Objectif : Analyse des logs par des techniques d'apprentissage automatique

- Partie 2 : comparaison - suite

Idée : visualiser graphiquement le Template ID en fonction du temps pour regarder l'évolution et corréler les anomalies.

Prétraitement : modification de csv2Deeplog pour faciliter le traitement et rajout des labels correspondant à chaque log :

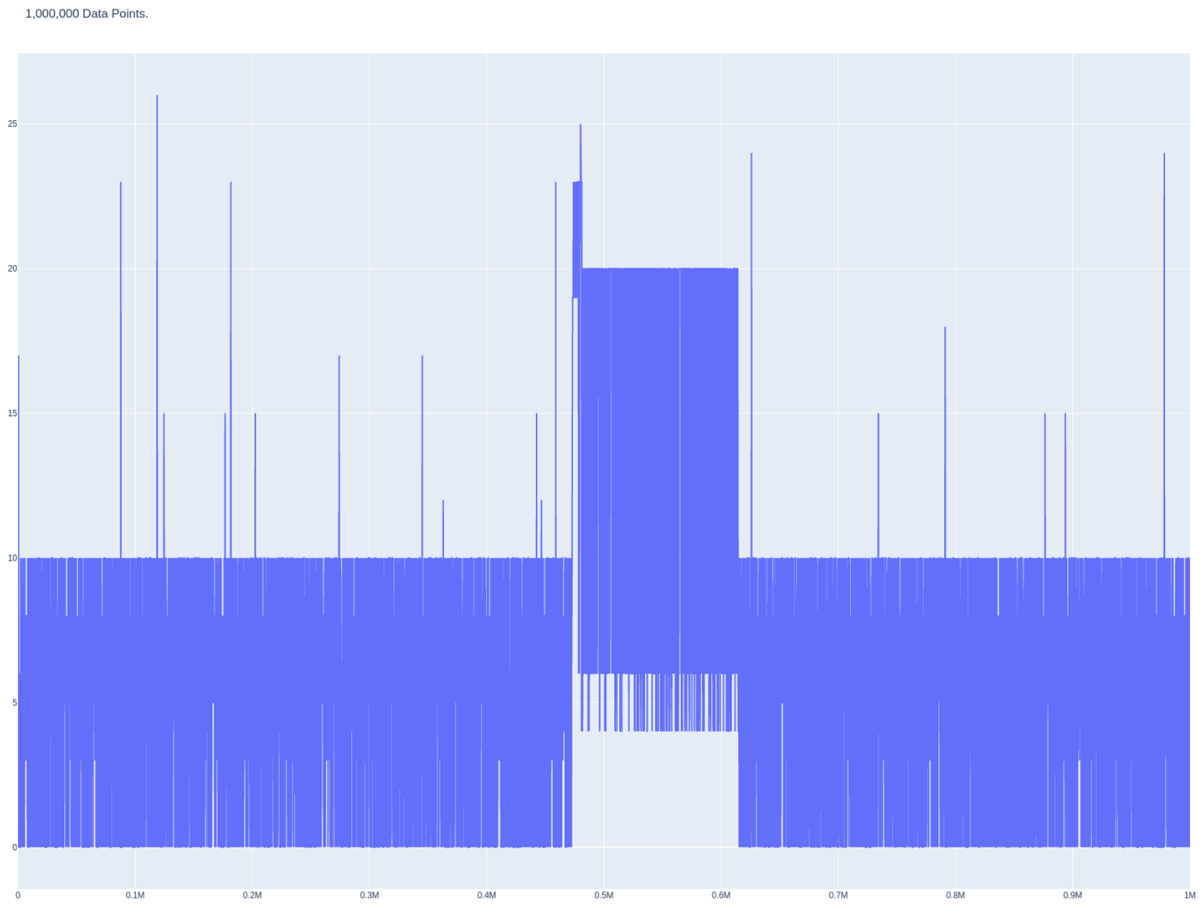
```
if args.format == "HDFS" or args.format == "Hadoop":
    struct_log = pd.read_csv(args.csvIn, engine="c", na_filter=False, memory_map=True)
    column_idx = {col: idx for idx, col in enumerate(struct_log.columns)}
    for _, row1 in enumerate(struct_log.values):
        match.append(re.findall(r"(blk_-?\d+)", row1[column_idx["Content"]]))

    for block_id_list in match:
        block_id = block_id_list[0]
        labeldata = label_data_dict[block_id]
        matched_labels.append(labeldata)


    with open(args.csvOut, 'w', newline='') as outfile:
        fieldnames = ["timestamp", "machine", "event", "anomaly"]
        writer = csv.DictWriter(outfile, fieldnames=fieldnames)
        writer.writeheader()

        # Loop
        for row in csv_reader:
            event_id = row[7].replace("E", "")

            timestamp = row[0]
            #print(int(row[0][0]))
            #session_dict[row[0]]["label"] = label_data_dict[row[0]]

            writer.writerow({"timestamp": timestamp, "machine": "", "even
```

et plot de Event_ID en fonction du temps complexe car plusieurs millions de données à traiter → Matplotlib impossible.

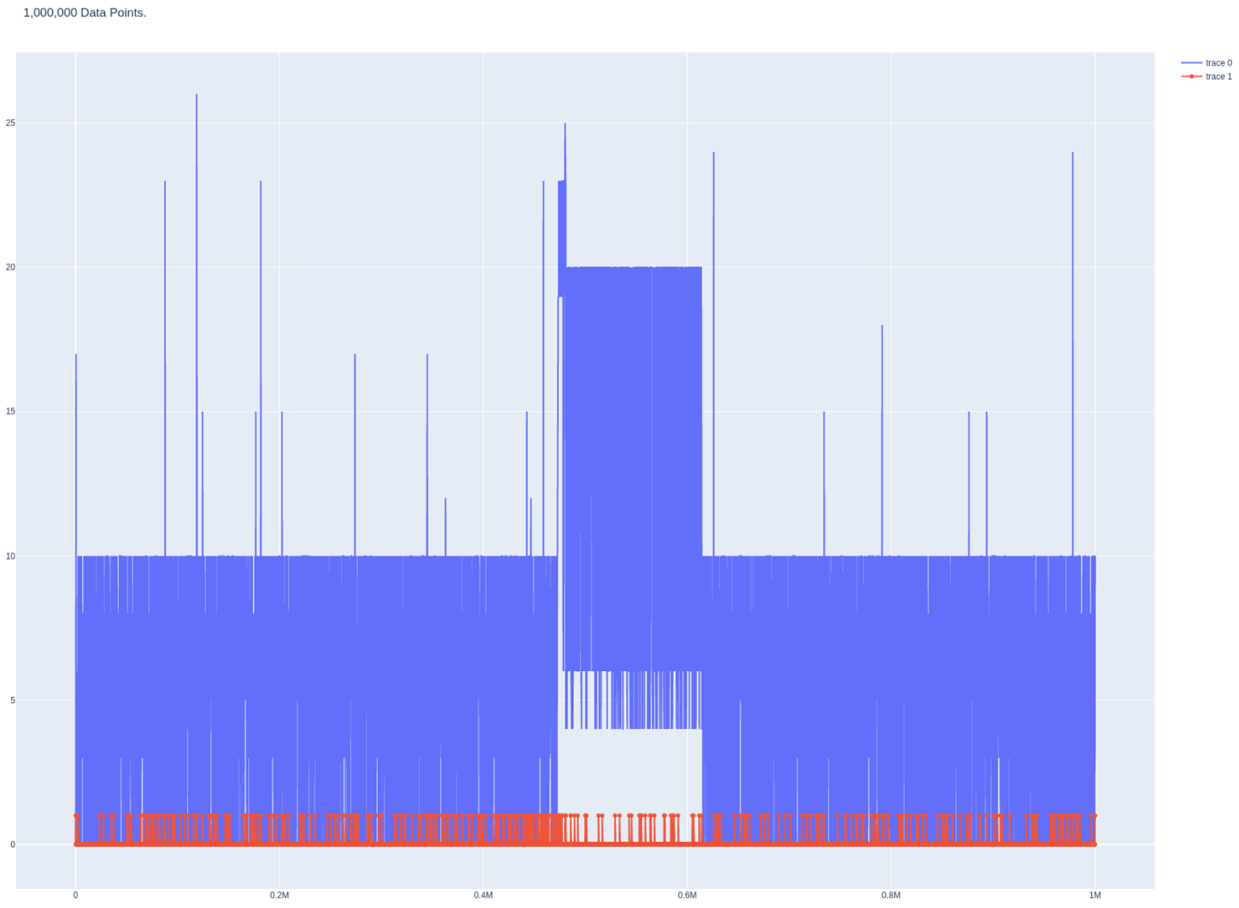
timestamp	machine	event	anomaly
1		0	0
2		3	0
3		0	0
4		0	0
5		5	0
6		5	0
7		8	0
8		8	0
9		5	0
10		8	0
11		10	0
12		10	0
13		10	0
14		0	0
15		0	0
16		3	0
17		12	0
18		0	0
19		0	0
20		0	1

Utilisation de la librairie Lenspy qui est capable de s'occuper de grandes séries temporelles.

Ce qui donne ceci :



Avec les labels superposés :



Utilisable ?

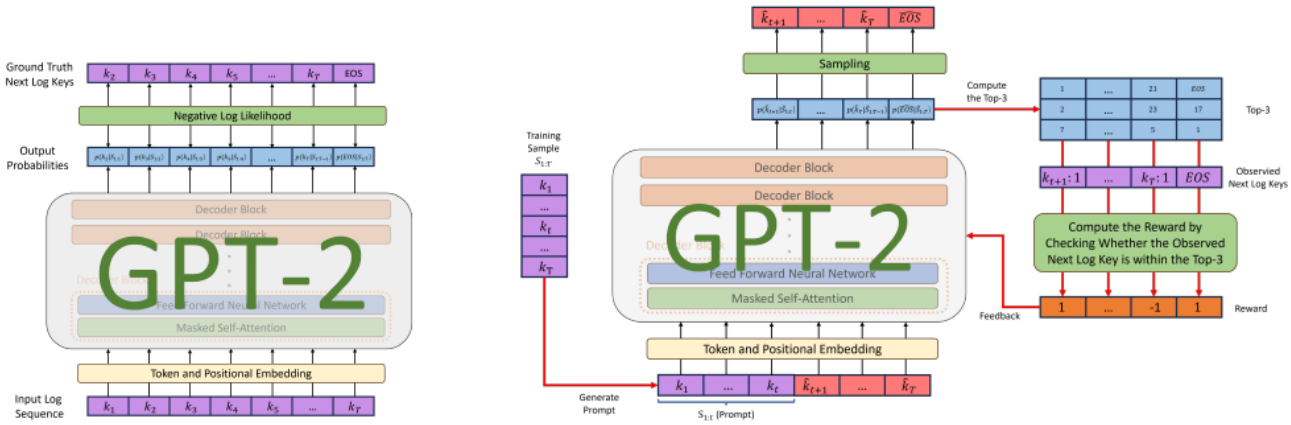
Suite : tests de tous les algorithmes sur tous les types de logs disponibles

Supervised													
Nom	HDFS	Apache	Hadoop	HealthApp	HPC	Linux	Mac	OpenSSH	OpenStack	Proxifier	Zookeeper	BGL	Thunderbird (10M)
LogRobust	96.00%	//	90.00%	//	//	//	//	//		//	//	97.01%	
CNN	95.59%	//	90.00%	//	//	//	//	//		//	//	96.23%	
LogBert	77.81%	//		//	//	//	//	//		//	//	89.47%	
LogGPT	99.33%	//		//	//	//	//	//		//	//	99.13%	
SVM	91.08%	//	88.00%	//	//	//	//	//		//	//	64.10%	
DecisionTree	99.53%	//	86.96%	//	//	//	//	//		//	//	6.75%	
Semi-supervised													
Nom	HDFS	Apache	Hadoop	HealthApp	HPC	Linux	Mac	OpenSSH	OpenStack	Proxifier	Zookeeper	BGL	Thunderbird (10M)
DeepCase	89.34%											XX	
PLELog	97.35%											88.91%	
Unsupervised													
Nom	HDFS	Apache	Hadoop	HealthApp	HPC	Linux	Mac	OpenSSH	OpenStack	Proxifier	Zookeeper	BGL	Thunderbird (10M)
PCA	41.92%		91.49%									40.20%	
DeepLog	84.71%	88.56%	42.04%	81.34%	12.86%	21.74%	14.49%	62.45%	60.26%	99.06%	64.05%	86.31%	
LogAnomaly	31.50%											XX	
AutoEncoder	86.42%											77.15%	
LogCluster	87.49%		XX									89.80%	
Iforest	38.14%		XX									32.50%	
InvariantsMiner	14.06%		XX									41.12%	

XX : erreur ou autre
// : absence de labels

→ nécessité d'adapter toutes les fonctions pour pouvoir les appliquer sur tous les logs, très chronophage et résultats potentiellement discutables.

Nouveauté : LogGPT [1]



(a) Pre-training (b) Fine-tuning

Excellents résultats.

Références :

[1] : Xiao Han, Shuhan Yuan, Mohamed Trabelsi. “LogGPT: Log Anomaly Detection via GPT ”. ArXiv 2309.14482, 2023.