A Report

on

Log Parser

by

Name of the student
Krishna Sharma

Registration No.
AP22110010128

Prepared in the partial fulfillment of the
Summer Internship Course

AT

Zeronsec
1st Floor, Plot-3, Navjivan Society - 2,
Ajwa Road, Vadodara, Gujarat 390006

SRM UNIVERSITY, AP

(July, 2024)

zeronsec

1st Floor, Plot-3, Navjivan Society - 2,
Ajwa Road, Vadodara, Gujarat 390006
www.zeronsec.com

Date: 15/07/2024

To Whom So Ever It May Concern

This is to certify that **Mr. Krishna Sharma** has successfully completed his internship program as "**AI/ML Trainee**" under the guidance & leadership of Mr. Mangalsing Patil. This internship duration was from 30-05-2024 to 13-07-2024. We wish him success in him future endeavours. During his tenure, we found him active and competent in executing all the assigned tasks and services were found to be satisfactory.

We wish him a bright future!

For, Zeronsec India Private Limited

ZERONSEC INDIA PVT. LTD.

Mr. Mangalsing Patil

Head – Research and Development

Format of Joining Report

SUMMER INTERNSHIP COURSE, 2024-25

JOINING REPORT

(To be sent within a week of joining to the University -Joining report to be uploaded online through Google Form circulated by Associate Dean, Practice School)

Date:

Name of the Student	Krishna Sharma
Roll No	AP22110010128
Programme (BTech/ BSc/ BA/MBA)	B. Tech
Branch	Computer Science and Engineering
Name and Address of the Internship Company [For research internship, it would be SRMAP]	Zeronsec 1st Floor, Plot-3, Navjivan Society - 2, Ajwa Road, Vadodara, Gujarat 390006 Telephone No: 9049048799 Email: mangal.p@zeronsec.com
Period of Internship	From 30th May to 13th Apri

I hereby inform that I have joined the summer internship on 30th May for the In-plant Training/ Research internship in the industry.

Date : 18th July 2024 Signature of the Student

CERTIFICATE FROM INDUSTRY MENTOR/HR
(FACULTY MENTOR FOR RESEARCH INTERNSHIP)

Certified that the above-mentioned student has joined our organization for the INTERNSHIP / INDUSTRIAL TRAINING / ACADEMIC ATTACHMENT in the industry / Organization.

Name of the Industry Mentor	Mangal Patil
Designation	Head Research and Development
Phone No (If any)	9049048799
Email	mangal.p@zeronsec.com
Signature & Date	 18/07/2024

Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this Log Parser project.

First and foremost, I am deeply indebted to the Vice Chancellor, Prof. Manoj K Arora, and the Dean, Prof. Ranjit Thapa, for providing me with the opportunity to undertake this internship at Zeronsec as part of the university's summer internship program. Their support and encouragement have been invaluable throughout this project.

I extend my heartfelt thanks to Mr. Mangalsing Patil, Industry Mentor, for his invaluable guidance, mentorship, and expertise. His insights and constructive feedback have been instrumental in shaping this project.

I am also grateful to Dr. Amit Kumar Mandal, Faculty Mentor, for his continuous support and encouragement. His academic guidance and mentorship have been crucial to the successful completion of this project.

I would also like to acknowledge the support and cooperation of the entire team at Zeronsec. Their expertise and collaborative spirit have enriched my learning experience.

Finally, I would like to thank my family and friends for their unwavering support and encouragement throughout this endeavour.

Thank you.

Abstract

This project presents the development and implementation of a machine learning-based log parser designed to efficiently extract structured information from unstructured log data. By leveraging a pre-trained RoBERTa model and employing fine-tuning techniques, the proposed model automates the process of identifying key fields within log entries, such as timestamps, line IDs, components, content, event IDs, and event templates. The model was trained on a diverse dataset of log data to ensure its ability to handle various log formats and structures. Experimental results demonstrate the model's effectiveness in accurately extracting desired information, surpassing traditional rule-based methods in terms of efficiency and accuracy. The automated log parsing process empowers security analysts to focus on higher-value tasks, leading to improved overall log analysis capabilities and enhanced cybersecurity operations.

Table of Contents

A brief introduction of the organization's business sector

Cybersecurity:

The cybersecurity industry encompasses a broad range of technologies, processes, and practices designed to protect networks, computers, programs, and data from attack, damage, or unauthorized access. As the digital landscape expands, so too do the threats faced by individuals and organizations alike.

Cybersecurity is a critical component of the modern economy, impacting sectors such as finance, healthcare, government, and retail. The industry is characterized by rapid technological advancements, evolving threat landscapes, and a constant need for skilled professionals.

Key areas within the cybersecurity industry include:

- Network security: Protecting computer networks from unauthorized access, use, modification, or disruption.
- Application security: Safeguarding software and applications from attacks.
- Data security: Ensuring the confidentiality, integrity, and availability of data.
- Cloud security: Protecting data, applications, and infrastructure in cloud environments.
- Identity and access management (IAM): Controlling access to systems and data.
- Cybersecurity consulting and services: Providing expert advice and support to organizations.

The Indian cybersecurity market has witnessed significant growth in recent years, driven by increasing digital adoption, rising cyber threats, and government initiatives. While the industry is still maturing compared to developed nations, it offers immense potential for growth and innovation.

Overview of the organization

1. Brief History

Though a specific founding date isn't available, Zeronsec is likely a young and dynamic cybersecurity company established within the last decade. As the digital landscape and cyber threats grew, Zeronsec probably identified a market gap for efficient and scalable cybersecurity solutions, particularly addressing the challenges of managing vast amounts of log data.

2. Business Size

With approximately 50 employees, Zeronsec falls under the category of a small to medium-sized enterprise (SME) within the cybersecurity industry. This size suggests a close-knit team fostering agility and a focus on personalized service. Despite its size, Zeronsec demonstrates a commitment to innovation, as evidenced by its development of machine learning-based solutions.

3. Product Lines (Services)

Zeronsec offers a comprehensive suite of cybersecurity products and services, making them a one-stop shop for organizations seeking to protect their digital assets. Here's a breakdown of their offerings:

- **Threat Intelligence Platforms**:
    - **threat-i:** Provides accurate, proactive, and actionable threat intelligence data, empowering security teams to anticipate and mitigate potential threats.
- **Security Information and Event Management (SIEM):**
    - **anrita:** Offers a comprehensive SIEM solution to collect, analyze, and centralize log data from various sources, enabling security personnel to readily identify and respond to security incidents.
- **Security Orchestration, Automation, and Response (SOAR):**
    - **ekasha:** Enhances the efficiency of Security Operations Centers (SOCs) through its SOAR technology. The platform automates routine tasks, allowing security teams to focus on complex threats and incident response.
- **Security Services:**
    - **Cloud Security:** Zeronsec provides expertise in securing cloud environments, helping organizations mitigate risks associated with cloud adoption.
    - **Managed Detection and Response (MDR):** Zeronsec offers MDR services, acting as an extension of an organization's security team, proactively monitoring for threats and providing a rapid response to security incidents.

- **SOC Operations:** Zeronsec might manage a client's Security Operations Center entirely or provide support in optimizing existing SOC operations.
- **Security Assessments:** Zeronsec conducts security assessments to identify vulnerabilities and security gaps within an organization's infrastructure.

4. Competitors

The cybersecurity market is highly competitive. Zeronsec's main competitors would likely include:

- **Established security firms:** Larger companies offering a broad range of cybersecurity solutions.
- **Specialized cybersecurity startups:** Smaller companies with niche expertise in specific areas of cybersecurity, potentially including threat intelligence or SOAR solutions.
- **Managed Security Service Providers (MSSPs):** Companies offering comprehensive security solutions, including SIEM, SOAR, and MDR, as a managed service.

To stand out, Zeronsec might leverage:

- Focus on innovative solutions like machine learning-based threat intelligence and log parsing.
- Agility as an SME.
- Personalized customer service.

Plan of Internship Program

Brief Introduction of the Department:

I was placed in the Cybersecurity department at Zeronsec for my internship. The department is responsible for safeguarding the organization's clients from cyber threats. Before my internship, the team primarily relied on manual log parsing techniques, using regular expressions (regex) and Grok patterns to extract relevant information from system logs. This information was then utilized by other team members for further analysis and security operations.

Start and End Dates of Internship:

My internship commenced on May 30, 2024, and concluded on July 13, 2024. The total duration of the internship was six weeks.

Duties and Responsibilities:

My primary responsibility during the internship was to develop a machine learning-based log parser model to automate the process of extracting information from system logs. The goal was to replace the existing manual, time-consuming methods with an efficient and accurate automated solution.

My role encompassed the following key tasks:

- Data Collection and Preparation: Gathering a diverse dataset of system logs from various sources, cleaning, preprocessing, and formatting the data for model training.
- Model Development: Researching and selecting suitable machine learning algorithms for log parsing. Implementing and fine-tuning the model to accurately extract relevant information from log data.
- Model Evaluation: Testing the model's performance on different log datasets and comparing its accuracy to the existing manual methods.
- Documentation: Creating comprehensive documentation detailing the project's methodology, results, and recommendations for implementation.

Through this project, I aimed to significantly improve the efficiency and accuracy of log analysis within the cybersecurity department, enabling quicker response times to security incidents.

Introduction

Log analysis is a critical function in cybersecurity, providing essential insights into system behaviour, potential threats, and security incidents. Traditional log analysis methods often rely on manual parsing techniques, such as regular expressions and Grok patterns, to extract meaningful information from raw log data. This approach is time-consuming, error-prone, and lacks scalability as log volumes continue to grow and log formats diversify.

To address these limitations, this project aimed to develop a machine learning-based log parsing model capable of automating the extraction of relevant information from diverse log formats. By eliminating the need for manual pattern creation and maintenance, the model aimed to significantly improve the efficiency and accuracy of log analysis.

The extracted information, produced by the log parser, would then be further analysed by cybersecurity analysts to identify potential threats, investigate security incidents, and gain actionable intelligence. The focus of this project was specifically on the automation of the log parsing process, leaving subsequent analysis tasks to domain experts. By developing a robust and adaptable log parsing model, this research contributes to enhancing the overall efficiency and effectiveness of cybersecurity operations.

Main Text

Assumptions Made:

- The availability of a sufficient quantity of diverse and labelled log data for model training.
- The suitability of a pre-trained RoBERTa model as a foundation for the log parsing task.
- The ability to effectively fine-tune the RoBERTa model for the specific requirements of log parsing.

Experimental Work/Data Collection:

- Data Acquisition: A dataset of log entries was collected, encompassing various system components and log formats to ensure model robustness.
- Data Preprocessing: Log data was cleaned, tokenized, and converted into a suitable format for model input. Labels were assigned to each log entry for the target fields (timestamp, line ID, component, content, event ID, event template).
- Data Splitting: The dataset was divided into training, validation, and testing sets for model development and evaluation.

Algorithm/Model Development:

- Model Architecture: A pre-trained RoBERTa model was fine-tuned for the log parsing task using multi-task learning to predict multiple target fields simultaneously.
- Training Process: The model was trained on the prepared dataset using an appropriate optimization algorithm and loss function. Hyperparameters were tuned to optimize model performance.

Results:

The fine-tuned RoBERTa model demonstrated promising performance in extracting desired information from log entries. Evaluation metrics, such as accuracy, precision, recall, and F1-score, were calculated to assess the model's effectiveness. The model exhibited strong performance in handling various log formats and extracting the specified fields with high accuracy.

Discussion and Interpretations:

The successful application of a pre-trained RoBERTa model for log parsing highlights the potential of transfer learning in this domain. The model's ability to handle diverse log formats and extract multiple fields simultaneously demonstrates its versatility. However, further

improvements could be achieved by exploring more advanced model architectures or incorporating domain-specific knowledge into the training process.

Outcomes

- Automated Log Parsing: The developed model successfully automates the extraction of key information from log entries, significantly reducing manual effort.
- Improved Efficiency: By streamlining the log parsing process, security analysts can focus on higher-value tasks and gain insights more rapidly.
- Enhanced Accuracy: The model's ability to accurately extract information from diverse log formats improves the reliability of subsequent analysis.

Conclusions

The fine-tuned RoBERTa model has proven to be an effective solution for automating log parsing. By leveraging transfer learning and multi-task learning, the model achieves high accuracy and efficiency in extracting desired information from log data. To further enhance the model's capabilities, exploring more complex model architectures and incorporating domain-specific knowledge could be considered. Continuous monitoring and retraining of the model are essential to adapt to evolving log formats and improve performance over time.

This project successfully developed a machine learning-based log parser capable of automating the extraction of valuable information from unstructured log data. The model's performance surpasses traditional methods, demonstrating the potential of machine learning in improving cybersecurity operations.

Appendices

Source Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import warnings
import torch
from torch.utils.data import DataLoader, Dataset
from transformers import RobertaTokenizer, RobertaModel, AdamW,
get_linear_schedule_with_warmup
from datetime import datetime
import numpy as np
from sklearn.metrics import precision_recall_fscore_support,
accuracy_score, confusion_matrix
import logging
import torch.nn as nn
from tqdm import tqdm, trange
import re
from sklearn.model_selection import KFold
import seaborn as sns
import matplotlib.pyplot as plt
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

df = pd.read_csv("dataset.csv")
df = df.head(22000)
df.info()
def preprocess_text(text):
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

def preprocess_df(df):
    features = ['log']
    target_labels = ['timestamp', 'LineId', 'Component', 'Content',
'EventId', 'EventTemplate']
    label_encoders = {}
    label_columns = []

    df['log'] = df['log'].apply(preprocess_text)
```

```
    for col in target_labels:
        le = LabelEncoder()
        df[col + '_encoded'] = le.fit_transform(df[col])
        label_encoders[col] = le
        label_columns.append(col + '_encoded')

        logger.info(f"Column: {col}")
        logger.info(f"Unique values: {len(np.unique(df[col +
'_encoded']))}")
        logger.info(f"Min: {df[col + '_encoded'].min()}, Max:
{df[col + '_encoded'].max()}")
        logger.info("---")

    return df, label_columns, label_encoders

df, label_columns, label_encoders = preprocess_df(df)
class LogDataset(Dataset):
    def __init__(self, logs, labels, tokenizer, max_len):
        self.logs = logs
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len
        self.num_labels = [len(np.unique(labels[:, i])) for i in
range(labels.shape[1])]

    def __len__(self):
        return len(self.logs)

    def __getitem__(self, item):
        log = str(self.logs[item])
        labels = self.labels[item]

        encoding = self.tokenizer.encode_plus(
            log,
            add_special_tokens=True,
            max_length=self.max_len,
            return_token_type_ids=False,
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt',
        )
```

```
        return {
            'log_text': log,
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'labels': torch.tensor(labels, dtype=torch.long)
        }
class MultiTaskRobertaModel(nn.Module):
    def __init__(self, model_name, num_labels):
        super(MultiTaskRobertaModel, self).__init__()
        self.roberta = RobertaModel.from_pretrained(model_name)
        self.dropout = nn.Dropout(0.1)
        self.classifiers =
nn.ModuleList([nn.Linear(self.roberta.config.hidden_size, num_label)
for num_label in num_labels])

    def forward(self, input_ids, attention_mask, labels=None):
        outputs = self.roberta(input_ids=input_ids,
attention_mask=attention_mask)
        sequence_output = outputs[0]
        pooled_output = self.dropout(sequence_output[:, 0])

        logits = [classifier(pooled_output) for classifier in
self.classifiers]

        if labels is not None:
            loss_fct = nn.CrossEntropyLoss()
            losses = [loss_fct(logit, label) for logit, label in
zip(logits, labels.T)]
            loss = sum(losses)
            return loss, logits
        return logits

print("Unique label values:", np.unique(df[label_columns].values))
print("Max label value:", np.max(df[label_columns].values))
print("Min label value:", np.min(df[label_columns].values))
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
num_labels = [len(np.unique(df[col])) for col in label_columns]
print("Number of labels for each task:", num_labels)
model = MultiTaskRobertaModel('roberta-base', num_labels)

logs = df['log'].values.tolist()
```

```
labels = df[label_columns].values

dataset = LogDataset(logs, labels, tokenizer, max_len=128)
device = torch.device('cuda' if torch.cuda.is_available() else
'cpu')
model.to(device)

num_epochs = 10
batch_size = 16
learning_rate = 2e-5
num_warmup_steps = 0
num_folds = 5

kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)
best_model = None
best_val_loss = float('inf')

for fold, (train_idx, val_idx) in enumerate(kf.split(dataset)):
    print(f"Fold {fold + 1}/{num_folds}")

    train_subsampler =
torch.utils.data.SubsetRandomSampler(train_idx)
    val_subsampler = torch.utils.data.SubsetRandomSampler(val_idx)

    train_loader = DataLoader(dataset, batch_size=batch_size,
sampler=train_subsampler)
    val_loader = DataLoader(dataset, batch_size=batch_size,
sampler=val_subsampler)

    model = MultiTaskRobertaModel('roberta-base',
num_labels).to(device)

    optimizer = AdamW(model.parameters(), lr=learning_rate,
correct_bias=False)
    total_steps = len(train_loader) * num_epochs
    scheduler = get_linear_schedule_with_warmup(optimizer,
num_warmup_steps=num_warmup_steps, num_training_steps=total_steps)

    for epoch in trange(num_epochs, desc="Epochs"):
        model.train()
        total_loss = 0
        all_preds = [[] for _ in range(len(label_columns))]
```

```
        all_labels = [[] for _ in range(len(label_columns))]

        progress_bar = tqdm(train_loader, desc=f"Epoch {epoch +
1}/{num_epochs}")
        for batch in progress_bar:
            optimizer.zero_grad()
            inputs = {
                'input_ids': batch['input_ids'].to(device),
                'attention_mask':
batch['attention_mask'].to(device),
            }
            labels = batch['labels'].to(device)
            loss, logits = model(**inputs, labels=labels)
            total_loss += loss.item()
            loss.backward()
            torch.nn.utils.clip_grad_norm_(model.parameters(),
max_norm=1.0)
            optimizer.step()
            scheduler.step()

            for i, logit in enumerate(logits):
                preds = logit.argmax(dim=-1).detach().cpu().numpy()
                all_preds[i].extend(preds)
                all_labels[i].extend(labels[:, i].cpu().numpy())

            progress_bar.set_postfix({'loss': f"{loss.item():.4f}"})

        avg_loss = total_loss / len(train_loader)

        model.eval()
        val_loss = 0
        val_preds = [[] for _ in range(len(label_columns))]
        val_labels = [[] for _ in range(len(label_columns))]

        with torch.no_grad():
            for batch in val_loader:
                inputs = {
                    'input_ids': batch['input_ids'].to(device),
                    'attention_mask':
batch['attention_mask'].to(device),
                }
                labels = batch['labels'].to(device)
```

```
                loss, logits = model(**inputs, labels=labels)
                val_loss += loss.item()

                for i, logit in enumerate(logits):
                    preds = logit.argmax(dim=-
1).detach().cpu().numpy()
                    val_preds[i].extend(preds)
                    val_labels[i].extend(labels[:, i].cpu().numpy())

        avg_val_loss = val_loss / len(val_loader)

        logger.info(f"Fold {fold + 1}, Epoch {epoch +
1}/{num_epochs}")
        logger.info(f"Training Loss: {avg_loss:.4f}, Validation
Loss: {avg_val_loss:.4f}")

        for i, (preds, labels) in enumerate(zip(val_preds,
val_labels)):
            precision, recall, f1, _ =
precision_recall_fscore_support(labels, preds, average='weighted')
            accuracy = accuracy_score(labels, preds)

            logger.info(f"Task {i+1} - Accuracy: {accuracy:.4f},
Precision: {precision:.4f}, Recall: {recall:.4f}, F1: {f1:.4f}")

            cm = confusion_matrix(labels, preds)
            plt.figure(figsize=(10, 8))
            sns.heatmap(cm, annot=True, fmt='d')
            plt.title(f'Confusion Matrix for Task {i+1}')
            plt.ylabel('Actual')
            plt.xlabel('Predicted')

plt.savefig(f'confusion_matrix_fold{fold+1}_task{i+1}_epoch{epoch+1}
.png')
            plt.close()

        if avg_val_loss < best_val_loss:
            best_val_loss = avg_val_loss
            best_model = model.state_dict()
            torch.save(best_model, './best_model.pth')
            print(f"New best model saved with validation loss:
{best_val_loss}")
```

```
print(f"Best model had a validation loss of {best_val_loss}")

model.load_state_dict(torch.load('./best_model.pth'))

tokenizer.save_pretrained('./tokenizer')
device = torch.device("cpu")
model = MultiTaskRobertaModel('roberta-base', [len(le.classes_) for
le in label_encoders.values()])
model.load_state_dict(torch.load('best_model.pth',
map_location=device))
model.eval()
tokenizer = RobertaTokenizer.from_pretrained('./tokenizer')
from collections import OrderedDict

def predict_log_details(log_text, model, tokenizer, label_encoders,
device):
    model.eval()
    inputs = tokenizer(preprocess_text(log_text),
return_tensors="pt", truncation=True, padding=True, max_length=128)
    inputs = {k: v.to(device) for k, v in inputs.items()}
    with torch.no_grad():
        logits = model(**inputs)

    predictions = [logit.cpu().numpy().argmax(axis=1)[0] for logit
in logits]
    confidence_scores = [torch.softmax(logit, dim=1).max().item()
for logit in logits]

    interpreted_predictions = OrderedDict([
        ('LineId',
label_encoders['LineId'].inverse_transform([predictions[1]])[0]),
        ('timestamp',
label_encoders['timestamp'].inverse_transform([predictions[0]])[0]),
        ('Component',
label_encoders['Component'].inverse_transform([predictions[2]])[0]),
        ('Content',
label_encoders['Content'].inverse_transform([predictions[3]])[0]),
        ('EventId',
label_encoders['EventId'].inverse_transform([predictions[4]])[0]),
```

```
        ('EventTemplate',
label_encoders['EventTemplate'].inverse_transform([predictions[5]])[
0])
    ])

    result = OrderedDict()
    result['input_log'] = log_text
    result['predictions'] = OrderedDict()

    for i, (key, value) in
enumerate(interpreted_predictions.items()):
        result['predictions'][key] = {
            'value': value,
            'confidence': f"{confidence_scores[i]:.4f}"
        }

    return result

new_log = "5,2016-09-28,04:30:31,Info,CBS,Ending TrustedInstaller
initialization.,E17,Ending TrustedInstaller initialization."
predicted_details = predict_log_details(new_log, model, tokenizer,
label_encoders, device)

def print_predictions(predictions):
    print("Input Log:")
    print(f"  {predictions['input_log']}")
    print("\nPredictions:")
    for key, value in predictions['predictions'].items():
        print(f"  {key}:")
        print(f"    Value: {value['value']}")
        print(f"    Confidence: {value['confidence']}")
        print()

print_predictions(predicted_details)
def batch_predict_log_details(log_texts, model, tokenizer,
label_encoders, device, batch_size=32):
    model.eval()
    all_predictions = []

    for i in range(0, len(log_texts), batch_size):
        batch_texts = log_texts[i:i+batch_size]
```

```
        batch_texts = [preprocess_text(text) for text in
batch_texts]
        inputs = tokenizer(batch_texts, return_tensors="pt",
truncation=True, padding=True, max_length=128)
        inputs = {k: v.to(device) for k, v in inputs.items()}

        with torch.no_grad():
            logits = model(**inputs)

        batch_predictions = [logit.cpu().numpy().argmax(axis=1) for
logit in logits]
        confidence_scores = [torch.softmax(logit,
dim=1).max(dim=1)[0].cpu().numpy() for logit in logits]

        for j, original_text in
enumerate(log_texts[i:i+batch_size]):
            predictions = [pred[j] for pred in batch_predictions]
            scores = [score[j] for score in confidence_scores]

            result = OrderedDict()
            result['input_log'] = original_text
            result['predictions'] = OrderedDict()

            for k, (key, le) in enumerate(label_encoders.items()):
                result['predictions'][key] = {
                    'value':
le.inverse_transform([predictions[k]])[0],
                    'confidence': f"{scores[k]:.4f}"
                }

            all_predictions.append(result)

    return all_predictions
batch_logs = ["03-17 16:13:38.811  1702  2395 D WindowManager:
printFreezingDisplayLogsopening app wtoken = AppWindowToken{9f4ef63
token=Token{a64f992 ActivityRecord{de9231d u0
com.tencent.qt.qtl/.activity.info.NewsDetailXmlActivity t761}}},
allDrawn= false, startingDisplayed =  false, startingMoved =  false,
isRelaunching =  false",
             "1117838976 2005.06.03 R02-M1-N0-C:J12-U11 2005-06-03-
15.49.36.156884 R02-M1-N0-C:J12-U11 RAS KERNEL INFO instruction
cache parity error corrected",
```

```
"2015-10-18 18:01:50,572 INFO [main]
org.apache.hadoop.yarn.event.AsyncDispatcher: Registering class
org.apache.hadoop.mapreduce.v2.app.commit.CommitterEventType for
class
org.apache.hadoop.mapreduce.v2.app.commit.CommitterEventHandler",
"081109 204842 663 INFO dfs.DataNode$DataXceiver:
Receiving block blk_1724757848743533110 src: /10.251.111.130:49851
dest: /10.251.111.130:50010",
"20171223-
22:15:29:950|Step_SPUtils|30002312|setTodayTotalDetailSteps=15140384
40000##7008##548365##8661##12456##27174269",
"360778 node-130 unix.hw state_change.unavailable
1141108031 1 Component State Change: Component \042alt0\042 is in
the unavailable state (HWID=2478)",
"Jun 15 02:04:59 combo sshd(pam_unix)[20892]:
authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser=
rhost=220-135-151-1.hinet-ip.hinet.net  user=root",
"Jul  1 09:04:37 authorMacBook-Pro symptomsd[215]:
__73-[NetworkAnalyticsEngine
observeValueForKeyPath:ofObject:change:context:]_block_invoke
unexpected switch value 2",
"Dec 10 07:08:28 LabSZ sshd[24208]: reverse mapping
checking getaddrinfo for ns.marryaldkfaczcz.com [173.234.31.186]
failed - POSSIBLE BREAK-IN ATTEMPT!",
"nova-compute.log.1.2017-05-16_13:55:31 2017-05-16
00:00:04.500 2931 INFO nova.compute.manager [req-3ea4052c-895d-4b64-
9e2d-04d64c4d94ab - - - - -] [instance: b9000564-fe1a-409b-b8cc-
1e88b294cd1d] VM Started (Lifecycle Event)",
"[10.30 16:49:08] chrome.exe -
proxy.cse.cuhk.edu.hk:5070 close, 1190 bytes (1.16 KB) sent, 1671
bytes (1.63 KB) received, lifetime 00:02",
"17/06/09 20:10:41 INFO storage.DiskBlockManager:
Created local directory at
/opt/hdfs/nodemanager/usercache/curi/appcache/application_1485248649
253_0147/blockmgr-70293f72-844a-4b39-9ad6-fb0ad7e364e4",
"1131566461 2005.11.09 dn700 Nov 9 12:01:01
dn700/dn700 crond(pam_unix)[2912]: session opened for user root by
(uid=0)",
"2016-09-28 04:30:31, Info                  CBS
SQM: Failed to start upload with file pattern:
C:\Windows\servicing\sqm\*_std.sqm, flags: 0x2 [HRESULT = 0x80004005
- E_FAIL]",
```

```
              "2015-07-29 19:13:27,721 - WARN
[RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] -
Connection broken for id 188978561024, my id = 1, error = ",
              ]
batch_predictions = batch_predict_log_details(batch_logs, model,
tokenizer, label_encoders, device)
def print_batch_predictions(predictions):
    for i, prediction in enumerate(predictions):
        print(f"Log {i + 1}:")
        print(f"  Input: {prediction['input_log']}")
        print("  Predictions:")
        for key, value in prediction['predictions'].items():
            print(f"    {key}:")
            print(f"      Value: {value['value']}")
            print(f"      Confidence: {value['confidence']}")
        print()

print_batch_predictions(batch_predictions)
```

References

- O’REILLY Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow Book
- https://www.geeksforgeeks.org/
- https://www.elastic.co/elasticsearch
- https://www.elastic.co/kibana
- https://www.youtube.com/
- https://huggingface.co/