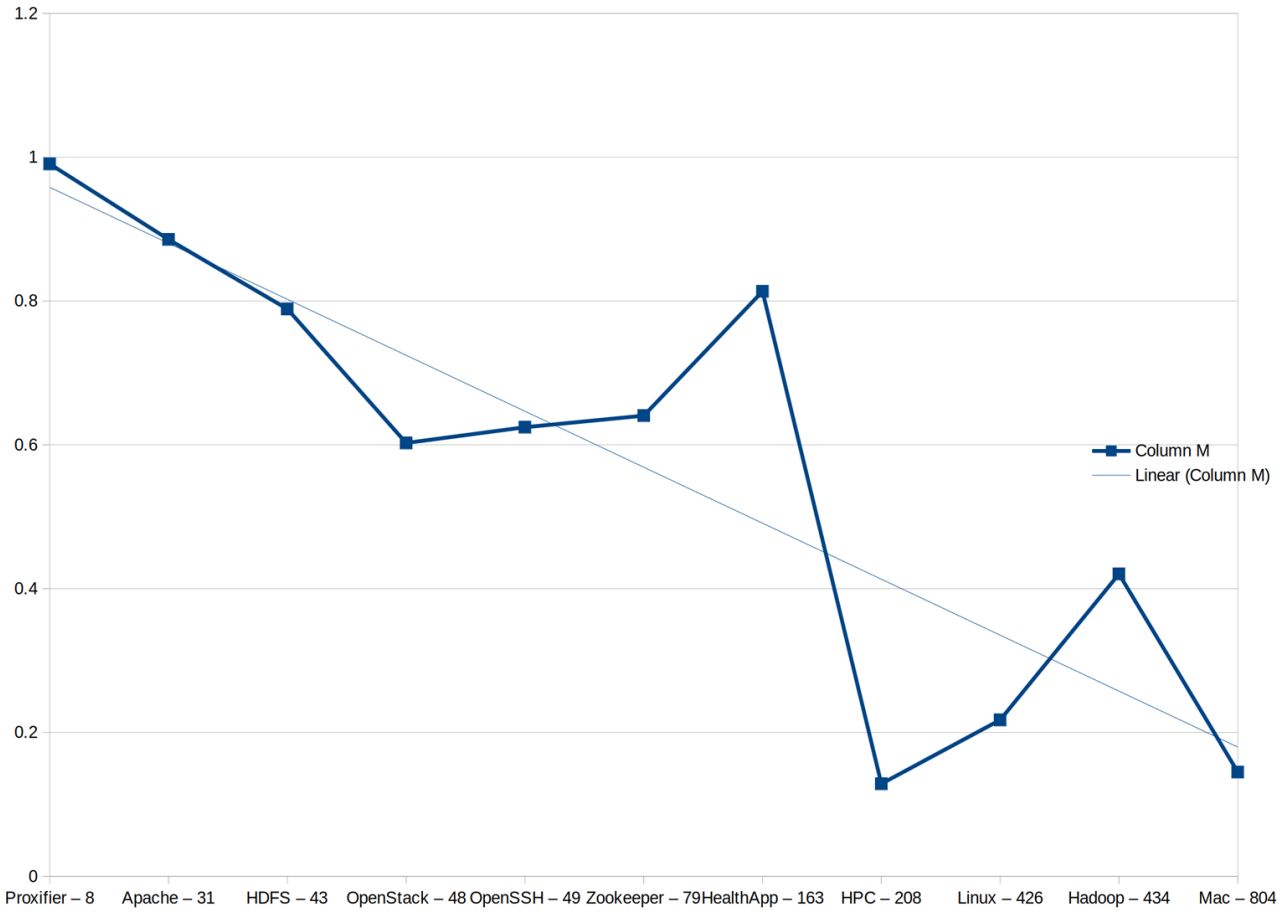
Objectif : Analyse des logs par des techniques d'apprentissage automatique

- Partie 2 : comparaison - suite

retest sur DeepLog [1] avec les fichiers complets de logs :

	DeepLog							
	F1					avg		classes
HDFS	0.7889	0.789	0.789	0.789	0.7889	**0.78896**		43
Apache	0.8594	0.861	0.9857	0.861	0.861	**0.88562**		31
Hadoop	0.4241	0.4183	0.4202	0.4194	0.4202	**0.42044**		434
HealthApp	0.8082	0.8139	0.8155	0.8155	0.8139	**0.8134**		163
HPC	0.1286	0.1286	0.1286	0.1286	0.1286	**0.1286**		208
Linux	0.2174	0.2174	0.2174	0.2174	0.2174	**0.2174**		426
Mac	0.149	0.1418	0.1447	0.1497	0.1391	**0.14486**		804
OpenSSH	0.6154	0.6154	0.6154	0.6607	0.6154	**0.62446**		49
OpenStack	0.6038	0.5977	0.6038	0.6038	0.6038	**0.60258**		48
Proxifier	0.9882	1	0.9882	0.9882	0.9882	**0.99056**		8
Zookeeper	0.6406	0.6406	0.6404	0.6406	0.6404	**0.64052**		79
BGL	0.3661	0.3671				**0.3666**		452

corrélation F1 / nombre de classes :



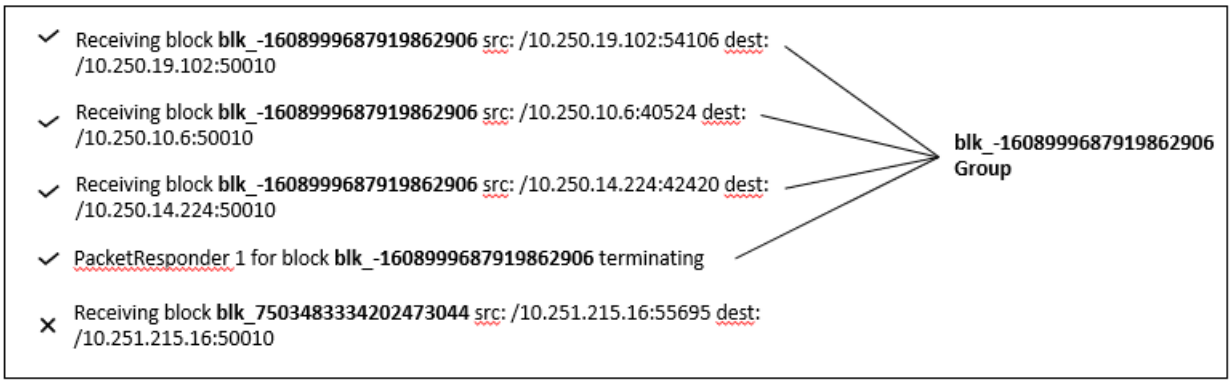
nombre de classes = nombre de template différents.

Problème : tous ne peuvent que faire HDFS sauf DeepLog car tous ont besoin des anomalies labélisées (déjà connues).

Essais de différents algorithmes sur HDFS :

CNN [2] :

Transformation HDFS vers suite logique connues et inconnues.



blk_-1608999687919862906 Group

[E3, E3, E1, E4, E6, E1, E1, E10, E12, E13, E12, E1, E1, E1]

blk_7503483334202473044 Group

[E10, E1, E1, E4, E3, E1, E1, E10, E1, E1, E1, E2, E20, E14, E1, E1, E1, E1, ...]

blk_4521112344874241126 Group

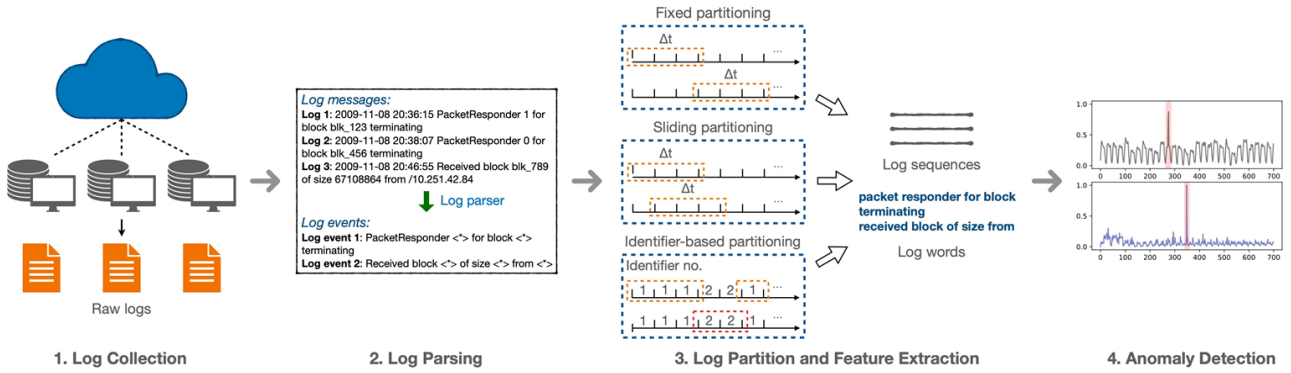
[E1, E1, E1, E1, E1, E40]

EX corressond au template structuré issu de la templatisation (Brain [3] ici)

→ ne fonctionne pas (erreur) mais création du set testable sur DeepCase [4] et DeepLog en txt.

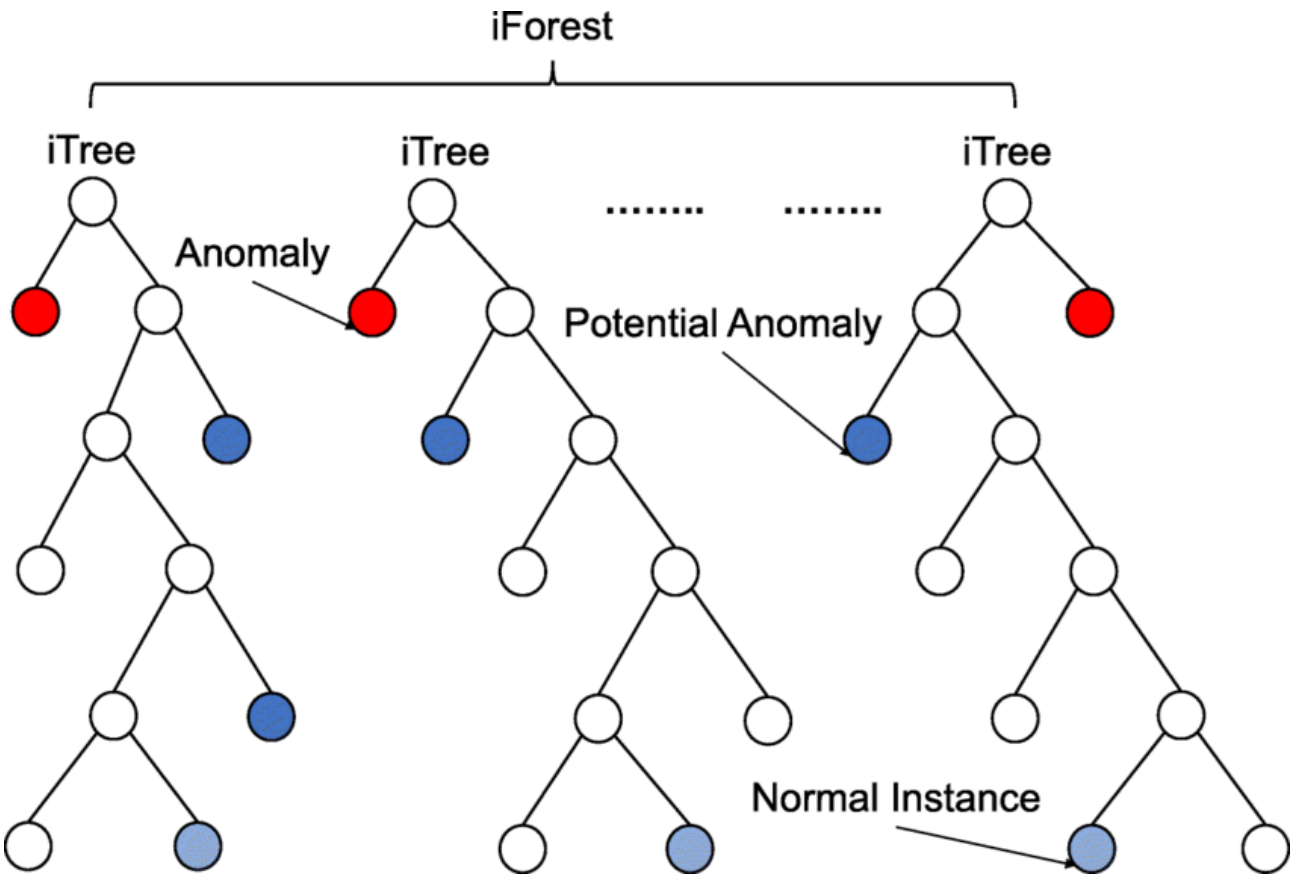
LogLizer [5] :
Créé par LogPAI, auteurs de LogHub.

Fonctionnement :

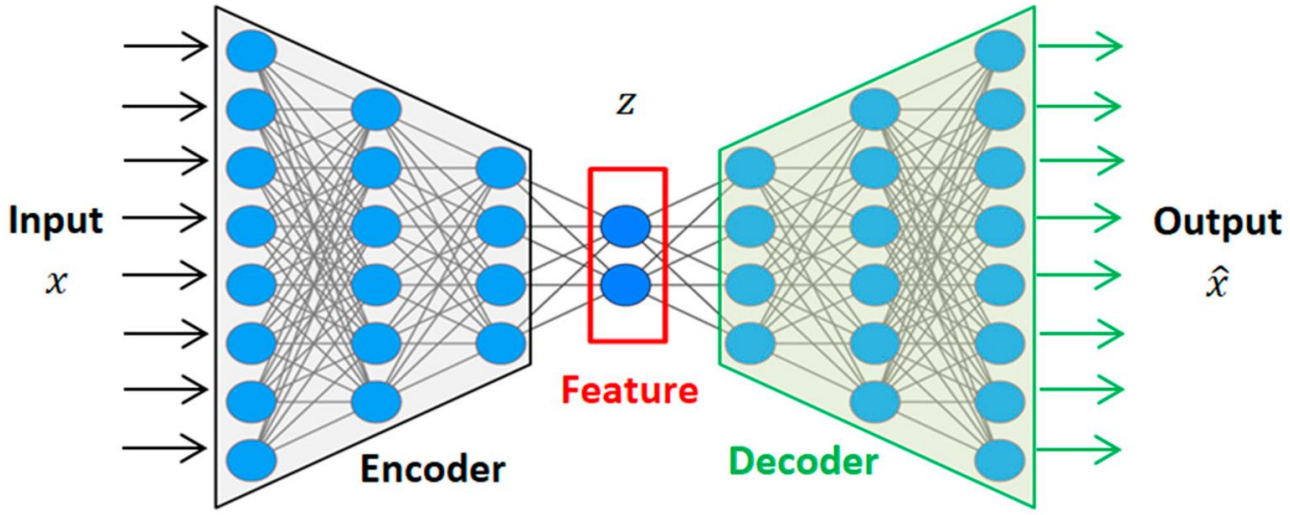


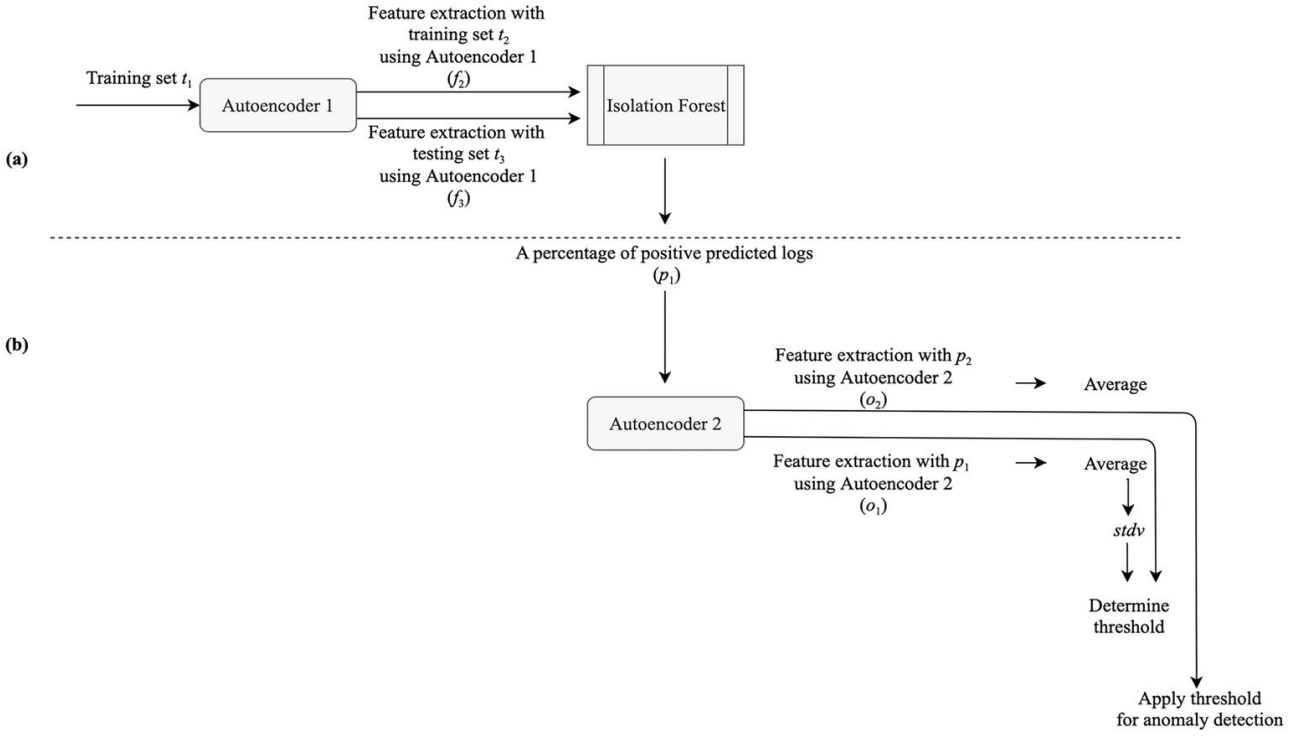
utilisation des labels sur HDFS également.
→ propose de comparer DeepLog (LSTM unsupervised), AutoEncoder* [6], LogRobust [7] (LSTM supervised) et CNN.

* AutoEncoder :
- Isolation Forest pour détecter des suitse positives plutôt que des anomalies



- Réseau AutoEncoder pour extraire les features





Résultats sur les 4 :

	autoencoder	CNN	LSTM sup	LSTM unsup
	0.88	0.936	0.941	0.731
	0.878	0.971	0.954	0.91
	0.879	0.962	0.961	0.843
	0.897	0.957	0.966	0.875
	0.882	0.959	0.956	0.872
	0.879	0.935	0.961	0.923
	0.869	0.969	0.959	0.818
	0.705	0.962	0.94	0.828
	0.879	0.949	0.958	0.824
	0.888	0.959	0.963	
	0.879		0.959	
	0.867		0.959	
	0.853		0.961	
avg	0.864230769	0.9559	0.956769231	0.847111111

Très bons résultats sur tous mais label sur HDFS nécessaire.
→ chercher sur AutoEncoder plus loin

Tests sur différents algorithmes classiques :

========== Model: PCA ==========
====== Model summary ======
n_components: 1
Project matrix shape: 43-by-43
SPE threshold: 1

Train validation:
====== Evaluation summary ======
Confusion Matrix: TP: 6735, FP: 166839, TN: 627, FN: 0
Precision: 3.880%, recall: 100.000%, **F1-measure: 7.471%**

Test validation:
====== Evaluation summary ======
Confusion Matrix: TP: 10103, FP: 389175, TN: 1582, FN: 0
Precision: 2.530%, recall: 100.000%, **F1-measure: 4.936%**

========== Model: IsolationForest==========
====== Model summary ======
Train validation:
====== Evaluation summary ======
Confusion Matrix: TP: 6073, FP: 12306, TN: 155160, FN: 662
Precision: 33.043, recall: 90.171, **F1-measure: 48.364**

Test validation:
====== Evaluation summary ======
Confusion Matrix: TP: 9155, FP: 28746, TN: 362011, FN: 948
Precision: 24.155, recall: 90.617, **F1-measure: 38.143**

========== Model: one class SVM ==========
====== Model summary ======
Train validation:
====== Evaluation summary ======
Confusion Matrix: TP: 19, FP: 167466, TN: 0, FN: 6716
Precision: 0.011, recall: 0.282, **F1-measure: 0.022**

Test validation:
====== Evaluation summary ======
Confusion Matrix: TP: 25, FP: 390757, TN: 0, FN: 10078
Precision: 0.006, recall: 0.247, **F1-measure: 0.013**

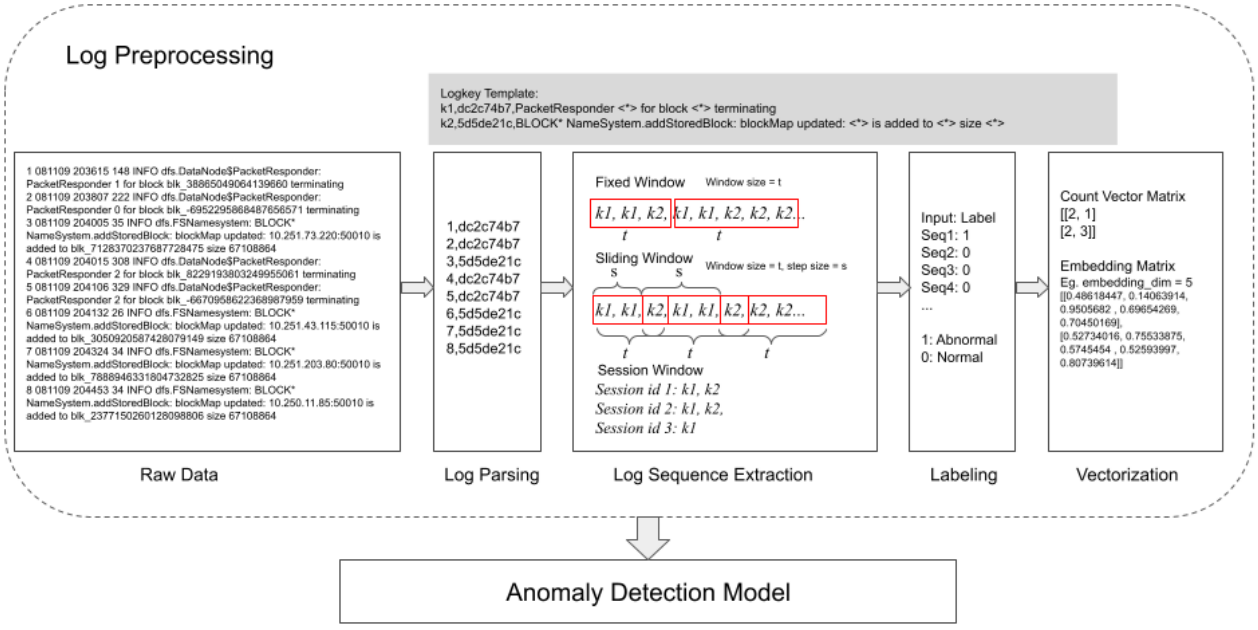
========== Model: LogClustering ==========
====== Model summary ======
Starting offline clustering...
Processed 1000 instances.
Found 4 clusters offline.

Train validation:
====== Evaluation summary ======
Confusion Matrix: TP: 2530, FP: 0, TN: 167466, FN: 4205
Precision: 100.000, recall: 37.565, **F1-measure: 54.614**

Test validation:
====== Evaluation summary ======
Confusion Matrix: TP: 3677, FP: 0, TN: 390757, FN: 6426
Precision: 100.000, recall: 36.395, **F1-measure: 53.367**

/!\ que HDFS également.

LogBert [8] :



results LogBert:
TP: 315, TN: 58094, FP: 79, FN: 1414
Precision: 79.95%, Recall: 18.22%, F1-measure: 29.67%
elapsed_time: 70.74110078811646

Suite : trouver des alternatives à DeepLog qui prennent tous les types de logs en entrée.

Suite : trouver des alternatives à DeepLog qui prennent tous les types de logs en entrée.
PLElog [9] :

Probabilistic Label Estimation

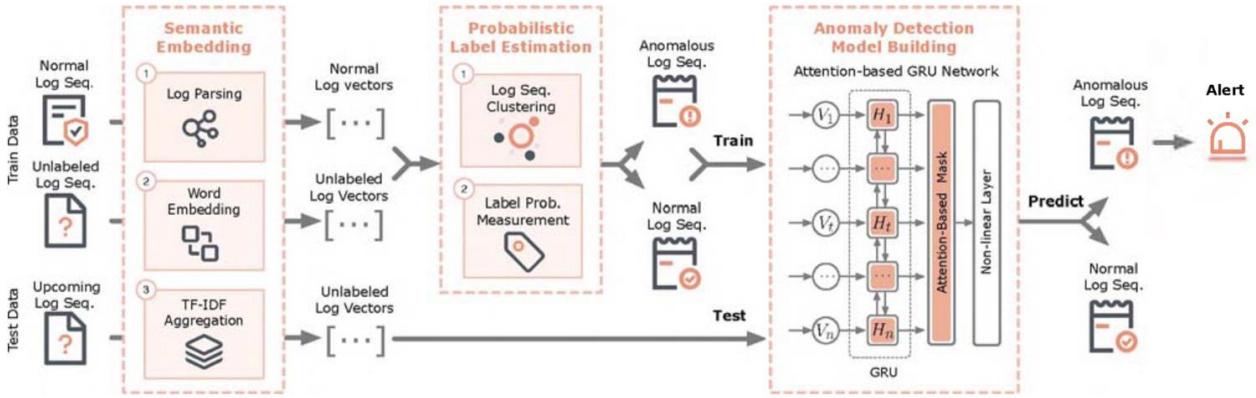


Fig. 2: Overview of PLELog

Sur HDFS complet :

=== Best Model ===
Start evaluating by threshold 0.500
TP: 3602, TN: 168721, FN: 121, FP: 75
Precision = 3602 / 3677 = 97.9603,
Recall = 3602 / 3723 = 96.7499,
F1 score = 97.3514

Récapitulatif sur HDFS :

Outil	DeepLog	DeepCase	AutoEncoder	CNN	PCA	Iforest	SVM	LogClustering	LogBert
F1	93.92%	89.34%	86.42%	95.59%	4.94%	38.14%	0.01%	53.37%	77.81%

Plutôt que de trouver quelque chose qui fonctionne partout, comment adapter les techniques déjà fonctionnelles sur les autres types de logs ?

Références :

[1] : Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS) (pp. 1285-1298).

[2] https://github.com/WraySmith/log-anomaly, 2021

[3] : S. Yu, P. He, N. Chen and Y. Wu, "Brain: Log Parsing With Bidirectional Parallel Tree," in *IEEE Transactions on Services Computing*, vol. 16, no. 5, pp. 3224-3237, Sept.-Oct. 2023, doi: 10.1109/TSC.2023.3270566.

[4] : van Ede, T., Aghakhani, H., Spahn, N., Bortolameotti, R., Cova, M., Continella, A., van Steen, M., Peter, A., Kruegel, C. & Vigna, G. (2022, May). DeepCASE: Semi-Supervised Contextual Analysis of Security Events. In 2022 Proceedings of the IEEE Symposium on Security and Privacy (S&P). IEEE.

[5] : Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R. Lyu. Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection. arXiv preprint, arXiv:2107.05908 (2021).

[6] : A. Farzad, T.A. Gulliver, “Unsupervised log message anomaly detection”, ICT Express, 6 (3) (2020), pp. 229-237

[7] : X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, et al., “Robust log-based anomaly detection on unstable log data,” in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 807–817, 2019.

[8] : H. Guo, S. Yuan, and X. Wu, “Logbert: Log anomaly detection via bert,”
arXiv preprint arXiv:2103.04475, 2021.

[9] : L. Yang *et al.*, "Semi-Supervised Log-Based Anomaly Detection via Probabilistic Label Estimation," *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, Madrid, ES, 2021, pp. 1448-1460, doi: 10.1109/ICSE43902.2021.00130.