Approche de traitement des logs pour la prédiction d'erreurs critiques

Myriam Lopez*, Marie Beurton-Aimar*
Gayo Diallo*,** Sofian Maabout*

*University of Bordeaux, LaBRI, UMR 5800, Talence, France
{myriam.lopez, marie.beurton, sofian.maabout}@labri.fr
**BPH INSERM 1219, Univ. of Bordeaux, F-33000, Bordeaux, France
gayo.diallo@u-bordeaux.fr

1 Introduction

La maintenance prédictive, d'importance capitale pour les fabricants (Hashemian (2011); Salfner et al. (2010)) permet d'une part de réduire les coûts liés à l'immobilisation des systèmes après dysfonctionnement et d'autre part anticiper des commandes des pièces de rechange. De nos jours, la plupart des machines modernes de l'industrie sont équipées de capteurs qui mesurent diverses propriétés physiques telles que la pression de l'huile ou la température du liquide de refroidissement. Après nettoyage et traitement, le signal issu de ces capteurs permet d'identifier au fil du temps les indicateurs d'un fonctionnement anormal (Wang et al. (2015)). La richesse de ces données permet de visualiser l'état du système dans le temps sous la forme d'une estimation de la durée de vie utile restante (Guo et al. (2017)). En parallèle, les machines livrent régulièrement des journaux consignant les différents événements qui permettent de suivre l'usage et les anomalies. Ainsi, la prédiction basée sur les événements est un sujet de recherche clé dans la maintenance prédictive (Wang et al. (2017); Gmati et al. (2019)).

Nous proposons, dans ce document, une approche de préparation des données, permettant d'entraîner un modèle de prédiction d'occurrence d'erreur critique. Elle repose sur l'exploitation de données historiques de journal associées aux machines. L'objectif est de prédire suffisamment tôt l'apparition d'un dysfonctionnement critique afin de faciliter les opérations de maintenance. L'approche est appliquée dans un contexte industriel réel et les performances empiriques obtenues montrent son efficacité.

Après avoir introduit les données et les paramètres clés dans la section suivante, nous détaillons notre méthodologie de préparation. Les expériences menées dans le cadre de nos travaux sont présentées ensuite, puis nous donnons un aperçu des travaux connexes. Enfin, nous concluons et donnons quelques indications pour les travaux futurs.

2 Collecte des données

Soit $\mathcal{F}$ un journal produit par une machine $\mathcal{M}$ où sont rapportées les erreurs émises par la machine suite à l'observation des valeurs d'un ensemble de capteurs. Nous supposons que

les erreurs sont enregistrées à intervalles de temps réguliers. Considérons un jeu d'erreurs $\mathcal{E} = \mathcal{L} \cup \mathcal{H}$ où $\mathcal{L}$ est l'ensemble des erreurs de faible criticité ℓ_j et $\mathcal{H}$ est l'ensemble des erreurs très critiques h_j. Notre objectif est de prédire l'occurrence des erreurs critiques (nous les appellerons erreurs cibles par la suite) selon l'occurrence des erreurs faiblement critiques (ou erreurs faibles). Chaque enregistrement dans $\mathcal{F}$ est une paire $\langle i, E \rangle$ où i est une estampille temporelle exprimée en jours, et $E \in \mathbb{N}^{|\mathcal{E}|}$ est un vecteur où $E[j]$ représente le nombre d'occurrences de l'erreur $e_j \in \mathcal{E}$ à l'estampille temporelle i. Nous utilisons l'exemple suivant tout au long de ce papier pour illustrer notre approche.

Exemple 2.1. Soit $\mathcal{E}$ un ensemble d'erreurs $\mathcal{E} = \{\ell_1, \ell_2, \ell_3, \ell_4\} \cup \{h_1, h_2\}$. La séquence suivante (tableau 1) décrit le contenu de 10 jours d'historique où chaque ligne est l'enregistrement associé à la journée i.

$Estampille$	ℓ_1	ℓ_2	ℓ_3	ℓ_4	h_1	h_2
1	0	12	6	1	0	0
2	0	0	3	2	0	0
3	0	1	4	1	1	1
4	1	0	1	2	0	0
5	0	1	1	2	0	0
6	0	1	1	1	0	0
7	0	1	1	0	0	1
8	1	0	1	8	0	1
9	0	0	6	1	1	0
10	1	0	7	1	1	0

TAB. 1: Exemple d'un journal d'événements log associés à leur estampille temporelle

Pour élaborer un modèle de prédiction et collecter les données, trois paramètres, illustrés dans la figure 1, sont appliqués successivement :

— **Intervalle Prédictif** : il décrit l'historique de données utilisé pour effectuer des prédictions. Sa taille en jours est définie par le paramètre PI. Les informations contenues dans cet intervalle sont rassemblées dans une structure appelée 'sac'.
— **Intervalle de Réactivité** : D'un point de vue pratique, prédire pour le lendemain présente peu d'intérêt. Aussi, une stratégie courante consiste à appliquer un intervalle de réactivité qui, au cours de l'apprentissage, agit sur le modèle comme une contrainte d'anticipation. La taille de l'intervalle, exprimée par le paramètre RI, contrôle le délais d'anticipation souhaité.
— **Intervalle d'Erreur** : Intuitivement, cet intervalle, dont la taille est définie par le paramètre EI, permet de prédire l'occurrence d'une erreur cible au cours d'un intervalle temporel donné, plutôt qu'à une unité de temps spécifique. Il introduit donc un degré d'incertitude sur la temporalité de la prédiction.

FIG. 1: Les trois paramètres appliqués pour la prédiction, produisant le sac B_1 de 3 jours

3 Méthodologie

L'objectif étant d'établir un modèle prédictif binaire, la condition préalable est de former des exemples représentatifs de l'historique, étiquetés OUI ou NON.

Definition 3.1. Soit B_i un sac et h_j une erreur cible. B_i est étiqueté OUI *ssi* l'intervalle $[i + PI + RI; i + PI + RI + EI - 1]$ contient une occurrence de h_j, il est étiqueté NON sinon.

Exemple 3.1. Soient $PI = 3, RI = 2$ et $EI = 2$, l'erreur cible h_1 et l'historique défini par le tableau 1. Pour définir l'étiquette du sac B_1, nous devons vérifier si l'erreur h_1 a été observée au cours des jours $[6; 7]$. Puisque ce n'est pas le cas, B_1 est étiqueté NON. Ainsi, en appliquant la définition ci-dessus, on obtient à partir de l'historique, les sacs B_1 et B_2, étiquetés NON et les sacs B_3 et B_4, étiquetés OUI.
Les sacs 5, 6, 7 et 8 ne peuvent être étiquetés car leur intervalles EI respectifs sont définis en dehors des limites de l'historique. Ils sont donc exclus de l'ensemble d'entraînement.

La configuration ci-dessus (sacs étiquetés) est proche de celle rencontrée dans le Multiple Instance Learning (MIL) Dietterich et al. (1997). Cependant, notre approche n'est pas conforme au MIL car la prédiction est basée sur le sac entier et non sur les exemples individuels. Ainsi, notre méthode alternative consiste à synthétiser l'information de chaque sac B_i, en maximisant les valeurs pour former un exemple unique appelé *méta-instance* [1]

Exemple 3.2. Soient $PI = 3$ et le sac B_1 défini entre $i = 1$ et $i = 3$. B_1 est synthétisé par le vecteur $\langle 0, 12, 6, 2\rangle$, c'est-à-dire que pour chaque ℓ_j nous gardons la valeur maximale dans B_1. Avec $RI = 2$, $EI = 2$, et l'erreur cible h_1 on obtient les *méta-instances* décrites dans le volet droit de la figure 2.

L'étape de pré-traitement est décrite par l'algorithme 1 où la séquence d'enregistrements T est obtenue à partir d'un journal de logs émis par une machine. Les enregistrements consécutifs sont rassemblés dans des sacs par fenêtre glissante de taille PI. Les sacs sont ensuite étiquetés puis leur contenu synthétisé en une *meta-instance*. La séquence T' obtenue en sortie de l'algorithme est utilisée comme donnée d'entrée du modèle d'apprentissage.

La complexité de la préparation des données est linéairement proportionnelle à la taille de la séquence : la boucle la plus extérieure est exécutée $O(|T|)$ fois. À chaque itération, les lignes de données PI sont synthétisées et l'étiquette est attribuée après identification du contenu de la ligne EI. Ainsi, la complexité globale est de $O(|T| \times (PI + EI))$.
On peut également noter que les itérations de la boucle extérieure peuvent être parallélisées puisqu'elles sont indépendantes les unes des autres.

1. Il existe d'autres méthodes de synthèse. Dans le cas d'usage présent, la fonction `MAX()` a été choisie car elle contribue aux bonnes performances de prédiction.

Estampille	ℓ_1	ℓ_2	ℓ_3	ℓ_4	h_1	h_2
1	0	12	6	1	0	0
2	0	0	3	2	0	0
3	0	1	4	1	1	1
4	1	0	1	2	0	0
5	0	1	1	2	0	0
6	0	1	1	1	0	0
7	0	1	1	0	0	1
8	1	0	1	8	0	1
9	0	0	6	1	1	0
10	1	0	7	1	1	0

Bag	ℓ_1	ℓ_2	ℓ_3	ℓ_4	Étiquette h_1
B_1	0	12	6	2	NON
B_2	1	1	4	2	NON
B_3	1	1	4	2	OUI
B_4	1	1	1	2	OUI

FIG. 2: Stratégie de synthèse des sacs utilisant la fonction `MAX()`

Algorithme 1 : DATAPREP

Entrées : Sequence T, EI, PI, RI, erreur cible h
Output : Tableau T' de *méta-instances* étiquetées.
début
  pour $i = PI + RI + 1$ *à* $|T| - EI + 1$ **faire**
    $s \leftarrow$ Synthétiser(T, $i - RI - PI$, $i - RI - 1$, $\mathcal{S}$)
    //un sac de taille PI commençant à l'index $i - RI - PI$ est synthétisé par une fonction $\mathcal{S}$, par exemple `MAX()`
    $s.label \leftarrow False$
    pour $j = i$ *à* $i + EI - 1$ **faire**
      si $T[j]$ *contient une occurrence de* h **alors**
        $s.label \leftarrow True$
    Ajouter s dans T'
  retourner T'

4 Expérimentations et résultats

4.1 Description du jeu de données

Nous avons recueilli les données brutes à partir d'un historique d'un an de journaux, chacun d'eux étant associé à une machine spécifique. [2]. Les machines partagent les même caractéristiques mécaniques mais elles ne sont pas soumises aux mêmes conditions d'utilisation. Avec une combinaison de paramètres (PI; RI; EI), chaque fichier journal est traité à l'aide de l'algorithme 1. Les séquences obtenues sont ensuite fusionnées pour constituer le jeu de données d'entrée de l'apprentissage. Celui-ci est partitionné en un jeu d'entraînement et un jeu de test avec un rapport 80/20, par échantillonnage stratifié.

2. Malheureusement, pour des raisons de confidentialité, nous ne pouvons pas fournir les données de cette étude

Pour chaque erreur cible, nous avons conçu plusieurs modèles, chacun d'eux obtenu par la combinaison de trois valeurs de paramètres $(PI; RI; EI)$. Nos recherches se sont concentrées sur 11 erreurs cibles, dont les jeux de données correspondants sont constitués de 193 *features* (erreurs faibles distinctes). En fonction de l'erreur cible, nous obtenons en moyenne 1000 échantillons, parmi lesquels de 4 à 30 % sont positifs. Bien qu'une technique de correction du déséquilibre de classe soit généralement recommandée dans un tel cas, nos expériences ont montré que le sous-échantillonnage n'a pas apporté de gain de précision substantiel [3]. Nous avons donc décidé de ne pas y recourir. Dans les expériences suivantes, la valeur par défaut pour les deux paramètres PI et RI est fixée à 7 jours. La valeur du paramètre RI, choisie par contrainte métier, représente la durée nécessaire pour effectuer une réparation sur le système.

4.2 Paramètres

Toutes les expériences ont été menées avec le logiciel RapidMiner [4]. Plusieurs algorithmes de classification ont été explorés. L'algorithme de réseau de neurones (NN) du *framework* H2o Candel et al. (2016) a surpassé les autres. On ne présente donc que ce dernier. Nous avons utilisé 4 couches de 100 neurones chacune, 1000 *epoch* et la *Cross-Entropy* comme fonction de coût. Les performances du modèle sont évaluées en termes de F1-score sur la classe positive, défini par la moyenne harmonique de la précision et du rappel.

4.3 Choix de la valeur du paramètre d'Intervalle d'Erreur (EI)

Le paramètre EI introduit une flexibilité pour les prévisions positives et une rigidité pour les prévisions négatives. En effet, une prédiction positive indique qu'au cours d'un intervalle de temps futur, une erreur critique peut se produire, tandis qu'une prédiction négative indique qu'*il n'y aura pas* d'occurrence dans cet intervalle. Intuitivement, pour les prédictions positives, on peut s'attendre à ce que plus EI est élevé, plus les prédictions du modèle sont correctes. Pour analyser cette hypothèse, nous avons fait varier EI de 1 à 4 jours et évalué pour chaque erreur cible, le F1-score de la classe positive. Les résultats sont présentés dans la figure 3.

Bien que le F1-score ne soit pas monotone par rapport à EI, dans l'ensemble, notre hypothèse initiale est confirmée : le F1-score augmente quand EI croît. Dans certains cas, le F1-score est nul (voir les erreurs E3 ; E4 ; E5 ; E10 et E11). Ceci s'explique par une valeur nulle du nombre de vrais positifs. Il est important d'observer qu'il n'y a pas de valeur de EI unique qui optimise le F1-score pour chaque erreur. Ceci plaide pour fixer une valeur de EI par erreur cible.

D'un point de vue applicatif, le taux de faux positifs (FPR) est également un indicateur clé [5]. La figure 4 montre l'évolution du FPR en fonction de EI. On observe qu'en général, le FPR tend à diminuer avec l'augmentation, expliquant les performances obtenues en figure 3.

L'augmentation de la taille de EI multiplie simultanément le nombre d'exemples positifs dans le jeu de données. Cela peut expliquer que nous obtenions de meilleures performances avec un EI plus important. Bien que ces performances puissent aussi dans certains cas masquer une sur-expression de faux positifs, le phénomène n'est pas présent ici, comme le montre la figure 4.

3. Par manque de place, les résultats ne seront pas détaillés dans cet article.
4. Rapidminer.com
5. $FPR = \frac{FP}{FP+TP} = 1 - precision$

FIG. 3: Évolution du F1-score en fonction de la taille du paramètre EI

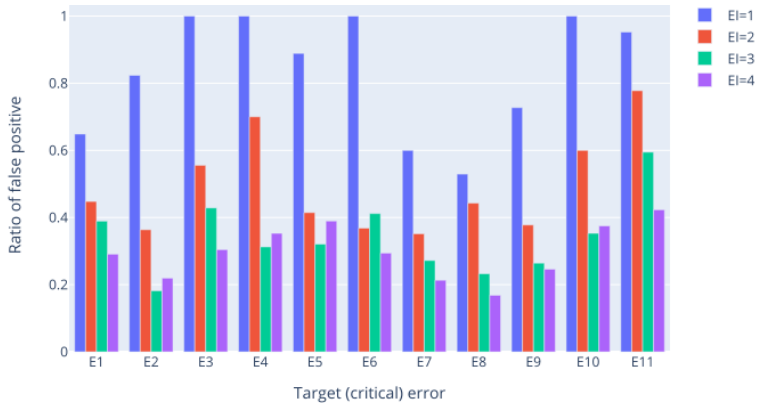

FIG. 4: Proportion de faux-positifs (fausses alarmes) en fonction de la taille du paramètre EI

5 Travaux relatifs

La maintenance prédictive a suscité de nombreuses recherches ces dernières années (Ran et al. (2019); Krupitzer et al. (2020); Zhang et al. (2019)). Les approches basées sur l'apprentissage supervisé, auxquelles notre étude se rapporte, permettent de répondre à ce besoin. À notre connaissance, les travaux de Sipos et al. (2014) et de Korvesis et al. (2018), appartenant à cette catégorie d'approche, sont les plus proches des nôtres.

Bien que similaire à la nôtre, la méthodologie proposée par Sipos et al. (2014) pour étiqueter les sacs, s'applique différemment selon qu'ils soient positifs ou négatifs. En effet, tandis que les sacs positifs sont agrégés par moyenne pour former une méta-instance, les sacs négatifs propagent leur étiquette individuellement sur les instances qu'ils contiennent. Par conséquent, le jeu de données obtenu correspond à un mélange de méta-instances (positives) et d'instances simples (négatives). Les auteurs choisissent la moyenne pour générer les méta-instances afin qu'elles soient, à l'échelle du sac, représentatives du comportement quotidien. En pratique, à

moins d'avoir un faible écart-type dans les sacs, un enregistrement peut être très différent de la moyenne du sac associé. Les estimations du modèle s'avéreront alors imprécises et fausseront les prédictions déduites des instances isolées. Dans notre cas d'usage, les écart-types sont élevés ce qui exclut l'utilisation de la moyenne. Par ailleurs, nous traitons les sacs positifs et négatifs de la même manière, ainsi, nous considérons que l'occurrence ou non d'une erreur critique s'explique par l'ensemble du sac associé, et non pas par des cas isolés de ce sac. Cela nous permet également de réduire le déséquilibre des classes. Dans Sipos et al. (2014), les prédictions sont basées sur les instances puis sont propagées aux sacs. Plus précisément, le modèle cherchera à prédire des instances dont les sacs ne sont pas classifiés. Si l'une des instances du sac est positive celui-ci est alors classé positif, négatif sinon. En somme, la classe du sac est prédite en fonction de la classe de ses instances, alors qu'en phase de préparation, ce sont les étiquettes des instances qui dépendent de celle du sac. Fondamentalement, les prédictions sont quotidiennes parce que l'occurrence d'une erreur est expliquée par l'information à l'échelle d'une journée, alors que nos besoins sont de combiner des événements sur une échelle de temps plus large. C'est aussi pour cette raison que, contrairement à Sipos et al. (2014), notre travail ne s'inscrit pas dans une approche d'apprentissage multi-instance (MIL), dont l'hypothèse principale stipule qu'un sac est positif *ssi* l'une de ses instances est positive Dietterich et al. (1997); Carbonneau et al. (2018) . Sur notre cas d'usage, la méthode de Sipos et al. (2014), ne nous a pas permis d'obtenir un F1-score supérieur à 0,2.

Un travail plus récent avec un processus de préparation des données similaire est celui de Korvesis et al. (2018). Ici le modèle de prédiction est basé sur une régression qui estime la probabilité d'occurrence future d'une erreur. Si cette probabilité est supérieure à un seuil fixé, une alerte est déclenchée. Ainsi, lors de la préparation des données, les étiquettes associées aux sacs sont des valeurs de probabilités, estimées à l'aide d'une fonction sigmoïde. Intuitivement, plus un sac est proche d'une erreur, plus sa probabilité est élevée. Leurs résultats ont démontré des performances largement supérieures à celles de Sipos et al. (2014). Nous pensons que cela est principalement dû à l'application elle même et qu'il n'y a pas de meilleure solution dans tous les cas. Notre solution est bien plus simple ce qui ne l'empêche pas d'atteindre des performances très satisfaisantes.

6 Conclusion et perspectives

Nous avons décrit une approche permettant d'exploiter les données des journaux pour prévoir les erreurs critiques qui peuvent provoquer des défaillances coûteuses de machines outil. Cette approche est basée sur l'agrégation des intervalles temporels qui précèdent ces erreurs. Combinée à un réseau de neurones, notre solution s'avère suffisamment précise. Son principal atout, par rapport à d'autres techniques, est sa simplicité. Même si nous ne prétendons pas qu'elle devrait fonctionner pour chaque situation similaire (prédiction basée sur des logs), nous pensons qu'elle pourrait être considérée comme une base avant d'essayer des options plus sophistiquées. Jusqu'à présent, nous avons conçu un modèle par erreur cible. À l'avenir, nous prévoyons d'analyser plus en profondeur les erreurs pour voir s'il serait possible de combiner différentes prédictions afin de réduire le nombre de modèles. En outre, nous souhaitons automatiser le réglage des paramètres : en donnant un (ensemble de) cible(s), trouver les valeurs optimales de PI, EI et RI de telle sorte que la performance du modèle appris soit maximisée.

Références

Candel, A., V. Parmar, E. LeDell, et A. Arora (2016). Deep learning with H2O. H2O AI Inc.

Carbonneau, M., V. Cheplygina, E. Granger, et G. Gagnon (2018). Multiple instance learning : A survey of problem characteristics and applications. *Pattern Recognit. 77*, 329–353.

Dietterich, T. G., R. H. Lathrop, et T. Lozano-Pérez (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell. 89*(1-2), 31–71.

Gmati, F. E., S. Chakhar, W. L. Chaari, et M. Xu (2019). A Taxonomy of Event Prediction Methods. In *Proc. of IEA/AIE conf.* Springer.

Guo, L., N. Li, F. Jia, Y. Lei, et J. Lin (2017). A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing 240*, 98–109.

Hashemian, H. M. (2011). State-of-the-art predictive maintenance techniques. *IEEE Trans. Instrum. Meas. 60*(1), 226–236.

Korvesis, P., S. Besseau, et M. Vazirgiannis (2018). Predictive maintenance in aviation : Failure prediction from post-flight reports. In *Proc. of ICDE Conf.*

Krupitzer, C., T. Wagenhals, M. Züfle, et al. (2020). A survey on predictive maintenance for industry 4.0. *CoRR abs/2002.08224*.

Ran, Y., X. Zhou, P. Lin, Y. Wen, et R. Deng (2019). A survey of predictive maintenance : Systems, purposes and approaches. *CoRR abs/1912.07383*.

Salfner, F., M. Lenk, et M. Malek (2010). A survey of online failure prediction methods. *ACM Comput. Surv. 42*(3), 10 :1–10 :42.

Sipos, R., D. Fradkin, F. Mörchen, et Z. Wang (2014). Log-based predictive maintenance. In *SIGKDD conference*, pp. 1867–1876. ACM.

Wang, C., H. T. Vo, et P. Ni (2015). An iot application for fault diagnosis and prediction. In *IEEE International Conference on Data Science and Data Intensive Systems*, pp. 726–731.

Wang, J., C. Li, S. Han, S. Sarkar, et X. Zhou (2017). Predictive maintenance based on event-log analysis : A case study. *IBM J. Res. Dev. 61*(1), 11.

Zhang, W., D. Yang, et H. Wang (2019). Data-driven methods for predictive maintenance of industrial equipment : A survey. *IEEE Systems Journal 13*(3), 2213–2227.

Summary

With the advent of Industry 4.0, failure anticipation is becoming one of the key objectives in industrial research. In this context, predictive maintenance is an active research area for various applications. This paper presents an approach to predict high importance errors using log data emitted by machine tools. It uses the concept of bag to summarize events provided by remote machines, available within log files. The idea of bag is inspired by the Multiple Instance Learning paradigm. However, our proposal follows a different strategy to label bags. Three main setting parameters are defined to build the training set allowing the model to fine-tune the trade-off between early warning, historic informativeness and forecast accuracy. The effectiveness of the approach is demonstrated using a real industrial application where critical errors can be predicted e.g., seven days before their occurence with high accuracy.