Log Anomaly Detection

Group 3
Sushanth Kulkarni UOH
Mahendar Byra UOH
Prabas Leti UOH
Sathwik Katta UOH

December 29, 2023

Contents

1 Abstract

This project addresses the task of log anomaly detection using a federated learning approach across four diverse datasets: HDFS, BGL, Spirit, and Thunderbird. The primary objective is to determine whether a sequence of related logs exhibits anomalous behavior. Initial efforts involve meticulous data preprocessing to ensure the datasets are prepared for training. Subsequently, we employ the "BertForSequenceClassification" model, leveraging its sequence analysis capabilities.

The unique federated learning framework is employed to train individual models on each dataset, ensuring privacy and security in the learning process. Once trained, these models are aggregated into a global model, representing a comprehensive understanding of log patterns across datasets. This global model is then utilized for effective log anomaly detection.

Notably, the project includes a comprehensive evaluation stage, computing vital metrics such as accuracy, precision, recall, and F1 score. These metrics serve as quantitative measures of the model's performance and effectiveness in detecting anomalies within log sequences.

The approach embraces the power of distributed learning to address log data's inherent diversity, providing a robust solution for anomaly detection. This project stands at the intersection of machine learning, cybersecurity, and distributed systems, contributing insights and methodologies applicable to various domains. The achieved results showcase the efficiency of the proposed federated learning model in identifying anomalous log sequences, laying the foundation for enhanced security in diverse log data environments.

2 Introduction

The solution approach to the problem involves mainly the following steps:

- Data Preprocessing
- Model Design
- Federated training

Data Preprocessing:
The data preprocessing phase encompasses log data collection from diverse sources, log parsing, and log grouping. Various grouping techniques, such as sliding window, fixed window, and session window, are employed to systematically organize log entries. Once the preprocessing is completed, the refined data is forwarded to the subsequent modeling stage.

Model Design:
The model design stage includes the creation, training, testing, and evaluation of the **BertForSequenceClassification** model at the local system. During this phase, the model is trained, and evaluation metrics are computed. This process is iteratively executed for each dataset. Once the model code is prepared, it proceeds to the subsequent step: Federated Training.

Federated Training:
The federated training phase entails establishing a federated client-server setup. The code received in the prior step is executed at each client. In this federated training stage, distinct models are trained using four different datasets. The aggregation step involves gathering all the models generated from the clients and aggregating them at the server. The resultant aggregated model is referred to as the Global Model, representing the final model capable of predicting outputs for diverse datasets.

3 Data Preprocessing

Daata preprosseing is primarily spilt into the following sub tasks:

- Find RAW log data, if unavailable find parsed log data.
- Remove empty and duplicate log entries from RAW log data.
- Parse RAW data into structured format, categorizing each entry into events.
- Group the parsed, structured log data into set of events called a sequence.

3.1 Find Data

The following datasets have been considered for anomaly detection:

1. HDFS
2. BGL
3. Thunderbird
4. Spirit

RAW and parsed datasets for HDFS and BGL were available at:

- https://github.com/logpai/loglizer/tree/master/data/HDFS
- https://zenodo.org/records/8115559
- https://github.com/logpai/loghub/tree/master

3.2 Parse Data

Log Hub only provides us 2k lines of raw data, which is not adequate, hence I've used 100k or more entries of log data available at **LogPai**.
Raw data is parsed using logparser python library and we use, **Drain** parser is primarily used.

3.3 Grouping parsed Data

The parsed data is now in a structured format, a key component here is *Event Id*! Each Event Id represent a particular static part of a log entry, we group these Event Ids, let the set of evenets be called a sequence, S_i. We generate all such S_i based on a grouping parameter.
Grouping parameters:

- HDFS : Session window protocol, **BlockId**
- BGL : Session window protocol, **Node Id**
- Thunderbird : Fixed Window, **Timestamp, window size: 900sec**
- Spirit : Fixed Window protocol, **Timestamp, Window size: 113sec**

3.4 Labelling Sequnence

The general idea: "*If all the events in a sequence are Normal, then the sequence is labelled 'Normal' else 'Anomalous'*"

4 Model specific tasks

The model design model mainly includes the following tasks.

- Feature Extraction and Data splitting.
- Model training.
- Model Evaluation and Evaluation metrics.

4.1 Feature Extraction and data splitting

Feature extraction involves the data division into input(X) and output(lables or y) and assigning 0's and 1's to Normal and Anomolous log sequences.
After data feature extraction , the Data is splitted into training and testing data which will be used in Training and Evaluation.

4.2 Model Training

The model training involves the following tasks:

- **Pretrained model and Tokenizer loading** :
  The pretrained BertForSequenceClassification and BertTokenizer are loaded from the transformers library.

- **Encodings and Datasets** :
  Using the Tokenizer and CustomDataset class the training and testing data is converted into encodings and datasets. These datasets will be used in the training.

- **Defining Training arguments and Trainer** :
  The training arguments decides how the models should be trained. It includes number of epochs ,batch size,etc.The trainer is the one which takes the datasets all trains the model.

- **Training and Saving the model** :
  The trainer calls train() method which trains the model.Then model.save_pretrained() is used to save the model.

4.3 Model Evaluation and Evaluation metrics

Once the model is trained, the test data is used to evaluate the model. In the evaluation , The following metrics are computed. These metrics decides model efficiency.

- Accuracy
- Precision
- Recall
- F1 score

The Evaluation metrics for BGL trained model for about 1000 data entries of 70,000 entries is :
Confusion matrix:

$$\begin{bmatrix} 3 & 0 \\ 0 & 37 \end{bmatrix}$$

Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0

Similarly the Evaluation metrics can be calculated for other datasets also.
Github link for training and evaluation code :
https://github.com/sushanthk-262/Log_anomaly_detection/tree/main/mahendar

5 Fedarated Learning

5.1 Federated Learning Process with Flower Client:

5.2 Initialization

Each participating device, known as a Flower client, initializes its own local model with random parameters.

5.3 Model Distribution:

The central server (aggregator) distributes the initial model parameters to all Flower clients.

5.4 Local Model Training:

Flower clients independently train their local models using their own local datasets and Training is done on the device without transmitting raw data to the central server.

5.5 Model Update:

After local training, each Flower client computes the difference between its local model and the global model obtained from the central server. This difference is known as the model update.

5.6 Model Update Transmission:

Flower clients send their model updates (not the raw data) back to the central server. This ensures privacy as raw data remains on the local devices.

5.7 Aggregation at Central Server:

The central server aggregates the received model updates from all Flower clients. This aggregation typically involves averaging the model updates to obtain a new global model.

5.8 Updated Global Model Distribution:

The central server distributes the updated global model parameters back to all Flower clients.

5.9 Iterative Training:

Steps 3-7 are repeated iteratively for multiple rounds. Each iteration refines the global model as Flower clients continue to train on their local datasets.

5.10 Convergence Check:

The federated learning process continues until the global model converges to a satisfactory state, or a predefined number of iterations is reached.

5.11 Final Global Model:

The final global model represents the collective knowledge learned from all Flower clients' local datasets.

5.12 Model Deployment:

The trained global model can be deployed for making predictions or inferences.

5.13 Privacy and Security Measures:

Throughout the process, privacy and security measures, such as encryption and differential privacy, may be employed to protect sensitive information.

By following all the steps i have tryed to aggregated local trained weights of the two sub trained models of the whole hdfs trained model and the results thati have got were :
for 3 rounds of aggregated weights :
for client-1 :

```
Fit history: {'loss': 0.08982961624860764} \\
    100% 10/10 [00:03<00:00, 2.71it/s]
    Now the global Eval accuracy: 0.90801932066679

Fit history: {'loss': 0.09201569110155106} \\
    100% 10/10 [00:02<00:00, 4.19it/s]
    Now the global Eval accuracy: 0.90801932066679

Fit history: {'loss': 0.09195733815431595} \\
    100% 10/10 [00:01<00:00, 7.14it/s]
    Now the global Eval accuracy: 0.90801932066679
```

for client-2 :

```
Fit history: {'loss':0.09417649358510971 }} \\
    100% 10/10 [00:03<00:00, 6.71it/s]
    Now the global Eval accuracy: 0.908007650077343

Fit history: {'loss': 0.09201569110155106 } \\
    100% 10/10 [00:02<00:00, 3.10it/s]
    Now the global Eval accuracy:0.908007650077343

Fit history: {'loss': 0.09207404404878616 } \\
    100% 10/10 [00:01<00:00, 10.04it/s]
    Now the global Eval accuracy:0.908007650077343
```

6 Conclusion

In conclusion, our project has successfully navigated the intricacies of data preprocessing for four distinct datasets—HDFS, BGL, Thunderbird, and Spirit. Employing federated learning, we trained these datasets across four discrete clients within a client-server framework, resulting in the creation of refined models at each client. Through meticulous aggregation at the server side, a robust global model emerged, encapsulating insights from the individual datasets. Different evaluation metrices are used to check the efficiency of the model. Notably, the model's versatility extends beyond its training datasets, demonstrating predictive capability across diverse data domains. Our project underscores the collaborative potential of federated learning, providing a scalable and privacy-preserving solution at the intersection of machine learning innovation and decentralized data collaboration. In essence, this strategy illustrates the significant impact of cooperative efforts, holding potential for future advancements in the field of federated learning applications.

7 Future Scope

In the preceding methodology, the Bert model is employed for both representation and classification purposes. In contrast, an alternative strategy utilizes Bert solely for representation learning. The representation Bert model is trained in a federated manner, contributing to the development of a comprehensive global representation model. This approach allows individual clients to establish their distinct classification models, thereby affording them a high degree of flexibility.

8 References

Project github link : https://github.com/sushanthk-262/Log_anomaly_detection
References :

- https://www.nathan-smith.net/project/log_anomaly/
- https://github.com/LogIntelligence/LogADEmpirical
- https://browse.arxiv.org/pdf/2202.04301.pdf