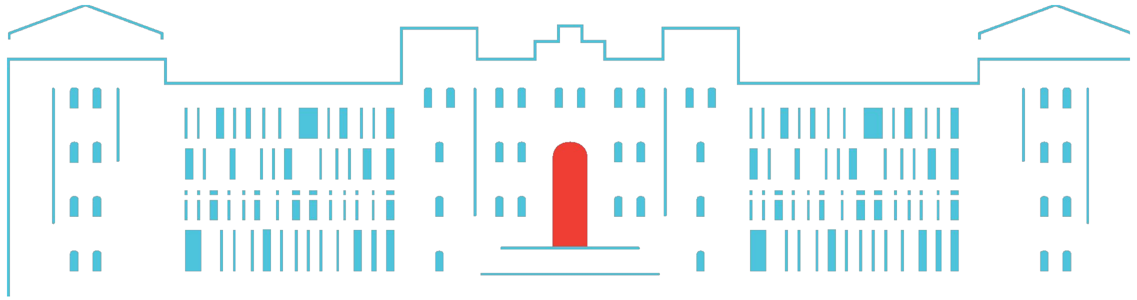


Team A3
Final Presentation

TUHH
Technische
Universität
Hamburg

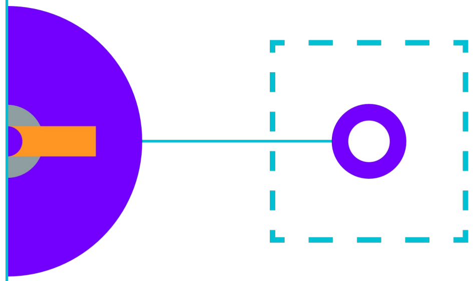


05.07.2024



Devarshi Kaushikkumar Shah
Gaurang Kishorbhai Sumara
Spoorthi Gudagunti
Varad Kulkarni

Line UP



What’s the project about?

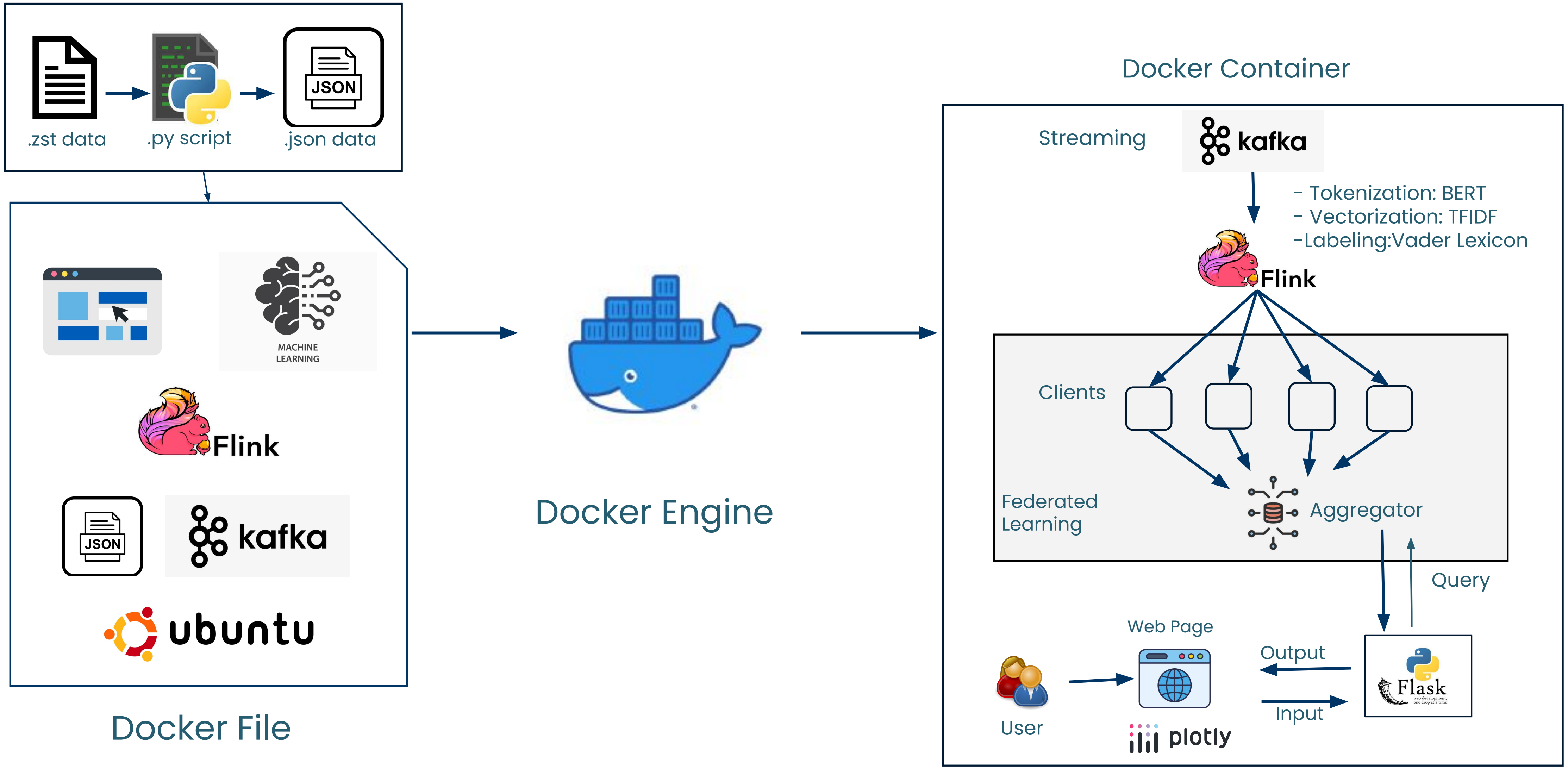
- Our project is about sentiment analysis based on Reddit comments
- What is Sentiment?

  Method used in NLP to understand opinions, emotions/attitude expressed towards specific keywords.
- In this project we are using Reddit data stream from the “The pushshift Reddit Dataset”, to train our Federated machine learning model to get “Sentiment” for user provided keywords

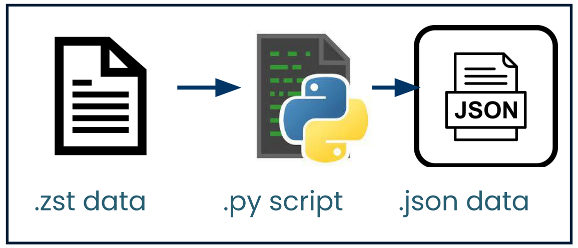
Solution Approach

- Our solution implements Streaming of Reddit comments to emulate “real world” scenarios. This is performed using Apache Kafka.
- Once we have “streams” of comments coming, we use Apache Flink for preprocessing of the data.
- Finally preprocessed data is provided to our federated machine Learning Model, to ensure distributed learning; as the dataset is quite large, and the number of comments being streamed is rather heavy.
- However, not entire dataset is used. To get the desired processing time on given resources.

Solution Architecture



Solution



- .zst file is decompressed, and then converted into .json with our custom script.
- This .json file is then pushed into the docker file for further processing.

Solution Continue

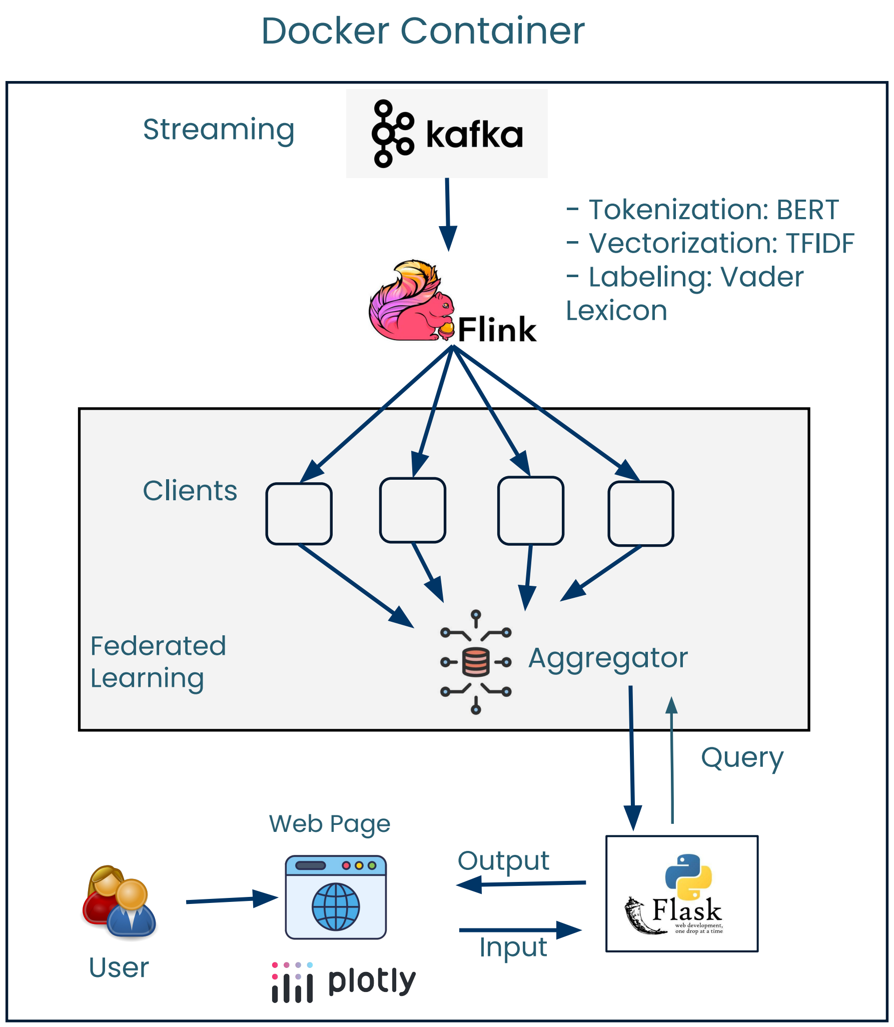
Docker File

- To eliminate the requirement of setting and configuring multiple software and their dependencies, the tech stack is built on a single Docker container; discarding “but it works on my machine” phenomenon.
- Ubuntu based Docker container installs and sets below modules:
  - miniconda, Apache Kafka, Apache Flink, Flink-kafka-connector, .json file
  - Other utilities such as curl, wget
  - Custom python and Bash scripts
  - Frontend packages like, Plotly, Dash and Flask for Frontend and Backend integration

Solution Continue

- conda.sh and trigger.sh combine automate below tasks:
  - Setup conda environment with various dependencies.
  - Triggers Zookeeper server, Kafka server and Producer, Flink cluster.
  - Performs health check of the services.
  - Triggers the Flink job.
  - Triggers Model’s Server, Clients, Global.
  - Initiate webpage, and port forward to the host machine.

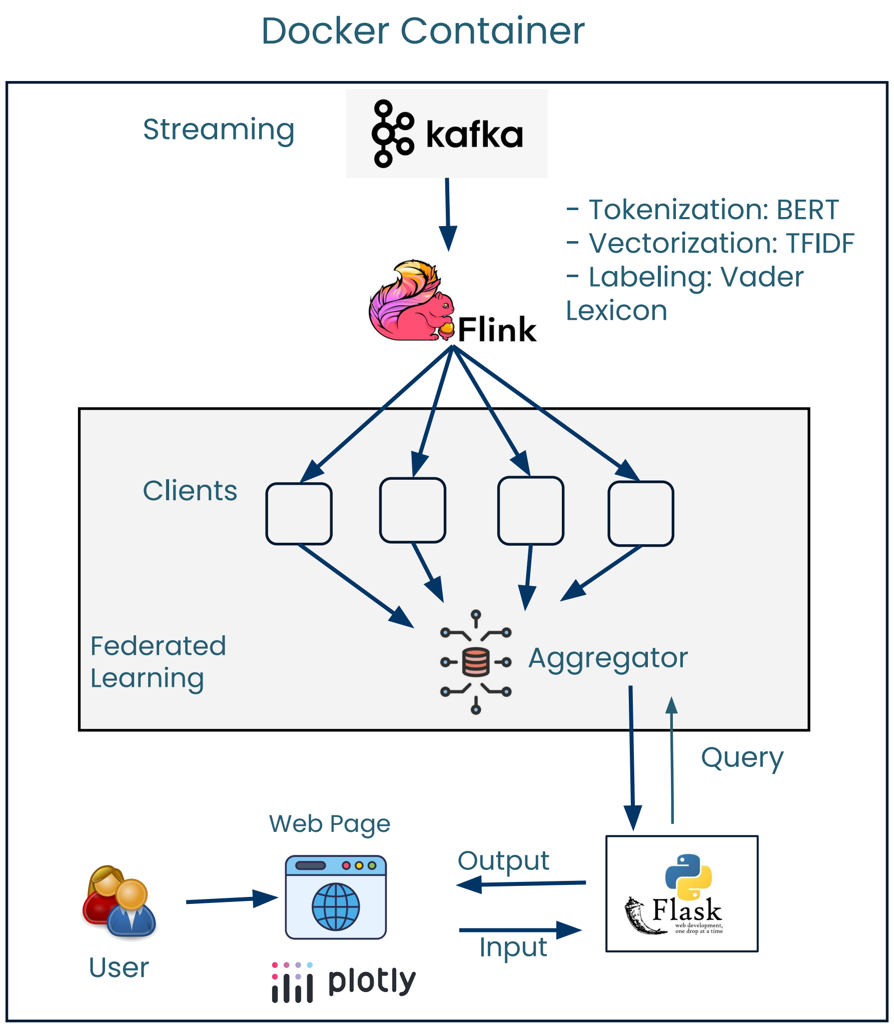
Solution Continue



Once all the services are healthy and running

- Kafka producer produces messages which in turn is streamed by kafka
- Flink consumes the stream and performs below action
    - Tokenization: BERT
    - Vectorization: TFIDF Vectorizer
    - Labelling: Vader Lexicon
    - Distribution of data
- Client models consumes a batch of 1000 messages (in 1 iteration) from their own Kafka topics, perform sentiment analysis; after which the weights and biases are forwarded to the global model.

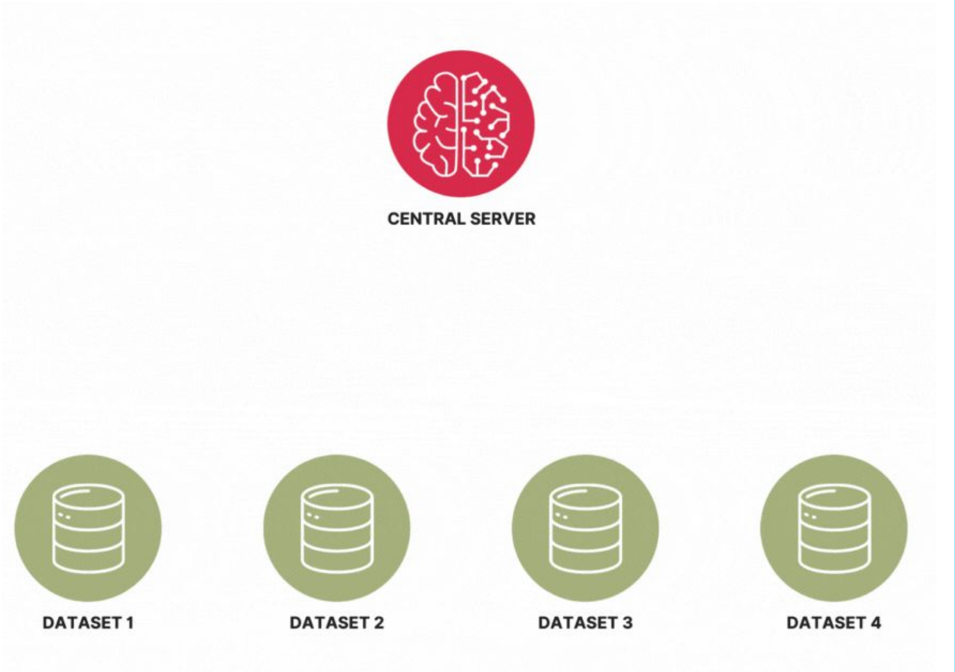
Solution Continue



- Then the Global model updates the local models by aggregating weights and biases.
- Flask integrates the backend with a web page, made with Plotly & Dash.
- The User inputs a text prompt on the webpage, which then being queried by the global model.
- Global model returns the sentiment based on ad hoc knowledge.

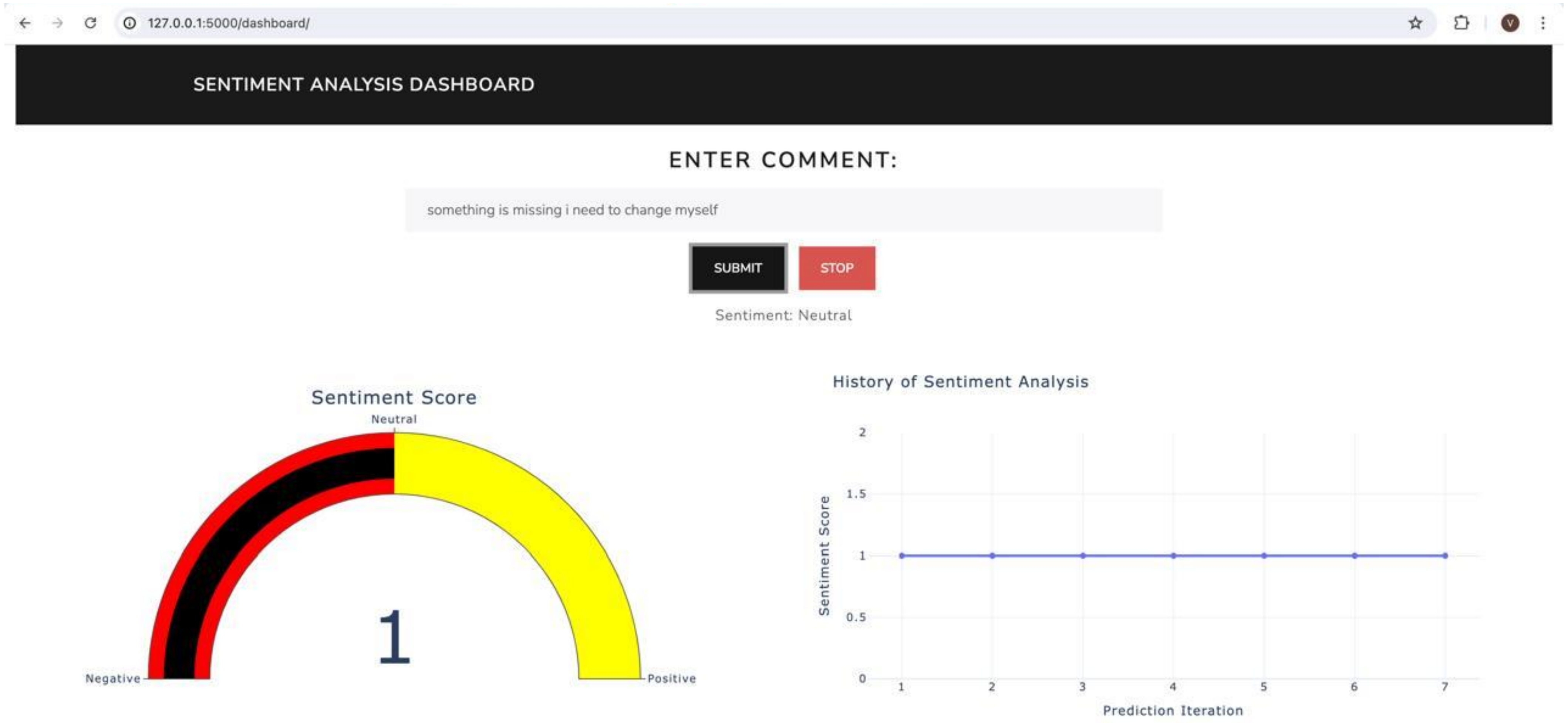
Solution Continue

- **Framework**: Pytorch Flower
- We have 4 Federated clients which are trained on **LSTM model**
- **Optimizer**: Adam Optimizer
- **Input Features**:
  - **input_ids**: These are the token IDs generated from the input text, used as input to the embedding layer of the LSTM.
  - **attention_mask**: These masks are used to indicate which tokens are actual words and which are padding, helping the model to ignore the padding tokens during training.
  - **tfidf_vector**: These vectors represent the input text using "Term Frequency-Inverse Document Frequency" (TF-IDF) scores, capturing the importance of words in the text.



Demo!

Sample Output



User Stories

Though the user stories created by us were divided among the team members, we worked together on almost all of the tasks.

- Google Cloud Setup, Docker, Bash, Markdown: **Devarshi**
- Data cleaning and preprocessing: **Gaurang**
- User Interface design and testing: **Spoorthi**
- Kafka, Flink, Federated Machine Learning Pipeline: **Varad Kulkarni**

Technical Issues

TUHH

1. Version incompatibility issues with below isolated softwares:
    - Kafka, Flink, Pyflink, Confluent Kafka, Pandas, Torch, Numpy & Torch Flower

2. Docker image:
    - Missing Dependencies in Alpine Linux, hence moved to Ubuntu Server 24.04 LTS on X86 running on Google Cloud
    - Pip installation failing due to Python version mismatch, hence moved to Miniconda

Future Scope

- Optimisation of pipeline, using multi-container architecture.
- Slimmer Docker image by replacing Ubuntu with other minimal Linux base image.
- We can have more federated clients, hyper parameter tuning, more capable pretrained model for precise sentiment.
- More, realtime metadata to be presented on the Webpage, such as Loss of the ad hoc iteration,
- Efficient tuning of Flink and Kafka according to the requirements of the project.

References

- Gitlab Code Repository: https://collaborating.tuhh.de/e-19/teaching/bd24_project_a3_a
- Docker Image: https://hub.docker.com/repository/docker/devarshikshah/bd24_project_a3_a/tags

QnA

TUHH

Thanks!