UQAC
Université du Québec
à Chicoutimi

DÉTECTION DES ÉCARTS DE TENDANCE ET ANALYSE PRÉDICTIVE POUR LE TRAITEMENT DES FLUX D'ÉVÉNEMENTS EN TEMPS RÉEL

PAR MASSIVA ROUDJANE

THÈSE PRÉSENTÉE À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI DANS LE CADRE D'UN PROGRAMME EN EXTENSION DE L'UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS EN VUE DE L'OBTENTION DU GRADE DE PHILOSOPHIÆ DOCTOR (PH.D.) EN INFORMATIQUE

QUÉBEC, CANADA

RÉSUMÉ

Les systèmes d'information produisent différents types de journaux d'événements. Les données historiques contenues dans les journaux d'événements peuvent révéler des informations importantes sur l'exécution d'un processus métier. Le volume croissant de ces données collectées, pour être utile, doit être traité afin d'extraire des informations pertinentes. Dans de nombreuses situations, il peut être souhaitable de rechercher des tendances dans ces journaux. En particulier, les tendances calculées par le traitement et l'analyse de la séquence d'événements générés par plusieurs instances du même processus servent de base pour produire des prévisions sur les exécutions actuelles du processus.

L'objectif de cette thèse est de proposer un cadre générique pour l'analyse des tendances sur ces flux d'événement, en temps réel. En premier lieu, nous montrons comment des tendances de différents types peuvent être calculées sur des journaux d'événements en temps réel, à l'aide d'un cadre générique appelé *workflow de distance de tendance*. De multiples calculs courants sur les flux d'événements s'avèrent être des cas particuliers de ce flux de travail, selon la façon dont différents paramètres de flux de travail sont définis.

La suite naturelle de l'analyse statique des tendances est l'usage des algorithmes d'apprentissage. Nous joignons alors les concepts de traitement de flux d'événements et d'apprentissage automatique pour créer un cadre qui permet le calcul de différents types de prédictions sur les journaux d'événements. Le cadre proposé est générique : en fournissant différentes définitions à une poignée de fonctions d'événement, plusieurs types de prédictions différents peuvent être calculés à l'aide du même flux de travail de base.

Les deux approches ont été mises en œuvre et évaluées expérimentalement en étendant un moteur de traitement de flux d'événements existant, appelé BeepBeep. Les résultats expérimentaux montrent que les écarts par rapport à une tendance de référence peuvent être détectés en temps réel pour des flux produisant jusqu'à des milliers d'événements par seconde.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

REMERCIEMENTS

Cette thèse est le résultat d'efforts et de persévérance de plusieurs années. Toutes mes expériences de vie ont fait que cette thèse soit entre vos mains aujourd'hui. Cela n'aurait certainement pas été possible sans l'aide, l'encouragement et la participation directe ou indirecte à la rédaction de celle-ci de on entourage. Avant de plonger dans les détails scientifiques, je tiens d'abord à rendre hommage aux personnes clés autour de moi.

Tout d'abord, j'ai eu le privilège d'être dirigée par monsieur Sylvain Hallé et codirigée par messieurs Raphaël Khoury et Djamel Rebaine. Je remercie tous les trois de m'avoir donné cette chance de puiser de leur expertise. Votre pédagogie, professionnalisme et patience avec moi ont fait que mon parcours se passe paisiblement et aboutisse à ce manuscrit.

Je remercie ma maman qui a toujours été là pour moi. Ses sacrifices, conseils et sa présence dans ma vie ont fait que ce travail soit possible.

Je tiens à remercier mes amis ici au Canada, en Algérie et en France qui ont essayé de m'égayer lorsque le bout du tunnel semblait plus loin et qui ont été là durant les moments les plus difficiles comme les plus joyeux. Je remercie également mes camarades et collègue du LIF : Asma, Xavier et Quentin. Pour tous les moments d'amitié et de partage qui ont fait que le voyage soit moins solitaire.

Merci

INTRODUCTION

On entend souvent que le monde actuel est devenu une «société d'information» et que, à travers une grande variété de domaines, on s'appuie fortement sur les données dans un éventail d'activités de plus en plus large. L'informatisation croissante de l'activité humaine, combinée aux progrès réalisés dans le domaine des capteurs numériques et des technologies de communication, a largement contribué à l'augmentation fulgurante de la production de données. De nos jours, même le plus petit message, la plus petite facture ou le moindre clic d'un visiteur dans une page sont stockés, classés et rendus disponibles pour des besoins d'analyse de plus en plus pointus.

Les estimations quant au volume de données produites annuellement varient. À titre d'exemple, la plateforme Flickr, un site web de partage de photos, a reçu en 2017 en moyenne 1,63 million de photos par jour [8], pour une taille de stockage quotidienne estimée à près de 4 téraoctets. En 2018, selon Statista [9], plus de 110 millions de comptes Instagram étaient actifs rien qu'aux États-Unis. Le réseau social le plus populaire, Facebook, était le premier à dépasser le milliard de comptes ouverts. En 2019, il comptait 2.4 milliards d'utilisateurs actifs chaque mois [10].

Plus globalement à l'échelle d'Internet, l'organisation Dell EMC (anciennement EMC), qui est spécialisée dans la vente de l'espace de stockage et dans le traitement et l'analyse des données, a estimé en 2014 que d'ici 2020, il se créerait 44 000 milliards de gigaoctets de données chaque année [11]. Cette augmentation de la quantité d'information enregistrée est heureusement accompagnée d'une augmentation des capacités de traitement et de stockage. Ainsi, le coût d'un gigaoctet est passé d'environ 500 000 dollars en 1981 à moins de 3 cents aujourd'hui [12].

Malgré toutes ces avancées technologiques au niveau de la production et du stockage de l'information, il n'en demeure pas moins que ces données, pour être utiles, doivent pouvoir être *traitées* afin d'en extraire une valeur ajoutée. De fait, le volume croissant de données sous différents formats et de sources diverses crée de nouveaux défis en matière de traitement. C'est pourquoi les entreprises, dans les dernières années, ont investi de manière significative dans les infrastructures qui facilitent la transformation, le rechargement et la réutilisation des données, à la fois au niveau du matériel, des logiciels d'entreposage de données, du personnel et de ses compétences.

Nombreuses sont les organisations qui ont su tirer profit de ces données collectées de diverses manières, leur permettant de réaliser des tâches qui, sans la présence de cette manne d'information, auraient été impossibles. Parmi les usages possibles de l'analyse de données, peuvent être mentionnés :

— l'ajout de valeur au processus de décision des entreprises [13]. Grâce à l'analyse des chiffres d'affaires et des résultats actuels, l'entreprise peut prendre des décisions pour l'achat ou la vente de produits, décider de l'investissement dans un marché, fixer les objectifs à long terme de l'entreprise ;

— la prévision des comportements des clients et des fluctuations du marché [14]. La science des données permet d'avoir une vision globale sur la clientèle d'une entreprise. Entre autres, elle permet de savoir si l'achat d'un produit est lié à l'âge des clients, à leur sexe ou bien à un autre facteur.

— la prestation d'une meilleure expérience pour les utilisateurs [15]. On cherche ici à adapter les produits et services de l'entreprise aux besoins des clients selon leur profil et leurs tendances d'achats.

— le suivi de l'évolution de l'état de santé d'un patient, dans le cas d'un hôpital [16]. Ceci permet d'avoir une idée précise sur l'historique du malade, retracer des maladies

génétiques et des maladies qui touchent des groupes de personnes, et parfois même prédire certaines maladies [17].

— la détection de comportements suspects [18], par exemple, en réalisant des profils de criminels et en les comparant à des individus sur les réseaux sociaux.

Tous ces exemples ont un point en commun : ils cherchent à faire ressortir, à partir d'une grande masse d'informations colligées, des tendances ou des « patrons » ayant un sens particulier selon le contexte. Une solution fréquemment utilisée pour l'extraction de l'information pertinente de ces différentes sources, et la réduction de l'effort nécessaire pour l'obtention d'un résultat de meilleure qualité, il s'appelle le *data mining*. Aussi connu sous le nom de forage de données, le data mining est la méthode qui permet d'extraire des connaissances à partir des données. Le data mining désigne toutes les techniques d'analyse, de transformation, de nettoyage et d'extraction de relations entre d'énormes quantités de données.

Les techniques regroupées sous le vocable de data mining sont très variées. Parmi celles-ci se trouvent le clustering [19], la classification [20], les arbres de décision [21], et les règles d'association [22]. Ces différentes techniques permettent entre autres de trouver des relations de corrélation entre les données, de faire de la prévision à partir de déductions faites sur les données connues, de chercher des groupes de données qui ont des caractéristiques communes et enfin, de faire un rangement ou une organisation dans les données. Plusieurs outils logiciels ont également été développés pour rendre le traitement plus facile et plus rapide. On pense entre autres à des produits comme RapidMiner [23], SAS [24], Weka [25], STATISTICA [26] ou R [27].

UN TYPE SPÉCIAL DE DONNÉES : LES JOURNAUX

Les données à la source du data mining peuvent être d'origines très diverses. On peut y compter par exemple des fichiers internes à l'entreprise tels que des tableaux d'évolutions,

l'ensemble des achats effectués par des clients, le prix des produits, les commentaires laissés par des utilisateurs, etc. Ces données peuvent être structurées, comme dans le cas d'une base de données, ou non structurées, comme dans le cas de commentaires textuels.

Parmi toutes ces sources d'information, il en existe un type particulier, en expansion et d'une grande importance, que sont les données issues d'une activité de **journalisation**. On dit qu'un *journal* (aussi appelé *log*) est une énumération de points de données se rapportant à un processus ou à une activité précise. Un log est typiquement produit en temps réel, et son contenu se modifie à mesure que se déroule le processus auquel il se rapporte. Lorsqu'il est lu, un log peut être vu comme un compte-rendu séquentiel d'observations à propos d'une tâche ou d'un système.

On peut trouver des logs dans pratiquement tout système dynamique. Ainsi, les systèmes d'information peuvent produire des journaux de différents types. Par exemple, les systèmes de gestion du flux de travail (*workflow management*), tels que Staffware [28] et InConcert [29], les systèmes CRM tels que FLOWer [30] et les plateformes ERP telles que SAP [31] produisent des journaux d'événements dans un format commun basé sur XML. Ces journaux contiennent de l'information sur les différentes tâches effectuées par chaque employé d'une entreprise, ainsi que les documents ou autres ressources qui sont échangés ou manipulés lors de l'exécution de ces tâches. De même, les systèmes de transactions financières tiennent également un journal de leurs opérations dans un format normalisé et documenté [32], comme c'est également le cas pour les serveurs Web tel qu'Apache [33] et Microsoft IIS [34]. Dans ce dernier cas, le journal collige l'ensemble des pages demandées par les visiteurs d'un site en y consignant diverses métadonnées telles que l'heure, l'adresse du demandeur et d'autres en-têtes provenant des requêtes HTTP reçues.

Dans un même registre, les moniteurs de réseau, tels que Orchids [35] ou Snort [36], peuvent également analyser des flux de paquets IP capturés par une interface, et dont les

différents en-têtes et champs peuvent être analysés ; cette suite de paquets peut être vue comme une forme particulière de journal, relatant cette fois-ci l'historique du trafic produit par un utilisateur. De la même manière, les différents systèmes d'exploitation produisent des logs en temps réel. Les événements se produisant lorsque le système s'exécute sont enregistrés afin de pouvoir faire un audit de l'état du système [37, 38]. Des exemples de fichiers de journalisation sont le journal application, le journal système, le journal installation ou le journal des connexions.

Les systèmes de santé sont une autre source importante de logs en temps réel [16]. En effet, des informations médicales sont recueillies par ces systèmes, à tout instant, sur l'état de santé des patients. Certains d'entre eux sont reliés à des appareils médicaux tels que l'équipement d'imagerie, l'équipement qui maintient les fonctions vitales ou les moniteurs médicaux [39, 40]. Ces informations sont toutes transmises en temps réel et ont besoin d'être traitées le plus vite possible.

Même les jeux vidéo en ligne sont devenus des producteurs de logs en temps réel [41, 42]. Dans ce contexte, des informations telles que le nombre d'échecs d'un joueur, les mouvements des personnages, le temps passé dans chaque niveau, les différentes décisions et étapes du jeu, y sont conservées. Ces informations sont importantes pour la détection de la source des erreurs, la prévision de mises à jour, l'ajout de nouvelles règles au jeu, ou la découverte de niveaux plus difficiles qu'anticipés.

Il existe également d'autres sources de données en temps réel qui peuvent être considérées comme des logs. Mentionnons : les marchés financiers [43], où le prix d'une action ne cesse de changer ; les transactions monétaires et les échanges de monnaie virtuelle comme le Bitcoin [44] ; les données environnementales [45] telles que la température ou la vitesse des vents ; les données spatiales , concernant la formation des étoiles et les différentes classes de galaxies ; les données issues de l'aviation [46], tels le suivi des vols et les données produites

par l'enregistreur de vol (la fameuse « boîte noire ») ; enfin, les données produites par les réseaux sociaux [47].

COMMENT TRAITER LES JOURNAUX ?

On l'a vu, les données produites sous la forme d'un journal abondent. Cependant, après leur collecte, celles-ci doivent être traitées et analysées pour en extraire une valeur ajoutée.

Dans le cas particulier des logs, l'analyse de la richesse des informations contenues dans ces journaux peut servir à plusieurs fins. Les journaux de processus métiers peuvent être utilisés pour reconstruire un flux de travail basé sur un échantillon de ses exécutions possibles [48] ; les journaux de bases de données financières peuvent être utilisés pour vérifier leur conformité à la réglementation [49] ; les activités suspectes ou malveillantes peuvent être détectées en étudiant les tendances dans les journaux d'un réseau ou d'un serveur [50] ; l'exécution enregistrée d'un jeu vidéo peut être analysée pour détecter la présence de bogues [42] ; l'historique des modifications d'un programme peut révéler des problèmes récurrents dans le code source [51]. Cette analyse peut prendre plusieurs formes : comparer des séquences d'événements à un workflow de référence, évaluer un ensemble de règles ou de propriétés formelles sur la séquence, ou surveiller l'apparition d'un motif ou d'une valeur prédéfinie.

Un élément important des logs est leur nature temps réel : leur contenu est produit dynamiquement par un système en cours d'exécution. Il est donc naturel de s'attendre à ce que leur traitement soit également effectué de la même manière : à mesure que de nouveaux points de données (des « événements ») sont ajoutés au log, le résultat de l'analyse ou du calcul s'ajuste et se met à jour continuellement. On parle alors d'un traitement en flux, également appelé traitement *streaming*. Ce mode de fonctionnement contraste avec les méthodes d'exploration de données plus traditionnelles fonctionnant en mode *batch* sur des journaux préenregistrés, et

qui calculent généralement un résultat une fois que le log entier est disponible, -c'est-à-dire souvent bien après la fin de l'exécution du processus que l'on observe.

Il existe bien dans la littérature des logiciels permettant de traiter des logs dont les événements arrivent progressivement. Les outils diffèrent, selon le domaine de provenance des données. Par exemple, une catégorie d'outils traite les événements dits complexes. Les plus connus de ces outils sont, Aurora [52], Borealis [7], STREAM [53], TelegraphCQ [54], SASE [55]. D'un autre côté, on trouve des logiciels appelés *moniteurs*. Le moniteur reçoit en entrée une propriété ou spécification qui devrait être respectée par le contenu du log (autrement dit la trace d'événements). À partir des informations contenues dans le log, le moniteur génère un **verdict** sur l'état du log : si la trace respecte la spécification, et s'il y a eu une violation de la spécification, quelles sont les actions spécifiées pour remédier à cette violation. Parmi les outils de monitoring, on peut citer : JavaMop[5], MarQ[56], Lola[57], Larva[58], RuleR [59].

Or, l'analyse des journaux s'intéresse de plus en plus à des problèmes faisant intervenir des éléments d'apprentissage automatique ou d'exploration de données : les utilisateurs sont de plus en plus intéressés par la recherche de modèles émergeant des ensembles de données, ce qui pourrait fournir des informations sur le processus en cours de journalisation et même déterminer le cours des actions futures. Un problème particulier lié aux journaux d'événements consiste à calculer des tendances sur une séquence d'événements et de détecter le moment où un log s'éloigne d'une tendance de référence donnée. Par exemple, une tendance peut être la durée moyenne d'exécution d'une tâche dans un processus métier, la distribution de la taille de paquets dans un flux de réseau ou le nombre de fois qu'une condition donnée est satisfaite dans une fenêtre glissante d'événements. La détection de déviations significatives de ces tendances peut révéler un incident, un dysfonctionnement ou un comportement indésirable du système observé, et donc fournir des informations exploitables à ses responsables.

Cependant, observer les tendances est un processus réactif : au mieux, une déviation est révélée au moment où elle se produit, et parfois même plus tard si l'analyse n'est pas effectuée en temps réel. Une seconde approche consiste donc à observer les événements passés et à en déduire certaines règles qui pourraient s'appliquer à des événements futurs - en d'autres termes, utiliser le passé afin de deviner de manière éclairée ce qui pourrait arriver par la suite. C'est ce que nous appelons l'analyse prédictive. Ce principe peut être appliqué de nombreuses manières. Par exemple, on peut utiliser l'analyse prédictive pour identifier les circonstances dans lesquelles un processus métier prend plus de temps que d'habitude à s'exécuter. L'application de ces règles apprises à l'exécution en cours d'un processus peut ensuite être utilisée pour prévoir son temps d'exécution. Si cela est jugé trop long, un responsable peut prendre des mesures préventives pour remédier à la situation. Le même flux de travail générique peut être utilisé pour établir des prévisions sur l'occurrence d'une attaque dans un réseau ou pour prédire la valeur d'un point de données particulier en fonction de ses valeurs antérieures. En fait, l'analyse prédictive sous diverses formes a été appliquée à un grand nombre de cas d'utilisation, allant de la détection de la criminalité [18], à la détection de la fraude [60], aux télécommunications [61], aux processus de soins de santé [62], aux médias sociaux [63], aux transports en commun [15], aux prévisions météorologiques et boursières [64], et à la prévision des pannes de superordinateurs [65].

Ainsi que mentionné, un aspect important de tous ces problèmes est que la déviation ou la prédiction doivent être calculées de manière continue : une source arbitraire génère progressivement des données, et une éventuelle déviation doit être détectée (ou une prédiction calculée) à mesure que les données sont ajoutées au log.

CONTRIBUTION

On pourrait croire que cette problématique, qui présente une parenté certaine avec les techniques de data mining, a été déjà étudiée et que des solutions existent pour le résoudre. Nous verrons en fait que peu de solutions existantes fournissent la combinaison appropriée de fonctionnalités de traitement de flux en temps réel et d'exploration de données nécessaire au traitement adéquat de ce type de question.

D'un côté, les outils de stream processing mentionnés plus tôt fonctionnent bien en temps réel ; cependant, le traitement qu'ils proposent se limite en général à l'application de fonctions d'agrégation sur des fenêtres glissantes d'événements, ou à l'application de fonctions simples sans égard à l'historique ou à la séquence des événements. De leur côté, les moniteurs, lorsqu'ils font de la surveillance en temps réel, induisent un surcoût inévitable d'exécution. Le surcoût est dû au code ajouté au système pour réaliser la surveillance. En effet, le temps d'exécution du programme surveillé et l'espace mémoire nécessaire pour son bon fonctionnement augmentent considérablement.

Les algorithmes de data mining, de leur côté, sont capables de réaliser des traitements approfondis et variés sur les données mais ne sont pas en streaming. En effet, à l'exception des algorithmes qui font de l'apprentissage incrémental, ces algorithmes traitent généralement les logs comme des vecteurs et doivent attendre que l'intégralité du log soit connue pour commencer le traitement. Ainsi donc, les traitements réalisés par la majorité des algorithmes de data mining, connus et largement utilisés, fonctionnent hors ligne et sur des sources de données préalablement enregistrées (en batch). Ils sont dans l'incapacité de traiter des flux de données en temps réel.

Le travail présenté dans cette thèse se situe donc à l'intersection de ces deux domaines. La question de recherche étudiée vise le développement, à la fois théorique et pratique, d'un

système permettant d'effectuer du traitement de logs arrivant en temps réel, en conjonction avec des algorithmes de data mining. Plus précisément, les objectifs que nous nous sommes fixés sont les suivants :

— Réaliser un workflow générique pour extraire de l'information des logs, arrivant en temps réel, tout en utilisant des techniques de data mining tel que le clustering.

— Mettre en œuvre le workflow sous forme de plusieurs exemples pratiques et plus évolués du workflow initial.

— Faire de l'analyse prédictive sur des traces d'événements en temps réel.

Le modèle proposé est une chaîne générique et de haut niveau d'unités de traitement d'événements de base combinées pour effectuer un calcul précis. Bien qu'un tel modèle puisse être implémenté dans tout système de traitement de flux d'événements, dans cette thèse, nous nous concentrons sur sa définition à l'aide de processeurs de flux fournis par le moteur de flux d'événements BeepBeep [66].

Comme son nom l'indique, une instanciation possible de ce flux de travail peut impliquer l'utilisation de techniques d'apprentissage machine, telles que les réseaux de neurones ou les arbres de décision, afin d'inférer une relation entre des caractéristiques calculées sur des fenêtres d'événements successifs et une "classe", définie par l'utilisateur, calculé à un moment ultérieur du journal.

ORGANISATION DE LA THÈSE

Le reste de cette thèse est organisé comme suit. Le chapitre 1 est consacré aux journaux. Une définition formelle du concept de log y est présentée ainsi que de nombreux exemples concrets de logs produits par différents types de systèmes. Il présente également des exemples de traitements pouvant être appliqués aux logs. Il servira à introduire différents exemples

d'utilisation de détection de tendances. À travers le chapitre nous passons en revue certains scénarios qui justifient le besoin de détection des déviations de tendances en temps réel.

Au chapitre 2, nous présentons une analyse de la littérature du domaine, en nous concentrant sur des techniques basées sur le runtime monitoring, le complex event processing, les statistiques et l'apprentissage machine. Nous présentons également une grande variété d'outils, à la fois open source et commerciaux, dont les capacités chevauchent les fonctionnalités de détection et d'extraction de tendances, et des fonctionnalités prédictives qui font l'objet de ce travail. Nous verrons que, si nombre de ces solutions peuvent effectuer une certaine forme de détection de tendance ou de prédiction sur les journaux d'événements, très peu peuvent le faire en mode continu, en fournissant et en mettant à jour un résultat en même temps que les événements d'entrée sont générés.

Une section est consacrée à la discussion des différentes mesures de distance. Le chapitre introduit également différentes techniques les plus utilisées et nécessaires à la compréhension du travail présenté. Pour chacune des techniques, des définitions, des exemples d'algorithmes et d'applications sont fournis. Une liste non exhaustive d'outils de data mining est également présentée.

Au chapitre 3, nous proposons un nouveau processus informatique appelé *trend distance workflow*. Ce flux de travail définit une séquence fixe d'étapes de calcul à exécuter sur un flux d'événements. Le modèle est très générique et fait abstraction de la plupart de ses fonctionnalités au moyen de paramètres pouvant être définis par l'utilisateur. On verra que de nombreuses tâches courantes de traitement et d'extraction de flux d'événements deviennent des cas particuliers de ce flux de travail, en fonction de la définition de ces paramètres.

Le premier modèle proposé est celui de la distance de tendance statique, où on donne tous les paramètres nécessaires pour que le flux corresponde à un flux concret. Des exemples

de distance de tendance statique sont discutés. Un deuxième modèle concernant les références multimodales est présenté. Dans cette section nous verrons que contrairement à la version précédente, plusieurs références peuvent être prises en compte pour un même flux. Certaines références dépendent du contexte, et d'autres références de tendance sont extraites de l'historique de la trace. Nous verrons également la technique d'extraction de la tendance qui permet d'utiliser de l'apprentissage machine pour générer des modèles de référence.

Au chapitre 4, nous introduisons ce que nous appelons un modèle de flux pour l'analyse prédictive. Tout d'abord, nous décrivons le flux de travail de prédiction statique, qui permet d'utiliser un flux d'événements comme base d'une prédiction fondée sur une fonction statique et prédéfinie qui est calculée sur une fenêtre glissante d'événements récents.

Ensuite, dans le même chapitre, nous montrons comment cette fonction peut elle-même être dérivée des journaux passés du même processus, en utilisant un deuxième type de flux de travail appelé apprentissage prédictif. Enfin, nous combinons les deux flux de travail précédents en un seul : le flux de travail de prédiction dit **auto corrélé** apprend une fonction de prédiction à partir des fenêtres précédentes du journal et utilise la fonction la plus récente pour calculer une prédiction sur la fenêtre actuelle de ce même journal.

Ces concepts, ainsi que de nombreux exemples de leur fonctionnement, ont été implémentés sous la forme d'un outil appelé *Pat The Miner*, lui-même un plug-in d'exploration de données pour le moteur BeepBeep. Au chapitre 5, nous décrivons cette mise en œuvre et rendons compte des expériences destinées à mesurer les performances de ces différents flux de travaux sous différents paramètres. Ces expériences révèlent qu'il est possible de calculer en temps réel les tendances et de détecter les écarts de tendance dans les journaux d'événements, jusqu'à des débits pouvant atteindre plusieurs milliers d'événements par seconde sur du matériel de base. Les résultats expérimentaux révèlent également que les prévisions et

l'apprentissage peuvent être calculés sur des journaux au rythme de milliers d'événements par seconde.

Ces résultats ouvrent la voie à l'analyse prédictive en temps réel dans un large domaine de cas d'utilisation. Le dernier chapitre de ce document est une conclusion de tout ce qui a été présenté. La seconde partie du même chapitre est consacrée à la proposition de travaux futurs.

La conclusion est divisée en deux sections. Nous avons dans un premier temps résumé les travaux que nous avons réalisés et, dans un deuxième temps, présenté des pistes de recherche et les limites des travaux réalisés.

Au cours de cette thèse et de la réalisation des objectifs fixés, trois articles scientifiques ont été publiés. Les références complètes sont données ci-dessous :

1. **Real-Time Data Mining for Event Streams** [67]

   M. Roudjane, D. Rebaïne, R. Khoury, S. Hallé. Real-time data mining for eventstreams. In *22nd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2018, Stockholm, Sweden, October 16-19, 2018*, pages 123–134. IEEE Computer Society, 2018.

2. **Predictive Analytics for Event Stream Processing** [68]

   M. Roudjane, D. Rebaïne, R. Khoury, S. Hallé. Predictive analytics for event stream processing. In *23rd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2019, Paris, France, October 28-31, 2019*, pages 171–182. IEEE, 2019.

3. **Detecting Trend Deviations with Generic Stream Processing Patterns** [69]

   M. Roudjane, D. Rebaïne, R. Khoury, S. Hallé. Detecting trend deviations with generic stream processing patterns. *Inf. Syst.*, 101 :101446, 2021.

Partie I

Notions de base et mise en contexte

CHAPITRE I
LES JOURNAUX D'ÉVÉNEMENTS ET LEURS TRAITEMENTS

Parmi toutes les données que peut produire un système d'information, ce travail se concentre sur un type particulier, que l'on appelle un journal, une trace ou un log. Dans ce chapitre, nous commençons notre étude par un grand bond en arrière et nous nous attardons particulièrement à ce concept, qui mérite certaines explications. Cette mise en contexte procédera en trois temps. D'abord, nous allons formellement définir le concept de trace d'événements et donner des exemples concrets de formats courants de représentation d'événements dans les journaux ; cette première section permettra de définir une terminologie de base ainsi que quelques conventions de notation. Ensuite, nous allons énumérer divers exemples de systèmes informatiques produisant des journaux d'événements. Dans un troisième temps, nous allons présenter des exemples concrets d'applications de l'analyse de journaux d'événements à des fins diverses.

Cette présentation de concepts nous permettra de passer à la quatrième et dernière section du chapitre, dans laquelle nous introduirons le problème concret faisant l'objet de cette thèse, à savoir la détection d'écarts de tendances et l'analyse prédictive sur les journaux d'événements. Il est important de noter que dans la présente thèse, nous traitons les tendances dans le sens large, par conséquent une tendance est une direction générale dans laquelle les données évoluent ou changent.

1.1 TRACES

Nous commençons par définir la terminologie et les concepts liés à la notion de trace. L'élément de base de tout système à base de traces s'appelle un **événement**. Un événement est un élément de données représentant toute action pouvant être accomplie par un programme

ou système informatique. Par exemple : retourner les résultats d'une recherche sur le Web, ajouter un utilisateur dans une base de données, lire ou écrire dans un fichier.

Selon P. Laplante [70], une *trace d'exécution* est une séquence ordonnée d'événements d'un certain type. Formellement, une trace d'exécution est une séquence potentiellement infinie d'événements [71]. Si Σ représente un ensemble d'événements possibles, la notation Σ^* représente l'ensemble des traces finies construites avec des événements de Σ. Σ^ω représente l'ensemble des exécutions infinies. Une trace par conséquent, sera notée par $\overline{\sigma}$. La formulation $\overline{\sigma} \preceq \overline{\sigma}'$ indique que $\overline{\sigma}$ est un préfixe de $\overline{\sigma}'$. Le i-ème événement de la trace $\overline{\sigma}$ est noté par $\overline{\sigma}[i]$. Comme la trace est une séquence potentiellement infinie, on écrit :

— $\overline{\sigma}_i$ pour désigner le suffixe de $\overline{\sigma}$ à partir du i-ème événement,

— $\overline{\sigma}^i$ pour représenter la concaténation de i copies de $\overline{\sigma}$. Dans ce cas, $\overline{\sigma}$ doit être de longueur finie.

Ce qui distingue un journal d'une source de données quelconque est son aspect séquentiel. Les événements qu'il contient sont placés dans une séquence ordonnée ; par conséquent, un journal représente souvent la consignation de mesures prises ou d'actions effectuées à des moments successifs dans le temps.

De fait, cet aspect séquentiel, voire temporel, va également se répercuter lorsqu'un traitement ou un calcul quelconque sera appliqué à ces journaux.

Le traitement réalisé sur les traces d'exécution peut être fait à deux moments différents. La vérification de l'exécution peut être faite en cours de l'exécution du programme surveillé ; dans ce cas on fait de la vérification de l'exécution *en ligne*. Dans cette perspective, un programme, qu'on nomme moniteur, qui lit les journaux d'exécution et qui génère un verdict est conçu de telle façon à ne pas influencer l'exécution du programme et en même temps recueillir les étapes de l'exécution de manière progressive et efficace. Les événements arrivant

au moniteur dans ce type de traitement sont consommés en temps réel, au fur et à mesure qu'ils se produisent, ils sont mis à la disposition du moniteur.

Le moniteur peut traiter les traces d'exécution d'une autre manière, c'est-à-dire traiter un ensemble fini d'exécutions pré-enregistrées ; dans ce cas on fait de la surveillance *hors ligne*. Le contenu des traces d'événements est connu à l'avance. Les traitements pouvant être faits sur les traces ne vont pas atteindre le programme lui-même, mais vont être sous forme de diagnostic de ce qui s'est passé ou va se passer [72].

Par définition, une trace peut être composée d'événements de type arbitraire. De fait, de nombreuses applications produisent des logs dont les événements sont consignés dans un format et selon une notation qui leur est unique –le plus souvent dans des fichiers texte. Nous en verrons quelques-uns plus loin dans ce chapitre. Dans cette section, nous nous concentrons d'abord sur un certain nombre de formats de représentation des logs qui sont génériques, partagés par plusieurs applications, et parfois soumis à un processus de standardisation. Ils seront abordés en ordre approximativement croissant de complexité. Cependant, dans la pratique, certains formats de représentation sont privilégiés pour stocker le contenu des événements et des traces. Dans cette section, nous décrivons les formats les plus courants.

Toute séquence de données brutes provenant d'une source quelconque peut être considérée comme un flux d'événement. Afin que ces données soient vues et traitées sous leur format de journal ou log, elles doivent être sauvegardées dans un support persistant. Dans ce qui suit, plusieurs types de fichiers pouvant représenter un journal de flux d'événement seront présentés.

1.1.1 FICHIER CSV

Le premier format est CSV, pour *comma separated values* , ou fichier de valeurs séparées par virgules. Un fichier CSV est un simple fichier texte dont les champs sont séparés par un caractère spécial, typiquement une virgule. Dans un tel fichier, chaque ligne du texte correspond à un événement. Chacune des valeurs sur cette ligne correspond à la valeur d'un attribut de l'événement. Ainsi, CSV impose naturellement une structure "tuples" pour représenter les événements. Ce type de fichier pourrait être importé sous forme d'un tableau par des tableurs tels que Microsoft Excel.

```
id,Prenom,Nom,email,Date,Pays,Code
100,Lily,Maroney,.@yopmail.com,2021-11-04,Sweden,VE
101,Rozele,Ferrell,.@yopmail.com,2020-11-09,Kiribati,
102,Tamqrah,Lail,.@yopmail.com,2022-01-20,Trinidad an
103,Sabina,Arvo,.@yopmail.com,2021-10-05,Niger,PS
104,Stacey,Gilmour,.@yopmail.com,2021-03-15,Niger,CN
105,Codie,Nunci,.@yopmail.com,2021-01-01,Cyprus,HU
106,Dari,Carri,.@yopmail.com,2021-06-17,Indonesia,TN
107,Beatriz,Daniele,.@yopmail.com,2020-08-28,Norfolk
108,Karly,Fabiola,.@yopmail.com,2022-03-12,Guinea-Bis
109,Rubie,Granoff,.@yopmail.com,2021-03-09,Qatar,AO
110,Ricky,Ricarda,.@yopmail.com,2021-01-06,Benin,AE
111,Ida,Giule,.@yopmail.com,2020-07-02,Lithuania,TF
112,Noelle,Bryna,.@yopmail.com,2020-04-01,Tunisia,ER
113,Elise,Kare,.@yopmail.com,2021-06-15,Switzerland,B
114,Lila,Kinnard,.@yopmail.com,2021-09-12,Libyan Arab
115,Moyna,Kaja,.@yopmail.com,2022-02-26,Eritrea,HT
```

FIGURE 1.1 : Exemple d'nn fichier CSV.

La Figure 1.1 nous montre un fichier CSV, dont les champs sont séparés par des virgules. La première ligne du fichier donne une description de chaque colonne. Comme on voit dans la figure, les attributs d'un fichier CSV peuvent être de différentes natures. Ils renferment le plus souvent des nombres ou des chaines de caractères.

1.1.2 TRACES MÉDICALES HL7

Le prochain format de représentation a été développé spécifiquement pour un domaine d'application précis. Health Level-7 ou HL7 [73] est une organisation qui développe des standards pour la représentation des documents cliniques. Elle s'occupe également de la mise en place d'un ensemble de normes internationales pour le transfert de données cliniques entre divers prestataires de soins de santé. Les normes développées par le HL7, portant le même nom que l'organisation, ont été adoptées par l'American National Standards Institute et l'Organisation Internationale de Normalisation (ISO).

Les informations envoyées à l'aide de la norme HL7 sont envoyées sous la forme d'un ensemble d'un ou de plusieurs "messages", chacun transmettant un enregistrement ou une information relative à la santé. Un message HL7 est composé de segments d'information organisés séquentiellement.

Un message HL7 est constitué de segments, champs et sous-champs. Un segment commence toujours par trois caractères spécifiques qui déterminent l'information contenue dans le segment, tel que l'information du patient, son parent le plus proche ou ses différentes visites à l'hôpital. Le segment est divisé par le caractère de délimitation (|) en champs. Le contenu du champ peut être un chiffre, nombre, caractère ou chaine de caractères et peut être subdivisé en sous-champs. Le caractère délimiteur utilisé dans les messages HL7 est spécifié dans le premier segment du message, dont le premier champ est MSH. Si le message HL7 utilise les délimiteurs par défaut, il commencera comme suit : MSH |. Les caractères contenus dans le deuxième champ du segment MSH définissent respectivement le caractère de séparation des : champs(|), des sous-champs (^), les champs qui se répètent (~), le début ou la fin d'une séquence d'échappement (\), et les sous-sous-champs (&).

Pour mieux comprendre la structure d'un message HL7, prenons comme exemple l'extrait de message de la Figure 1.2. Chaque ligne du message (segment) commence par trois lettres significatives qui décrivent le type d'information que le segment contient.

```
1 MSH|^~\&|EPIC|EPICADT|SMS|SMSADT|202003120115|SECURITY|ADT^A01|0706123|D
2 |2.4|
3 PID||5742810^^^2^ID
4 1|343670|| JOE^WILLIAM^| JOE^WILLIAM^|19601001|M||B|688 ROUTE A
5 AVE^^ MAVILLE ^QC^ 7H2B1^CA ||(123)456-7890|||M|NON|500002314~2230095|
6 NK1||ANN^MARIE^^^^|SPO||(123)456-7890||EC|||||||||||||||||||||||||
7 PV1||O|155 ~351~C~PMA^^^^^^^^^|||351^KAREN
8 SMITHJOHN^LINDA ^^^^^^^|||||||||
9 ||4902291|||||||||||||||||||||||202003120115
```

FIGURE 1.2 : Extrait d'un message au format HL7.

Dans la Figure 1.2, on voit 4 segments d'informations. L'en-tête de message commence par le segment MSH, abréviation de *message header*, qui décrit le type de message, la version et d'autres informations pertinentes. Le segment PID (Patient ID) contient les informations personnelles du patient comme son nom, identifiant, et son adresse. Le segment NK1 (*next of kin*) contient les informations relatives au parent le plus proche du patient. Enfin le segment PV1 (patient visit), qui contient les informations relatives au séjour du patient dans l'hôpital tel que le nom du docteur assigné.

On peut voir HL7 comme un raffinement du format CSV. Mis à part quelques détails cosmétiques (comme la virgule remplacée par une barre verticale), le format permet à plusieurs types d'événements de coexister dans un même journal ; ceux-ci peuvent avoir une structure différente. Un champ peut également être divisé en sous-champs, une chose qui n'est pas

possible (du moins, de manière standardisée) avec CSV. En contrepartie, ce format, bien que plus riche, ne trouve aucune utilisation hors du domaine médical.

1.1.3 FICHIER XML

Un fichier avec l'extension XML est un fichier écrit en Extensible Markup Language qui est un langage de balisage. Les fichiers XML [74, 75] sont des fichiers texte bruts qui encodent la structure d'un document et le stockage des données. Le langage est largement utilisé pour la représentation de structures de données arbitraires telles que celles utilisées dans les services Web.

Un *tag* ou *balise* est le texte contenu entre les deux caractères "<" et ">", la balise de départ est écrite < balise >, une balise de fin est exprimée comme suit < / balise>. Un *élément* est constitué de la balise de départ, de la balise de fin et tout ce qui est contenu entre les deux. Un élément peut contenir des éléments enfants, cette imbrication d'éléments donne finalement une structure arborescente au document. Un *attribut* est un couple nom-valeur contenu à l'intérieur de la balise de départ.

Dans la Figure 1.3, un exemple de balise pourrait être <root>, un élément est tout ce qui se trouve entre <adresse> < / adresse>, y compris les deux balises. Enfin, un exemple d'attribut est `Prénom` dans la balise de départ <personne>.

Contrairement à CSV, XML permet aux éléments d'être imbriqués les uns dans les autres. C'est donc une structure en **arbre** qui est donnée aux éléments. Chaque élément peut être vu comme un noeud de l'arbre, et un élément imbriqué dans un autre est représenté par la relation parent-enfant.

```xml
<root>
  <personne
  Prénom="Claudina"
  Nom="Persse"
  email="Kristina.Clara@yopmail.com"
  />

  <adresse>
    <ville>Luxembourg (city) </ville>
    <pays>Chad </pays>
  </adresse>

  <random>10.654</random>
  <bool>false</bool>
  <date>2020-10-03</date>

</root>
```

FIGURE 1.3 : Forme du fichier XML.

1.1.4 FICHIER XES

XES acronyme pour (eXtensible Event Stream) [76] est un standard développé par Eindhoven University of Technology. XES définit une grammaire dont le but est de fournir une méthodologie unifiée, simple, flexible, extensible et expressive pour représenter les logs d'événements et les flux d'événements occurrents dans les systèmes. Il est basé sur les balises XML spécifiques pour les logs d'événements. Ce standard a été développé initialement pour faire du process mining. Néanmoins, son format est adapté également pour le data mining et toute analyse statistique. Le XES a été approuvé par l'IEEE, en 2016, comme standard qui accomplit l'interopérabilité dans les journaux d'événements et les flux d'événements [77, 78].

Comme on peut le voir dans l'exemple de balises XES donné dans la Figure 1.4, XES maintient la structure standard de représentation d'un log d'événement. Le log contient des traces qui elles-mêmes contiennent des séquences d'événements. Des attributs de différents types peuvent être insérés dans n'importe lequel de ces trois niveaux. La diversité des types

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes.version="2.0" xes.features="arbitrary-depth" xmlns="http://www.xes-standard.org
    /">
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.
        xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <global scope="trace">
        <string key="concept:name" value=""/>
    </global>
    <global scope="event">
        <string key="concept:name" value=""/>
        <date key="time:timestamp" value="1970-01-01T00:00:00.000+00:00"/>
        <string key="system" value=""/>
    </global>
    <classifier name="Activity" keys="concept:name"/>
    <classifier name="Another" keys="concept:name system"/>
    <float key="log attribute" value="2335.23"/>
    <trace>
        <string key="concept:name" value="Trace number one"/>
        <event>
            <string key="concept:name" value="Register client"/>
            <string key="system" value="alpha"/>
            <date key="time:timestamp" value="2009-11-25T14:12:45:000+02:00"/>
            <int key="attempt" value="23">
                <boolean key="tried hard" value="false"/>
            </int>
        </event>
        <event>
            <string key="concept:name" value="Mail rejection"/>
            <string key="system" value="beta"/>
            <date key="time:timestamp" value="2009-11-28T11:18:45:000+02:00"/>
        </event>
    </trace>
</log>
```

FIGURE 1.4 : Forme d'un fichier XES. [1]

d'attributs rend la représentation des métadonnées de ce format plus expressive. Le nombre d'attributs contenus dans le log, trace et événement n'est pas prédéfini. Chaque événement doit être dans un élément <event>, ses attributs sont chacun dans des éléments dont le nom représente le type, et il y a des métadonnées extérieures aux logs dans une section <global>.

La librairie OpenXES a été développée comme implémentation complète et *open source* du standard XES sur Python [79].

1.1.5 FICHIER JSON

JavaScript Object Notation ou JSON est un format de fichier textuel lisible par l'humain et facilement interprétable par la machine [80]. Il est utilisé pour les échanges entre navigateurs

et serveurs. Sa force réside dans le fait que n'importe quel objet du langage JavaScript peut être converti en document JSON. Il est également indépendant de tout langage de programmation. Il utilise néanmoins des conventions familières à tous les langages de programmation. On voit la forme d'un document JSON dans la Figure 1.5.

```
{ "Programme": {
     "id": "3081",
     "nom": "Sciences et technologies de l'information",
     "Étudiants":{

        "Informatique": [
           {"Nom":"Personne 1", "Cours":"Mathématique"}
           {"Nom":"Personne 2", "Cours":"Lecture"}
           {"Nom":"Personne 3", "Cours":"Rédaction"}
        ]
     }
 }
}
```

FIGURE 1.5 : Données exprimées en JSON

Un fichier JSON est un tableau associatif entre un mot clé, par exemple identifiant ou nom, et une valeur. Les types de base d'un fichier JSON peuvent être un type primitif tels qu'un nombre, caractère, une chaine de caractères ou un booléen. Ils peuvent être une liste de type primitif, un autre objet JSON tel que l'élément "Étudiants", ou bien une liste d'objets JSON tels que l'élément "Informatique". L'imbrication d'objets de type primitifs avec des objets de JSON rend cette notation très légère et facile à d'utilisation.

Il existe un important recoupement entre l'expressivité de JSON et celle de XML ; on peut sans difficulté convertir des documents entre ces deux formats. En raison de la présence d'imbrication, JSON est généralement associé à une structure en arbre tout comme XML.

1.1.6 FICHIER CTF

Les formats présentés jusqu'ici sont tous basés sur des fichiers textes représentables sous forme de chaînes de caractères.

De son côté, le format de trace commun (Common Trace Format) a été développé par Efficios [1]comme un format de trace **binaire** qui permet de générer des traces depuis n'importe quelle application ou système C/C++.

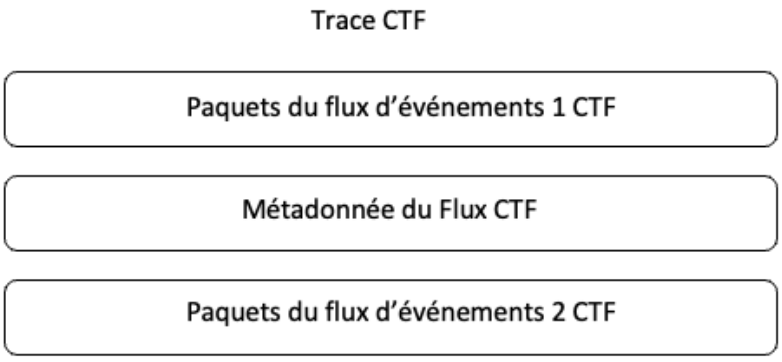


FIGURE 1.6 : Composants d'une trace CTF.

Une trace en CTF est composée de plusieurs flux d'événements binaires ; un flux est une concaténation de paquets comme on le voit dans la Figure 1.6. À son tour, un paquet (*stream packet*) est composé de plusieurs éléments : un en-tête, un contexte, des événements concaténés, comme le montre la Figure 1.7.

L'en-tête ou *trace packet header* contient la même information pour tous les événements contenus dans la trace, tels que l'identifiant de la trace et l'identifiant du flux (*stream*). Le *stream packet context*, quant à lui, contient des informations relatives au flux d'événements

1. www.efficios.com/ctf

telles que la taille du contenu, la taille du flux, l'horodatage du début et de fin de l'événement, le chiffrement appliqué s'il y a lieu, etc.

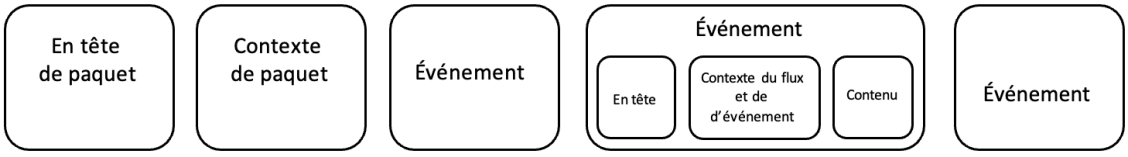


FIGURE 1.7 : Flux d'événements en CTF.

Chaque événement est composé des informations suivantes : l'en-tête (event header) qui contient l'identifiant de l'événement, le contexte de flux (*stream event context*) qui est appliqué à tous les événements du flux, le contexte de l'événement (*event context*) qui contient des informations relatives à l'événement en cours, et enfin le contenu de l'événement (*payload*) qui contient des champs spécifiques à un type d'événement donné.

CTF représente un changement majeur par rapport aux autres formats présentés jusqu'ici. D'abord, comme on l'a déjà mentionné, il s'agit d'un format binaire et non textuel, permettant potentiellement une représentation plus compacte de l'information. Ensuite, il standardise un certain nombre de métadonnées à propos d'un log dans des en-têtes obligatoires. De façon plus importante, il permet par défaut l'encapsulation de plusieurs séquences d'événements dans une unique structure de données, ce qui ne peut être effectué qu'au moyen de marqueurs non standardisés dans les autres formats. Le format laisse également beaucoup de latitude aux champs contenus dans un événement, qui comptent comme types possibles tous les types primitifs du langage C.

Nous verrons dans la prochaine section que plusieurs systèmes informatiques produisent des logs dans un format qui leur est propre. Cependant, dans la très grande majorité des cas, ces formats spécifiques peuvent conceptuellement être assimilés à l'une ou l'autre des

représentations que nous avons décrites. Par exemple, on verra qu'un serveur web comme Apache consigne des entrées qui, bien que ne respectant pas strictement la codification CSV, sont représentées dans un format qui lui est équivalent. On peut donc affirmer que cette présentation des traces possibles est un portrait relativement global des types d'événements que l'on peut rencontrer dans l'étude des journaux produits par des systèmes informatiques.

1.2 SOURCES DE TRACES

Maintenant que nous avons pris soin de décrire clairement ce qu'est un journal et comment ses événements peuvent être représentés, nous nous tournons vers différents exemples de systèmes d'information générant des traces dans l'un ou l'autre de ces formats.

L'objectif de la présente section est de faire remarquer au lecteur la grande diversité de scénarios dans lesquels les données produites ou recueillies peuvent être assimilées à une notion de flux d'événements. Par conséquent, toute technique d'analyse ou de traitement s'appliquant à des flux de manière générique pourra immédiatement être portée à l'un ou l'autre de ces scénarios avec un minimum d'adaptations requises (ce qui sera le cas de la contribution de la présente thèse) .

Nous divisons cette section en quatre catégories selon l'origine des événements. On s'intéressera dans l'ordre aux logs applicatifs, aux traces produites par l'activité réseau, aux processus d'affaires et à l'exécution de programmes instrumentés.

1.2.1 LOGS APPLICATIFS

La journalisation permet un suivi et une gestion efficace des activités et performances du serveur, ainsi que la détection de problèmes éventuels.

Apache Le serveur HTTP Apache fournit des fonctionnalités de journalisation très complètes et flexibles. Un journal Apache est un enregistrement des événements qui se sont passés depuis la requête initiale jusqu'à la fermeture de la connexion.

Deux types de journaux sont stockés par Apache : le journal des accès et le journal des erreurs. Le journal d'accès contient les requêtes qui arrivent au serveur. Ce journal peut contenir des informations sur ce que les gens voient, le statut de la requête (succès ou échec). Ce journal consigne l'historique horodaté de toutes les ressources demandées par chaque utilisateur, et cela constitue une manne d'informations potentielles (par exemple, pour reconstruire l'itinéraire de visite d'un utilisateur unique).

La Figure 1.8 nous montre un exemple de journal d'accès. L'information contenue dans le fichier est de gauche à droite l'adresse IP du client, l'identité du client (le trait d'union indique que cette information est indisponible), l'identifiant de l'utilisateur ayant demandé le document, l'heure de la réception de la requête, la ligne de la requête client placée entre guillemets, le code de statut retourné au client par le serveur et enfin la taille de l'objet retourné au client.

```
10.0.0.2 - John [12/Dec/2017:22:25:30 -0700]  "GET /apache_test.pdf HTTP/1.0" 200 4300
```

FIGURE 1.8 : Exemple d'un journal d'accès Apache.

Le journal des erreurs contient toutes les erreurs survenues lors du traitement des requêtes par le serveur. La Figure 1.9 donne une idée du format de ce journal.

Tout comme le journal d'accès, le journal d'erreurs contient l'hodotage, l'adresse IP du client et le type d'erreur rencontrée. Dans notre exemple, le fichier demandé n'existe pas. Différents types d'erreurs peuvent être répertoriés dans ce journal tel que : une erreur de

```
[Fri Jan 21 20:04:20 2018] [error] [client 127.0.0.1] File does not exist: /var/ww/test.pdf
```

FIGURE 1.9 : Exemple d'un journal d'erreurs Apache.

connexion de l'utilisateur, un déni de connexion, un fichier introuvable, emplacement erroné du fichier, page introuvable, serveur non joignable, pour ne citer que celles-ci.

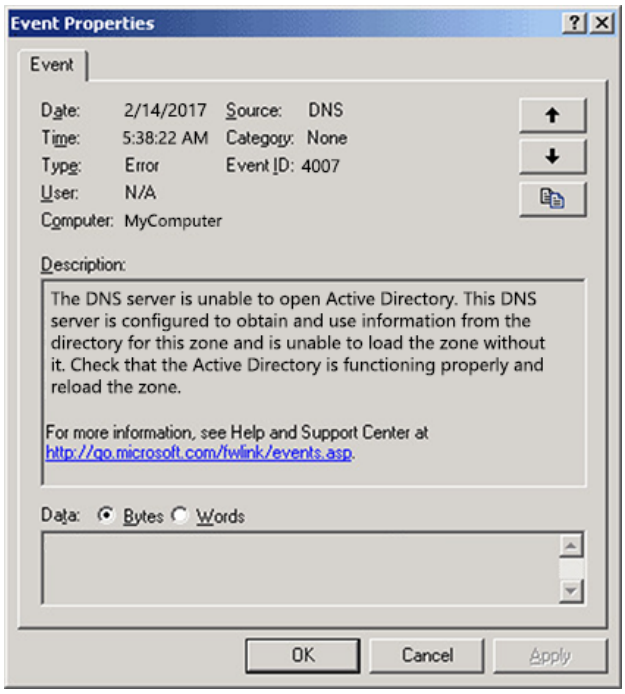


FIGURE 1.10 : Exemple d'un événement.

Active Directory *Active Directory* (AD) est l'annuaire mis en œuvre par Microsoft pour les systèmes d'exploitation Windows à partir de Windows 2000 [81]. Son objectif est de fournir un service d'authentification et d'identification dans un réseau d'ordinateurs Windows. Les éléments répertoriés par AD sont les comptes des utilisateurs connectés, les différents postes du réseau, les serveurs, et tout autre élément du réseau administré. Le serveur qui

héberge l'annuaire AD est appelé contrôleur de domaine. La Figure 1.10 illustre un exemple d'événement du journal des événements du serveur DNS sur Windows Server 2003.

Les informations recueillies sont stockées sur un ou plusieurs contrôleurs de domaines dans des bases de données. Les objets sont organisés de manière hiérarchique en trois grandes catégories : ressources, services et utilisateurs. Chaque objet représente une seule entité et ses attributs. Il est possible d'afficher les détails et les instances d'un événement donné dans Event Viewer. La Figure 1.11 illustre les instances d'un événement du serveur DNS.

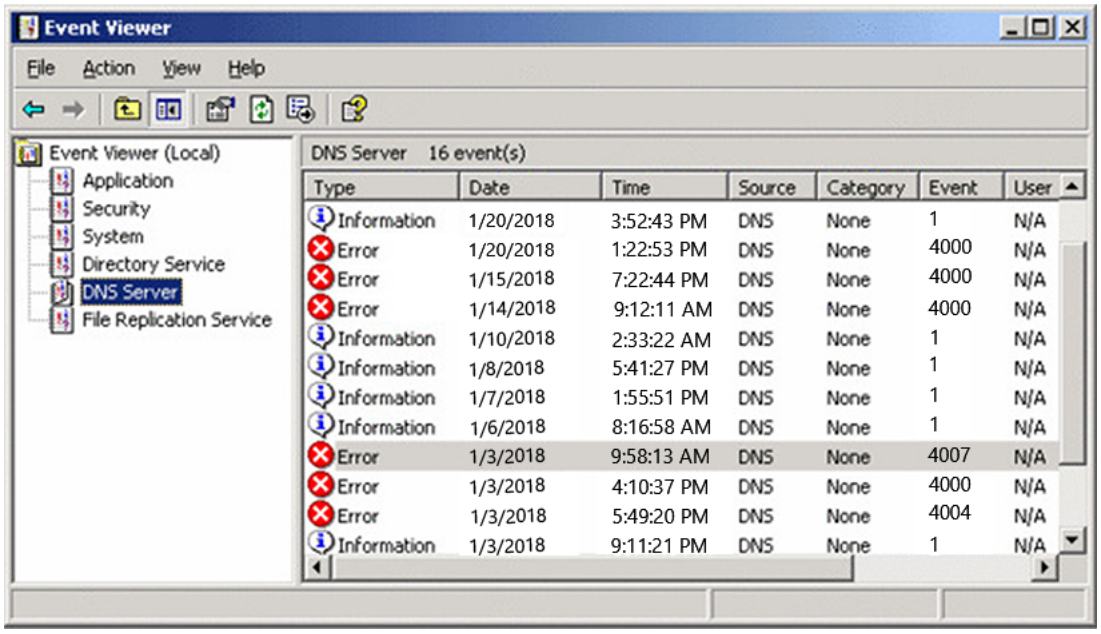


FIGURE 1.11 : Attributs d'un événement.

Active Directory conserve énormément d'informations sur le système : login/logout, modifications au compte, activation/arrêt de services système, problèmes matériels, état de la batterie, etc. L'analyse des journaux AD permet de faire du diagnostic, par exemple lors de pannes.

D'autres types de traces résultent d'autres types d'activités telles que la trace de l'exécution d'un système d'exploitation (Windows, Linux, Android) ; la Figure 1.12 ci-dessous est obtenue par le Event Viewer qui est une console de gestion de Microsoft. Cet outil est inclus dans le système d'exploitation Windows. Il permet une visualisation des événements et donne la possibilité de surveiller l'état du système pour résoudre les problèmes quand ils surviennent.

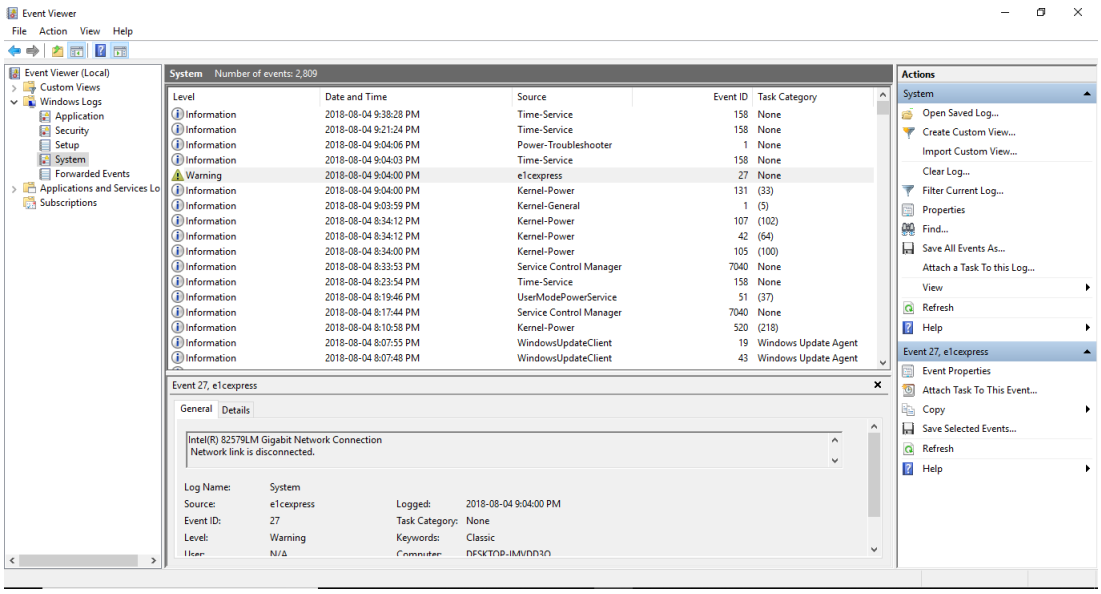


FIGURE 1.12 : Logs d'un système Windows.

MySQL Un autre type de journaux est celui généré par un serveur de bases de données tel que MySQL. Plusieurs fichiers de journalisation sont mis à la disposition de l'utilisateur. Ils peuvent être retrouvés dans le dossier données de **mysqld**.

Parmi les journaux générés, on retrouve entre autres le **journal des erreurs** qui, comme pour Apache, décrit les problèmes rencontrés du démarrage jusqu'à l'arrêt de MySQL. On voit dans la figure 1.13 un exemple de sortie de journal de requête dans lequel sont indiquées l'adresse du client, les connexions établies et les requêtes traitées.

```
051743   12:15:10      1 Connect    root@localhost on MyComputer
051743   12:15:23      1 Query      SHOW search_path
051743   12:15:34      2 Query      SELECT * FROM Employee
051820   23:13:04     13 Query      SELECT * FROM Tasks
051820   23:13:17     13 Query      SELECT * FROM Tasks where id = 3
```

FIGURE 1.13 : Exemple de journal de requête MySQL.

De son côté, le **journal de requêtes** décrit les connexions établies et les requêtes exécutées, le **journal des mises à jour** et le **journal des binaires** enregistrent les déclarations qui engendrent un changement.

SYSLOG

Les sources de journaux d'événements dans le système d'exploitation Linux varient. Il peut s'agir de logs provenant du système, des services ou bien des différents types d'applications qui sont en cours d'exécution. Tous ces logs sont enregistrés dans un dossier spécial, nommé `/var/log`. Linux dispose de plusieurs systèmes de logs, dont les plus importants sont : `/var/log/syslog` et `/var/log/messages` où sont stockés les données d'activité de système. `/var/log/auth.log` et `/var/log/secure` où sont stockés les événements en relation avec les ouvertures de sessions, les actions de l'utilisateur root et toute action en relation avec la sécurité et confidentialité. Finalement, `/var/log/kern.log` enregistre les événements se produisant dans le noyau et les erreurs et avertissements. Le serveur web Apache et Mysql, décrits précédemment, stockent leurs logs respectivement dans les annuaires `/var/log/apache2` et `/var/log/mysql`.

Syslog [82] est un standard, et un protocole, qui définit le service de journalisation dans un système informatique.

```
Feb 20 23:33:00 server2 sshd[12345] : Failed password for root from 127.0.0.1 port 22 ssh2
```

FIGURE 1.14 : Exemple d'un message Syslog.

Dans la Figure 1.14, on voit la forme d'un message sous le format standardisé de Syslog [83]. Le message formaté commence par l'en-tête qui contient la date et l'heure d'occurrence de l'événement. Ensuite, le nom de la machine source du log, le processus qui a causé l'événement, le niveau de priorité de l'événement. Le format du log résultant peut toujours être modifié à partir du fichier de configuration syslog.conf.

LIBRAIRIES DE LOGGING

Tout type d'application peut produire des logs de son exécution par l'insertion d'instructions de logging directement dans le code. Cela peut servir au développeur à déboguer au cours du développement, ou à fournir des informations de diagnostic qui peuvent aider à comprendre les causes d'une panne. On peut souvent activer la production de ces messages par un paramètre à la ligne de commande.

À titre d'exemple, Apache LOG4J est une librairie de logging qui fait partie de l'Apache logging Services. Elle fait partie des différentes librairies de logging de Java. Il y a sept niveaux de journalisation intégrés dans Log4j : OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ce dernier étant le niveau donnant le plus d'informations détaillées. Log4j permet à l'utilisateur de spécifier son propre format de niveau de log. Un outil de génération de niveaux de logs personnalisés est intégré à la librairie pour permettre de générer des logs complémentaires aux niveaux intégrés ou même de les remplacer.

Il existe également un type de journalisation dite détaillée (en anglais *verbose*). En effet, cette approche permet de contrôler le niveau de détails contenus dans les enregistrements et par conséquent les données contenues dans les fichiers de journalisation.

L'exemple 1.15 ci-dessous montre comment des instructions de logging sont insérées dans un programme.

```
final static Logger logger = Logger.getLogger(nomDeClasse.class);

logger.debug("Message de niveau debug");
logger.error("Message de niveau erreur");
```

FIGURE 1.15 : Exemple d'instruction de logging.

On appelle des méthodes de l'objet «logger», selon le niveau. Le logger n'impose aucun format : chaque message est une chaîne de caractères quelconque que le développeur est libre de formater comme il le souhaite. Le fichier résultant n'est pas nécessairement du CSV, même si la plupart du temps le format lui ressemble. C'est aussi le développeur qui décide des emplacements dans son programme où de telles instructions sont insérées.

Chacune de ces applications a son propre format de journalisation. Les journaux obtenus s'apparentent tous au format CSV. On obtient le même format, soit des événements par ligne avec des champs dans chaque colonne.

1.2.2 TRACE RÉSEAU

Un autre domaine dans lequel les données peuvent être modélisées sous forme de logs d'événements est celui des réseaux. En effet, dans le protocole TCP/IP, la transmission se fait par paquets. Chaque paquet IP est une combinaison de plusieurs champs qui décrivent l'information envoyée. Le paquet IP est constitué d'un en-tête IP, de l'adresse IP de la source,

de l'adresse IP de destination, et des données IP. Il est important de mentionner que l'en-tête IP a une structure fixe, composée de plusieurs champs qui contiennent diverses informations telles que la longueur totale, la durée, etc.

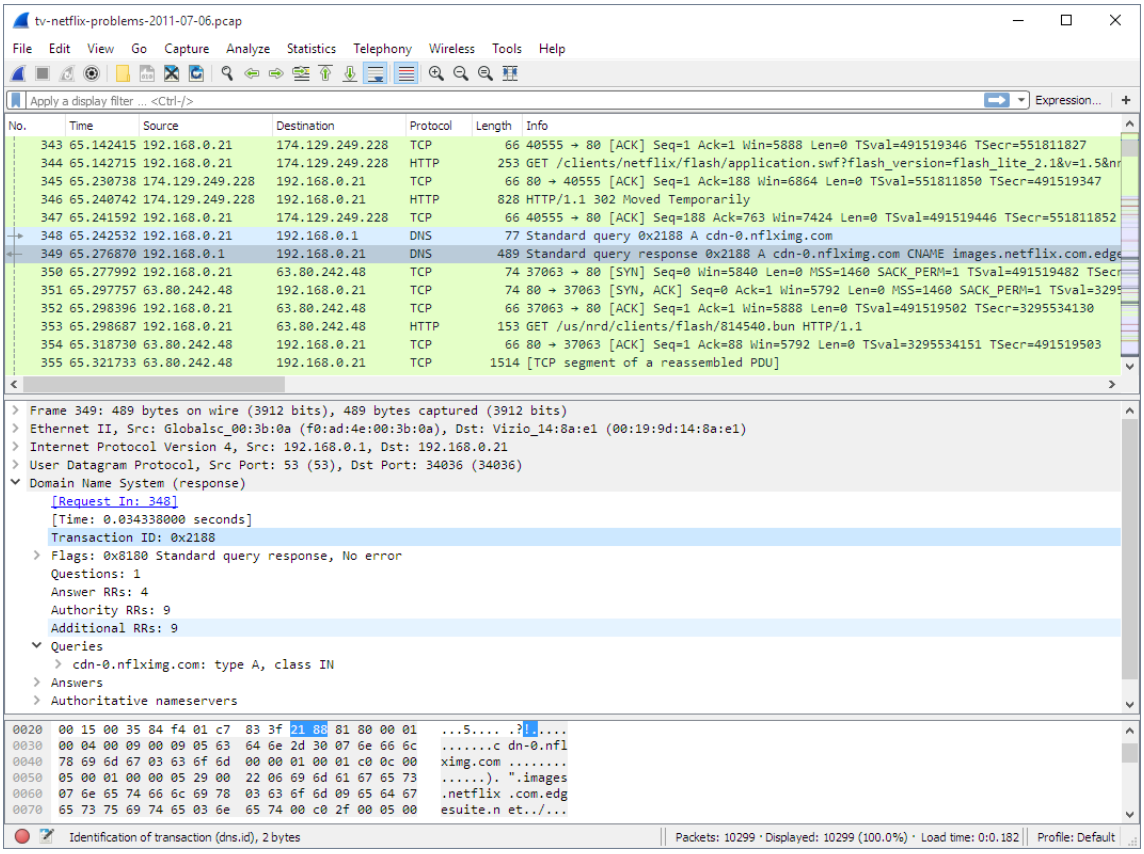


FIGURE 1.16 : Trace de paquets réseau. [2]

Les paquets envoyés dans le réseau peuvent être capturés en utilisant des outils tels que Wireshark [84]. La Figure 1.16 est une capture de réseau avec cet outil. On voit également la forme de la console de l'outil et les différentes parties de la capture. La première partie (sélectionnée en vert) énumère la liste des paquets IP capturés, ce qui constitue ici la trace. La seconde partie donne les détails du paquet (donc de l'événement) sélectionné. La troisième partie affiche le contenu du paquet IP sélectionné en hexadécimal.

Wireshark contient un filtre spécial pour les paquets du Protocole DNS. On reconnait le trafic de ce protocole car il est toujours surligné en bleu pâle. Les paquets DNS sont de deux types : demande et réponse. Une demande peut contenir plusieurs requêtes. La structure de données d'une demande informe le serveur DNS du type de la requête, du nombre de questions qu'elle contient, et les données contenues dans la requête. La réponse contient exactement les mêmes champs, mais elle contient un flag de réponse et un nombre non nul de réponses.

Dans une même capture réseau sont entremêlés des paquets relatifs à plusieurs applications et plusieurs protocoles. La Figure 1.16 montre que le paquet affiché (numéro séquentiel 349) est la réponse à une requête contenue dans le paquet précédent (348), et qu'elle correspond au protocole DNS.

Comme on peut le voir, la structure du paquet IP en champs fixes permet de le convertir sous forme CSV. Le fichier résultant serait du texte brut décrivant les différents champs, et dont les valeurs sont séparées par des virgules. La donnée du paquet peut contenir des données d'une application qui serait elle-même sous un format précis. Un exemple de cela pourrait être le protocole HTTP, pouvant être codifié sous un format JSON dont les différents champs sont contenus dans la partie donnée du paquet IP.

L'analyse d'un flux de paquets est pratique pour, par exemple, détecter des attaques, différencier un trafic normal d'un trafic suspect, établir un profil de trafic selon le moment de la journée afin de prévoir les ressources à fournir, ou encore diagnostiquer des problèmes ou des pannes dans un réseau. Il existe déjà des outils pour l'analyse de ce type de flux, tels que les systèmes de détection d'intrusion (ou IDS, *Intrusion detection Systems*). Une méthode avancée de l'examen et de gestion du trafic réseau s'appelle le *deep packet inspection*. Elle est utilisée pour filtrer les paquets pour localiser et identifier ou bloquer des paquets spécifiques. Le *deep packet inspection* est utilisé principalement pour la détection d'intrusion [85].

1.2.3 PROCESSUS D'AFFAIRES

Les processus d'affaires sont un autre domaine où les données peuvent être modélisées sous forme de logs d'événements. Un processus d'affaires ou *business process* [86] est une activité ou un ensemble d'activités liées et formant une structure spécifique qui répond à un besoin commercial. Un business process peut généralement être visualisé de deux façons différentes. La première est comme un organigramme d'une séquence d'activités, la seconde est comme une matrice de processus d'une séquence d'activités.

BPMN (Business Process Model and Notation) [87] est un standard permettant aux organisations de modéliser leurs activités internes sous forme de business process, et de les partager avec une notation standard. La modélisation graphique fournie comme résultat permet d'assurer que toutes les parties collaborant dans ce processus sont sur la même longueur d'onde quant à la signification de chaque partie du processus. La Figure 1.17 montre un exemple du résultat de la modélisation d'un business process avec BPMN, et le résultat graphique obtenu.

Il existe plusieurs systèmes (Business Process Engines ou "*workflow management systems*") qui permettent de gérer ces processus tels que Staffware [28] et InConcert [29], les systèmes CRM tels que FLOWer [30] et les plateformes ERP telles que SAP [31]. Ces outils se chargent de l'interprétation, l'exécution et la gestion des processus exprimés en BPMN.

Comme tout processus, les business process laissent des traces de leurs exécutions, qui décrivent les différentes étapes par lesquelles le processus est passé. La Figure 1.17, tirée de [88], donne un exemple de l'exécution d'un tel processus. Un *business process engine* centralisé consigne chaque transition effectuée dans le modèle BMPN avec des métadonnées. Par exemple, l'action "Check Car Reservation" pourrait produire un événement qui contient le nom et le montant des soumissions de chaque expéditeur ayant été contacté.

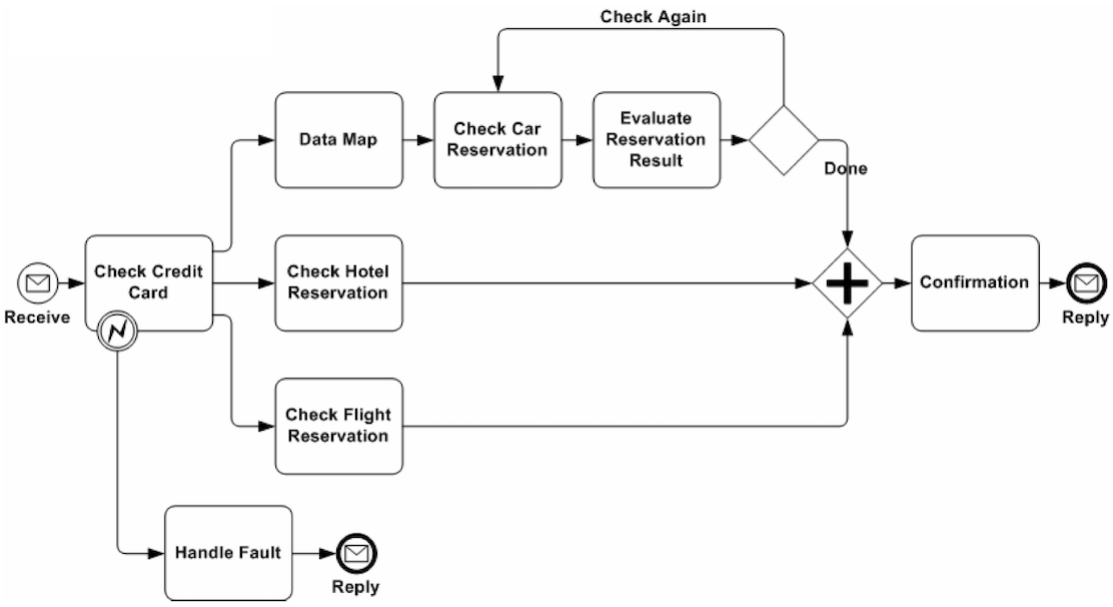


FIGURE 1.17 : Exemple d'un business process.

Les processus sont modélisés sous le format BPMN, mais les exécutions quant à elles sont représentées sous un format XES décrit dans la sous-section 1.1.4.

Un système de gestion de dossier est un cas particulier des *business process*. La partie suivante donne un exemple d'application d'un type spécifique de systèmes de gestion, les fichiers sous format HL7.

Les traces d'événements issues du système d'information médicale, les fichiers HL7, peuvent être analysées pour obtenir des informations pertinentes sur les changements inattendus dans les valeurs de données des clients. Un type particulier de *BP engine* gère les processus médicaux. Une approche a été présentée en utilisant les techniques du *Complex Event Processing* où les événements produits contiennent des données médicales et sont décrits sous le format HL7 [16].

1.2.4 EXÉCUTION DE PROGRAMMES INSTRUMENTÉS

Une famille importante de sources de logs est les traces de bas niveau, produites par l'exécution d'un programme, formées d'appels de méthodes ou d'autres types d'instructions. L'analyse de traces de ce type s'appelle exécution de programmes instrumentés ou *Runtime Monitoring*.

Un **traceur** est un outil qui permet d'enregistrer des événements survenant durant l'exécution d'un système. Les traceurs les plus connus sont capables d'enregistrer au sein d'une même trace des événements survenant dans le système d'exploitation et dans les applications en espace utilisateur.

Un point de trace peut être inséré statiquement ou dynamiquement dans une application en espace utilisateur ou dans le noyau du système d'exploitation. Les points de trace peuvent être activés ou désactivés à tout moment pendant la durée de vie du système. Lorsqu'un point de trace est atteint lors de l'exécution, une sonde, connectée à celle-ci, est chargée d'écrire les données dans des tampons gérés par le traceur. Dans chaque traceur, il y a une partie dont la seule tâche est d'écrire les informations collectées sur un périphérique. La quantité de données générées par les traceurs peut être importante.

Une *trace de l'espace utilisateur* offre une vue du comportement d'un système proche du code applicatif. Pour générer une telle trace, il faut instrumenter chaque application. Une *trace du noyau offre* une vue de plus bas niveau. Elle peut révéler toutes les interactions entre les applications et le système d'exploitation.

Une force des principaux traceurs est qu'ils ont un impact minime sur le comportement des systèmes sur lesquels ils sont utilisés. Cette caractéristique fait d'eux des outils de choix pour déboguer des erreurs de synchronisation, où la moindre perturbation du système peut empêcher le comportement fautif de se reproduire. Grâce à leur faible surcoût, ils peuvent

surveiller un système en production sur une longue période, jusqu'à ce que l'erreur ou l'événement recherché se manifeste.

Divers types de logs ont été définis dans la section précédente. Ils sont issus de différents domaines et sont utilisés pour résoudre différentes problématiques. Les logs en question sont tous obtenus grâce à l'utilisation d'un logiciel spécifique pour le traçage. Dans ce qui suit, on présentera quelques-uns de ses traceurs.

Linux Trace Toolkit Next Generation (LTTng) [2] : LTTng [37] est un package pour tracer les noyaux du système d'exploitation Linux, les applications et librairies, et est également très pratique pour le débogage.

Event Tracing for Windows (ETW) [3] : Event Tracing for Windows est un traceur intégré au système d'exploitation Windows. Il permet le traçage des événements noyau fournis par Microsoft. Les développeurs peuvent aussi générer des événements à partir de leurs applications.

DTrace [89] est un logiciel conçu pour Solaris pour la détection des problèmes en temps réel. Depuis, il a été élargi à d'autres systèmes d'exploitation tels que FreeBSD et Mac OS X. Tout comme LTTng, il permet de tracer le noyau et l'espace utilisateur et vise un impact minimal sur la performance. Contrairement aux autres outils, Dtrace permet de spécifier des actions à effectuer lorsqu'un événement survient.

Les traceurs peuvent révéler quelles fonctions d'une application consomment du temps processeur, de la mémoire, des accès réseaux, etc. Ils fournissent également des détails pour chaque occurrence d'un événement. Cela permet, par exemple, de faire de la surveillance et de la vérification.

2. https://lttng.org/

3. https://docs.microsoft.com

Les outils précédents produisent des traces de bas niveau, généralement composées d'appels système. On peut également tracer des programmes au niveau de l'API d'un langage de programmation, au moyen d'une technique appelée *programmation orientée aspect* (POA) [90, 91]. Selon les langages utilisés, les librairies POA s'appellent AspectJ (Java), AspectC (C), ou AspectC++ (C++).

Un aspect est utilisé, contrairement à un module technique, pour décrire un aspect technique selon l'endroit de son insertion dans le code. Ceci permet entre autres de tracer le résultat de l'exécution d'un programme, en ayant une vision claire de toutes les méthodes qui y ont été exécutées.

Un aspect est défini sur une classe, en utilisant soit des méthodes telles que *Around* dans l'AspectJ, pour contrôler la classe. Cette annotation permet d'intercepter des points de coupure dans la classe, à savoir les méthodes de la classe avant et après leur exécution.

La POA peut-être utilisée, entre autres, pour faire de la journalisation. L'aspect permet, par exemple, de créer et renvoyer les noms des méthodes qui s'exécutent et leurs arguments. Elle permet également d'insérer des instructions plus avancées à des moments précis lors de l'exécution du programme. Il est à noter que, pour AspectJ, il existe également plusieurs annotations qui permettent l'interception des méthodes à différents moments. Les plus connues et plus utilisées des annotations sont **@Before**, **@After**, **@AfterReturning**, **@AfterThrowing** et enfin **@Around**.

Parmi les applications de l'analyse des données produites par un traceur, une attention spéciale doit être accordée au *runtime verification*.

Runtime verification [72] ou *runtime monitoring* sont les deux noms donnés à la discipline d'analyse de l'exécution des systèmes informatiques. Cette approche se base sur l'extraction de l'information du système au moment de son exécution pour l'analyser afin de

détecter et éventuellement réagir aux violations des propriétés. Les spécifications en *runtime verification* sont exprimées en machines à états finis, en logique temporelle ou autre langage de description de prédicats sur les traces. Dans le domaine du *runtime monitoring* un **moniteur** est tout simplement un périphérique qui lit une trace finie et génère un certain verdict.

La *runtime verification* peut être utilisée dans plusieurs optiques : la surveillance des politiques de sécurité [92], les tests [93], la vérification et validation, la modification du comportement du programme, etc. Même si la *runtime verification* couvre moins de terrain que le *model-checking*, elle permet d'obtenir des résultats moins coûteux et plus précis, étant donné qu'elle traite l'exécution en question et non pas une modélisation du programme entier.

Un exemple intéressant de l'utilisation du *runtime monitoring* sur les jeux vidéo sera présenté dans la sous-section suivante.

1.2.5 PROGRAMMES ÉVÉNEMENTIELS

Les programmes événementiels sont une catégorie particulière de systèmes caractérisés par une boucle s'exécutant en permanence, et dans laquelle l'état du programme peut être mis à jour en fonction d'entrées produites par l'utilisateur. Des exemples notables de programmes événementiels sont un grand nombre de jeux vidéo, où chaque itération de la boucle peut typiquement modifier la position de différents personnages, l'état du niveau dans lequel ils évoluent, ainsi que d'autres variables internes. À noter que cet état peut évoluer même sans intervention du joueur [3].

L'état du jeu à tout moment est constitué d'un cycle d'activités récurrentes telles que le choix du personnage qu'on peut changer à tout moment, la décision de quitter le jeu, de retourner vers le menu et les différents dialogues. La mise à jour des paramètres du jeu se fait grâce à la *boucle de jeu*. En effet, la boucle de jeu peut être exploitée pour produire des

événements, qui sont des captures de l'état du jeu et des personnages qu'il contient à chaque itération de celle-ci.

Le déroulement du jeu devient ainsi une suite d'événements qui crée des séquences continues. C'est ainsi que sont générées les traces d'événements dans les jeux vidéo. On verra plus en détail dans la suite du document comment ces traces sont utilisées pour la détection de bogues dans les jeux vidéo.

Les traces d'événements issues des jeux vidéo peuvent être utiles pour la détection d'intrusion dans les jeux. Un travail a été fait dans ce sens. Varvaressos *et al.*[3] proposent une architecture de monitoring en temps réel. Contrairement aux techniques connues pour la surveillance des jeux vidéo, ce travail exploite la boucle du jeu en y insérant l'instrumentation. Avec toute mise à jour de la boucle, le jeu renvoie un événement contenant les données de l'état actuel du jeu qu'on appellera *instantanée (snapshot)*.

Dans ce contexte, le moniteur et le jeu sont séparés, et sont contenus dans deux processus indépendants qui communiquent via des messages XML. La Figure 1.18 montre un exemple de XML qui génère un message basé sur l'état actuel du jeu. Les propriétés surveillées sont exprimées sous forme de contrainte de logique temporelle appliquée aux instantanées du jeu.

On voit que ceci est un cas particulier de *runtime monitoring*. Les événements dans les jeux sont des instantanés de l'état interne du programme.

1.2.6 FLUX DE CAPTEURS

Un grand nombre de systèmes sont composés de capteurs mesurant différentes caractéristiques de l'environnement, et les lectures à différents moments dans le temps constituent

```
<message>
 <characters>
  {% for pingu in characters.pingus %}
   <character>
    <id>{$pingu.id}</id>
    <action>{$pingu.action}</action>
    <isalive>{$pingu.isalive}</isalive>
    <position>
     <x>{$pingu.x}</x>
     <y>{$pingu.y}</y>
    </position>
    <velocity>
     <x>{$pingu.velx}</x>
     <y>{$pingu.vely}</y>
    </velocity>
    <groundtype>{$pingu.groundtype}</groundtype>
   </character>
  {% endfor %}
 </characters>
 <overhead>{$overhead}</overhead>
 <deadlyvelocity>{$deadlyvelocity}</deadlyvelocity>
 <timestamp>{$timestamp}</timestamp>
 <messagenumber>{$messagenumber}</messagenumber>
</message>
```

FIGURE 1.18 : Boucle d'un jeu vidéo. [3]

donc des traces au sens où on l'entend dans cette thèse. Parmi les exemples de travaux ayant étudié ces flux, on mentionne les habitats intelligents [94].

Caldas et al. [95] présentent un résumé des études qui ont appliqué des algorithmes d'intelligence artificielle (IA) pour analyser la marche humaine à partir de données de capteurs inertiels, en vérifiant qu'ils peuvent soutenir l'évaluation clinique.

En agriculture aussi les capteurs sont largement utilisés. En effet, en se basant sur un réseau de capteurs sans fil, Muangprathub et al. [96] ont proposé un système d'arrosage optimal des cultures agricoles. Le système proposé est composé d'un équipement de capture et

collection de données, la gestion de données. Les prédictions se font au niveau de l'application web, le dernier est un téléphone intelligent qui permet de contrôler l'arrosage.

1.3 APPLICATIONS DE L'ANALYSE DES JOURNAUX

Dans la présente section, il sera question de présenter des scénarios d'application dans différents domaines des logs et des exemples de traitements possibles.

1.3.1 ÉCARTS DE TENDANCE

On a mentionné au passage à la section 1.2 différentes applications de l'analyse des flux d'événements spécifiques à chacun des scénarios abordés. Dans cette section, on s'intéresse à la notion de tendance, soit une mesure globale calculée sur un flux d'événements ou une partie de ce flux, et plus particulièrement à la notion d'écart de *tendance*. Nous allons énumérer différentes situations simples dans lesquelles cette notion peut s'appliquer. Selon le contexte, on verra que la détection d'un écart peut correspondre à des réalités très diverses. En d'autres termes, ce même concept général peut se décliner de plusieurs manières et avoir plusieurs utilités.

JOURNAUX DE PROCESSUS MÉTIER

Dans un premier exemple, considérons un processus métier (*Business process*) simple décrivant le traitement d'une demande d'indemnisation au sein d'une compagnie aérienne (Figure 1.19, tiré de van der Aalst [4]). Un journal dans un tel scénario serait généralement composé d'événements appartenant à plusieurs instances du processus à différents stades d'achèvement. En supposant que chaque événement contienne, entre autres, un identifiant pour l'instance de processus à laquelle il appartient, il devient possible de diviser ce journal en

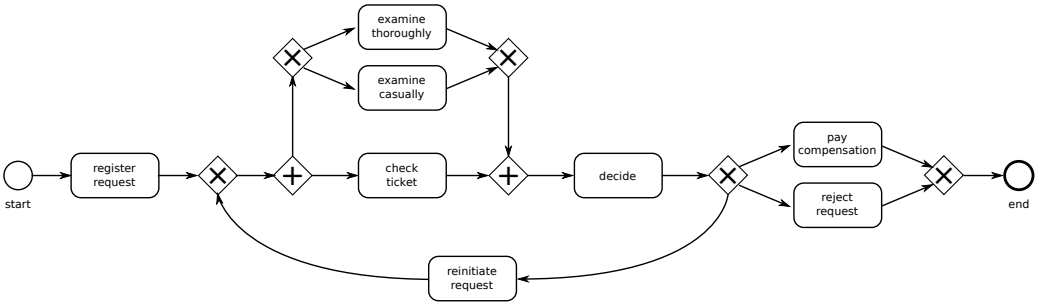


FIGURE 1.19 : Une représentation simple d'une demande de processus d'indemnisation à l'aide de la notation BPMN [4].

plusieurs tranches, chacune ne conservant que les événements correspondant à un identifiant d'instance unique.

Une tâche possible pouvant être exécutée sur chacune de ces tranches consiste à vérifier la conformité aux règles de processus métiers. Un exemple de ce type de règle pourrait être le suivant : « une demande doit être examinée de manière approfondie si le montant demandé est supérieur à 100 ». Ces règles peuvent être exprimées dans diverses notations largement étudiées, et le processus de vérification de ces règles est appelé *contrôle de conformité* (par exemple [97, 98, 99, 100, 101]). Cependant, il peut également être intéressant de calculer et d'observer diverses autres tendances sur ces tranches, et en particulier d'être averti lorsqu'un processus dévie d'une tendance de référence donnée auparavant. Par exemple :

Scénario 1. Être averti lorsque la durée de traitement d'une demande (du début du traitement jusqu'à la prise de décision) est plus longue de plus de 25% par rapport à une certaine borne fixée B.

Le schéma de la Figure 1.20 représente les durées de traitement des demandes de 6 clients. Supposons que la borne de temps fixée à partir de laquelle une alarme sera déclenchée est B.

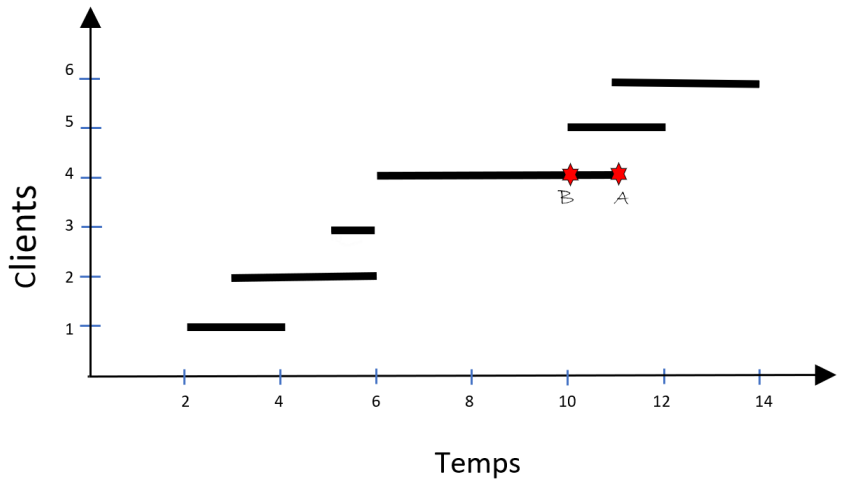


FIGURE 1.20 : Exemple de durée de traitement d'une demande par client.

Dans ce premier exemple, fixons $B = 4$. On remarque que toutes les durées de traitement sont relativement courtes sauf pour le client 4. Sa durée de traitement totale atteint 5 à $t = 12$ (représenté par le point A), elle dépasse, par conséquent, la borne fixée de plus de 25%.

Contrairement aux règles de conformité habituelles, une telle notification n'indiquerait pas nécessairement une « violation » stricte, mais plutôt une indication qu'un événement inhabituel pourrait se produire et nécessiterait une plus grande attention. Par exemple, un message spécial pourrait être envoyé aux clients, indiquant un retard dans le traitement de leur commande. La frontière entre la stricte conformité et une simple « notification » peut être encore plus floue, en remplaçant une référence fixe par une tendance calculée à partir de l'historique du processus, comme dans l'exemple suivant :

Scénario 2. Être averti lorsque la durée d'une demande est supérieure de plus de 25% à la durée moyenne des demandes du mois dernier.

Reprenons l'exemple de la Figure 1.20. Pour ce deuxième exemple, supposons que le « mois dernier » est représenté par la période $t \in [0,6]$ et que la durée moyenne de traitement d'une demande durant cette période était $B = 2$. Le mois suivant, au temps $t = 10$, la durée de

traitement du client 4 est égale à 3, au point B. Cette durée est supérieure à la durée moyenne du mois passé $B = 2$ de plus de 25%, une alerte est donc déclenchée.

Il est à noter que cette fois-ci, la tendance de référence n'est pas basée sur une limite fixe, mais est plutôt calculée à partir d'un ensemble d'instances précédemment enregistrées du même processus. Bien entendu, des calculs similaires pourraient être effectués sur des aspects du processus autres que la durée : le nombre de reprises, le montant remis au client, etc. De plus, cette requête s'éloigne encore un peu plus de la conformité à une règle stricte : une requête dont la durée est plus longue que la normale n'est pas forcément incorrecte, mais peut néanmoins nécessiter un examen plus approfondi.

1.3.2 DOSSIERS MÉDICAUX ÉLECTRONIQUES

Nous passons maintenant au domaine de la gestion des dossiers médicaux, où les événements sont des messages exprimés dans le format HL7 [102].

On a vu que les messages HL7 peuvent être générés à partir de différentes sources : équipements médicaux produisant des résultats de tests, logiciel de gestion des patients permettant d'enregistrer les actes et procédures médicaux individuels, bases de données sur les médicaments, etc. Pour un patient donné, la fusion de toutes ces différentes sources produit une longue séquence de messages HL7 pouvant être assimilée à un flux d'événements.

Une étude de Berry et Milosevic [16] a identifié plusieurs situations dans lesquelles on cherche à détecter des modifications importantes et inattendues dans les valeurs des données susceptibles de compromettre la sécurité des patients. Or ces modifications ne sont rien d'autres qu'une forme d'écart de tendance. Dans ce contexte, une règle générale, qui peut s'appliquer à tout champ numérique, identifie le moment où une valeur de donnée commence à s'écarter de sa tendance actuelle, par exemple :

Scénario 3. Avertir l'utilisateur lorsqu'un champ de données observé se situe à trois écarts types au-dessus ou en dessous de sa moyenne.

Pour mieux illustrer le scénario 3, prenons un exemple. Soit les régions A et B dans la Figure 1.21 partie du log capturé. Selon le scénario 3, une alerte sera lancée au point A, car sa valeur se situe à plus de trois écarts types au-dessus de la moyenne. Le point B, quant à lui, ne déclenchera pas d'alerte, car sa valeur est dans l'intervalle de tolérance, c'est-à-dire que la valeur de B se situe à moins de 3 écarts types de la moyenne.

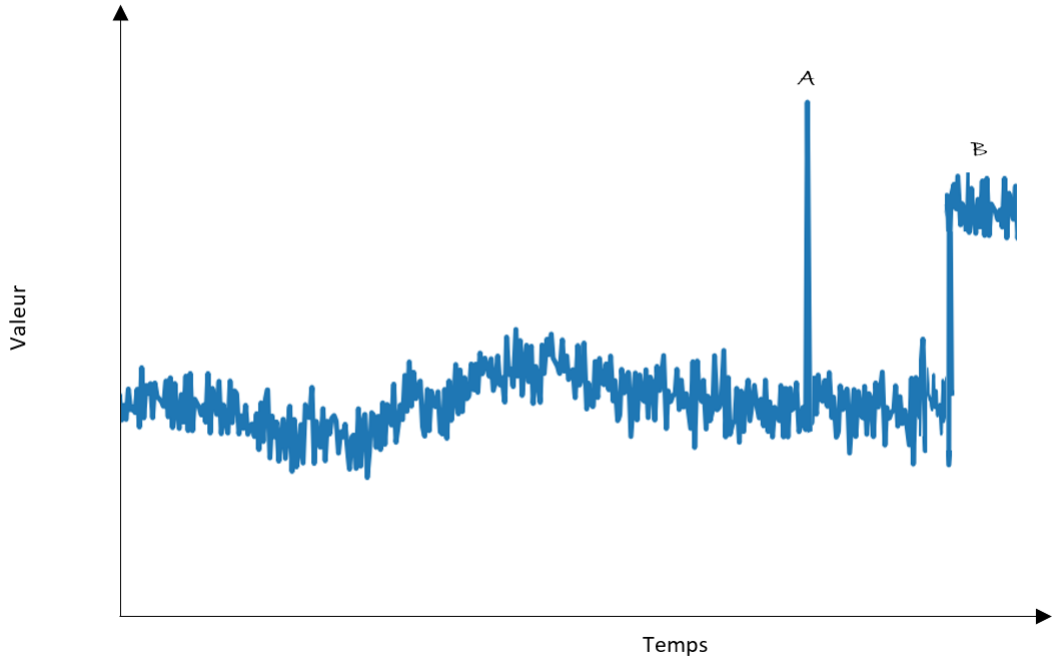


FIGURE 1.21 : Exemple d'un flux d'événements.

Encore une fois, il ne s'agit pas d'une violation, mais plutôt d'une alerte sur l'état du processus surveillé. Par exemple, si un médecin surveille la pression artérielle de l'un de ses patients de près. Une notification de déviation peut être envoyée à celui-ci, lorsque la pression artérielle du patient s'éloigne de la valeur moyenne.

De telles requêtes peuvent être rendues plus complexes et corréler les valeurs trouvées dans plusieurs événements. Par exemple :

Scénario 4. Avertir l'utilisateur lorsque deux points de données successifs sur trois se trouvent à plus de deux écarts types de la moyenne du même côté de la ligne moyenne.

Toujours sous les mêmes conditions de l'exemple précédent, et en suivant les consignes du scénario 4, on remarque que le point A dans la Figure 1.21 ne déclenche pas d'alerte, car c'est le premier point à dépasser de 2 écarts types la ligne moyenne. Cependant, une alerte sera lancée au point B, car on a deux points successifs à dépasser la ligne moyenne de plus de deux écarts types.

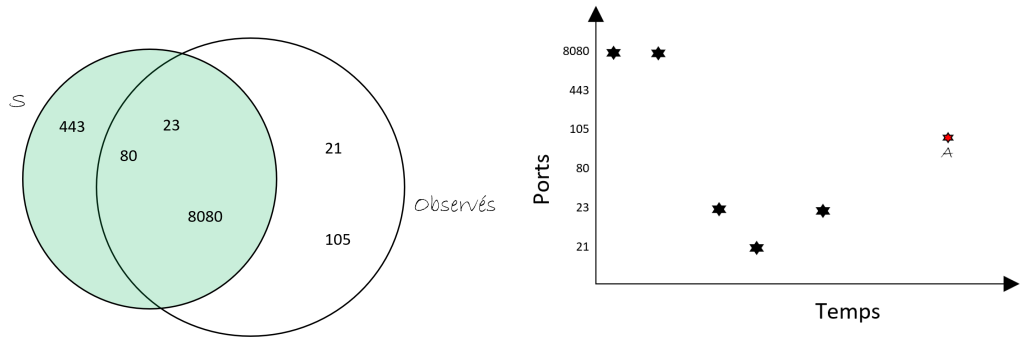
Un exemple d'application serait l'analyse des données de l'ECG d'un patient. Ceci permet de détecter des troubles cardiaques tels que des palpitations, une arythmie (battements du cœur irréguliers), ou bien des battements de cœur anormalement élevés. Dans ce contexte, une notification de déviation sera envoyée si deux prises successives sur trois sont trop loin de la moyenne.

Bien que notre scénario ne le spécifie pas, cette agrégation peut être calculée sur une « fenêtre glissante », telle que les 100 derniers événements ou les événements de la dernière heure. Cette fois, on note que les requêtes ne relient pas la tendance du journal à une référence fixe, ni même à une tendance calculée sur un ensemble de journaux antérieurs. On essaie plutôt de comparer les valeurs actuelles à une tendance calculée à partir de l'historique du journal lui-même. Nous verrons plus tard que ce processus s'appelle *auto-corrélation*.

1.3.3 SÉCURITÉ DES SYSTÈMES

Le calcul des tendances sur les flux d'événements peut également être utile à des fins de sécurité, en traitant des journaux provenant de différentes sources. Comme premier exemple, on peut considérer une capture en temps réel de paquets réseau comme un flux d'événements et surveiller les déviations de diverses métriques calculées sur ce flux [103]. Un système peut garder une trace des ports TCP apparaissant dans les paquets capturés sur une certaine période et agréger ces ports dans un ensemble. Cet ensemble peut ensuite être comparé à une référence et une alarme peut être déclenchée lorsque les deux ensembles sont suffisamment différents. Cela peut conduire à une requête de tendance telle que la suivante.

Scénario 5. Envoyer une notification lorsque l'ensemble des ports TCP vus au cours de la dernière heure diffère d'au moins 5% d'un certain ensemble de référence de ports *S*.



(a) Ensemble de ports de référence et observés. (b) Ports connectés.

FIGURE 1.22 : Exemple de scénarios 5 et 6.

Soit S= {443, 80, 8080, 23} un ensemble des ports TCP de référence. Le schéma 1.22 représente l'ensemble des ports TCP sur lequel se connecte notre système durant la dernière heure. Des connexions à deux ports ne faisant pas partie de l'ensemble de référence ont été enregistrées, les ports TCP 21 et 105. À l'aide de l'index de similarité de Jaccard, on

calcule la distance entre notre ensemble de référence S et l'ensemble des ports observés durant la dernière heure : $J(S,O) = \frac{|S \cap O|}{|S \cup O|} = \frac{3}{6} = 50\%$. Les connexions de la dernière heure sont différentes de plus de 5% de l'ensemble de référence S. Une alarme est déclenchée.

En supposant que S soit un échantillon du trafic vérifié comme étant « normal », une différence peut indiquer qu'une activité suspecte se produit sur le réseau. Cette fois, on note que la tendance calculée sur le flux n'est pas une valeur numérique, mais un ensemble d'éléments discrets (ports TCP). Il s'agit d'un autre exemple où l'auto-corrélation peut être utilisée : au lieu de calculer un ensemble de références statique S, on pourrait utiliser un ensemble de ports observés à un moment donné dans le passé (par exemple, l'heure précédente ou la même heure la veille).

Les autres journaux système peuvent être traités de la même manière. Par exemple, Microsoft Active Directory maintient un journal de toutes les sessions authentifiées sur un réseau de stations de travail Windows [38]. Les événements de connexion/déconnexion du même utilisateur ou du même ID de processus peuvent être agrégés dans des sessions, et les tendances de ces sessions peuvent être utilisées pour détecter des activités inhabituelles ou suspectes. Par exemple, les sessions peuvent être regroupées dans des catégories prédéfinies en fonction de leur durée (moins d'une minute, 1 à 5 minutes, etc.). La fréquence relative des sessions pour chaque intervalle de durée sur une période donnée constitue une distribution discrète, que l'on peut ensuite comparer à une distribution de référence :

Scénario 6. Envoyer une notification lorsque la distribution des durées de session est différente d'au moins k en comparaison à une distribution de référence.

Soit P la distribution de référence des durées de connexions associées aux ports de référence, telle que : $P(8080, 443, 23, 80) = (0.25, 0.1, 0.15, 0.5)$. Par conséquent, durant une session de connexion 25% du temps on utilise le port 8080, 10% on se connecte via le port

443, 15% du temps via le port 23 et enfin la moitié du temps la connexion se fait par le port 80. Soit k la limite de tolérance de dépassement des durées, telle que $k = 30$ minutes.

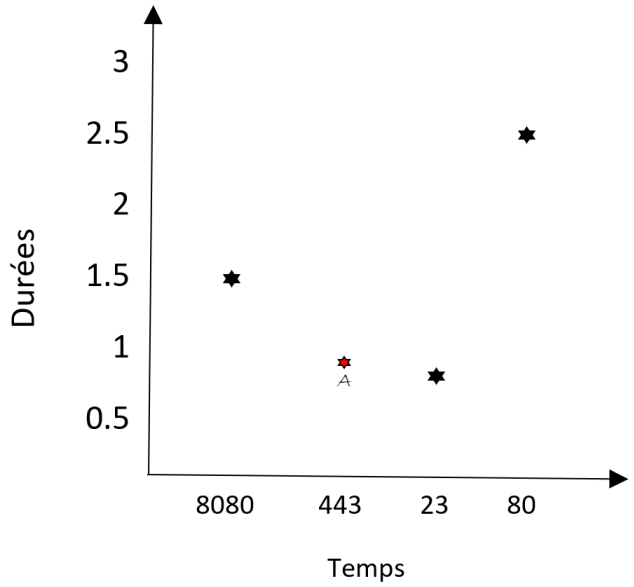


FIGURE 1.23 : Exemple de durée de connexion pour la session ouverte.

La session représentée dans la Figure 1.23 a duré 6 heures. Si la session était conforme à la référence P, les durées de connexion sur les ports seraient les suivantes (8080, 443, 23, 80)=(1.5, 0.6, 0.9, 1.5). Les valeurs observées sont (8080, 443, 23, 80)=(1.5, 1.1, 0.9, 2.5). On remarque une augmentation de la durée de connexion via le port 443, et par conséquent une déviation de plus de 30 minutes au niveau des ports 443 et 80. Une alarme est donc déclenchée.

Cette fois, la tendance calculée n'est ni un scalaire ni un ensemble, mais un tableau associatif mettant en relation des catégories et des nombres réels. De toute évidence, la « distance » entre deux de ces tableaux devrait être définie de manière appropriée en fonction du contexte. Entre autres, on pourrait cumuler la somme des différences entre les valeurs de tableau pour chaque catégorie.

1.4 ANALYSE PRÉDICTIVE

Cette section sera consacrée à la notion de prédiction et l'analyse prédictive, qu'elle soit sur un flux ou une séquence particulière du flux. Nous verrons quelques situations simples qui permettront de mettre en contexte la notion de prédiction sur des flux d'événements complexes.

Scénario 1. Prédire la prochaine valeur dans un log en utilisant la moyenne des i événements précédents.

Prenons l'exemple du log de la Figure 1.24. Le log comprend trois instances entrelacées (jaune, bleu et vert). Chaque instance est composée de trois événements x, y et z. Supposons que la valeur que contient chaque événement est sa durée. On remarque que celle-ci varie d'une instance à l'autre.

Essayons de prédire la prochaine valeur dans ce log en utilisant le raisonnement de ce premier scénario. Premièrement, il faut fixer $i = 5$, on prend en compte uniquement les 5 derniers événements, c'est-à-dire $\langle z, y, x, z, y \rangle$. Supposons que : $\langle z = 6, y = 5, x = 5.5, z = 4, y = 2 \rangle$, la moyenne des 5 événements précédents est égale à 4.5. On prédit alors que le prochain événement qui apparaîtra dans le log aura une durée de 4.5.

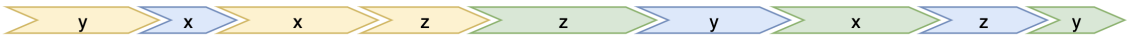


FIGURE 1.24 : Durée des événements d'un log.

Scénario 2. Apprendre les durées d'un événement x à travers toutes ses valeurs passées dans toutes les instances du log, ensuite utiliser cette fonction apprise pour prédire la prochaine valeur de x.

Reprenons la même Figure 1.24, mais cette fois supposons que le log provinne d'un site web où les trois principales actions que l'utilisateur peut faire sont : x : se connecter au site web, y : s'inscrire au site web et z : payer dans le site web. On voudrait prédire, à partir des données de durées des actions, quelle serait la durée de la prochaine connexion, c'est-à-dire l'événement x. Pour suivre le raisonnement du scénario 2, on ne prendra cette fois en considération que les événements x précédents. Supposons que $x_{bleu} = 3$, $x_{jaune} = 6$, $x_{vert} = 5$. Le résultat de la fonction apprise serait approximativement ce qu'on voit dans la Figure 1.25. Ce résultat est obtenu en faisant une extrapolation du modèle appris afin d'avoir une prédiction de la prochaine valeur.

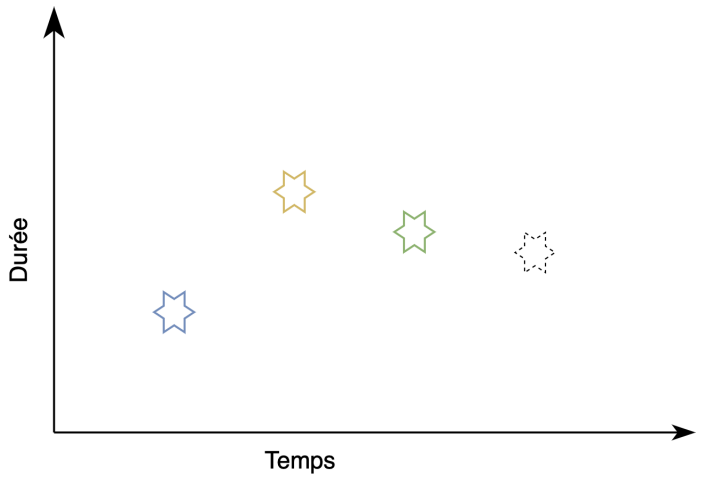


FIGURE 1.25 : Résultat de la fonction d'apprentissage pour le scénario 2 de prédiction.

Scénario 3. Pour des processus entrelacés, prédire la prochaine valeur à apparaitre pour chacune des instances en cours.

On a vu que les événements complexes sont des processus entrelacés, où les événements de plusieurs processus sont mélangés. L'objectif de ce scénario est de montrer qu'il est possible

de diviser le flux d'événements en sous-flux pour traiter chacun des processus individuellement. Supposons que nos deux processus sont des demandes de prêts bancaires. Deux instances de demandes peuvent avoir lieu simultanément (ou presque) et les durées de traitement peuvent différer d'un dossier à l'autre. L'approche du scénario 3 propose de diviser le processus de traitement de demande selon les dossiers clients, et faire des prédictions à partir des événements du processus de chaque individu. La division du flux initial et la prédiction sur chaque processus individuel se font comme illustré à la Figure 1.26.

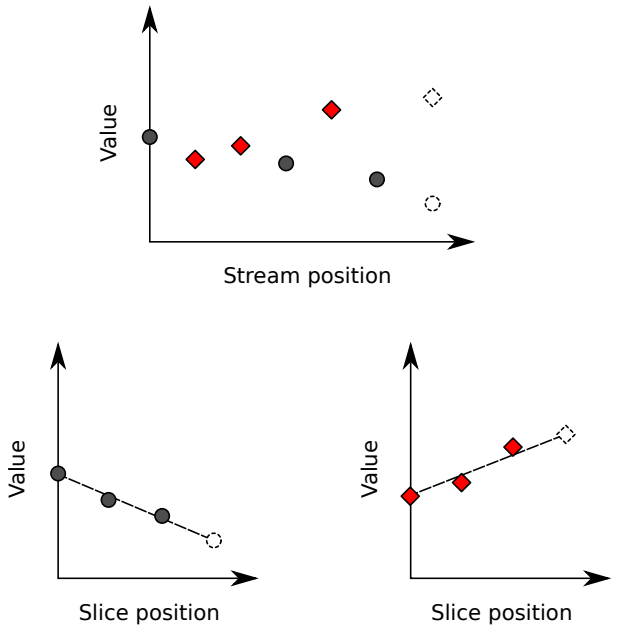


FIGURE 1.26 : Prédiction sur des processus entrelacés.

Scénario 4. Prédire la prochaine valeur d'un flux en utilisant une fonction de régression linéaire générée à partir des *m* derniers événements de l'instance du log.

Nombreux sont les processus pouvant être prédits grâce à l'application d'une simple régression linéaire. La nouveauté dans ce qui est proposé dans le scénario 4 est qu'il s'agit de réaliser cette prédiction non pas de manière statique, mais de façon dynamique, soit prédire

la prochaine valeur à partir des derniers m événements de façon continue. Supposons que la valeur qu'on cherche à prédire est le prix d'une action dans le marché boursier. Comme il s'agit d'une valeur assez volatile, il serait plus intéressant de ne regarder que l'historique proche des prix. La représentation graphique d'un tel traitement serait semblable aux deux graphiques du bas de la Figure 1.26.

Scénario 5. À partir d'une suite d'événements, générer une fonction de prédiction de l'événement prochain le plus probable. Cette fonction peut par la suite être utilisée pour faire les prédictions statiques de l'événement prochain le plus probable.

Lorsqu'on visite un site web, on fait un nombre de clics sur le site avant de le quitter. Une information assez importante pour l'administrateur du site web est de savoir quelles sont les pages visitées le plus souvent et comment amener les clients à s'abonner ou acheter selon le type de business. Une information capitale pour réaliser ceci est de pouvoir prédire à l'avance où va se diriger le client dans le site, autrement dit son prochain clic. Une façon de prédire le prochain clic du client est de générer une fonction de prédiction à partir des anciennes visites de ce même client. On aura ainsi un profil de client, on saura par conséquent «mieux» le guider pour qu'il arrive à la page qu'on souhaiterait qu'il visite.

Scénario 6. Générer un arbre de décision appris à partir des événements du flux et d'un vecteur de caractéristiques pour rendre un verdict si un événement a duré k fois plus que la borne fixée.

Un exemple d'application de ce scénario pourrait être sur les durées de connexions au serveur. Supposons qu'une adresse IP se connecte à un serveur et prenne plus de 5 fois la durée maximale fixée (ou prédite). Une notification devrait être envoyée à l'administrateur

pour l'aviser qu'il y a un risque de sécurité. L'arbre de décision contient des probabilités qui sont interprétées en verdict $\top$ ou $\bot$ comme on le voit dans la Figure 1.27.

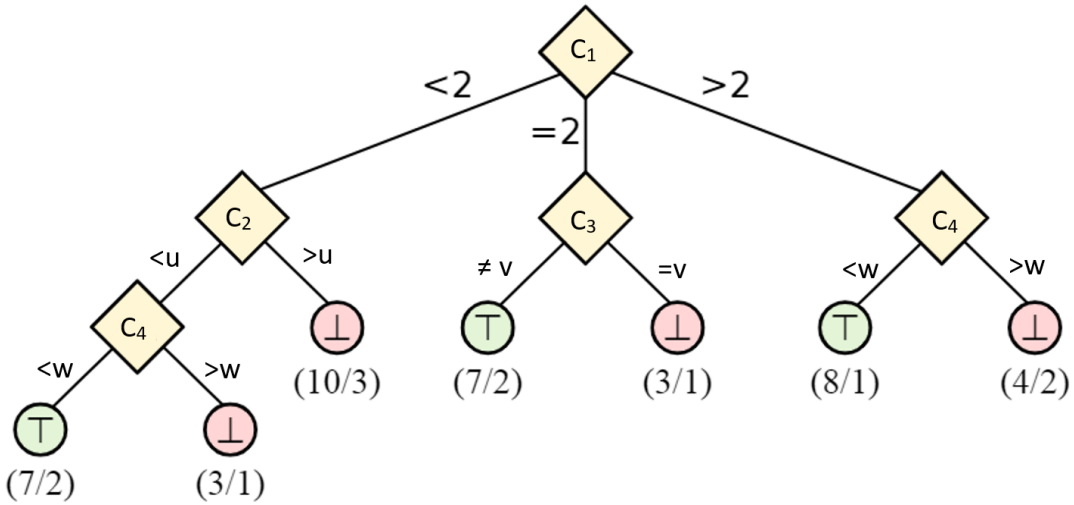


FIGURE 1.27 : Exemple d'arbre de décision pour le scénario 6.

Supposons que c_1 représente la durée totale de connexion d'une adresse IP au serveur, c_2 représente la durée de connexion au port 80, c_3 représente la durée de connexion au port 21 et c_4 représente la durée de connexion au port 110. La durée maximale autorisée sur les ports 80, 21 et 110 est u, v, w respectivement. Une fois l'arbre de décision construit, à l'arrivée d'une nouvelle donnée, dans ce cas une nouvelle durée de connexion, l'algorithme suit le chemin de l'arbre qui correspond au profil de la nouvelle donnée et retournera la feuille correspondante. Le résultat est composé du verdict (si oui ou non il y a dépassement de la durée) et de la probabilité associée au verdict.

1.5 CONCLUSION

Ce chapitre a servi à présenter la notion de flux d'événement et à donner de nombreux exemples de structures de données et de formats standardisés étant utilisés comme tels (§1.1).

On a également présenté un grand nombre de systèmes informatiques produisant des flux d'événements pour des raisons diverses (§1.2). On a également identifié deux notions générales se rapportant au concept de tendance sur un flux d'événements, soit la détection d'écarts de tendance (§1.3) et l'élaboration de prédictions basées sur ces tendances (§1.4). On a présenté plusieurs exemples simples de situations où ces deux concepts correspondent à différents usages en santé, en sécurité informatique, en analyse des processus métier.

Les exemples présentés dans ce chapitre ont plusieurs points communs, même s'ils proviennent de domaines différents. Tout d'abord, ils impliquent le calcul d'une certaine « valeur globale » sur une séquence d'événements, que nous avons appelée une tendance. Deuxièmement, ils nécessitent un retour d'information en temps réel : un écart de tendance doit être détecté dès que possible ; cela exclut la possibilité d'un traitement par lots hors connexion sur un journal précédemment enregistré. Troisièmement, la « notification » qu'ils envoient ne représente pas nécessairement une erreur, mais plutôt une indication qu'un important événement pourrait nécessiter une plus grande attention. Quatrièmement, ils introduisent la possibilité de jumeler la tendance et l'écart calculé avec des techniques de prédiction pour donner une idée de ce qui pourrait se passer dans le futur proche. Cette dernière notion est particulièrement importante, car elle permettrait d'altérer les actions futures si elles ne nous convienent pas.

Partie II

Revue de la littérature

CHAPITRE II
REVUE DE LA LITTÉRATURE

Ce chapitre donne un portrait de toutes les notions rattachées à cette thèse, où l'on cherche à détecter ou prédire des écarts de tendance sur des logs en mode streaming. Afin de bien structurer la thèse, on dresse, dans ce chapitre, un inventaire des concepts importants et de ce qui existe pour chacun des « morceaux » de cet énoncé : les streams, la détection d'écarts et la prédiction.

Ce chapitre sera divisé en 4 sections, chacune d'elles servira à mieux aiguiser la vision du travail fait durant cette thèse. Nous débuterons par l'introduction des outils d'analyse de flux d'événements. Ensuite nous parlons de la détection des anomalies, mesures de tendance et métriques de distance, pour enfin introduire dans la section 3 l'analyse prédictive et ses différentes méthodes. La section 4 va faire une synthèse de ce qui a été présenté, ainsi qu'identifier les aspects qui n'ont pas été étudiés et qui justifient la contribution de la thèse.

2.1 OUTILS D'ANALYSE DE FLUX D'ÉVÉNEMENTS

Dans cette première section, on verra des techniques et outils permettant d'analyser différents types d'événements. Dans un premier temps, on aborde l'analyse des logs de façon générale. Ensuite en verra deux grandes familles d'outils d'analyse de logs : le *runtime verification* et les outils permettant de faire de l'analyse dynamique de logs, le traitement des événements complexes *CEP* pour l'analyse d'événements provenant de plusieurs sources simultanées et ayant un format plus complexe. La section sera conclue par la présentation du *process mining* et de quelques outils spécialisés dans l'analyse des processus métier.

2.1.1 ANALYSE DE LOGS

Une première catégorie de solutions regroupe des logiciels et des bibliothèques spécialisés dans le traitement des journaux d'événements, soit en temps réel, soit sur des fichiers préenregistrés. Certains systèmes d'analyse de journaux offrent des capacités de filtrage de base de type Grep, telles que Snare [104], EventLog Analyzer [105], Splunk [106] et Lumberjack [107]. Cependant, ces outils génériques offrent des capacités de recherche et de filtrage des événements relativement de base. À titre d'exemple, la Figure 2.1 montre l'interface graphique pour la saisie de filtres et d'alarmes personnalisés dans un outil d'analyse de journal commercial appelé ManageEngine EventLog Analyzer. Un utilisateur a la possibilité de sélectionner des événements dans un journal selon des conditions simples sur certains de ses champs. Les requêtes prédéfinies, disponibles dans un autre onglet, sont des fonctions de traitement codées en dur dont le fonctionnement de base peut être légèrement modifié en ajustant une poignée de paramètres.

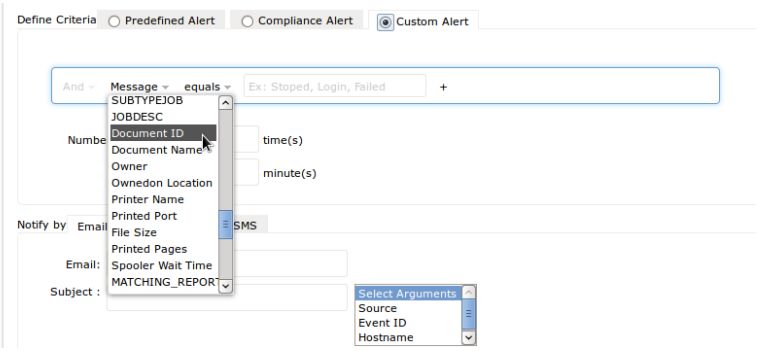


FIGURE 2.1 : Saisie d'une requête personnalisée dans EventLog Analyzer de ManageEngine.

Des solutions plus complètes peuvent être divisées en deux familles, selon l'objectif final de ce traitement de journal. Il existe des systèmes qui permettent d'énoncer des conditions sur des flux d'événements qui se répondent par vrai ou faux : le *runtime verification*, qui correspond à la première famille d'outils qu'on verra dans la sous-section 2.1.2. Ensuite, on élargit sur les systèmes permettant d'accomplir des calculs sur ces flux, qui sont la deuxième

famille d'outils, traitant les événements complexes (*Complex Event Processing*), présenté à la sous-section 2.1.3. Les deux familles seront vues dans les sections suivantes et des exemples d'outils seront donnés et leur fonctionnement expliqué en plus de détails.

2.1.2 RUNTIME VERIFICATION

Au chapitre 1, une définition de la vérification dynamique (*Runtime Verification*) a été donnée. Dans la section qui suit, nous allons introduire quelques outils qui implémentent ce paradigme pour réaliser la vérification de l'exécution en temps réel. Il est à noter que la plus grande partie des outils présentés sont des logiciels libres.

JAVAMOP

La programmation orientée monitoring (Monitoring-Oriented Programming, MOP) a été introduite en 2003 par Chen et al. [108]. MOP est un cadre formel pour le développement et l'analyse des logiciels. L'objectif principal de la programmation orientée monitoring est de réduire l'écart entre les spécifications formelles et l'implémentation d'un programme, et ce, en vérifiant que l'implémentation respecte les spécifications. Il a été décrit plus en détail dans des travaux ultérieurs tels que [5, 109, 110].

Les spécifications de la programmation orientée monitoring sont composées de formalismes logiques tels que la **Logique Temporelle Linéaire** ou LTL, qui est largement utilisée dans la vérification formelle. Les expressions LTL se construisent en combinant trois types d'éléments essentiels : un ensemble de propositions atomiques **P**, des opérateurs logiques tels que $\neg$, $\vee$ et $\wedge$, et des opérateurs temporels modaux dont les principaux sont : **X** : Suivant ou *next* en anglais ; **U** : jusqu'à, de *Until* en anglais ; **F** : Éventuellement, de *finally* en anglais ; **G** : toujours, de *globally* en anglais. L'objectif est de décrire sous quelles conditions une spécifi-

cation est vraie. Il y a une famille d'outils permettant de faire de la vérification d'exécution, qui se base sur des formules de logique temporelle pour faire le monitoring, tel que MonPoly [111] qui sera décrit dans la suite de cette section.

Une expression LTL peut prendre différentes formes ; en voici quelques exemples. Si p et q sont deux formules LTL, $G\neg p$, pUq et Xp sont toutes des formules LTL. L'expression $G\neg p$ décrit une séquence d'événements où p n'est jamais vraie. La formule pUq stipule que p doit rester vraie tant et aussi longtemps que q n'est pas vraie. La dernière expression Xp, indique que la formule p doit être vraie à partir du second événement de la trace.

Dans MOP, les propriétés désirées sont spécifiées grâce aux formalismes logiques, similaires à ceux vus ci-dessus. Des réactions aux violations des propriétés spécifiées peuvent également être ajoutées. Les différentes implémentations de MOP permettent d'intégrer automatiquement les spécifications dans l'application à surveiller, grâce à un moniteur généré à partir de celles-ci. La forme d'une annotation MOP est donnée dans la Figure 2.2. Dans la structure de l'annotation, le nom et les attributs de la spécification sont d'abord définis. Ensuite, le corps de la spécification est décrit. La dernière partie de l'annotation permet de définir les violations et la manière avec laquelle les gérer.

Les annotations sont introduites comme des commentaires dans le langage hôte. Elles commencent par un nom optionnel, suivi d'un nom décrivant sa logique sous-jacente. Les attributs sont optionnels ; ils sont suivis du corps de l'annotation. La dernière partie de celle-ci se divise en deux parties : un gestionnaire de violation et un gestionnaire de validation.

JavaMOP [5, 112] est un outil de développement qui supporte le paradigme MOP. Il contient des implémentations d'algorithmes pour synthétiser des moniteurs à partir de spécifications . Il donne également la possibilité aux utilisateurs d'incorporer de nouveaux formalismes pour les utiliser plus tard dans les spécifications. JavaMOP a une architecture

extensible, qui permet de synthétiser des moniteurs pour les différentes spécifications et les piloter afin d'assurer l'exactitude du comportement pendant l'exécution du programme.

```
... (Java code A) ...
/*@ FTLTL
   Predicate red : tlc.state.getColor() == 1;
   Predicate green : tlc.state.getColor() == 2;
   Predicate yellow : tlc.state.getColor() == 3;
   // yellow after green
   Formula : [](green -> (! red U yellow));
   Violation handler : ... (Java "recovery" code) ...
@*/
... (Java code B) ...
```

FIGURE 2.2 : Une annotation JavaMOP [5].

La Figure 2.2 montre les annotations JavaMOP qui décrivent un feu de signalisation soumis à la propriété « le feu jaune vient après le feu vert ». La spécification est exprimée en logique temporelle linéaire [113]. JavaMOP prend cette annotation et génère automatiquement le moniteur correspondant à la spécification. Un exemple de code du moniteur en Java est montré à la Figure 2.3.

```
... (Java code A) ...
switch(FTLTL_1_state) {
case 1:
   FTLTL_1_state = (tlc.state.getColor() == 3) ? 1 :
     (tlc.state.getColor() == 2) ? (tlc.state.getColor() == 1) ? -2 : 2 : 1; break;
case 2:
   FTLTL_1_state = (tlc.state.getColor() == 3) ? 1 :
     (tlc.state.getColor() == 1) ? -2 : 2; break ;
}
if (FTLTL_1_state == -2) { ...(Violation Handler)... }
// Validation Handler is empty
... (Java code B) ...
```

FIGURE 2.3 : Le moniteur généré pour l'annotation de la Figure 2.2 [5]

TRACEMATCHES

Tracematches [114, 115] est un formalisme de spécifications implémenté comme une extension de AspectJ [116]. AspectJ est une extension créée pour le langage de programmation Java pour permettre de faire de la programmation orientée aspect. Elle permet aux utilisateurs de spécifier via des expressions régulières des propriétés à vérifier en temps réel. Les expressions sont faites avec des variables libres sur la trace d'exécution dynamique.

La programmation orientée aspect permet d'ajouter des fonctionnalités dites *advices* dans différents points d'un programme. Les *advices* sont exécutés lors de l'occurrence d'événements dits *jointpoints*. Le *pointcut* définit les points de coupure du programme, en d'autres termes le *pointcut* définit les endroits où les *advice* seront appliquées.

Les événements d'intérêt sont définis en utilisant les standards de AspectJ, de telle sorte qu'un *tracematch* puisse être considéré comme un pointcut plus expressif. Un *tracematch* définit un patron accompagné d'un bloc de code qui sera exécuté si la trace correspond au patron. Un tracematch est constitué de trois parties : les déclarations de symboles (événements importants), un patron sur les symboles, et un code à exécuter. Une correspondance (match) se produit lorsqu'un suffixe de la trace actuelle, restreinte aux symboles déclarés, est un mot du langage spécifié dans le patron.

```
1  tracematch(Iterator i) {
2      sym hasNext before:
3          call (* java.util.Iterator+.hasNext()) && target (i);
4      sym next before:
5          call (* java.util.Iterator+.next()) && target (i);
6
7      next next {System.err.println("Trouble with "+i);} }
```

FIGURE 2.4 : Exemple de tracematch.

Tracematches exécute deux fonctions. La première est le filtrage de la trace actuelle pour la faire correspondre aux symboles. Une fois la correspondance faite, la seconde fonction de Tracematches est des lier les variables issues de la correspondance.

Prenons un exemple tiré du travail de Bodden et al.[115], illustré dans la Figure 2.4. L'objectif du tracematch est de réaliser la vérification HasNext du tracematch. On commence par déclarer les symboles d'itération. Ensuite, on trouve la déclaration des symboles hasNext et next qui se charge de capturer les deux méthodes respectivement. L'erreur contenue dans la dernière ligne de code n'est déclenchée que lorsque deux next consécutifs.

MONPOLY

Monpoly est un outil de vérification de propriétés logiques et temporelles. Il se base sur un langage appelé MFOTL. Il a été développé pour pouvoir à la fois formaliser les relations de premier ordre entre les données et spécifier des propriétés temporelles. En effet, le langage proposé est une extension de LTL et des opérateurs d'agrégation courants dans les langages de requête de base de données comme SQL. Un algorithme de monitoring pour ce langageSes a également été proposé. Monpoly traite chaque événement comme une petite base de données sur laquelle les requêtes seront appliquées. En plus des opérateurs temporels standards, «Globalement» $\Box_{\varphi}$ qui signifie qu'à partir de ce moment φ doit rester vraie pour tous les suffixes de trace à venir, «Éventuellement» $\Diamond_{\varphi}$, c.-à-d. φ doit rester vraie pour certains suffixes de la trace à partir de maintenant. Des paramètres peuvent être ajoutés à ces opérateurs pour exprimer quelques assertions de base et spécifier un intervalle de temps. La formule $\Diamond_{[a,b]}\varphi$ signifie qu'à partir du moment présent, la propriété φ devrait être vraie à un moment donné entre le moment a et le moment b. Pour représenter le passé, une version noire ($\blacklozenge$ $\blacksquare$) des deux opérateurs a été introduite.

La formule ci-dessous est une illustration du langage proposé pour formuler la politique de détection de fraude suivante : un utilisateur ne peut retirer plus de 10000$ de sa carte de crédit, durant une période de 30 jours.

$$\Box \forall u.\, \forall s.\, [\mathsf{SUM}_a\ a.\, \blacklozenge_{[0,31)}\ withdraw(u,a)](s;u) \rightarrow s \preceq 10000\,.$$

Tout d'abord, la formule indique que l'agrégation devrait rester vraie, tout le temps et pour chaque utilisateur et pour toutes les sommes retirées. La somme (SUM) s de tous les retraits durant les 30 derniers jours, est calculée pour chaque utilisateur u. Une relation binaire est définie entre l'utilisateur u et la somme s de tous ses retraits durant les 30 derniers jours. Celui-ci doit rester inférieur à 10000, sinon il y a une violation de la politique.

RULER

RuleR [59] est un système de vérification de trace, basé sur des règles conditionnelles, dans lequel de nombreuses logiques temporelles utilisées pour la vérification à l'exécution peuvent être compilées. Un programme de surveillance «RuleR» comprend un ensemble de règles nommées, sachant qu'une occurrence positive ou négative d'un nom de règle ou d'un nom d'observation est appelée un «littéral». Une règle peut être active ou inactive et contient deux parties : une première partie dite «conditionnelle», aussi appelée antécédent, qui peut être un ensemble de littéraux conjonctifs et une seconde partie dite «corps de la règle» qui est un ensemble disjonctif d'ensembles de littéraux conjonctifs.

C'est grâce au corps de la règle active que sont définies les règles et les observations à conserver dans l'état suivant, sachant que l'observation actuelle remplit la partie condition. Barringer et al. [59] montrent comment peuvent être codées, dans RuleR, les formules logiques temporelles linéaires pour la surveillance de traces finies. Pour mieux comprendre la

représentation des règles RuleR, prenons un exemple : soit r une règle et a une observation. Considérons la règle suivante :

$$r :\rightarrow a, r \tag{2.1}$$

où r de la partie droite de la règle représente la condition, qui dans ce cas est vide. La partie gauche de la règle représente le corps de la règle. Si r est active, la règle stipule qu'on doit observer a et r reste active pour la prochaine observation. Cette composition exprime que a doit toujours être valide pour toutes les observations futures sinon il y aura une contradiction avec la condition.

Prenons un deuxième exemple : soit r' une règle RuleR, a et b deux observations. Considèrons la règle suivante :

$$r' :\rightarrow a, r'|b \tag{2.2}$$

Si r' est active, on peut avoir deux scénarios où cette règle sera respectée. Soit a est observée et r' reste active pour la prochaine observation, ou bien b est observée. Si aucune des deux conditions n'est valide, la trace ne respecte donc pas la spécification.

RuleR est une version plus simple de Eagle [117], un outil antérieur, plus facile à utiliser à des fins de surveillance. Eagle a été introduit comme une logique temporelle basée sur des règles générales pour la spécification des moniteurs d'exécution.

LOLA

LOLA [57] est un langage de spécification basé sur les flux d'événements. LOLA évalue un modèle et donne un flux en sortie. Le flux de sortie dépend du flux donné en entrée. La spécification en LOLA comprend N variables d'entrée ou variables indépendantes t_i et M variables de sortie ou variables dépendantes $s_j = e_j(t_1, \ldots, t_n, s_1, \ldots, s_m)$. Les e_j sont des expressions de flux. LOLA contient des bool et float, des opérateurs numériques et des opérateurs logique & et | et condition *si* dénotée `ite`.

Une expression peut être un simple `True` ou `False`, comme elle peut être une composition d'expressions de fonctions d'agrégation sur des fenêtres : somme, moyenne, largeur, max, min, etc. Cette imbrication d'expressions permet de modéliser et formuler des problèmes plus complexes.

Soit t_1 t_2 deux variables booléennes, t_3 un flux de variables de type entier et s_i des spécifications LOLA. On voit ci-dessous un exemple des différentes spécifications qu'on peut formuler à l'aide de LOLA, extrait de l'article original [57] :

$$s_1 = \textbf{true}$$

$$s_2 = t_3$$

$$s_3 = t_1 \vee (t_3 \leq 1)$$

$$s_4 = ((t_3)^2 + 7) \, mod \, 15$$

$$s_5 = \texttt{ite}(s_3, s_4, s_4 + 1)$$

$$s_6 = \texttt{ite}(t_1, t_3 \leq s_4, \neg s_3)$$

$$s_7 = t_1[+1, \textbf{false}]$$

$$s_8 = t_1[-1, \textbf{true}]$$

$$s_9 = s_9[-1, 0] + (t_3 \, mod \, 2)$$

La variable s_1 désigne un flux dont la valeur est vraie à toutes les positions, tandis que s_2 désigne un flux dont les valeurs sont les mêmes que celles de t3 à toutes les positions. Les variables peuvent également être obtenues en évaluant leurs expressions à chaque position du flux, les flux de s_3 à s_6 sont des exemples d'une telle approche. Les valeurs d'une variable de flux peuvent être extraites à partir d'un flux spécifique suivant un certain ordre. À titre d'exemple les deux variables s_7 et s_8 dont les valeurs sont égales aux valeurs du flux pour t_1 décalées d'une position, la seule différence entre les deux flux est que la dernière position de s_7 prend la valeur par défaut faux et la première position de s_8 prend la valeur par défaut true. Le flux spécifié par s_9 compte le nombre d'entrées impaires dans le flux t_3 en accumulant $t_3 \, mod \, 2$.

Kohl et al [118] présentent un exemple d'application pour les spécifications LOLA, une formalisation des émissions de gaz des véhicules automobiles et le monitoring en temps réel des spécifications de la régulation européenne. La Figure 2.5 montre un exemple d'expression LOLA pour le calcul du percentile 95 de la vitesse temps accélération positive d'un véhicule *va* sur une fenêtre de *n* événements. La forme de la fenêtre d'agrégation contient la fonction d'agrégation `percentile95`, une expression qui décrit la fenêtre d'événement $(-n, 0)$ et la variable sur laquelle on fait le calcul *va*, lorsque la condition `a_is_positive` est vraie.

Afin de déclencher une alarme, la fonction *trigger* est utilisée, comme on le voit dans la Figure 2.6.

```
output float va95~ := percentile95(va[-n:0 | a_is_positive])
```

FIGURE 2.5 : Spécification de percentile en LOLA

```
trigger is_valid_test & emission_limits_exceeded
```

FIGURE 2.6 : Exemple de Trigger en LOLA

Une des conditions de la régulation européenne sur les émissions de gaz est la limitation de la vitesse de circulation des véhicules dans un milieu urbain à 60 km/h ou moins. La spécification formalisée en LOLA exprimant cette limite est la première ligne de la Figure 2.7 où on déclare la variable `is_urban` sous condition que la vitesse `v` régulée soit respectée. La seconde ligne calcule la vitesse moyenne en milieu urbain `u_avg_v` en utilisant une fenêtre glissante de taille *N* sur le flux, lorsque la condiction `is_urban` est vraie.

```
output bool  is_urban := v <= 60 // 6.3
output float u_avg_v  := avg(v[-N:0 | is_urban])
```

FIGURE 2.7 : Exemple de formalisme de spécification en LOLA

MARQ

L'outil MarQ [56] se base sur les automates à événements quantifiés *(Quantified event automata QEA)* pour faire du monitoring. Il convient à la surveillance hors ligne et à la surveillance en ligne de programmes Java, à l'aide d'AspectJ pour l'instrumentation. MarQ traite des événements paramétriques. Un tel événement est composé d'une paire de nom

d'événement et d'une liste de valeurs de données. Une trace paramétrique est une séquence finie d'événements paramétriques. Une propriété paramétrique désigne un ensemble de traces paramétriques. MarQ se démarque par l'utilisation de la technique *parametric trace slicing* [119] qui le place parmi les outils de monitoring les plus rapides.

Les automates à événements quantifiés (QEA) sont une liste de variables quantifiées avec un automate à événements. Dans un QEA, les variables peuvent être quantifiées ou libres. Un QEA peut avoir zéro ou plusieurs variables qui sont quantifiées de manière universelle ou existentielle. Toutes les variables d'un automate QEA qui ne sont pas quantifiées sont dites libres. Un automate à événements est une machine à états finis avec des transitions étiquetées avec des événements paramétriques de symboles, où les valeurs de données peuvent être remplacées par des variables.

L'outil `QEABuilder` permet de spécifier un QEA comme entrée du système MarQ. Un objet générateur est utilisé pour ajouter des transitions et enregistrer des informations sur la quantification et les états.

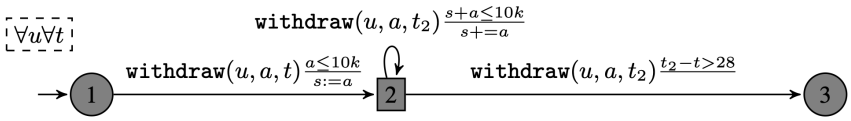


FIGURE 2.8 : Exemple de formalisme de spécification QEA.

La Figure 2.8 est une illustration de la propriété de retrait (*Withdraw*). Elle indique que sur une période de 28 jours, un utilisateur ne peut retirer plus de 10000$ de son compte. On peut voir sur la Figure une façon de représenter cette propriété où u est un utilisateur et t est l'instant de début d'une période de 28 jours. Pour chaque utilisateur u et à tout

moment t le montant retiré par l'utilisateur $\texttt{withdraw}(u,a,t)$ devrait être inférieur à 10000 et l'accumulation de tous les retraits devrait être inférieure à 10000.

Mentionnons au passage que LARVA [58] est un autre outil utilisant le concept de représentation des spécifications par automate.

2.1.3 TRAITEMENT DES ÉVÉNEMENTS COMPLEXES

La seconde catégorie d'outils considérés est celle qui traite des événements complexes (*Complex Event Processing*) ou CEP [120]. Le traitement des événements complexes consiste essentiellement à traiter les flux d'événements, provenant possiblement de plusieurs sources simultanément, afin de générer de nouveaux flux d'événements d'un niveau d'abstraction supérieur. Traditionnellement, les systèmes CEP ont tendance à être plus proches des systèmes de bases de données, et nombre d'entre eux empruntent des termes et une syntaxe à des langages de bases de données tels que SQL.

Plus précisément, C. Luckham définit le traitement des événements complexes comme tout traitement, suivi et analyse de flux de données sur des événements afin de tirer une conclusion et d'en extraire les événements les plus significatifs [120]. Le terme «complexe» est associé à cette méthode, car elle combine des données provenant de différentes sources dans le but de déduire à partir de ces événements simples, d'autres, qui seraient plus complexes. La Figure 2.9 montre l'architecture basique d'un système suivant cette philosophie.

Les événements peuvent d'être de différentes natures, allant d'articles de presse, à des publications sur les réseaux sociaux, de flux boursiers ou d'autres types de données. Un événement peut aussi représenter un changement de valeur comparée à un seuil, un changement d'état ou toute autre observation. Le terme nuage d'événements (*event cloud* en anglais) est

aussi répandu. Il s'agit du terme qui désigne une suite d'événements dont l'ordre est le fruit des relations les liant les uns aux autres.

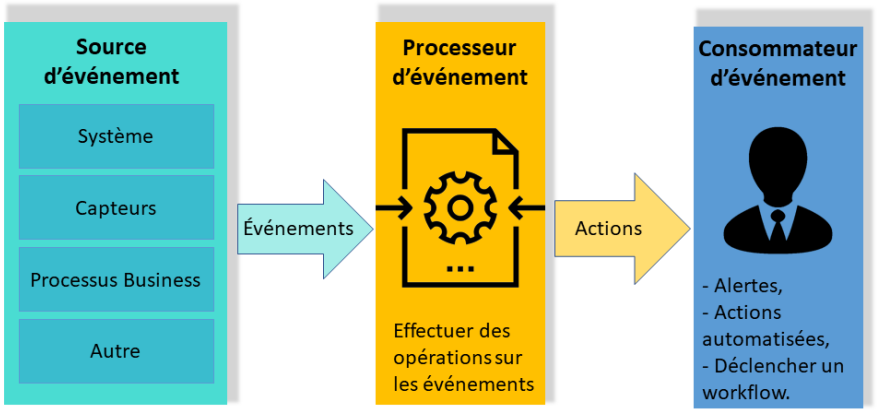


FIGURE 2.9 : Architecture du traitement d'événements

Dans la suite, nous présentons les outils les plus représentatifs de cette catégorie.

TELEGRAPHCQ

La première version de Telegraph [121] a été conçue pour prendre en charge un système de requêtes pour les données web détaillées, le FFF (*Federated Facts and Figures*). Telegraph est une suite de technologies pour le traitement continu des requêtes adaptatives. Les technologies sous-jacentes à cet outil ont été développées dans le but de permettre une adaptation aux graphes de flux individuels de données.

Pour illustrer une requête en TelegraphCQ, prenons l'exemple de la requête dite «instantanée » (*Snapshot query*) de la Figure 2.10. Cette requête se charge de sélectionner le prix de fermeture de l'action de Microsoft «MSFT» durant les cinq premiers jours de bourse.

```
SELECT closingPrice, timestamp
FROM ClosingStockPrices
WHERE stockSymbol = 'MSFT'
for (; t==0; t = -1 ){
    WindowIs(ClosingStockPrices, 1, 5);
}
```

FIGURE 2.10 : Exemple de la requête *instantanée* en TelegrapgCQ [6].

La seconde version de Telegraph est TelegraphCQ. Elle se concentre sur le traitement d'un grand nombre de requêtes continues sur des flux de données à volume élevé et variable. TelegraphCQ comporte deux nouvelles composantes : la première est le CACQ (*Continuously Adaptive Continuous Queries over Streams* [122]), une extension qui prend en charge plusieurs requêtes continues, et PSoup [123] qui elle-même est une extension de CACQ. PSoup permet de prendre en charge l'accès aux données déjà arrivées et à la connectivité intermittente. L'ajout de ces deux extensions à la version initiale de Telegraph a permis l'obtention de meilleurs résultats et a augmenté les performances de l'outil.

STREAM

STREAM [124] est un système de gestion de flux de données (*Data Stream Management System*), conçu pour faire face à des débits élevés de données et à un grand nombre de requêtes continues grâce à une allocation et une utilisation soigneuse des ressources. STREAM prend en charge un langage de requêtes déclaratives et des plans d'exécution de requêtes flexibles.

D'une manière générale, les résultats des requêtes continues sont transmis sous forme de flux de données de sortie, comme ils peuvent également être transmis sous forme de résultats relationnels qui sont mis à jour au fil du temps. Lors de l'enregistrement d'une requête continue à l'aide du langage de requête déclaratif CQL [125], cette requête est compilée dans ce qu'on appelle un plan de requête. Un plan distinct est généré pour chaque requête, puis le système fusionne les plans ayant des sous-plans correspondants ou similaires.

```
Select Istream(Close.item_id)
From Close[Now], Open[Range 5 Hours]
Where Close.item_id = Open.item_id
```

FIGURE 2.11 : Exemple de la requête STREAM

La requête STREAM illustré dans la Figure 2.11 est un exemple tiré de Arasu et al. [125]. Elle a pour objectif d'extraire les identifiants *item_id* pour toutes les transactions fermées après cinq heures de leur ouverture. Autrement dit, la requête permet d'extraire les transactions contenues dans la table *Close* qui ont durées dans la table *Open* pendant cinq heures ou moins. La commande `Istream` se charge d'extraire les tuples et les envoyer sous forme de stream.

Un plan de requête s'exécute en continu et est composé (1) des opérateurs de requête, similaires à ceux d'un SGBD traditionnel ; (2) des files d'attente inter-opérateurs, également similaires à l'approche adoptée par certains SGBD traditionnels ; et (3) des synopsis, utilisés pour maintenir l'état associé à chaque opérateur.

SASE

SASE [55] exécute des requêtes d'événements complexes sur des flux d'événements en temps réel. Il est orienté vers la surveillance d'applications de surveillance basées sur le RFID.

Les requêtes d'événements complexes exécutent un filtrage et une corrélation des événements. Afin de les faire correspondre à des modèles spécifiques. Les événements pertinents sont transformés en de nouveaux événements de plus haut niveau, afin d'être utilisés dans des applications de surveillance externes.

```
PATTERN SEQ(TaskStart a,CPU b,TaskFinish c,CPU d)
WHERE a.taskId = c.taskId AND
      b.nodeId = a.nodeId AND
      d.nodeId = a.nodeId AND
      b.value > 95% AND
      d.value <= 70% AND
     skip_till_any_match(a, b, c, d)
WITHIN 15 seconds
RETURN a, b, c
```

FIGURE 2.12 : Exemple de la requête SASE [6]

Un exemple de requête SASE est illustré dans la Figure 2.12. La requête commence par la commande `PATTERN` qui définit le patron des événements qui seront observés, la commande `WHERE` permet d'exprimer les conditions qui doivent être respectées par le flux d'événements, `WITHIN` la fenêtre sur laquelle on observe les restrictions et enfin `RETURN` indique la valeur qui sera retournée.

L'application de surveillance est notifiée immédiatement à la réception des événements pertinents, grâce à l'exécution basée sur les flux de ces requêtes. Des mesures de sécurité peuvent ensuite être exécutées pour éviter toute perte d'information et tout résultat à effet néfaste.

BOREALIS

Borealis [7] est un moteur de traitement de flux distribué. La collection de requêtes continues soumises à Borealis, dites « diagramme de requêtes », peut être considérée comme un grand réseau d'opérateurs. Le traitement au niveau de ces opérateurs est distribué sur de multiples sites. Un serveur Borealis est exécuté sur chaque site.

Comme le montre la Figure 2.13, la principale composante d'un site est le processeur de requêtes (QP) qui constitue le cœur du processus d'exécution des requêtes à mono-site. Les flux d'entrée sont introduits dans le QP et les résultats sont extraits des files d'entrée/sortie. Une seconde composante est un nœud Borealis. Afin de prendre des actions en collaboration, les modules contenus dans les nœuds Borealis communiquent avec leurs homologues sur d'autres nœuds Borealis.

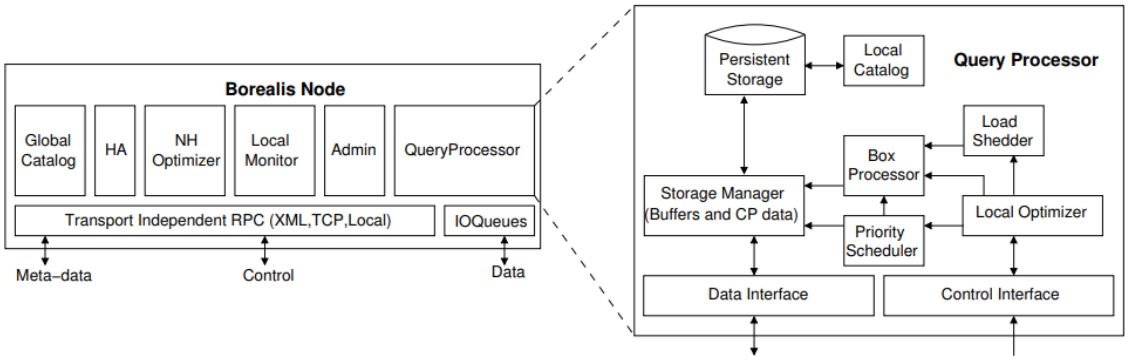


FIGURE 2.13 : Architecture de Borealis [7].

SIDDHI

Siddhi [126] est un moteur de requête utilisé dans le processeur d'événements complexes WSO2, un moteur open source pour l'analyse en temps réel des événements. Il prend en charge les modèles de détection d'événements classiques tels que les filtres, les fenêtres, les jointures et les modèles d'événement, ainsi que des fonctionnalités plus avancées telles que les partitions d'événements et l'association des valeurs d'une base de données à des événements. En termes de capacités d'interrogation, Siddhi prend en charge le calcul de fonctions d'agrégation typiques (somme, moyenne) sur des fenêtres et peut également exprimer une forme de patrons séquentiels sur des événements.

La représentation des événements dans Siddhi se fait par une structure de tuple. L'architecture de l'outil consiste en des processeurs connectés. Les événements arrivant passent à travers l'ensemble des processeurs liés par des files d'attente. Chaque processeur est composé d'un *exécuteur* qui renvoie un booléen qui indique si oui ou non l'événement arrivant est compatible avec le processeur [6]. Si un processeur reçoit plusieurs flux d'événements simultanés, Siddhi envoie tous les événements dans une même file d'attente. Il attribue un identifiant à chaque événement pour reconnaître sa provenance et respecter sa priorité lors du traitement.

```
select f.symbol, p.accountNumber, f.accountNumber
from pattern [every f=FraudWarningEvent2 ->
p=PINChangeEvent2(accountNumber = f.accountNumber)]
```

FIGURE 2.14 : Exemple d'une requête Siddhi [6].

La Figure 2.14 donne un exemple de requête dans le langage de Siddhi. Dans celle-ci, il est exigé qu'un événement p doive suivre un autre événement f et que leurs attributs accountNumber soient identiques ; c'est de cette façon que f et p sont liés. Lorsqu'une séquence

d'événements satisfait ces contraintes, un événement de sortie appelé PINChangeEvent2 contenant le symbole et le numéro de compte identifiant ce modèle est généré.

ESPER

Esper [4] offre à la fois un langage, un compilateur et un moteur de traitement en temps réel d'événements complexes. Il vise le développement rapide d'applications qui traitent de larges volumes d'information. Esper offre un langage appelé *Event Processing Language*, une extension de SQL qui supporte des fonctionnalités pour la reconnaissance de patrons et des fenêtres sur des flux d'événements.

Les événements d'Esper peuvent contenir des données imbriquées [6]. Ils peuvent aussi être une composition d'événements. Chaque portion de requête est associée à un contexte qui trie les événements et les met dans des ensembles appelés *context partitions*.

```
select a.custId, sum(a.price + b.price)
from pattern [every a=ServiceOrder ->
b=ProductOrder(custId = a.custId)
where timer:within(1 min)].win:time(2 hour)
where a.name in ('Repair', b.name)
group by a.custId
having sum(a.price + b.price) > 100
```

FIGURE 2.15 : Exemple d'une requête Esper [6].

À titre d'exemple, la requête de la Figure 2.15 sélectionne un prix total par client, sur des paires d'événements arrivant durant les 2 dernières heures, où le nom des services est filtré et la somme des prix doit être supérieure à 100.

4. http://www.espertech.com

CAYUGA

Cayuga [127] est un système complexe de surveillance des événements pour les flux de données à grande vitesse. Il fournit un langage de requête simple pour la composition de requêtes (avec état) avec un moteur de traitement de requête basé sur des automates à états finis non déterministes avec des tampons.

Une requête en Cayuga comporte trois parties.

1. la clause `SELECT` choisit les attributs à inclure dans les événements de sortie,
2. la clause `FROM` décrit un modèle d'événements devant être mis en correspondance,
3. la clause `PUBLISH` attribue un nom au flux de sortie résultant.

Le filtrage, les sommes et les moyennes font partie des fonctionnalités de manipulation de données supportées par le moteur.

Par exemple, l'expression suivante est une requête Cayuga qui crée un événement de sortie lorsqu'il existe au moins dix événements d'entrée dont l'attribut `summary` contient le mot «iPod» dans le même intervalle de 24 heures :

```
SELECT * FROM
 (SELECT *, cnt AS 1 FROM
   FILTER {contains(summary,''iPod'')=1}(webfeeds))
   FOLD {TRUE, cnt>10 AND dur<1 day, cnt AS cnt+1}
     (SELECT * FROM
     FILTER {contains(summary,''iPod'')=1}(webfeeds))
PUBLISH ipod popularity
```

FIGURE 2.16 : Exemple d'une requête Cayuga [6].

AUTRES OUTILS

Il existe une panoplie d'autres outils qui entrent tous dans le même domaine et qui utilisent les mêmes techniques que ceux décrits précédemment, parmi lesquels peuvent être cités VoltDB [128] et PIPES [5]. De la même manière, Cordies [129], Flink [130], LogCEP [131], StreamQRE [132], TESLA [133], et SPA [134] sont d'autres outils de traitement d'événements et d'analyse des journaux. Ils offrent tous des fonctionnalités de nature similaire à celle de l'un des outils décrits ci-dessus. Une comparaison en profondeur de toutes ces solutions sort du cadre de cette thèse. Le lecteur est invité à consulter un récent rapport technique pour plus de détails sur ce sujet [6].

Également en dehors de cette étude, il existe des systèmes périphériques au traitement réel des flux d'événements, tels que des « courtiers en événements » (en anglais *event brokers*) tels qu'Apache Kafka [135], Flume [136], Samza [137], S4 [138], Spark [139], Storm [140], ZeroMQ [141], etc.

La plupart de ces systèmes peuvent être décrits comme à usage général : ils fournissent des langages de requête pour effectuer des tâches courantes sur les flux d'événements, tels que le calcul des moyennes, le filtrage ou la jointure de flux. D'autres travaux [142] proposent l'utilisation d'un système CEP à usage général (dans ce cas, Esper), et pour écrire des requêtes correspondant à une activité anormale dans un réseau. Les requêtes proposées ne sont pas génériques et s'appliquent spécifiquement à l'activité du réseau. Certains autres travaux se sont concentrés spécifiquement sur le traitement de flux basé sur les fenêtres, qui se concentre sur l'évaluation efficace des fonctions sur les fenêtres glissantes d'événements. Un exemple notable est le système SABRE [143], qui est un moteur de traitement de flux évaluant les

5. https://github.com/umr-dbs/xxl

requêtes SQL en streaming de manière parallèle en utilisant des cœurs CPU et GPGPU. D'autres travaux se concentrant sur les variantes de streaming de SQL incluent [144].

Les outils présentés jusqu'ici sont pour la grande majorité des logiciels libres développés par des chercheurs universitaires. Il existe également des dizaines d'outils commerciaux revendiquant des fonctionnalités de CEP avec des niveaux de détail très variables et pour lesquels il est difficile de fournir des informations sans une documentation détaillée et une mise en œuvre librement disponible. Nous ne mentionnons qu'en passant Amazon Kinesis [145], StreamBase SQL [146], StreamInsight [147], Progress Apama [6], Coral8 [7], IBM WebSphere Business Events [8], IBM Streaming Analytics [9], et Oracle Stream Analytics [10].

Les liens entre CEP et RV ont été étudiés par S. Hallé [148], dans lequel il a été avancé que, alors que le CEP fournissait des fonctions plus riches de traitement et d'agrégation des données, RV pouvait exprimer des propriétés plus souples liées au séquencement d'événements dans un journal. Hormis les fonctions d'agrégation de base (telles que la moyenne et la somme), la plupart des moniteurs d'exécution et des outils CEP ne fournissent aucune autre fonction de traitement de données dans leurs langages respectifs et sont donc incapables de traiter au moins certains des cas d'utilisation décrits au chapitre 1.

2.1.4 PROCESS MINING

Une catégorie spéciale est réservée aux outils et techniques du domaine de l'exploration de processus [4] (en anglais *Process Mining*) , qui peut en quelque sorte être vue comme un croisement des deux types de solutions précédents. L'exploration de processus aborde

6. https://www.progress.com/
7. http://www.coral8.com/
8. http://www-01.ibm.com/software/integration/wbe/
9. https://www.ibm.com/cloud/streaming-analytics
10. https://www.oracle.com/technetwork/middleware/complex-event-processing

spécifiquement la question des journaux d'événements et peut également impliquer une forme d'exploration de données sur ces journaux. Dans sa forme originale, l'exploration de processus se définit comme la « découverte automatique des informations d'un journal des événements » [149]. Le meilleur outil représentatif de cette catégorie est le framework ProM [150], qui fait l'objet d'un développement actif depuis plus de dix ans. ProM est un framework extensible qui prend en charge de nombreuses techniques d'exploration de processus d'affaires. Il propose trois types de plug-ins : les plug-ins de *découverte* calculent les sorties uniquement en fonction des données contenues dans les journaux des événements. Les plugins de *conformité* comparent les données de log à un modèle externe. Enfin, les plugins d'*extension* tirent parti à la fois des données d'un log et des modèles. RapidProm [151] est une extension de RapidMiner [152] qui s'interface avec ProM. Il qui permet d'organiser les étapes de traitement en un flux de travail, - c'est-à-dire un graphe des étapes de traitement, où la sortie d'une étape peut être envoyée comme entrée à la suivante.

Parmi d'autres outils, iDHM [153] est un outil de découverte de processus interactif prenant en compte les données qui combine des techniques de classification et des méthodes heuristiques pour la découverte simultanée de données et du flux de contrôle du processus. Sa méthode de découverte de processus utilise des techniques de classification et permet de distinguer les chemins peu fréquents du bruit aléatoire. MPE [154] est un outil interactif qui intègre l'exploration de processus multiperspective. BupaR [155] est une suite open source de paquetages R qui peuvent être utilisés pour explorer et visualiser les données d'événement et les processus de surveillance. Enfin, Apromore [156] est une plate-forme open-source d'analyse de processus métier.

L'extraction de processus, comme son nom l'indique, se concentre sur les processus. De nombreuses opérations d'exploration de processus tentent de déduire un modèle de processus à partir des données de journal et travaillent à partir de ce modèle par la suite. Cependant,

nous avons vu au chapitre 1 comment les tendances calculées dans les exemples ne font pas référence à un modèle de processus. Pour certains d'entre eux, aucun modèle de ce type ne pourrait même être raisonnablement trouvé. Ppar exemple, un flux de paquets TCP / IP ou des sessions Active Directory. De plus, comme pour les logiciels d'exploration de données, nous ne connaissons pas de systèmes d'exploration de processus qui peuvent générer des informations en temps réel sur un journal d'événements traité en mode continu.

2.2 DÉTECTION D'ANOMALIES

Dans la présente section, on étudiera les techniques visant à détecter des anomalies dans des ensembles de données. Dans un premier temps on aborde les techniques générales de détection de valeurs aberrantes (2.2.1). On verra, ensuite, qu'une de ces techniques est basée sur le principe d'un calcul de distance entre une tendance et une valeur, qui est l'approche retenue dans la thèse, c'est pourquoi on fait ensuite un inventaire de mesures de tendances pour des types de données variés (2.2.2), ainsi que des métriques de distance que l'on peut employer (2.2.3).

2.2.1 DÉTECTION DES VALEURS ABERRANTES

Les données aberrantes [20] sont des points significativement distants du reste des observations d'une même instance. Ces données ont des caractéristiques très différentes par rapport à l'ensemble dans lequel elles sont contenues [157].

Une donnée aberrante peut être due à la variabilité du phénomène étudié, indiquer une erreur dans les mesures, comme des erreurs de transmission ou un défaut dans l'appareil de mesure, et finalement, elle peut être le fruit du hasard. Ces données peuvent indiquer un

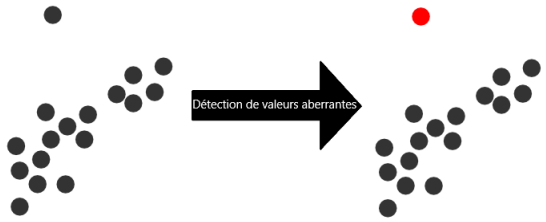


FIGURE 2.17 : Illustration de données aberrantes.

comportement frauduleux ou une erreur humaine. Parfois une variable peut sembler aberrante alors qu'elle ne constitue qu'une valeur extrême d'une population.

Les méthodes de détection des valeurs aberrantes peuvent être classées en trois catégories principales : supervisées, non supervisées et semi-supervisées. Dans la détection *supervisée*, un ensemble de données d'apprentissage est utilisé pour étiqueter les différentes instances. Le modèle prédictif résultant est utilisé pour classer les échantillons en données normales ou anormales [158]. La détection *semi-supervisée* utilise une partie des données non étiquetées comme un ensemble de données d'apprentissage pour étiqueter uniquement la classe normale [159, 160]. Enfin, la dernière catégorie est *non supervisée* : ce dernier type de méthodes de détection des valeurs aberrantes ne nécessite pas d'ensemble de données d'apprentissage, et essaie plutôt de détecter des anomalies en utilisant des données non étiquetées [161, 162].

Pour mieux comprendre et différencier les méthodes de détection, prenons l'exemple suivant : soit β_i la durée d'une session de connexion pour une adresse IP donnée et β un ensemble de données représentant toutes les durées β_i. La détection *supervisée* des données aberrantes consisterait à suivre le schéma suivant :

1. Calculer deux intervalles de durées de connexion et leur associer respectivement les étiquettes «durée normale» et «durée anormale»

2. Écrire un algorithme qui classifie toutes les données entrantes en l'une des catégories spécifiées.

Dans ce type de détection, on oblige l'algorithme à suivre la directive de la spécification et il n'y a pas de possibilité, pour l'algorithme, de créer de nouvelles étiquettes.

Lors de la détection *semi-supervisée* des données aberrantes, on construit un modèle qui connaît le comportement «normal» des données. Dans notre exemple, une spécification de l'intervalle de durée normale de session sera rédigée et le modèle pourra donc détecter toutes les sessions respectant la spécification. Toute durée ne respectant pas la spécification sera considérée «anormale» et aberrante.

La détection *non supervisée* n'utilise pas de données étiquetées. Par conséquent, le modèle suppose que la majorité des durées de connexions sont «normales» et essayera de détecter les valeurs qui sont «loin» du reste des données.

Il existe différentes façons de traiter les valeurs aberrantes dans la catégorie non supervisée, en fonction de la nature des données et des hypothèses émises à leur sujet. Plusieurs groupes de méthodes sont contenus dans la branche dite non supervisée, comme les méthodes paramétriques et non paramétriques [163]. Face à des ensembles de données complexes, d'autres groupes de méthodes apparaissent, chacune pour un type spécifique de données : ensembles de données de haute dimension [164], ou ensembles de données spatiales [165, 166].

Une classe particulière de travaux s'est concentrée spécifiquement sur la détection d'anomalies ou de valeurs aberrantes dans les flux d'événements. Bien que l'utilisation de techniques statistiques sur les flux de données soit étudiée depuis plus de deux décennies [167, 168, 169, 170, 171, 172], l'accent mis sur la détection des anomalies est plus récent. La « détection des valeurs aberrantes à tout moment » est le problème de déterminer à tout moment si un objet dans un flux de données est anormal. Il peut être retracé aux travaux d'Aggarwal [173],

qui ont posé les fondements mathématiques du problème. Plusieurs approches algorithmiques pour une telle détection ont été proposées au cours des quinze dernières années, qui supposent soit que les flux de données sont produits à un rythme fixe [174, 175, 176] (permettant à l'algorithme d'ajuster sa précision en fonction du budget de temps disponible) ou non [177].

Maintenant que les différentes techniques de détection des valeurs aberrantes ont été catégorisées, un dernier point intéressant à considérer sont les différentes approches pour la détection des valeurs aberrantes. En effet, l'utilisation de chacune de ces techniques peut différer selon l'approche choisie. Dans ce qui suit, nous verrons quelques approches existantes.

APPROCHES STATISTIQUES

Les tests statistiques considèrent toutes les données ne suivant pas le modèle statistique sous-jacent comme une valeur aberrante [178].

Un exemple de méthode pour cette approche est la cote Z. La cote Z est un concept fondamental en statistique. Il indique le nombre d'écarts types qui sépare le point de données de la moyenne. L'application de la transformation cote Z met en évidence la distance des données par rapport à la moyenne en fonction de l'écart type. Une cote Z de 3 voudrait dire que la donnée se trouve à 3 écarts types près de la moyenne. On note :

$$Z_i = \frac{x(i) - \mu}{\sigma}.$$

Ou μ est la moyenne de l'échantillon et σ est son écart type et $x(i)$ la ième donnée.

Si les données sont distribuées selon une loi normale, alors, 68.2% des données seraient à 1 cote Z près de la moyenne, 95% seraient à 2 cotes Z près et 99.7% des données seraient à

3 cotes Z de la moyenne. Typiquement, on juge qu'un point de données qui se trouve à plus de 3 cotes Z loin de la moyenne, il est fort probable qu'il s'agisse d'une donnée aberrante.

APPROCHES PAR PROXIMITÉ

Les approches basées sur la distance reposent sur la détection d'une valeur aberrante en analysant son voisinage [179].

Donnons comme premier exemple le Local Outlier Factor (LOF). Cette méthode se base sur le concept des régions locales ou densité locale. La région ou localité est déterminée par le nombre de voisins les plus proches du point de donnée [180]. La densité locale du point d'intérêt est calculée et est comparée à la densité locale des voisins. Si la densité locale du point d'intérêt est sensiblement inférieure comparativement à celle des voisins ceci serait une indication que le point d'intérêt est isolé et par conséquent une valeur aberrante.

La fonction de densité locale est définie comme l'inverse de la moyenne. $K-distance(x_i)$ est la distance qui sépare le point x_i de son k-ème voisin le plus proche. $N_k(A)$ représente l'ensemble des points qui se trouvent dans le rayon de la K-distance. La distance d'accessibilité est le maximum entre la $K-distance(x_j)$ est la distance entre x_i et x_j est formellement donnée par :

$$DA_k(x_i,x_j) = max\{K-distance(x_j), distance(x_i,x_j)\}. \tag{2.3}$$

La mesure de distance choisie dépend de la problématique traitée.

La densité d'accessibilité locale (DAL) est l'inverse de la moyenne de la distance d'accessibilité entre un point x_i et tous ses voisins. Plus la moyenne de la distance d'accessibilité est élevée, moins il y aura de points présents dans le périmètre k autour du point x_i. En d'autres termes, plus la valeur de DAL est petite, plus le groupe de voisins est éloigné.

$$DAL_k(x_i) = \frac{1}{\sum_{x_j \in N_k(x_i)} \frac{DA_k(x_i, x_j)}{|N_k(x_i)|}}. \tag{2.4}$$

Pour déterminer si un point est une valeur aberrante, on calcule le ratio de la moyenne DAL des voisins du point au DAL du point en question. Le facteur d'aberrance local (FAL) (*Local Outlier Factor*) est décrit formellement par :

$$FAL_k(x_i) = \frac{\sum_{x_j \in N_k(x_i)} DAL_k(x_j)}{|N_k(x_i)|} \times \frac{1}{DAL_k(x_i)}. \tag{2.5}$$

Intuitivement, on divise la moyenne des densités d'accessibilité locale des voisins de x_i sur la densité de x_i ($DAL_k(x_i)$). Si ce facteur est proche de 1, ceci indique que la valeur est à l'intérieur d'un cluster de données. Plus précisément :

— Si $FAL_k(x_i) = 1$: distance similaire aux voisins ;

— Si $FAL_k(x_i) < 1$: densité plus élevée que les voisins ;

— Si $FAL_k(x_i) > 1$: densité inférieure à celle des voisins, valeur aberrante.

Un autre exemple de cette approche est Isolation Forest. Contrairement aux autres techniques de détection d'anomalies, *isolation forest* [181] traite les anomalies et non les instances normales. Dans la littérature, la majorité des techniques analysent les instances, retrouvent les étiquettes «normales» et ensuite jugent tout ce qui dévie ou qui est loin de ce dernier ensemble comme «anomalie». L'algorithme *Isolation forest* met à son avantage le fait que les instances aberrantes sont peu nombreuses et qu'elles sont différentes du reste des instances. Elles sont par conséquent plus faciles à isoler. L'algorithme crée des arbres d'isolation en continuant à diviser les données à des seuils et attributs aléatoires jusqu'à ce que chaque point soit isolé.

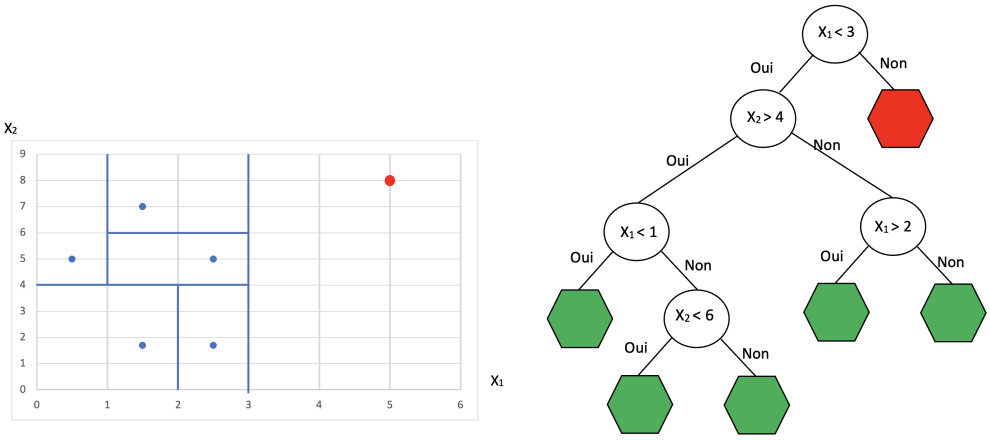


FIGURE 2.18 : Procédure *isolation forest*.

Les anomalies sont, ensuite, définies comme les points ayant des longueurs de chemin les plus courtes. La longueur du chemin est définie comme le nombre d'arêtes qu'un élément x_i traverse de la racine jusqu'aux feuilles de l'arbre (aussi appelé nœuds externes), autrement dit, il s'agit du nombre de divisions par lesquelles l'élément passe pour se retrouver seul et isolé.

Soit x_1 et x_2 deux attributs choisis aléatoirement par l'algorithme pour faire la division. Supposons que $x_1 \in [1,5]$ et $x_2 \in [2,10]$. Le résultat d'une seule itération de l'algorithme *isolation forest* pour la détection de valeurs aberrantes est donné dans la Figure 2.18. Cette division sera répétée et la longueur moyenne du chemin est calculée sur tous les arbres obtenus et pour tous les sommets. Dans le cas de la Figure 2.18, le point ($x_1 = 5, x_2 = 8$) est détecté comme une valeur aberrante.

APPROCHE BASÉE SUR LES CLUSTERS

Les approches basées sur les clusters [182] considèrent comme aberrantes toutes les instances qui ne sont pas contenues dans un groupe de données fortement liées, qui sont loin du cluster le plus proche ou qui sont contenues dans un cluster considéré comme aberrant.

Un exemple de méthode pour cette approche est appelé *Outlier Detection using In-degree Number* (ODIN). Cette méthode de détection de valeurs aberrantes suggère de calculer le nombre d'ensembles de voisins les plus proches auxquels appartient un point donné. Plus haute est la valeur, plus grande est la probabilité que le point se situe dans une région dense. À l'inverse, si le nombre d'ensembles auxquels appartient le point est petit, ceci peut être dû au fait qu'il se situe dans une région vide de l'espace des données. En d'autres termes, le point est une valeur aberrante.

Nous verrons que les deux dernières approches sont utilisées dans cette thèse, à savoir les approches basées sur les clusters et les approches basées sur la distance.

2.2.2 MESURES DE TENDANCE

De manière très générale, le problème qui nous intéresse est de déterminer lorsqu'un flux d'événement dévie d'une certaine tendance. Ceci entraîne, implicitement, le calcul d'une «tendance», que l'on va comparer avec un autre objet en mesurant, de manière abstraite, sa «distance» avec la tendance calculée.

Pour cette raison, il faut dans l'état de l'art donner un inventaire des mesures de tendance existant pour divers types de données, et également des fonctions qui peuvent être utilisées comme mesures de distance auxquelles on fera référence plus tard dans la thèse. Ce sera l'objectif des sections 2.2.2 et 2.2.3.

On va regrouper les définitions selon le type de données manipulé : d'abord des nombres scalaires, ensuite des structures composées comme des ensembles ou des histogrammes, et ensuite des mesures de tendances spécifiques à des données séquentielles.

MESURES SCALAIRES

Par souci de complétude, rappelons d'abord brièvement les mesures de tendance classiques sur des ensembles de nombres [183]. Le moment d'ordre r dans $\mathbb{N}$ d'une variable aléatoire X est un indicateur de sa dispersion. Le moment d'ordre r ordinaire est défini, s'il existe, par :

$$m_r = \mathbb{E}(X^r).$$

Le moment d'ordre 1 est l'espérance : $\mu = m_1 = \mathbb{E}(X)$. Le moment centré d'ordre r est défini, s'il existe, par :

$$\mu_r = \mathbb{E}([X - \mathbb{E}(X)]^r).$$

Le moment centré d'ordre 2 est la variance : $V(X) = \mu_2 = \mathbb{E}[(X-\mu)^2]$.

MESURES NON SCALAIRES

Les mesures précédentes s'appliquent sur des ensembles de nombres, et retournent également en sortie un nombre ; c'est pourquoi on les a appelées les mesures de tendance «scalaires». Il existe également d'autres fonctions qui peuvent être vues comme des mesures de tendance dans une acception plus large (à savoir le «résumé d'un certain aspect d'un objet donné»), et qui produisent en sortie autre chose qu'un scalaire.

Un premier exemple est un histogramme, soit une fonction $h : E \to \mathbb{R}$ associant un ensemble d'éléments à un nombre. Si on a un ensemble de valeurs numériques, on peut créer des catégories correspondant à des intervalles de valeurs, et produire une fonction donnant le nombre de valeurs appartenant à chaque intervalle (d'où le nom «histogramme»). Si on a une séquence de symboles pris dans un ensemble connu, on peut extraire comme tendance une fonction h donnant le nombre d'occurrences de chaque symbole. On peut également considérer la fréquence relative d'occurrence en normalisant les valeurs de la fonction de telle sorte que leur somme soit égale à 1.

L'autre exemple est un graphe $G = (S, A)$, où S est l'ensemble des sommets représentants les objets et A l'ensemble d'arêtes ou d'arcs représentants des relations symétriques ou asymétriques. Si on a un ensemble d'objets avec des relations qui les lient, on peut créer une représentation de ces derniers sous forme d'un graphe, qui peut être considéré comme mesure de tendance sur cet ensemble.

MESURES SUR DES SÉQUENCES

Une autre catégorie de mesures s'intéresse aux séquences ou aux sous-séquences d'une trace donnée. Cette catégorie prend en entrée des séquences de lettres, de syllabes ou de mots et extrait des mesures permettant une exploration plus approfondie desdites séquences.

N-grammes est un concept du Natural langage processing. Il s'agit d'une séquence de N mots. Par exemple, «intelligence artificielle» est un 2-grammes, «les réseaux de neurones est un 4-grammes». Dans toute langue il y'a des expressions, des suites de mots, que l'on rencontre plus souvent que d'autres. Savoir la fréquence d'apparition des séquences de mots ou la probabilité d'apparition du prochain mot dans une séquence est très utile. On pourrait alors

prédire la suite d'une phrase. Ceci aide dans les logiciels de corrections de fautes d'orthographe de grammaire par exemple.

Un modèle N-grammes permet de savoir l'occurrence d'un mot, ou d'un symbole, en se basant sur les N_1 mots le précédant dans la séquence.

Soit E un ensemble contenant une séquence ADN, (l'ADN humain est constitué d'un alphabet de 4 lettres). Si $E = \{ACTCGGCGCA\}$ et N_i l'ensemble de diagrammes obtenu lorsqu'on fixe $N = i$, alors le 1-gramme, le 2-grammes et le 3-grammes sont respectivement :

— $N_1 = \{A, C, T, C, G, G, C, G, C, A\}$,

— $N_2 = \{AC, CT, TC, CG, GG, GC, CG, GC, CA\}$,

— $N_3 = \{ACT, CTC, TCG, CGG, GGC, GCG, CGC, GCA\}$.

Les N-grammes peuvent servir, entre autres, à détecter les anomalies dans les systèmes d'exploitation [184, 185].

MESURES AGRÉGÉES

On peut calculer un ensemble de mesures élémentaires et les assembler dans un vecteur. Si chaque mesure de base correspond, comme mentionné précédemment, à un «aspect d'un objet», alors ce vecteur devient une tendance «multi-aspect».

À cet égard, l'extraction des caractéristiques (*Feature extraction*) [186] est une technique qui permet la transformation des données d'entrée en un ensemble d'attributs. Cette technique permet de résumer l'ensemble des données reçues en entrée en un vecteur représentatif d'attributs. Une mesure agrégée peut, ainsi, être vue comme un vecteur de caractéristiques au sens où on l'entend en data mining : chaque objet considéré est «résumé» par un vecteur de valeurs.

L'extraction des caractéristiques est largement appliquée dans plusieurs disciplines parmi lesquelles la reconnaissance de formes [187] et la reconnaissance des patrons [188].

2.2.3 MÉTRIQUES DE DISTANCE

Un des principaux objectifs de cette thèse est le calcul de la distance de tendance. La section précédente a introduit les différentes mesures de tendances, cette section sera consacrée aux métriques de distance. Une première façon de détecter les anomalies est de comparer deux tendances : une référence et une observée. Dans la présente section, on introduit différentes métriques de calcul de distance.

La distance entre deux objets indique leur degré de similarité. De ce fait, mesurer la similarité entre les deux objets revient à mesurer la distance qui les sépare. Le choix de la mesure de distance est spécifique au problème traité, à l'objectif de l'étude et aux résultats attendus.

MÉTRIQUES SCALAIRES

La mesure de similarité peut être définie comme une fonction $d : D \times D \rightarrow \mathbf{R}^{+}$, appliquée sur un ensemble d'objets et ayant certaines propriétés.

Dans ce qui suit, on donne une liste non exhaustive des mesures de similarité les plus populaires. Soit $x = (x_1, x_2, \ldots, x_n)$ et $y = (y_1, y_2, \ldots, y_n)$ deux vecteurs représentants deux objets ou instances pour lesquels on veut calculer la similarité [189].

Distance de Minkowski [190] Cette distance est une métrique dans un espace vectoriel normé. C'est une généralisation de plusieurs métriques de distance. Elle est définie comme

suit :

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}, p \in \mathbf{N}.$$

Distance de Manhattan Aussi connue sous l'appellation de distance de valeur absolue. Elle peut être vue comme la distance entre deux points dans un réseau routier urbain. Elle est obtenue quand on remplace p par 1 dans la distance de Minkowski. La formulation mathématique de cette distance est la suivante :

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|.$$

Distance euclidienne Cette mesure de distance est la plus commune. La distance euclidienne calcule la racine carrée de la différence entre les coordonnées d'une paire d'objets. Elle fait partie de la famille de distance de Minkowski. Elle est obtenue lorsque p=2 :

$$d_E(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^2 \right)^{1/2}.$$

Distance de Chebychev Cette distance mesure la valeur absolue de la différence entre les coordonnées d'une paire d'objets. Elle est obtenue en évaluant la limite lorsque $p \to \infty$ de la distance de Minkowski. Formellement, cette distance est représentée comme suit :

$$d_C(x,y) = \max_i |x_i - y_i|.$$

MÉTRIQUES NON NUMÉRIQUES

Les types de données rencontrées dans les événements complexes ne sont pas toujours sous forme de scalaires. C'est pour cela que la section suivante sera consacrée à l'exposition de quelques métriques non scalaires pour la mesure de similarité entre deux objets, tel que le

coefficient de Jaccard, les métriques sur les histogrammes, la distance de Levenshtein et la distance entre graphes.

Coefficient de Jaccard L'index de similarité de Jaccard est défini pour deux ensembles A et B comme suit :

$$J(A,B) \triangleq \frac{|A \cap B|}{|A \cup B|}. \tag{2.6}$$

L'index renvoie une valeur comprise entre 0 (les deux ensembles sont disjoints) et 1 (les deux ensembles sont identiques). Lorsqu'il est appliqué à des ensembles de N-grammes, un faible coefficient de Jaccard indique un nombre élevé de N-grammes différents ; fixer le seuil à une valeur $t \in [0,1]$ indiquerait à quel moment un ensemble de N-grammes diffère «trop» des N-grammes de référence.

Prenons par exemple deux ensembles $A = \{a,b,c,d,e,j\}$ et $B = \{a,c,e,i,m\}$, le coefficient de Jaccard pour ces deux ensembles est :

$$\begin{aligned} J(A,B) \triangleq & \frac{|A \cap B|}{|A \cup B|}, \\ = & \frac{|\{a,c,e\}|}{|\{a,b,c,d,e,j,i,m\}|}, \\ = & \frac{3}{8} = 0.375. \end{aligned}$$

Métriques sur les histogrammes

On a vu qu'un histogramme est une fonction $h_i : E \to \mathbb{R}$ associant un objet quelconque $e \in E$ à un nombre réel. Les nombres réels représentent la distribution de la classe correspondante e. Cette technique permet de représenter, entre autres, les couleurs dans une image par la distribution des pixels dans un espace de couleur.

Une catégorie de mesure de similarité entre deux histogrammes se base sur la comparaison des mêmes classes dans les deux histogrammes.

La distance de forme de Minkowski qui est une généralisation de la distance de Minkowski pour les histogrammes est définie par :

$$d_{L_r}(h,h') \triangleq \left(\sum_{e \in E} |h(e) - h'(e)|^r \right)^{1/r}. \tag{2.7}$$

Une autre métrique est appelée la distance χ^2, définie comme suit :

$$\delta_{\chi^2}(h,h') \triangleq \sum_{e \in E} \frac{(h(e) - m(e))^2}{m(e)}, \tag{2.8}$$

où $m(e) \triangleq \frac{h(e)-h'(e)}{2}$. Intuitivement, cette distance mesure à quel point il est peu probable qu'un histogramme représente une population tirée de la population représentée par l'autre. Discuter et comparer les distances d'histogramme sort du cadre de cette thèse. On retrouvera une présentation détaillée des métriques d'histogramme dans Rubner et al. [191].

Distance de Levenshtein [11] Cette métrique détient son nom du mathématicien soviétique Vladimir Levenshtein, qui l'a proposé dans son article [192], en 1965. La distance de Levenshtein est une généralisation de la distance de Hamming [193]. Elle mesure la distance entre deux séquences comme étant le nombre minimum de modifications que subit chaque caractère individuel d'un mot (suppression, remplacement, insertion) pour obtenir un autre mot. Cette mesure est connue en informatique formelle et est utilisée pour calculer la différence entre deux séquences.

11. https://fr.wikipedia.org/wiki/Distance_de_Levenshtein

Formellement, la distance de Levenshtein pour deux chaines de caractères a et b est définie comme suit :

$$lev(a,b) = \begin{cases} \max(|a|,|b|) & \text{si } \min(|a|,|b|) = 0, \\ lev(a-1,b-1) & \text{si } a[0] = b[0], \\ 1 + \min \begin{cases} lev(a-1,b) \\ lev(a,b-1) \\ lev(a-1,b-1) \end{cases} & \text{sinon.} \end{cases}$$

où a et b sont deux chaines de caractères, $a-1$ la chaine a sans la première lettre, $b-1$ la lettre b sans sa première lettre, *lev* est la fonction Levenshtein $|a|$ et $|b|$ sont la cardinalité de a et b respectivement.

Exemple :

La distance de Levenshtein entre « sommeil » et « soleil » est le nombre minimum de changements apporté aux deux séquences pour qu'ils se ressemblent, et est égale à 2, comme suit :

— Remplacer « m » et « l » : so**m**meil → so**l**meil

— Supprimer le deuxième « m » : som**m**eil → sol eil

Graph edit distance

La distance entre deux graphes (*graph edit distance*) est une mesure de similarité entre graphes [194]. Il est possible de transformer un graphe en un autre en appliquant un nombre fini de séquences d'opérations d'édition (de modifications). Cette mesure est égale à la séquence d'opérations permettant de réaliser la transformation et ayant le moindre coût. Les opérations de transformation élémentaires sont : l'insertion d'un nouveau sommet dans

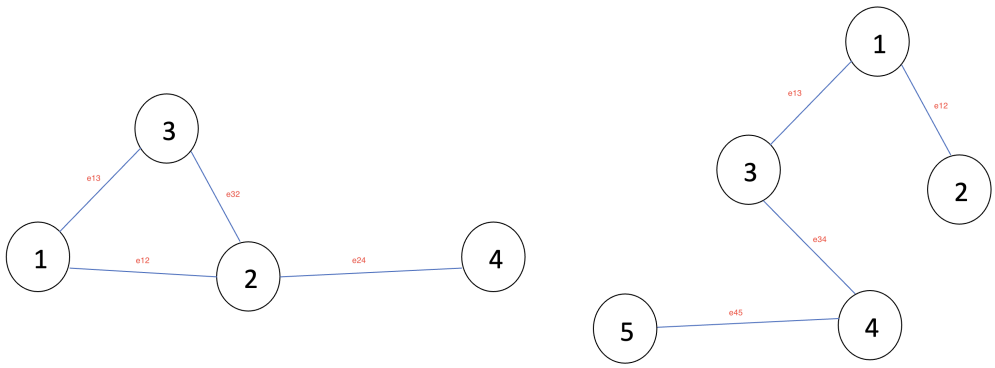
le graphe, suppression d'un seul sommet du graphe, substitution de sommet, insertion d'une nouvelle arête entre une paire de sommets, suppression d'une seule arête entre une paire de sommets, substitution d'une arrête ou de son étiquette.

Étant donné un ensemble d'opérations sur un graphe, la distance entre deux graphes g_1 et g_2 est formellement définie, dans sa forme la plus simple, comme suit :

$$DEG(g_1, g_2) = min_{(e_1,\ldots,e_k)\in P(g_1,g_2)} \sum_{i=1}^{k} c(e_i).$$

Où $c(e_i) \geq 0$ est le coût de l'opération d'édition e_i, et $P(g_1, g_2)$ représente l'ensemble des chemins d'édition, transformant g_1 en g_2.

Par exemple, supposons que le coût de chaque opération de modification est $c(e_i) = 1$. On veut transformer le graphe g_1 au graphe g_2 dans la Figure 2.19



(a) Graphe g_1 (b) Graphe g_2

FIGURE 2.19 : Exemple de transformation de graphes.

Le coût de la transformation du graphe g_1 en g_2 est $DEG(g_1, g_2) = \sum_{i=1}^{5} c(e_i) = 5$, qui est la somme des transformations suivantes :

— e_1 : insertion de l'arrête $e34$ entre les sommets 3 et 4 ;

— e_2 : insertion du sommet 5 ;

— e_3 : insertion de l'arrête $e45$ entre les sommets 4 et 5 ;

— e_4 : Supression de l'arrête $e32$ entre les sommets 2 et 3 ;

— e_5 : Supression de l'arrête $e24$ entre les sommets 2 et 4.

2.3 ANALYSE PRÉDICTIVE

Cette thèse vise non seulement à détecter des anomalies sur des flux d'événements, mais également à effectuer certaines prédictions. C'est pourquoi on dresse un portrait succinct des techniques existantes en termes de prédiction. Par «prédiction», on entend déduire à partir des flux de données historiques ou passées, avec une certaine probabilité, des flux qui vont se produire dans le futur.

2.3.1 PRÉDICTION PAR RÉGRESSION

La régression est l'une des techniques les plus anciennes pouvant être utilisée de manière prédictive. Elle permet de modéliser une relation numérique entre variables dépendantes et indépendantes. Dans la régression classique, la sortie est une fonction qui «ajuste» intuitivement un ensemble de points de manière à ce que l'erreur soit minimisée, selon certaines métriques. Les relations sont généralement modélisées à l'aide de fonctions de prédicteur linéaire, et les paramètres de modèle inconnus sont estimés à partir des données. La régression linéaire est la première et la plus rigoureusement étudiée et utilisée dans la pratique [195]. La raison en est la simplicité de traitement de la relation linéaire avec les paramètres inconnus. Une fois que la fonction de régression est connue, elle peut ensuite être évaluée en des points autres que ceux utilisés pour la calculer, et donc joue le rôle de «prédicteur».

En statistique, l'analyse de régression désigne le modèle mathématique qui établit le lien entre les valeurs d'une variable donnée et les valeurs d'autres variables (prédicteur ou

variables indépendantes). L'exemple le plus connu de régression est peut-être l'identification de la relation entre la taille et le poids d'une personne, affichée dans des tableaux obtenus en utilisant l'équation de régression, évaluant ainsi un poids idéal pour une taille spécifiée.

Le type le plus simple et plus connu de régression est la régression linéaire [196]. Ce type de régression a pour but de chercher une relation linéaire entre une variable expliquée et une variable explicative. Formellement, la régression linéaire est décrite par la formule suivante [197], pour un individu i :

$$Y = X\mathscr{B} + \mathscr{E}. \tag{2.9}$$

Où $Y = \langle y_1, \ldots, y_m \rangle$ et y_i est la variable expliquée (dépendante), $X = \langle x_{i,1}, x_{i,2}, \ldots, x_{i,n} \rangle$ et $x_{i,j}$ variable explicative (indépendante), $\mathscr{E} = \langle \varepsilon_1, \varepsilon_2, \ldots \varepsilon_m \rangle$ et ε_i représente l'erreur ou le bruit, $\mathscr{B} = \langle \beta_1, \beta_2, \ldots, \beta_n \rangle^T$ et les β_i sont des coefficients. Intuitivement, y_i est une variable aléatoire dont la valeur dépend d'autres variables notées par $x_{i,j}$, le bruit est la mesure de l'erreur d'approximation de la valeur de y_i.

Il existe deux types de régression linéaire : la régression linéaire multiple décrite par la formule 2.9 et la régression linéaire simple qui peut être considérée comme un cas particulier de la première catégorie.

Retrouver $\hat{Y} = X\hat{\mathscr{B}}$, l'estimation de la variable Y (aussi dite valeur prédite de Y) revient à retrouver les paramètres du modèle, ou paramètres estimés, $\hat{\beta}_j$. Il existe différentes techniques pour calculer l'erreur commise lors de l'estimation parmi lesquelles le calcul de la différence suivante : $\mathscr{E} = Y - \hat{Y}$.

Il existe un second type de régression dit *régression non linéaire* [198]. On obtient ce type de modèle lorsque les termes ne sont pas linéaires. La majeure différence entre le modèle

linéaire et non linéaire réside dans les techniques de mesure de l'erreur d'estimation. Plus de détails sur la différence entre les deux types de régression peuvent être retrouvés dans [199, 200].

Il existe également différentes variantes de régression à savoir la régression robuste [201, 202] ; la régression polynomiale ; la régression logistique ; la régression non paramétrique [203] (une comparaison avec les méthodes paramétriques peut être retrouvée dans [204]) ; et efin la régression locale.

Les applications de cette méthode statistique dans l'exploration de données se fait dans différents domaines. Le commerce et la finance sont deux domaines où la régression est utilisée pour prédire, par exemple, les montants des ventes de nouveaux produits en fonction des dépenses publicitaires. La régression est encore largement utilisée dans le marché boursier, entre autres, pour la prédiction des séries chronologiques des indices boursiers [205]. L'étude [206] a réalisé une comparaison entre deux différentes techniques de prédiction à savoir les réseaux de neurones et la régression dans le marché boursier d'Istanbul.

La météorologie est un autre domaine connu pour sa large utilisation des techniques de régression pour la prédiction, entre autres, des vitesses et des directions du vent, des températures, et les effets environnementaux tels que la relation entre la concentration de l'ozone et la condition météorologique [207]. Un autre exemple intéressant de cette pratique, qui s'applique à la fois à la météorologie et aux énergies renouvelables, serait celui des éoliennes, pour mesurer et prédire la quantité d'énergie obtenue par la force du vent [208].

2.3.2 CLASSIFICATION

La classification est le processus de placement d'un objet spécifique dans un ensemble de catégories, en fonction des propriétés de l'objet [20]. Une fonction de classification est

produite en associant les caractéristiques d'un objet à une classe. Lorsqu'un nouvel objet est considéré, la prédiction est la classe que lui associe le classificateur en fonction de ses caractéristiques. La contribution de la thèse n'utilise que de façon superficielle ces techniques, c'est pour cette raison qu'on ne va que donner un bref aperçu de celles-ci.

La classification moderne est basée sur quatre composants fondamentaux :

- Classe : la variable dépendante du modèle représentant l'étiquette mise sur l'objet après sa classification.
- Prédicats : les variables indépendantes du modèle, ce sont les caractéristiques et attributs des données à classer et c'est en se basant sur ceux-ci que la classification sera effectuée.
- Ensemble de données d'apprentissage : il s'agit de l'ensemble de données contenant des valeurs pour les deux composants précédents. Il est utilisé pour *entraîner* le modèle afin qu'il puisse reconnaître la classe appropriée, en fonction des prédicats disponibles.
- Ensemble de données de test : il contient de nouvelles données qui seront classées par le modèle (classificateur) construit. Il permet d'évaluer la performance du modèle et la précision de la classification.

La classification est considérée comme un processus à deux phases. Durant la première phase, dite phase d'entraînement, un modèle est construit en se basant sur un sous-ensemble des données (l'ensemble d'apprentissage). Une fois le modèle construit, la seconde phase est entamée, celle de classification appelée *phase de test*. En se basant sur l'ensemble de données de test, on procède à la classification des différents éléments en utilisant le modèle (classificateur) construit.

ARBRES DE DÉCISION

Une représentation naturelle du problème de l'apprentissage à partir d'un ensemble d'instances indépendantes serait par un arbre de décision [21]. Les nœuds d'un arbre de décision correspondant au test d'un attribut particulier. Généralement, le test sur un nœud compare une valeur d'attribut avec une constante. Cependant, certains arbres comparent deux attributs ou utilisent une fonction d'un ou de plusieurs attributs. Les nœuds feuilles donnent une classification qui s'applique à toutes les instances qui atteignent la feuille, ou un ensemble de classifications, ou une distribution de probabilité sur toutes les classifications possibles. Pour classer une instance inconnue, on parcoure l'arbre vers le bas en fonction des valeurs des attributs testés dans les nœuds successifs. Lorsqu'une feuille est atteinte, l'instance est classée en fonction de la classe affectée à la feuille.

Pour illustrer les arbres de décision dans leur forme la plus simple, prenons l'exemple d'une personne qui veut prendre la décision de l'activité qu'elle fera durant cette fin de semaine selon la météo. Un arbre de décision que cette situation pourrait générer est celui de la Figure 2.20.

Une option très intéressante consiste à laisser une méthode d'apprentissage automatique prendre la relève dans l'arbre de décision, après sa construction manuelle. La construction manuelle d'arbres de décision est un bon moyen d'avoir une idée de l'activité fastidieuse consistant à évaluer différentes combinaisons d'attributs à scinder, mais elle s'avère une tâche ardue.

Il existe plusieurs algorithmes dans la littérature qui font la construction d'arbres de décision parmi lesquels ID3 [209], C4.5 (dont un livre a été consacré à ce puissant outil de résolution [210]), J48, etc.

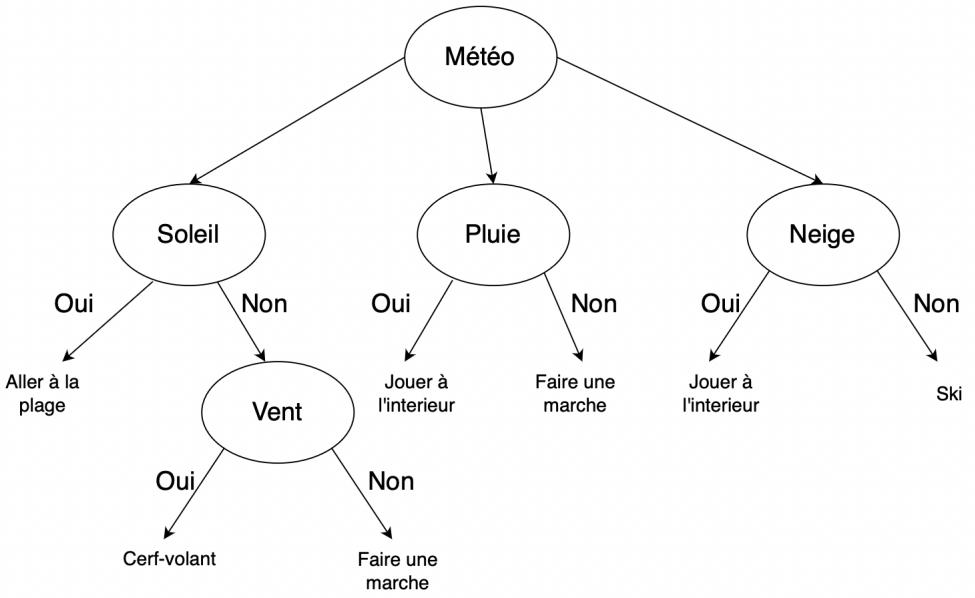


FIGURE 2.20 : Exemple d'un arbre de décision.

K-NEAREST NEIGHBOR

Partant d'un proverbe français bien connu «*dis-moi qui tu fréquentes, je te dirais qui tu es*», la méthode «K plus proche voisins», très utilisée en classification, peut être introduite à partir de cette idée. Dans le domaine de la reconnaissance de formes, l'algorithme *K-Nearest Neighbors* représente cette méthode de classification, dans laquelle un nouvel objet est étiqueté en fonction de ses (K) voisins les plus proches [211].

Le principe de l'algorithme KNN est simple. En effet, étant donné un ensemble d'apprentissage et un nouvel objet à classifier, la «distance» entre le nouvel objet et les objets d'apprentissage est d'abord calculée, et les k objets les plus proches sont choisis. Comme on le voit dans la Figure 2.21, il s'agit de classifier un élément nouveau selon les informations qu'on a sur ses voisins les plus proches, en particulier leur étiquette. Le nouvel élément prendra donc l' étiquette de la majorité de ses voisins [189].

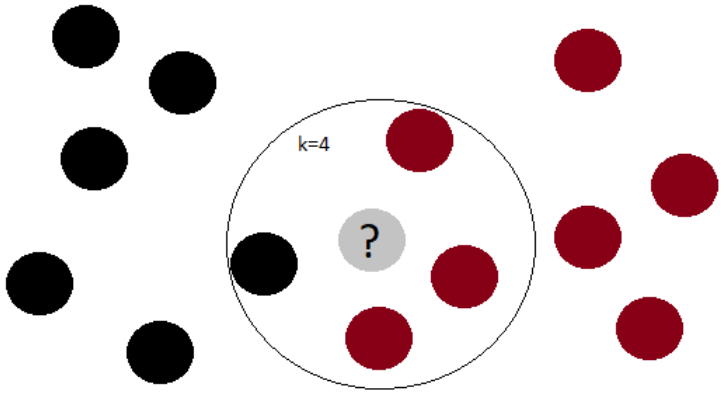


FIGURE 2.21 : K plus proches voisins.

Pour construire l'algorithme, nous avons besoin des éléments suivants :

- Un ensemble d'apprentissage ;
- Une fonction de distance pour calculer la similarité entre les objets ;
- La valeur de k, le nombre d'objets appartenant à l'ensemble de données d'apprentissage, sur lequel on se basera pour obtenir la classification d'un nouvel objet.

Dans son approche naïve, à partir des trois exigences citées ci-dessus, un nouvel objet, non classé, sera classé, par l'algorithme KNN, selon les étapes suivantes :

1. Calculer la distance entre tous les éléments de l'ensemble d'apprentissage et le nouvel objet ;
2. Identifier les k objets les plus proches (k voisins les plus similaires), en prenant en compte les distances calculées dans la première étape ;
3. Attribuer l'étiquette la plus fréquente parmi les k éléments les plus proches de cet objet (vote majoritaire).

L'algorithme du k-plus proche voisin est l'un des plus simples de tous les algorithmes d'apprentissage, puisqu'il consiste simplement à classer un objet par le vote majoritaire de ses voisins.

Lorsque la taille de l'ensemble des données d'apprentissage est importante, l'approche naïve de cet algorithme est intensive en calcul. De nombreuses variantes d'algorithmes de plus proches voisins ont été proposées au cours des années, en cherchant généralement à réduire le nombre d'évaluations de distance et à devenir ainsi plus faciles à traiter.

Un inconvénient du vote majoritaire est que les classes ayant les objets les plus fréquents tendent à dominer la décision concernant la classification d'un nouvel objet. L'alternative peut consister à considérer un « système de vote pondéré », c'est-à-dire pondérer chacun des k voisins les plus proches, en choisissant le poids w en fonction de la distance entre le nouvel objet et le voisin correspondant.

La technique de KNN dépend entièrement du paramètre k, des méthodes variées permettent de sélectionner cette valeur afin d'optimiser les performances de l'algorithme.

LES RÈGLES DE LA CLASSIFICATION

Une représentation très simple et intuitive des connaissances utilise les règles de classification [212]. Une règle est composée d'une condition, qui comprend un ou plusieurs tests, et d'une conclusion, qui associe une instance à la classe à laquelle elle appartient.

Dans son étude, Quinlan [213] propose 4 méthodes de simplification d'arbres de décision parmi lesquelles figurent les règles de classification. Dans son texte, l'auteur affirme qu'il s'agit d'une méthode d'élimination rapide de chemins non concluants, c'est-à-dire qui ne satisfont pas les conditions initiales.

Les règles de classification sont exprimées sous la forme suivante : si l'attribut a est dans la classe A et l'attribut b est dans la classe B, alors l'instance z appartient à la classe Z. À chaque classe correspond un ensemble d'attributs pertinents. Une fois que l'ensemble des règles générées à partir de l'ensemble de données est trié par classes, elles constituent une liste de règles utilisée pour représenter les informations de classification complètes sur l'ensemble de données.

Une autre catégorie d'outils de data mining digne de mention est celle des Algorithmes permettant de faire de l'apprentissage en ligne (apprentissage incrémental). En particulier, les algorithmes dits VFDT *Very Fast Decision Tree algorithm* [214]. VFDT est une technique utilisée dans l'exploration de flux de données pour générer un arbre de décision pour la classification. Pour déterminer quand un nœud feuille doit être remplacé par un nœud branche, il utilise l'inégalité de Hoeffding. L'algorithme est conçu pour gérer des attributs discrets et est conçu pour fonctionner avec des flux de données. L'arbre de décision est créé en remplaçant continuellement les nœuds feuilles par des nœuds de branche.

L'analyse en ligne massive (MOA) [215] est un projet open source développé à l'Université de Waikato en Nouvelle-Zélande, conçu pour extraire de grands flux de données en temps réel. MOA est utilisé pour créer et évaluer des algorithmes pour l'exploration de flux de données. Il est particulièrement utile pour les applications où les données sont générées en continu et rapidement, telles que la surveillance des médias sociaux, la détection des intrusions sur le réseau et l'analyse des marchés financiers. MOA fournit également une interface graphique qui permet de visualiser les flux de données et les résultats de l'analyse. Il est conçu pour être facile à utiliser et pour permettre le prototypage rapide de nouveaux algorithmes.

Il existe également d'autres techniques prédictives, qu'on n'abordera pas en détail, telles que les séries temporelles [64, 216], les modèles bayésiens [217], les réseaux de neurones

[218], les machines à vecteurs de support [219], et les forêts aléatoires [220] pour ne citer que celles-ci.

2.4 CRITIQUE DES SOLUTIONS EXISTANTES

Ce chapitre a pour objectif de donner une vue globale des outils et des techniques impliquée dans le traitement des logs d'événements afin d'y extraire de l'information. La première section a exposé le panorama de familles d'outils d'analyse de logs. On a ensuite introduit des techniques d'analyse d'anomalie et le lien que cette notion a avec l'analyse de logs et surtout l'information supplémentaire que ce type d'analyse apporte comparativement à ce qui est déjà existant. Finalement, un ensemble de techniques de prédiction, pouvant servir et ayant servi dans notre démarche, a été présenté.

Les outils de CEP fournissent un traitement riche et des fonctions d'agrégation sur des données arrivant progressivement. La *runtime verification* permet d'exprimer des propriétés booléennes sur des séquences de logs. Néanmoins, les outils de *runtime verification* et les outils de CEP ne fournissent pas de fonctions plus avancées que les traitements de base telle que le calcul d'une moyenne ou d'une somme. Plusieurs exemples de logs cités précédemment requièrent des traitements plus avancés avec des algorithmes puissants. Aucun des outils cités précédemment ne pourrait prédire le comportement futur des traces qu'il reçoit ou bien ranger les traces dans des catégories bien définies. En d'autres termes, aucun traitement relatif au forage de données (*data mining*) et à l'application d'algorithmes d'extraction d'information ne peut être appliqué par ces outils.

De leur côté, les systèmes développés pour faire du data mining, offrent des fonctionnalités avancées d'exploration de données, de prédiction, de classification, etc. Cependant, aucun de ses systèmes n'est conçu pour traiter des logs d'événements en continu. En effet, ces

outils considèrent toutes les données à traiter comme une source statique et dont le contenu est connu à l'avance.

Comme les deux autres catégories d'outils, les systèmes de data mining n'ont pas la capacité de traiter les exemples vus précédemment. Dans le meilleur des cas, un journal d'événement serait représenté par un tableau ou une liste, à l'exception des algorithmes qui traitent de l'apprentissage incrémental. Cette représentation fait perdre aux données leur aspect *streaming*.

Il convient de noter, toutefois, que ces algorithmes s'appliquent à la détection de valeurs aberrantes par des séries de valeurs exclusivement numériques. Nous avons vu dans les exemples du chapitre 1 qu'il existe d'autres types de tendances qui ne sont pas représentées par des nombres, ni même par des valeurs scalaires. De plus, aucune des solutions proposées, lorsqu'elles fournissent une implémentation accessible au public, ne peut s'intégrer dans les moteurs de traitement de flux existants, limitant leur éventuelle interaction avec ces systèmes. Les réseaux de capteurs ont également fait l'objet de techniques spécifiques à ce cas d'utilisation [221, 222]. Le lecteur est renvoyé à une enquête récente sur ce sujet particulier [223].

Il existe, en effet, différents volets ou domaines qui traitent des traces d'événements. Chaque domaine traite une facette des données sans forcément exploiter toute la complexité de l'information contenue dans les logs. La Figure 2.22 servira d'outil de référence pour bien visualiser et situer chacune de ces catégories, selon son apport.

Selon le type des données traitées, différentes techniques ressortent. Comme on voit sur le graphique, lorsqu'on traite des données statiques, on fait appel à des techniques d'extraction de données avec du data mining via des fonctions arbitraires. Lorsque les données se présentent sous forme de vecteurs, on utilise des techniques de calcul numériques à travers des outils tels

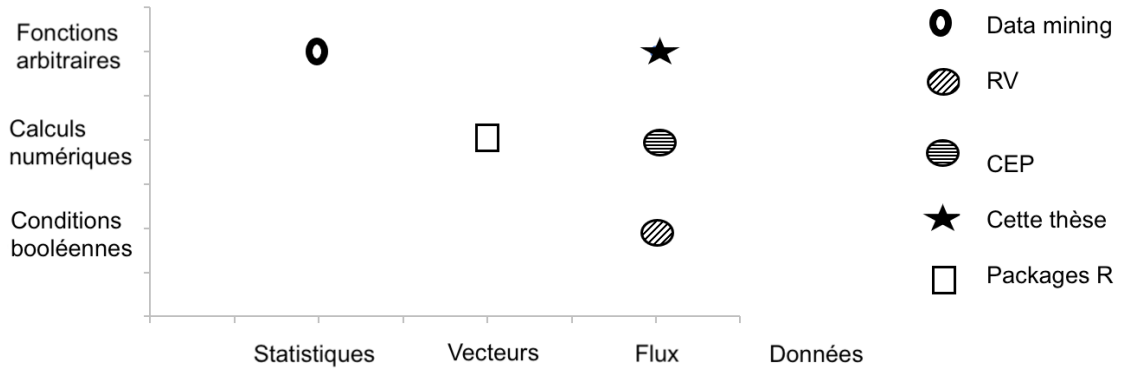


FIGURE 2.22 : Récapitulatif des comparaisons et situation de la thèse.

que *packages R* pour faire l'analyse. Finalement, lorsque les données se présentent sous forme de flux d'événements, il existe deux techniques principales (comme vu durant ce chapitre) pour le traitement des flux : utilisation des conditions booléennes avec des techniques de *runtime verification* et calculs numériques avec des techniques de *complex event processing*.

Cette thèse vient se situer une couche au-dessus pour ajouter un niveau de complétude à l'information extraite et permettre d'obtenir des résultats plus détaillés et complexes sur les caractéristiques des flux. On propose une nouvelle approche du traitement des logs d'événements ainsi que deux types d'analyses distinctes. La première approche vise la détection, qui permet d'extraire les tendances résidentes dans les journaux d'événements via des mesures de tendance. Ensuite, ces tendances sont utilisées comme référence pour détecter et mesurer les écarts des «événements futurs» de la tendance établie, et ce, grâce aux différentes mesures de distance. La seconde approche tire profit des techniques de prédiction, régression et classification, pour extrapoler l'information disponible et effectuer certaines prédictions sur les traces. L'objectif est de prouver qu'il est possible d'appliquer des techniques avancées d'extraction de l'information et d'apprentissage machine sur des logs afin d'extraire des tendances des flux et faire de l'analyse prédictive sur les traces et événements du log.

Partie III

Contribution

CHAPITRE III

DÉTECTION DES ÉCARTS DE TENDANCE SUR LES FLUX D'ÉVÉNEMENTS

Sur la base des observations précédentes, nous décrivons dans ce chapitre diverses techniques permettant de calculer des tendances sur une séquence d'événements tirés d'une source arbitraire ou d'un ensemble de telles séquences. Pour la plupart, les techniques que nous présentons se concentrent sur un traitement *en continu*. Cela signifie que les événements des journaux d'entrée peuvent être reçus un par un en temps réel et que le calcul des tendances correspondantes ou des écarts par rapport à une tendance est produit à la volée. Ceci est en contraste avec les méthodes dites *batch*, qui prendraient un journal d'événements préenregistré dans son ensemble et produiraient un résultat pour le journal entier en une seule opération. La solution proposée va nous permettre de résoudre concrètement chacun des scénarios présentés en section 1.3. Pour répondre à cette problématique, nous proposons un modèle formel sous forme de workflow générique permettant d'extraire des tendances des flux d'événements et de détecter les écarts de celles-ci.

3.1 MODÈLE DE TRAITEMENT DE FLUX D'ÉVÉNEMENTS

Avant de proposer une solution aux lacunes identifiées, on doit définir un modèle abstrait permettant de décrire des transformations sur des flux d'événements génériques. [12]

3.1.1 PROCESSEURS

Les processeurs sont des unités de calcul qui transforment des flux d'entrée en flux de sortie. Un processeur est *stateful* dans le sens ou les résultats obtenus après le traitement

12. Les définitions sont accompagnées des pictogrammes de BeepBeep pour permettre une meilleure visualisation. Néanmoins, BeepBeep n'est pas l'unique outil permettant d'implémenter les workflows proposés, d'autres outils tels qu'Apache Spark permettent également de le faire.

dépendent des événements reçus. Chaque processeur sera représenté par une boîte représentant la fonctionnalité qu'il remplit. Une chaine de processeurs est formée en reliant un ou plusieurs processeurs avec des tubes appelés *pipes*.

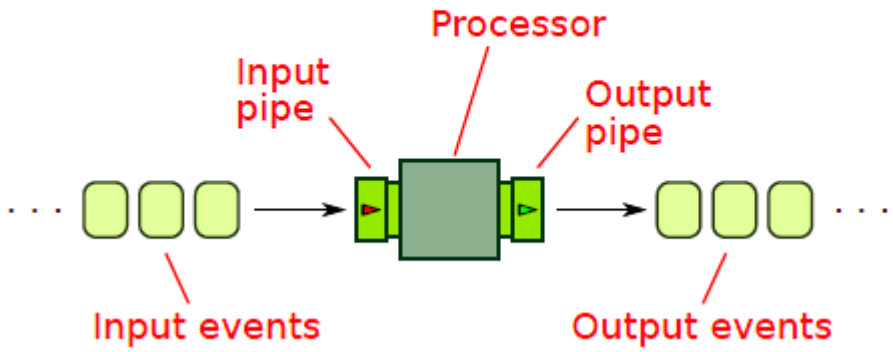


FIGURE 3.1 : Représentation graphique d'un processeur.

La Figure 3.1 est une représentation simple d'un processeur. Il prend en entrée un flux d'événements et donne comme résultat un flux d'événements de sortie. On voit également la représentation graphique des *pipes* dont on a parlé précédemment, qui relient les différents processeurs.

Une définition formelle des processeurs a été donnée par Bédard et Hallé [224]. Nous reprenons les définitions qui s'y trouvent et les réutiliserons pour définir formellement les processeurs utilisés dans le cadre de cette thèse.

Rappelons que Σ est l'ensemble d'événements $\{\sigma_1, \sigma_2 \ldots\}$, $\overline{\sigma}$ est une trace d'événements et $\vec{v} = \langle \overline{\sigma_1}, \ldots, \overline{\sigma_n} \rangle$ est un vecteur de flux d'événements. En prenant en considération les définitions données à la section 1.1, un processeur est formellement défini comme une fonction $\pi : (\Sigma_1 \times \ldots \times \Sigma_m)^* \rightarrow (\Sigma_1' \times \ldots \times \Sigma_n')^*$; Sous la condition que $\vec{v} \preceq \vec{v}'$ implique $\pi(\vec{v}) \preceq \pi(\vec{v}')$.

Un processeur est une fonction π qui prend en entrée un ensemble de flux d'événements et qui renvoie un flux d'événements en sortie après avoir appliqué une ou des transformations.

EXEMPLES DE PROCESSEURS

La définition d'un processeur permet d'en créer différents types selon l'usage général que l'utilisateur veut en faire. Dans cette partie, nous verrons quelques exemples.

— `Trim` : le rôle de `Trim`, illustré dans la Figure 3.2, est de supprimer un nombre déterminé d'événements dès le début d'un flux. Le nombre d'événements à supprimer est spécifié dans le constructeur. Formellement `Trim` est décrit comme suit : $\pi(\vec{v}) \triangleq \langle \vec{v}[k], \vec{v}[k+1], \vec{v}[k+2], \ldots \rangle, k \in N$.

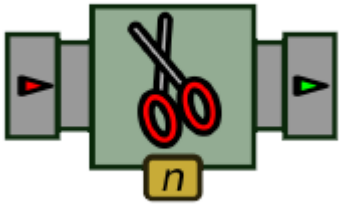


FIGURE 3.2 : Représentation de la fonction `Trim`.

— `Decimate` : ce processeur, montré par la Figure 3.3, rejette les événements d'un flux d'entrée à des intervalles périodiques. Cette tâche peut être réalisée de deux manières : sur la base d'un nombre déterminé d'événements, ou bien basé sur un intervalle de temps fixe. Formellement ce processeur est de la forme suivante : $\pi(\vec{v}) \triangleq \langle \vec{v}[0], \vec{v}[k], \vec{v}[2k], \ldots \rangle, k \in N$.

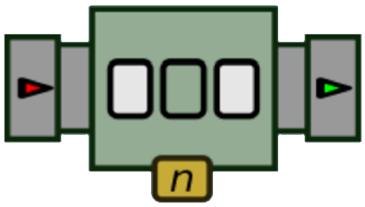


FIGURE 3.3 : Représentation de la fonction `Decimate`.

— `Fork` : l'objectif du processeur `Fork`, qu'on peut voir dans la Figure 3.4, est de diviser le flux original en plusieurs copies identiques. Cette division permet d'effectuer plusieurs calculs séparés sur le même flux. Formellement : $\pi(\vec{v}) \triangleq \langle \vec{v}, \ldots, \vec{v} \rangle$.

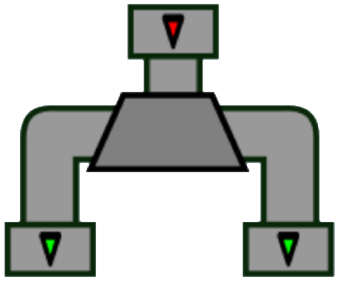


FIGURE 3.4 : Représentation de la fonction `Fork`.

— `Cumulative` : comme son nom l'indique, ce processeur , illustré dans la Figure 3.5, calcule une "somme" cumulative sur les valeurs reçues jusqu'à présent. La fonction f à appliquer sur le flux a deux arguments. Si l'on note la valeur précédente renvoyée par le processeur a, la prochaine valeur renvoyée par le processeur, à l'arrivée d'un nouvel événement b, sera une fonction de a et b notée $f(a, b)$.

Le processeur `Cumulative` est formellement décrit comme suit : soit une fonction $f : \Sigma^2 \to \Sigma$ et une valeur initiale $\sigma_0 \in \Sigma$. Alors :

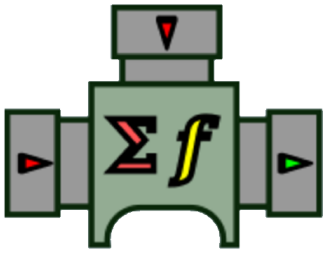


FIGURE 3.5 : Représentation de la fonction `Cumule`.

$$\pi(\langle\sigma\rangle) \triangleq \langle f(\sigma_0, \sigma)\rangle$$
$$\pi(\langle\overline{\sigma}\cdot\sigma\rangle) \triangleq \pi(\langle\overline{\sigma}\rangle)\cdot\langle f(\pi(\langle\overline{\sigma}\rangle)[-1], \sigma)\rangle$$

Où $\pi(\langle\overline{\sigma}\rangle)[-1]$ est le dernier événement produit par π sur le flux $\overline{\sigma}$.

— `Filter` : le processeur de filtrage nommé `Filter` , permet à un utilisateur de garder ou supprimer des événements d'un flux d'entrée de manière arbitraire.

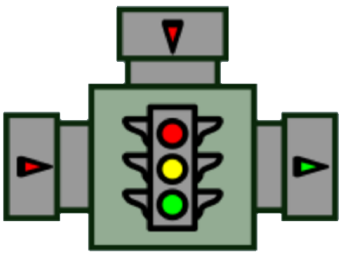


FIGURE 3.6 : Représentation de la fonction `Filter`.

Dans sa forme la plus simple, qu'on voit dans la Figure 3.6, un `Filter` a deux *pipes* d'entrée et une *pipe* de sortie. Le premier canal d'entrées comprend le flux d'événements à filtrer et le second canal reçoit des booléens. Si la valeur du booléen à la position *n*

est *true*, l'événement à la position n dans le flux d'entrées est envoyé à la sortie. La formulation mathématique de ce processeur est la suivante : $\pi : (\Sigma \times \{\top, \bot\})^* \to \Sigma^*$.

— `Window` : ce processeur, illustré dans la Figure 3.7, est l'un des plus complexes. Il effectue un calcul sur une fenêtre d'événements du flux d'entrée. Son fonctionnement est illustré par exemple : supposons que nous voulons calculer la somme des événements reçus sur une fenêtre coulissante de largeur 3. Chaque séquence de trois événements successifs s'appelle une fenêtre ou *window* et donnera lieu à un résultat. Le premier événement de sortie est la somme des événements de 0 à 2, la seconde sortie est la somme des événements de 1 à 3, etc. Paramétré par un autre processeur $\pi' : \Sigma^* \to \Sigma'^*$, `Window` est formellement défini par : $\pi(\langle \sigma_0 \dots \sigma_n \rangle) \triangleq \langle \pi'(\langle \sigma_0 \dots \sigma_{k-1} \rangle)[-1], \pi'(\langle \sigma_1 \dots \sigma_k \rangle)[-1], \dots, \pi'(\langle \sigma_{n-k} \dots \sigma_n \rangle)[-1] \rangle$.

FIGURE 3.7 : Représentation de la fonction `Window`.

— `Slice` : le processeur `Slice`, illustré dans la Figure 3.8, permet de séparer un flux en plusieurs sous-flux et effectuer le même calcul séparément pour chacun de ces sous-flux.

3.1.2 CHAINE DE PROCESSEURS

Relier les sorties d'un processeur aux entrées d'autres crée une chaine de processeurs qui réalisent les calculs voulus. Toute sortie de processeur peut être connectée à l'entrée du suivant,

FIGURE 3.8 : Représentation de la fonction `Slice`.

du moment que le type de sortie du premier corresponde au type d'entrée du second. Un processeur peut recevoir plusieurs types d'événements de différentes sources. Par conséquent, le résultat final pourrait aussi être d'un type quelconque.

L'arité d'un processeur est reconnue au nombre de *pipes* qui entrent (ou sortent de lui). Une convention de couleur est mise au point pour distinguer les types d'événements reçus par les *pipes* à partir de la couleur de celles-ci, comme on voit dans la Figure 3.9.

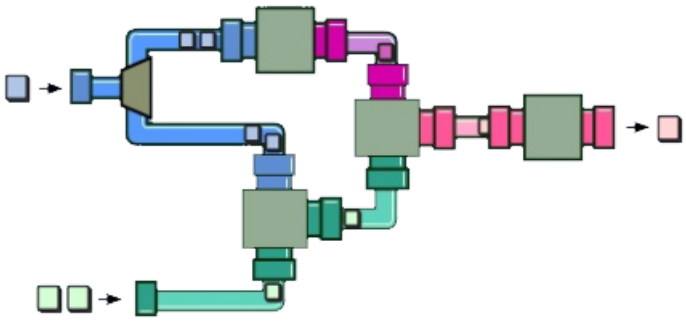
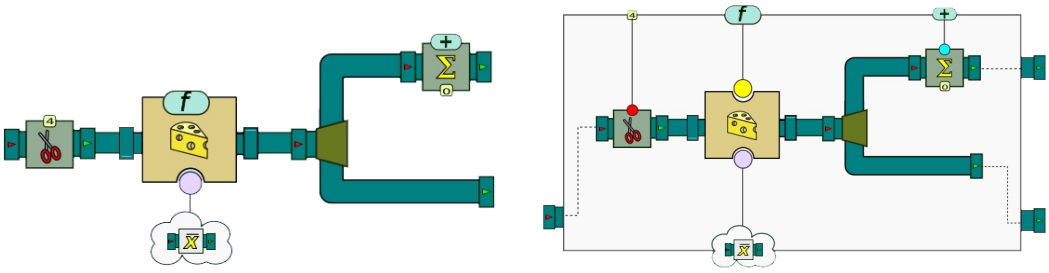


FIGURE 3.9 : Représentation abstraite d'une chaine de processeurs.

3.1.3 GROUPE DE PROCESSEURS

Le groupe de processeurs est un type particulier de processeurs, appelé `Group-Processor`. Il permet d'encapsuler une chaine de processeurs, rendant possible de les manipuler comme s'il s'agissait d'un seul processeur. Le `GroupProcessor` est une boite noire dont le contenu est masqué n'exposant que les *pipes* d'entrée et de sortie situées aux extrémités de la chaine de processeurs le constituant.

La Figure 3.10 illustre une chaine de processeurs reliés par des *pipes* et la même chaine de processeurs encapsuler dans un même `GroupProcessor`.



(a) Exemple d'une chaine de processeurs. (b) Encapsulation sous la forme d'un groupe paramétrisé.

FIGURE 3.10 : Exemple d'un groupe.

Ce modèle, basé sur le concept de boîtes de traitement et de *dataflow*, est compatible avec le modèle de calcul interne de plusieurs systèmes existants : Apache Storm Trident, BeepBeep [66], Aurora [52], Borealis [7], TelegraphCQ [121]. Toute architecture fournissant les éléments décrits ici suffit à faire fonctionner les patterns que l'on va décrire dans la suite.

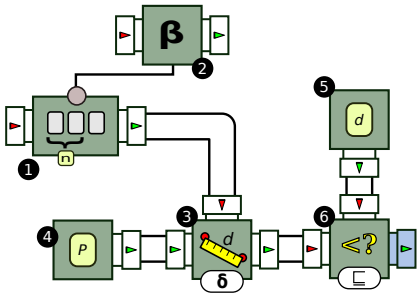


FIGURE 3.11 : Le workflow de distance de tendance statique.

3.2 DISTANCE DE TENDANCE STATIQUE

Nous introduisons maintenant une technique générique pour calculer les tendances sur une séquence d'événements tirée d'une source arbitraire et pour détecter si la tendance calculée dévie d'une certaine «référence». Cette technique est mieux illustrée sous la forme d'un flux de travail, comme le montre la Figure 3.11. Dans cette figure, chaque zone représente une unité abstraite de calcul sur une ou plusieurs séquences d'événements. Le pictogramme dans chacune de ces boîtes décrit les fonctionnalités particulières de la boîte correspondante.

Tout d'abord, un flux d'événements est reçu et envoyé dans un calcul à fenêtre glissante (boîte 1 de la Figure 3.11). Par convention, les événements d'entrée arrivent à gauche et les nouveaux événements de sortie sont produits et sortent à droite de la boîte. Un tel calcul nécessite deux paramètres : la largeur de la fenêtre (appelée n) et un calcul β à effectuer sur chaque fenêtre, représenté par la boîte 2. Concrètement, le calcul de la fenêtre glissante crée une fenêtre des n premiers événements reçus ($e_0, e_1, \ldots, e_{n-1}$). Il donne ensuite cette fenêtre d'événements au calcul β ; la sortie renvoyée par β est le premier événement à être généré par la boîte 1. Le calcul «glisse» ensuite d'un événement et crée une nouvelle fenêtre composée des événements e_1 à e_n. Il donne cette fenêtre à β, dont la valeur de retour est le deuxième événement à être généré par la boîte 1, et ainsi de suite.

Le résultat final est que, à partir d'un flux d'événements en entrée, la boîte 1 crée un nouveau flux, constitué de l'application de β sur des fenêtres successives de largeur n. Intuitivement, β représente le calcul d'une «tendance» sur une fenêtre qui glisse sur le flux d'entrée. Comme le but du processus est de détecter si le flux d'entrée présente un «écart» par rapport à une référence, les tendances calculées seront comparées à celle-ci. C'est l'objet de la boîte 3, qui évalue ce que l'on appelle la métrique de distance. Il lui faut deux arguments : le premier est le flux de tendances calculé par la boîte 1, le second est un flux de valeurs de référence, fourni par la boîte 4. Dans le scénario le plus simple, le motif de référence ne change pas dans l'ensemble du flux d'entrée et la boîte 4 renvoie simplement le même motif de référence P perpétuellement.

Pour chaque paire (P, p), où P est la valeur de référence et p est une tendance calculée par la boîte 1, une métrique de distance $\delta(P, p)$ est évaluée et sa valeur d_p est renvoyée en tant que sortie de la boîte 3. Intuitivement, δ est une fonction qui estime «à quelle distance» la valeur p se trouve de la référence P. Le flux de travail est censé générer une notification lorsque cette distance dépasse un seuil spécifique. C'est la tâche de la boîte 6, qui compare la distance calculée d_p à une valeur de seuil fixe d (fournie par la boîte 5). La fonction $\sqsubseteq$ est appelée *fonction de comparaison* ; $\sqsubseteq(d, d_p)$ renvoie $\top$ (vrai) lorsque d_p est «supérieure» à la valeur de seuil maximale d.

Le résultat étant que le flux de travail de la Figure 3.11 reçoit un flux d'événements d'entrée arbitraire $e_0, e_1, \ldots$ et produit en sortie une séquence de valeurs booléennes $b_0, b_1, \ldots$. Dans des conditions normales, ce flux de travail génère la valeur $\bot$ (faux) à plusieurs reprises. L'occurrence de la valeur $\top$ indique une anomalie : techniquement, on peut en déduire que si $b_i = \top$, alors :

$$\sqsubseteq(d, \delta(P, \beta(e_i, e_{i+1}, \ldots, e_{i+n-1}))) = \top \tag{3.1}$$

En d'autres termes, la tendance calculée sur la fenêtre d'événements e_i à e_{i+n-1} est à une distance supérieure à d de la référence P.

Nous appelons le flux de travail de la figure 3.11 le flux de travail de *distance de tendance statique*. Comme on peut le constater, ce flux de travail de base nécessite un certain nombre de paramètres pour correspondre à un calcul concret. Nous pouvons les résumer formellement comme suit. Soit E l'ensemble des événements à partir desquels est créé le flux d'entrée d'origine, et soit $\mathbb{B}$ l'ensemble des valeurs booléennes $\{\top, \bot\}$. Alors :

— $n \in \mathbb{N}$ est un entier positif représentant la largeur de la fenêtre glissante sur laquelle la tendance est calculée

— $\beta : E^n \to T$ est la fonction de tendance. À partir d'une fenêtre de n événements dans E, elle calcule une tendance $t \in T$. L'ensemble T représente l'ensemble des valeurs de tendance possibles pouvant être renvoyées par β.

— $P \in \mathbb{P}$ est un objet appelé motif de référence

— $\delta : \mathbb{P} \times T \to D$ est une fonction appelée métrique de distance. Elle compare une tendance de référence $P \in \mathbb{P}$ et une tendance calculée $t \in T$ et renvoie une distance $d \in D$, pour un ensemble de distances D.

— $d \in D$ est une valeur de distance appelée *seuil de distance maximum*

— $\sqsubseteq : D^2 \to \mathbb{B}$ est la fonction de comparaison de distance. En général, on s'attend à ce que $\sqsubseteq$ induise une relation d'ordre $\leq$ sur D, définie par $d \leq d' \Leftrightarrow \sqsubseteq (d, d') = \top$.

Nous appelons une *configuration* du flux de travail une combinaison spécifique de définitions pour chacun de ces paramètres.

3.2.1 EXEMPLES DE DISTANCE DE TENDANCE STATIQUE

Un premier avantage du flux de travail à distance de tendance statique est qu'il est très générique. Selon la façon dont nous définissons ces sept paramètres (l'ensemble E plus les six paramètres ci-dessus), le flux de travail de distance de tendance statique peut représenter de nombreux calculs courants sur des flux d'événements. Nous donnons quelques exemples dans ce qui suit.

EXEMPLE 1 : MOYENNE

Comme premier exemple, nous supposons que $E \triangleq \mathbb{R}$ est un flux de valeurs numériques arbitraires. Nous définissons $\beta : \mathbb{R}^n \to \mathbb{R}$ comme :

$$\beta(e_0, \ldots, e_{n-1}) \triangleq \sum_{i=0}^{n-1} \frac{e_i}{n} \tag{3.2}$$

La fonction β calcule la moyenne d'une fenêtre glissante de n événements. Définissons $\mathbb{P} \triangleq \mathbb{R}$ et $\delta(x,y) : \mathbb{R}^2 \to \mathbb{R}^+$ tels que $\delta(x,y) = |x-y|$. Soit $\sqsubseteq \triangleq \leq$ la relation plus petit ou égal sur les nombres réels. Pour une largeur de fenêtre donnée n, une référence $p \in \mathbb{R}$ et un seuil de distance $d \in \mathbb{R}$, le flux de travail de distance de tendance statique détecte chaque fois que la moyenne des n derniers événements diverge de plus de d par rapport à la valeur de référence p.

La boîte β peut être implémentée en tant que chaîne de processeurs principaux, comme illustré à la Figure 3.12. Un flux de valeurs numériques est divisé en deux parties : le flux supérieur est envoyé à un processeur `Cumulate` qui calcule leur somme cumulée. Le flux inférieur est donné à un processeur `TurnInto` qui transforme tout événement d'entrée en constante 1. Ce flux de «1» est ensuite ajouté à un autre processeur `Cumulate`. Les paires de

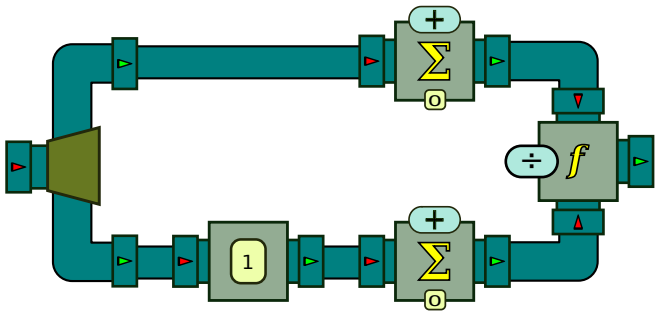


FIGURE 3.12 : La chaîne de processeurs pour calculer la moyenne courante sur un flux d'événements.

nombres produites par les flux supérieur et inférieur sont divisées, ce qui calcule effictivement la *moyenne cumulative* de toutes les valeurs d'entrée reçues jusqu'à présent.

EXEMPLE 2 : VECTEUR DES MOMENTS

L'exemple précédent peut être généralisé à des moments statistiques arbitraires. Si X est une séquence de valeurs numériques, le k-ième moment (noté $E^k[X]$) peut être défini comme suit :

$$\frac{1}{|X|}\sum_{x\in X} x^k \tag{3.3}$$

Pour rappel, le premier moment, $E^1[X]$, représente la moyenne de l'échantillon. On peut définir une fonction $\beta_m : \mathbb{R}^n \to \mathbb{R}^m$ telle que $\beta_m(e_0,\ldots,e_{n-1}) = (E^1,E^2,\ldots,E^m)$, où E^k désigne le k-ème moment d'échantillon de l'ensemble $\{e_0,\ldots,e_{n-1}\}$. La tendance calculée par la fonction β_m est un vecteur des m premiers moments statistiques. Au lieu d'un nombre unique, le motif de référence $P \in \mathbb{P}$ devient également un vecteur de nombres réels $p = (p_1,\ldots,p_m)$, en posant $\mathbb{P} \triangleq \mathbb{R}^m$. Il décrit les moments statistiques attendus des valeurs contenues dans le flux d'entrée.

La fonction de distance $\delta : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ doit maintenant être définie sur des paires de vecteurs $\overline{x} = (x_1, \ldots x_m)$ et $\overline{y} = (y_1, \ldots y_m)$. Il existe un grand nombre de métriques de distance sur un espace vectoriel m-dimensionnel, dont beaucoup peuvent être décrites comme des cas particuliers de la distance de Minkowski [190] vu au Chapitre 2.

EXEMPLE 3 : DISTRIBUTION DE FRÉQUENCES

Les calculs statistiques ne sont pas les seuls calculs de tendance pouvant générer un vecteur de valeurs numériques. Par exemple, soit E un ensemble fini de q symboles discrets arbitraires $\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_q\}$. Définissons $\beta : E^n \rightarrow [0,1]^q$, tel que $\beta(e_1, \ldots, e_n) \triangleq (c_1, \ldots, c_q)$ de telle sorte que c_i est la fréquence du symbole ε_i (c'est-à-dire le nombre d'occurrences de ε_i dans $\{e_1, \ldots, e_n\}$ divisé par n). Par conséquent, $(c_1, \ldots, c_q)$ représente la distribution des symboles dans une fenêtre de n événements. Supposons que $q = 2$ et que $E = \{a, b\}$. Le vecteur $(3/10, 7/10)$ indiquerait une distribution de référence où 30% des symboles dans une fenêtre de largeur n sont des a et 70% sont des b.

Là encore, cette tendance peut être mise en œuvre sous forme d'une chaîne de processeurs, comme illustrée à la Figure 3.13. Cette chaîne utilise le processeur `Slice`. Comme nous l'avons vu, il sépare le flux d'entrée en plusieurs sous-flux en fonction de la valeur de sortie d'une fonction appelée *fonction de découpage en tranches*. Dans le présent exemple, la fonction de découpage est simplement la fonction identité (Id) ; cela aura pour effet de créer des sous-flux pour chaque symbole distinct. Chacun de ces sous-flux est introduit dans une instance distincte de la boîte supérieure, qui calcule et affiche le nombre de symboles reçus jusqu'à présent. La dernière boîte applique une fonction appelée N, qui prend la sortie du processeur de tranches (un tableau associatif entre les symboles et le nombre d'occurrences) et normalise les valeurs, produisant le vecteur souhaité.

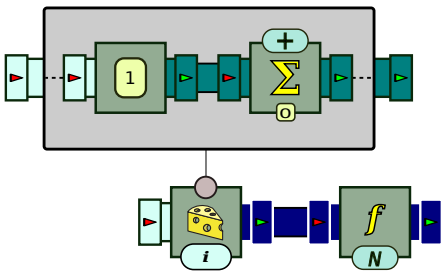


FIGURE 3.13 : La chaîne du processeur pour calculer la distribution des symboles sur un flux d'événements.©Massiva Roudjane

Symbole	Fréquence
a	$3/10$
b	$7/10$

TABLEAU 3.1 : Un exemple simple d'un tableau associatif entre les symboles et leurs fréquences relatives. ©Massiva Roudjane

En utilisant la distance de Chebychev $\hat{\delta}_\infty$ comme métrique de distance et $p = (p_1, \ldots, p_m)$ (où $p_i \in [0,1]$ pour tout i) comme tendance de référence, le flux de travail de distance de tendance statique détectera chaque fois qu'un symbole ε_i apparaît à une fréquence qui diverge de plus d'un seuil d de sa distribution de référence.

Dans ce contexte, la distance de Chebychev peut également être interprétée comme une distance d'histogramme. On peut légèrement modifier la définition de N sur la Figure 3.13, de manière à renvoyer un tableau associatif $H : E \to \mathbb{R}$ entre chaque symbole et sa fréquence relative, similaire à celle présentée dans le tableau 1. H est donc interprétée comme une fonction, où $H(e)$ désigne la fréquence relative du symbole $e \in E$.

Une fonction de distance générale pour deux histogrammes $H, H' : E \to \mathbb{R}$ est la distance de forme de Minkowski ou χ^2 définies dans la section 2.2.3.

EXEMPLE 4 : N-GRAMMES

Les tendances ne doivent pas nécessairement être basées sur des valeurs numériques, ni même renvoyer des valeurs numériques. Un moyen possible de caractériser un flux d'événements consiste à analyser la succession d'étiquettes d'événements qui se produisent. Soit $e_0, e_1, \ldots$ un flux de telles étiquettes. Un N-grammes est un N-uplet $\langle e_i, e_{i+1}, \ldots, e_{i+N} \rangle$ pour un certain $i \geq 0$. Autrement dit, il s'agit d'une liste de N événements successifs dans le flux. Un ensemble de N-grammes peut être créé en accumulant les N-grammes commençant à la position 0 puis 1, etc. Par exemple, si $\overline{e}$ est le flux a, b, c, a, b, a, fixer $N = 2$ produira l'ensemble S de digrammes suivant :

$$\{\langle a, b \rangle, \langle b, c \rangle, \langle c, a \rangle, \langle b, a \rangle\} \tag{3.4}$$

À noter que puisque S est un ensemble, le digramme $\langle a, b \rangle$ n'est inclus qu'une fois, même s'il apparait deux fois dans le flux.

L'accumulation de l'ensemble de N-grammes (pour une valeur appropriée de N) a été considérée comme un moyen efficace de créer un «résumé» d'un flux, en particulier dans les flux générés à partir d'une activité présentant des modèles réguliers. Dans un tel cas, les mêmes blocs d'événements successifs sont susceptibles de se produire fréquemment. En revanche, un écart par rapport au comportement normal sera détecté grâce à la présence de N-grammes différents de l'ensemble de référence. En fait, l'utilisation de N-grammes a été suggérée pour la détection de comportements anormaux en vision par ordinateur [225] et dans les processus des systèmes d'exploitation [226].

La Figure 3.14 montre comment un ensemble de N-grammes peut être calculé en tant que chaîne de processeurs. Un processeur de fenêtre de largeur N reçoit pour instruction

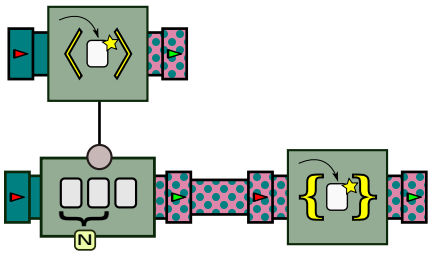


FIGURE 3.14 : La chaîne de processeurs pour calculer l'ensemble des N -grammes sur un flux d'événements.

d'accumuler les événements en entrée dans une liste. Cette liste, qui est en fait un N-gramme, est ensuite sortie et envoyée à un processeur appelé `PutInto`, qui accumule ces listes dans un ensemble. Le résultat final est une chaîne de processeurs qui produit un ensemble constamment mis à jour de tous les N-grammes vus jusqu'ici dans le flux d'entrée.

Étant donné que la tendance calculée est maintenant un ensemble, il est nécessaire de disposer d'une mesure de distance appropriée aux ensembles. Un candidat naturel pour une telle distance est l'index de similarité de Jaccard 2.2.3.

Cependant, étant donné que le processeur β dans le modèle de distance de tendance est évalué sur une fenêtre de largeur m, il est possible que cette fenêtre ne présente qu'un petit sous-ensemble de tous les N-grammes inclus dans la référence. Cela donnerait un faible coefficient de Jaccard, mais n'indiquerait pas nécessairement que la séquence d'événements observées diverge de la référence. Par conséquent, comme mesure de distance alternative, on pourrait définir :

$$J'(A,B) \triangleq \frac{|A-B|}{|B|}. \tag{3.5}$$

Cette métrique compte plutôt la proportion d'éléments dans A qui ne sont pas dans la référence B, par rapport à la taille totale de B.

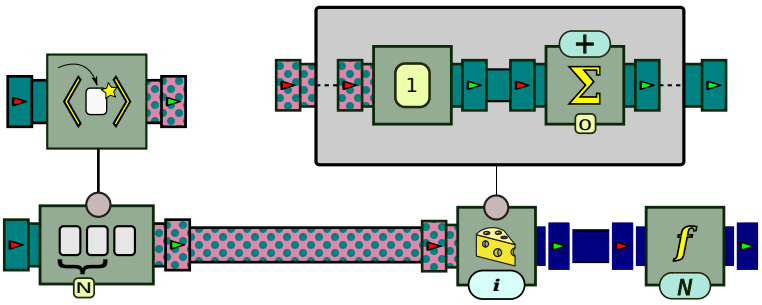


FIGURE 3.15 : Calcul de la distribution de fréquence de *N*-grammes sur un flux d'événements.

Divers calculs de tendance peuvent également être combinés. Par exemple, on peut calculer des *N*-grammes sur un flux d'événements, puis considérer la distribution de fréquence de ces *N*-grammes, plutôt que de les accumuler dans un ensemble. Cela peut facilement être réalisé en connectant la sortie du calcul de *N*-gramme (illustré à la Figure 3.14) en tant qu'entrée d'une chaîne de processeurs calculant la distribution de fréquence des symboles (Figure 3.13). Dans ce cas, les «symboles» sont chacun des *N*-grammes individuels. La chaîne de processeurs résultante est illustrée à la Figure 3.15.

EXEMPLE 5 : TENDANCES SUR LES TRANCHES

Dans certains cas, une seule source d'événements peut en réalité contenir plusieurs flux entrelacés, chaque flux correspondant au scénario d'un objet ou d'une entité particulière. Ceci se produit en particulier dans les journaux de processus d'entreprise, qui stockent les événements relatifs à plusieurs instances du même processus, chacun à un moment différent de son exécution. Par exemple, un journal des instances pourrait contenir un événement *Register* indiquant le début d'une nouvelle instance i du processus, suivi d'un événement *Reject* indiquant la fin d'une autre instance i'. Dans ces cas, il ne fait aucun sens de corréler directement ces événements sans les séparer d'abord en «sous-flux» ne contenant que les événements d'une seule instance de processus.

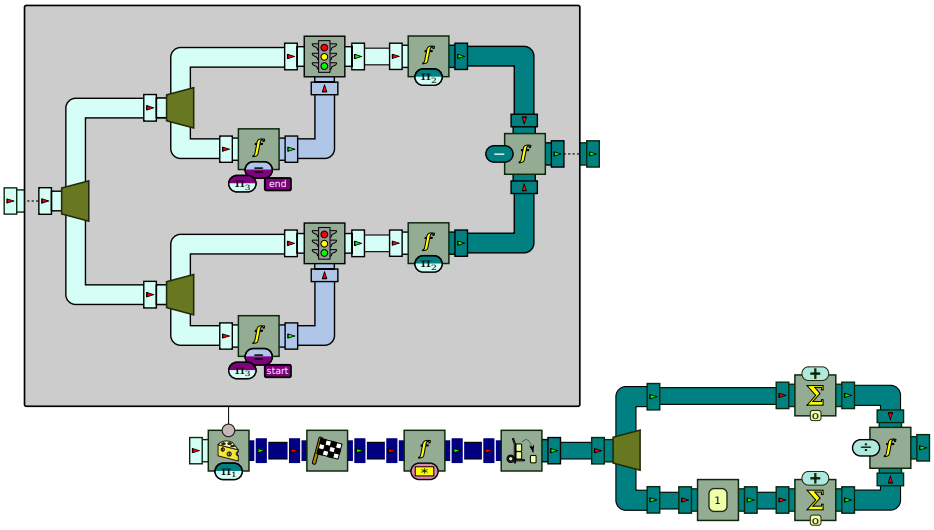


FIGURE 3.16 : La chaîne de processeurs pour calculer la durée moyenne des instances de processus.

Encore une fois, le processeur `Slice` est approprié pour ce type de traitement . Supposons que l'ensemble d'événements E soit composé de triplets $(p,t,\ell) \in \mathbb{N} \times \mathbb{N} \times L$, où p est un identificateur numérique pour les instances de processus, t est un horodatage et ℓ est l'étiquette d'un événement (tel que Register) extrait d'un ensemble L. Supposons également qu'il existe des fonctions $\pi_1 : E \to \mathbb{N}$, $\pi_2 : E \to \mathbb{N}$ et $\pi_3 : E \to L$, chargées d'extraire respectivement l'horodatage, l'identificateur de processus et l'étiquette d'un événement. Un processeur `Slice` peut utiliser π_1 pour diviser le log en sous-flux pour chaque instance de processus.

Sur chaque sous-flux, supposons que nous souhaitons calculer la durée totale du processus. Celle-ci peut être obtenue en comparant les horodatages des événements de début et de fin de cette instance. Le résultat final est la chaîne illustrée à la Figure 3.16. En un mot, les heures de début et de fin de chaque instance de processus sont extraites et soustraites. Une fois que le dernier événement de la fenêtre est reçu, les valeurs du tableau associatif sont extraites. Ces valeurs sont sorties une par une et envoyées dans une chaîne calculant la moyenne (identique à

celle de la Figure 3.12). Le résultat final de cette chaîne de traitement est un flux d'événements, chacun contenant la durée moyenne des processus terminés.

3.3 MOTIF DE RÉFÉRENCE MULTIMODAL

Les valeurs de référence que nous avons vues jusqu'à présent sont appelées *unimodales* : elles consistent en un seul nombre, vecteur ou distribution. Pour ne pas déclencher une alarme, le flux d'événements considéré doit rester proche de cette valeur de référence unique. Cependant, il existe des situations où la référence est faite de plusieurs valeurs : celles-ci sont appelées *multimodales*.

EXEMPLE 6 : MOTIF DE RÉFÉRENCE BIMODAL

Supposons que le flux de symboles discrets de l'exemple 3 puisse suivre l'une de deux distributions possibles : 30% de a et 70% de b, ou l'inverse. Ceci peut être modélisé par deux modèles de référence : les vecteurs $(3/10, 7/10)$ et $(7/10, 3/10)$. Il n'est pas possible de savoir à l'avance à laquelle de ces deux distributions le flux étudié est conforme, mais on peut souhaiter qu'il reste proche de l'une ou l'autre de ces distributions.

Ceci est le premier exemple où les ensembles $\mathbb{P}$ et T diffèrent. Ici, $T \triangleq \mathbb{R}^q$, alors que $\mathbb{P} \triangleq 2^{\mathbb{R}^q}$. En d'autres termes, la tendance calculée est une distribution, tandis que le motif de référence est un *ensemble* de distributions. Ceci, à son tour, a un impact sur la fonction de distance δ. Une façon possible de la définir est la suivante : $\delta_{\delta'} : 2^{\mathbb{R}^q} \times \mathbb{R}^q \to \mathbb{R}$ est tel que :

$$\delta_{\delta'}(\{\overline{p}_1, \ldots, \overline{p}_\ell\}, \bar{t}) = \min_{i=1}^{\ell} \{\delta'(\overline{p}_i, \bar{t})\}. \tag{3.6}$$

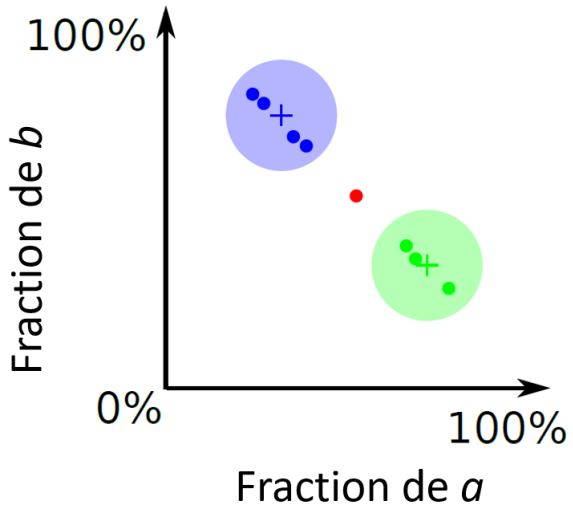


FIGURE 3.17 : Un exemple d'un motif de référence bimodal.

Cette fonction prend en paramètre une autre métrique de distance $\delta' : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$; par exemple, il peut s'agir d'une des métriques d'espace vectoriel que nous avons présentées dans l'exemple précédent. Intuitivement, la distance entre un vecteur de tendance calculé $\bar{t}$ et un ensemble de vecteurs $\{\overline{p}_1, \ldots, \overline{p}_\ell\}$ est définie comme la plus petite distance entre $\bar{t}$ et l'un des $\overline{p}_i$, selon la métrique de distance dans l'espace vectoriel δ'.

Revenons à notre exemple. Utilisons pour la métrique de distance $\hat{\delta}_2$, la distance euclidienne classique, et définissons comme seuil la valeur $d = 0,07$. Comme l'illustre la Figure 3.17, chacun des deux vecteurs de référence peut être décrit sous forme de points dans un espace à deux dimensions. Avec la distance métrique spécifique et le seuil de distance utilisé ici, ils représentent le centre de deux disques avec un rayon de $0,07$. Chaque fenêtre de 10 événements peut également être représentée sous forme de point dans ce graphique, correspondant à la fréquence des a et b qu'elle contient. Les fenêtres situées dans les disques bleu et vert sont «suffisamment proches» de l'un des vecteurs de référence, tandis que la fenêtre des événements représentée par le point rouge est trop éloignée de l'un ou de l'autre.

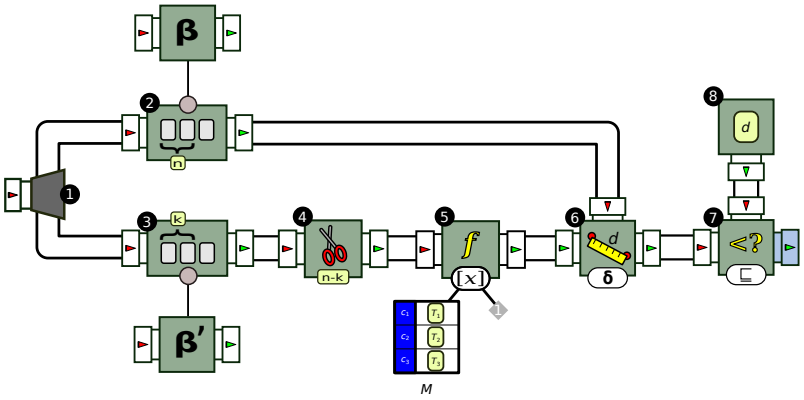


FIGURE 3.18 : Le modèle multimodal contextuel.

Le terme «multimodal» est emprunté aux statistiques. Ainsi, la Figure 3.17 peut-être vue comme un exemple de fonction de densité de probabilité pour une valeur numérique unique ayant une distribution bimodale.

3.4 MOTIF DE RÉFÉRENCE DÉPENDANT DU CONTEXTE

Dans le flux de travail précédent, la tendance calculée sur la fenêtre pouvait être comparée à plusieurs tendances de référence possibles. Le choix de la référence sur laquelle la comparaison est faite est indépendant de la fenêtre elle-même : le motif de référence le plus proche, p, est choisi et la distance à la valeur p est renvoyée. Nous appelons ce modèle de distance *sans contexte* pour cette raison.

En revanche, il existe des situations dans lesquelles la tendance actuelle ne doit pas être comparée à n'importe laquelle des références possibles, mais plutôt à une référence spécifique. En d'autres termes, le flux d'événements actuel est utilisé non seulement pour calculer une tendance sur une fenêtre, mais également pour choisir laquelle des multiples tendances de référence doit être utilisée comme référence pour cette fenêtre. C'est ce que nous appellerons un modèle *multimodal contextuel*.

Ce schéma est illustré à la Figure 3.18. Le flux d'événements en entrée est divisé en deux copies (boîte 1) ; sur le chemin du haut, une tendance est calculée par une boîte $\beta : E^n \to T$ sur une fenêtre de n événements, comme précédemment (boîte 2). Sur le chemin du bas, une deuxième tendance est calculée par une autre boîte $\beta' : E^k \to C$ sur une fenêtre de $k \leq n$ événements (boîte 3), le flux de sortie est coupé de ses $n-k$ premiers événements. Cela signifie que le i-ème événement de sortie dans le flux supérieur est la tendance calculée par β sur la fenêtre d'événements $i-n$ à i, alors que le même événement dans le flux inférieur est la tendance calculée par β' sur la fenêtre d'événements $i-k$ à i. La tendance calculée par le processeur β' est désignée par un terme spécial : nous l'appelons le *contexte*. On note $C = \{c_1, \ldots, c_\ell\}$ l'ensemble des valeurs de contexte possibles pouvant être renvoyées par β' pour une fenêtre d'événements.

Ce contexte est ensuite envoyé à une boîte (boîte 5) à qui il appartient de choisir la tendance à utiliser. La boîte se fait passer en paramètre un tableau ($M : C \to T$) qui associe chaque contexte possible c_i à une tendance de référence T_i. Étant donné une tendance $c \in C$, le processeur extrait simplement la valeur associée à c dans M. Le reste de la chaîne fonctionne comme auparavant. La distance entre la tendance de référence sélectionnée et la tendance calculée est observée (boîte 6) et comparée à un seuil prédéfini à l'aide d'une fonction de comparaison (boîtes 7 et 8).

Nous donnons ci-après quelques exemples de contextes pouvant être pris en compte lors du choix d'une référence. Comme premier exemple, considérons un flux d'événements $E \triangleq \mathbb{N} \times \mathbb{N}$ provenant d'un moniteur de trafic réseau. Une fois par minute, un événement $e = (t, b)$ est émis, contenant un horodatage (t) et la bande passante (b, en octets) consommée à la dernière minute. Définissons l'ensemble des tendances $T \triangleq \mathbb{Q}$ et utilisons comme processeur de tendance β la moyenne mobile de la composante de bande passante de chaque événement

sur une fenêtre de largeur n. Définissons le contexte comme $C \triangleq \mathbb{B}$ et le processeur $\beta' : E \to C$ tel que :

$$\beta'((t,b)) \triangleq \begin{cases} \top & \text{si } t \text{ est un jour de semaine} \\ \bot & \text{sinon} \end{cases} \tag{3.7}$$

Le processeur β' regarde simplement l'horodatage d'un événement et détermine s'il a eu lieu pendant le week-end ($\bot$) ou non ($\top$) ; ceci sera utilisé comme contexte (dans ce cas, la largeur de la fenêtre du processeur de contexte est définie comme $k = 1$). À partir de là, il est possible de créer une correspondance $M : C \to T$ entre un contexte et une tendance de référence. Supposons dans ce cas que $M(\top) = 10^6$, et $M(\bot) = 10^4$. Utilisons la distance de Manhattan de dimension 1 comme fonction de distance et $d = 10^2$ comme valeur de seuil. Un modèle de tendance contextuelle instancié de cette manière déclenchera une alarme chaque fois que la bande passante moyenne sur n événements ne se situe pas entre $10^6 \pm 10^2$ octets pendant les jours de la semaine et $10^4 \pm 10^2$ octets les week-ends.

Il convient de noter qu'il s'agit d'une situation qu'un flux de travail multimodal standard ne pourrait pas facilement capturer. En effet, on pourrait fournir un ensemble de deux tendances de référence $\{10^4, 10^6\}$; toutefois, si on utilise $\delta_{\delta'}$ comme métrique de distance, aucune alarme ne sera déclenchée avec une largeur de bande de 10^6 octets pendant les week-ends ou de 10^4 octets pendant les jours de semaine. Dans les deux cas, la largeur de bande observée est suffisamment proche de l'une des références et il n'existe aucun mécanisme permettant d'ajouter des conditions sur laquelle ces tendances devrait s'appliquer réellement. Si par contre on prend plusieurs références en fonction du contexte, on donne plus de réalisme à la comparaison, et plus de champs d'application à la méthode.

De même, au lieu de calculer une référence globale pour tous les jours de la semaine, une référence pourrait être calculée pour chaque jour indépendamment (le contexte ici serait

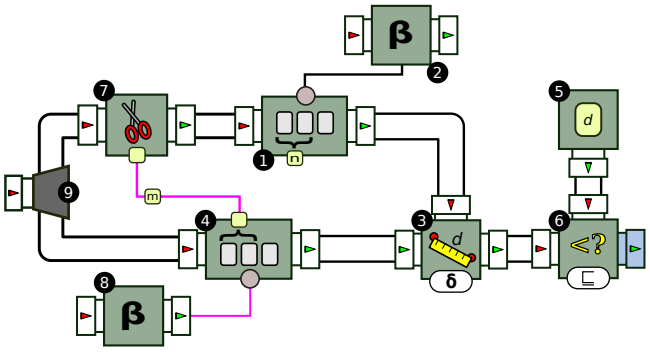


FIGURE 3.19 : Le flux de travail de distance de tendance auto corrélé.

le jour). La notion de contexte peut également être utilisée pour définir des *exceptions* à une tendance générale. Par exemple, on pourrait utiliser une tendance de référence différente pour des jours de la semaine considérés comme «spéciaux», tels que le vendredi fou ou d'autres jours fériés.

3.5 DISTANCE DE TENDANCE AUTO CORRÉLÉE

Comme son nom l'indique, le modèle de distance de tendance statique s'attend à ce qu'un motif de référence fixe $P \in \mathbb{P}$ soit donné à l'avance. Jusqu'ici, nous avons laissé de côté la question de savoir comment obtenir ce motif de référence. Dans cette section, nous présentons une première méthode pour calculer un tel motif de référence, en comparant la tendance calculée sur la fenêtre en cours à une autre tendance calculée sur la même séquence d'événements, mais plus éloignée dans le passé. Dans un tel contexte, une alarme est déclenchée lorsque le flux présente une tendance trop différente de celle observée précédemment. Comme il n'y a pas de motif de référence fixe et que les écarts de tendance observés sur un flux se rapportent au flux lui-même, nous appelons cette technique *distance de tendance auto corrélée*.

La Figure 3.19 illustre le flux de travail de distance de tendance auto corrélé. Les boîtes 1–3 et 5–6 ressemblent au schéma de distance de tendance statique. La différence réside en amont dans la façon dont les tendances «actuelle» et «de référence» sont extraites de la séquence d'événements en entrée. Cette séquence est tout d'abord divisée en deux copies (boîte 9). La copie la plus haute est coupée de ses premiers m événements, comme indiqué par la boîte 7. Ceci fait en sorte que les flux entrants dans les boîtes 1 et 4 sont décalés de m : tandis que la boîte 4 reçoit le flux d'événements $e_0, e_1, \ldots$, la boîte 1 reçoit le flux $e_m, e_{m+1}, \ldots$ Ces deux boîtes appliquent alors le même calcul β sur une fenêtre glissante : la boîte 1 sur une fenêtre de largeur n, et la boîte 4 sur une fenêtre de largeur m. La sortie de β sur ces deux fenêtres est ensuite envoyée à la métrique de distance (boîte 3), et le reste du processus se déroule de la même manière que le flux de travail de distance de tendance statique que nous avons introduit précédemment.

Comme auparavant, la métrique de distance (boîte 3) est alimentée par une séquence de paires de tendances (p,t), où $t \in T$ est la tendance calculée sur la dernière fenêtre d'événements de largeur n. Toutefois, la tendance de référence $p \in \mathbb{P}$ est désormais également calculée à partir d'une fenêtre d'événements du même flux. Soit k le nombre d'événements reçus du flux d'entrée jusqu'à présent, avec $k \geq m+n$. En raison de la présence de la boîte de découpage (7), on peut observer que, lorsque la boîte 1 applique β sur une fenêtre des n derniers événements ($\{e_{k-(n-1)}, e_{k-(n-2)}, \ldots, e_k\}$), la boîte 4 applique β sur une fenêtre des m événements précédents ($\{e_{kn-(m-1)}, e_{kn-(m-2)}, \ldots, e_{kn}\}$). En d'autres termes, la mesure de distance compare la tendance calculée à partir des n derniers événements à une référence calculée à partir des m événements qui les précèdent.

Le reste du flux fonctionne de manière similaire au flux de travail de distance de tendance statique. Le flux de travail auto corrélé peut être instancié de différentes manières, en fonction de la manière dont on donne des valeurs à sept paramètres :

— E est l'ensemble des événements d'entrée et est défini comme auparavant

— $n \in \mathbb{N}$ est la largeur de la fenêtre utilisée pour calculer la tendance «présent»

— $m \in \mathbb{N}$ est la largeur de la fenêtre utilisée pour calculer la tendance «passé»

— $\beta_m : E^m \to \mathbb{P}$ et $\beta_n : E^n \to T$ sont les deux calculs de tendance à appliquer sur les fenêtres «passé» et «présent» , respectivement.

— $\delta : \mathbb{P} \times T \to D$ est la métrique de distance, définie comme auparavant

— $d \in D$ est le seuil de distance

— $\sqsubseteq : D^2 \to \mathbb{B}$ est la fonction de comparaison de distance

Dans de nombreux cas, il est intéressant d'appliquer le même calcul à la fois aux fenêtres passées et présentes. Dans un tel cas, $m = n$, $\beta_m = \beta_n$ et $\mathbb{P} = T$. De plus, la plupart des exemples présentés dans la section précédente peuvent être convertis en flux de travail auto corrélés de manière simple. Par exemple, pour l'exemple 1 on peut faire en sorte de déclencher une alarme lorsque la moyenne des n dernières valeurs numériques présente une différence supérieure à d avec la moyenne des n valeurs qui les précèdent.

Dans le cas de l'exemple 2, utilisons la distance de Chebychev $\hat{\delta}_\infty$ comme métrique de distance et un seuil de distance de $d \in \mathbb{R}$. Le flux de travail de distance de tendance auto corrélée détectera chaque fois que l'un des m moments statistiques calculés sur les n derniers événements diffère de plus de d du même moment statistique calculé sur les n événements précédents. En utilisant une autre mesure de distance et une autre fonction de comparaison, il est également possible de spécifier un seuil distinct pour chaque instant. Soit $D \triangleq \mathbb{R}^m$ et définissons $\delta : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ tel que $\delta((x_1,\ldots,x_m),(y_1,\ldots,y_m)) \triangleq (z_1,\ldots,z_m)$ si et seulement si $z_i = |x_i - y_i|$ pour chaque $i \in [1,m]$. Cette métrique calcule la distance de Manhattan de chaque paire de composantes vectorielles. Le seuil de distance $d \in \mathbb{R}^m$ peut maintenant définir une valeur pour chaque composant. Enfin, la fonction de comparaison

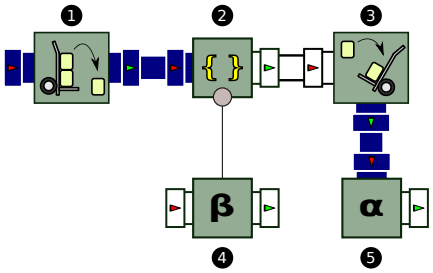


FIGURE 3.20 : Le flux de travail d'extraction des tendances.

$\sqsubseteq: \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{B}$ peut être définie de telle sorte que $\sqsubseteq((x_1,\ldots,x_m),(y_1,\ldots,y_m)) = \top$ si et seulement si $y_i > x_i$ pour un $i \in [1,m]$.

L'exemple 3, qui calcule une distribution de fréquence de symboles, peut être transformé en un flux de travail auto corrélé d'une manière similaire. En fait, le seul exemple qui ne peut pas être facilement transformé en autocorrélation est l'exemple 4, qui implique un motif de référence multimodal. En effet, puisque β_n calcule une seule tendance dans la fenêtre *présente*, si nous posons $\beta_m = \beta_n$, une seule tendance sera également calculée dans la fenêtre *passée*. Ceci est conforme à l'hypothèse (raisonnable) selon laquelle un log d'événements suit une distribution ou une autre, mais pas plusieurs distributions à la fois.

3.6 EXTRACTION DE TENDANCE

Le cas des modèles de référence multimodaux est beaucoup plus susceptible de se produire lors de la compilation des tendances calculées à partir de plusieurs logs. Dans cette section, nous introduisons un deuxième mécanisme permettant d'obtenir une tendance de référence, cette fois en calculant les tendances individuelles à partir d'un ensemble de séquences d'événements préalablement obtenues. Cela donne lieu à un autre flux de travail, appelé *flux de travail d'extraction de tendance*.

Ce flux de travail est illustré à la Figure 3.20. L'entrée dans ce flux de travail est un *ensemble* fini de logs d'événements préenregistrés. Cet ensemble est *décomposé* par la boîte 1. Le processeur responsable de cette action est `Unpack` ; il prend en entrée une trace de listes et retourne les événements individuels de chaque liste un par un. Chaque journal est ensuite envoyé à la boîte 2, qui le rejoue à la fonction de tendance β (boîte 4). Chaque événement généré par la boîte 2 est le résultat de l'application de β sur un journal d'événements complet ; ceci correspond à la tendance globale extraite de ce journal. Les tendances de chaque journal sont regroupées dans un ensemble qui est ensuite envoyé à la boîte 5. Cette boîte applique une fonction α à l'ensemble des tendances. Intuitivement, α peut être considérée comme une fonction d'agrégation permettant de calculer une tendance globale à partir des tendances individuelles obtenues de chaque journal. Une fois calculée, cette tendance globale peut ensuite être utilisée comme motif de référence P dans le flux de travail de distance de tendance statique original.

Comme tous les flux de travail vus jusqu'à présent, le flux de travail d'extraction de tendance peut être instancié de différentes manières, cette fois en définissant trois paramètres :

— E est l'ensemble des événements dans les journaux d'entrée ; notons E^* une trace d'éléments de E

— $\beta : E^* \to T$ est la fonction d'extraction de tendance, qui prend un journal dans E^* et calcule une tendance $t \in T$.

— $\alpha : 2^T \to T'$ est la fonction d'agrégation de tendance ; elle prend en entrée plusieurs tendances de T et calcule une tendance globale $t' \in T'$

Le fonctionnement du flux de travail d'extraction de tendance peut être vu comme une étape préliminaire sur un ensemble de journaux pré collectés, pour permettre ensuite de détecter des déviations sur un nouveau flux d'événements. Nous donnerons ci-après quelques

exemples de fonctions de tendance et d'agrégation pouvant être utilisées, en fonction du contexte.

3.6.1 MOYENNE

Nous commençons par un exemple simple où $E \triangleq \mathbb{R}$ et les logs d'événements sont constitués de valeurs numériques scalaires. Nous définissons $\beta : \mathbb{R}^* \to \mathbb{R}^2$ tel que, pour $\langle e_0, e_1, \ldots, e_n \rangle \in \mathbb{R}^*$, $\beta(\langle e_0, e_1, \ldots, e_n \rangle) = (x, y)$ si et seulement si $x = \sum_{i=0}^{n} \frac{e_i}{n}$ et $y = n$. En d'autres termes, pour chaque log, β calcule la moyenne des valeurs dans le log ainsi que sa longueur. On peut ensuite définir une fonction d'agrégation $\alpha : 2^{\mathbb{R}^2} \to \mathbb{R}$, telle que :

$$\alpha(\{(x_1, y_1), \ldots, (x_m, y_m)\}) \triangleq \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} y_i} \tag{3.8}$$

La fonction α calcule ainsi la moyenne des valeurs moyennes dans chaque journal, pondérée par la longueur de chaque journal.

3.6.2 CLUSTERING

Un exemple plus intéressant concerne les modèles de référence multimodaux. Soit $E \triangleq \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_q\}$, un ensemble de q symboles discrets, comme dans l'exemple 1 de la section 3.2.1. Définissons $\beta : E^* \to [0,1]^q$ comme une fonction qui, pour chaque journal, calcule le vecteur de la fréquence relative de chaque symbole du journal. Ceci est un exemple où différents journaux peuvent présenter différentes distributions de symboles. Dans ce cas, un *algorithme de clustering* peut être utilisé pour déterminer quels sont les vecteurs les plus représentatifs des diverses distributions.

De manière plus formelle, on rappelle qu'un algorithme de clustering prend en entrée un ensemble *I* de vecteurs de dimension *m* et renvoie en sortie un autre ensemble *O* de vecteurs de dimension *m*. Cet ensemble est tel que la distance entre chaque vecteur de *I* et le vecteur le plus proche dans *O* est minimisée, selon une métrique de distance δ. Si les vecteurs dans *I* se trouvent dans un petit nombre de «clusters» relativement disjoints, un algorithme de cluster tente de trouver ce que l'on appelle les *centroïdes*.

Ceci est illustré dans la Figure 3.17, où chacun des points du graphique représente la fréquence relative de *a* et de *b* dans un journal. Comme on peut le constater, il existe de nombreux journaux dans lesquels la fréquence relative de *a* et de *b* est proche de 30 % / 70 % (points bleus), et de nombreux journaux dans lesquels la fréquence relative de *a* et *b* est proche de 70 % / 30 % (points verts). Un algorithme de clustering, étant donné cet ensemble de points, pourrait trouver deux clusters et évaluer leurs centroïdes comme les vecteurs $(3/10, 7/10)$ (symbole "+" bleu) et $(7/10, 3/10)$ (symbole "+" vert).

Ceci complète l'exemple 4, en montrant comment un motif de référence multimodal peut être obtenu à partir d'un ensemble de journaux préexistants. On pourrait utiliser comme fonction $\alpha : 2^{\mathbb{R}^m} \rightarrow 2^{\mathbb{R}^m}$ l'un des algorithmes de classification mentionnés au chapitre 2. La sortie du flux de travail d'extraction de modèle devient alors un ensemble de vecteurs de référence de dimension *m*, qui peuvent être utilisés comme base pour instancier un flux de travail de distance de tendance statique.

À noter que ce flux de travail est encore une fois très générique, il ne dépend que des définitions données à E, β et α. Dans le cas où α est un algorithme de classification, les vecteurs réels extraits des journaux d'entrée peuvent également être arbitraires. Les moments statistiques et les distributions de fréquence ne sont que deux exemples de la vaste gamme de caractéristiques numériques que l'on peut calculer sur une séquence d'événements.

Bien que cet exemple utilise le clustering, il diffère des travaux précédents sur le clustering de flux de données, tels ceux de Guha *et al.* [170]. Le problème ici est de créer des grappes à partir de fonctionnalités extraites de plusieurs journaux, alors que les travaux précédents étaient axés sur la création de grappes à partir des points de données provenant d'un seul flux.

3.7 TENDANCE DU DEUXIÈME ORDRE

Un problème récurrent lors de l'utilisation de mesures basées sur les tendances est le nombre potentiellement élevé de *faux positifs* qu'ils peuvent générer [227]. Dans ce contexte, un faux positif est interprété comme un écart de tendance dépassant le seuil prédéfini, sans pour autant être considéré comme représentatif d'une situation qui mérite d'être examinée de près. C'est particulièrement le cas lorsqu'une seule tendance est calculée sur le flux d'entrée, alors que la présence d'un comportement anormal, dans le contexte, ne peut pas être déterminée en analysant cette tendance unique.

À cet égard, reprenons l'exemple d'une requête de tendance du scénario 5 présenté au chapitre 1. On rappelle que détecter un écart de 5% dans l'ensemble des ports peut ne pas être utile pour distinguer le trafic normal et anormal. Il peut plutôt être utile de surveiller le nombre de ports ouverts ou le volume de données par port. Si seul l'ensemble des ports connectés était considéré, la plupart des professionnels de la sécurité seraient concernés par un seul port dans une plage spécifique (par exemple, ssh/22) qui serait ouvert, alors que ce n'est pas le cas. En d'autres termes, un seul écart de tendance peut ne pas être considéré comme suspect, mais il mérite toute notre attention s'il est combiné à des alarmes déclenchées par d'autres calculs de tendance sur les mêmes données d'entrée.

À strictement parler, notre modèle de distance de tendance original peut être adapté à un tel cas. On peut imaginer un processeur β composé de deux sous-processeurs, dont la

tendance est ensuite fusionnée dans une structure de données composée sous forme d'un couple. De même, une fonction de distance spéciale opérant sur les deux composantes du couple peut également être définie, de sorte qu'une alerte soit déclenchée en fonction d'une condition faisant référence aux deux éléments du tableau. Bien que possible, cette solution est au mieux maladroite : les tendances doivent être calculées sur la même fenêtre, et une nouvelle fonction de distance doit être conçue chaque fois qu'un ensemble différent de processeurs β est considéré.

Une façon plus simple de gérer ce scénario consiste à considérer le processus de détection d'écart comme une opération en plusieurs étapes. Dans ce contexte, un nombre arbitraire de tendances est calculé sur le flux d'entrée d'origine. Chacune d'entre elles utilise sa propre taille de fenêtre, sa métrique de distance et son seuil, et suit le modèle de distance de tendance normal, tel que décrit précédemment. L'important est de voir la sortie de ces processeurs comme un nouveau flux, qui peut lui-même être utilisé comme source d'un nouveau modèle de distance de tendance.

La Figure 3.21 illustre le processus résultant. Le flux d'entrée d'origine est d'abord divisé en plusieurs copies, une pour chacun des processeurs de distance de tendance. Chacun de ces processeurs peut utiliser une boîte de tendance distincte β_1, …, β_n et générer un flux de booléens ; les valeurs aux positions correspondantes dans chacun des flux de sortie sont fusionnées dans un tableau. Le résultat est un flux de tableaux de booléens, que nous appellerons la tendance du «premier ordre». Intuitivement, une valeur de $\top$ au i-ème composant d'un tableau indique un écart de la i-ème tendance de i dépassant le seuil qui lui a été spécifié.

La tendance «de second ordre» est simplement une tendance calculée sur ce flux de tableaux. Comme tout autre modèle de distance de tendance, la seconde tendance est calculée sur sa propre fenêtre, qui peut être différente de la largeur de la fenêtre utilisée dans les

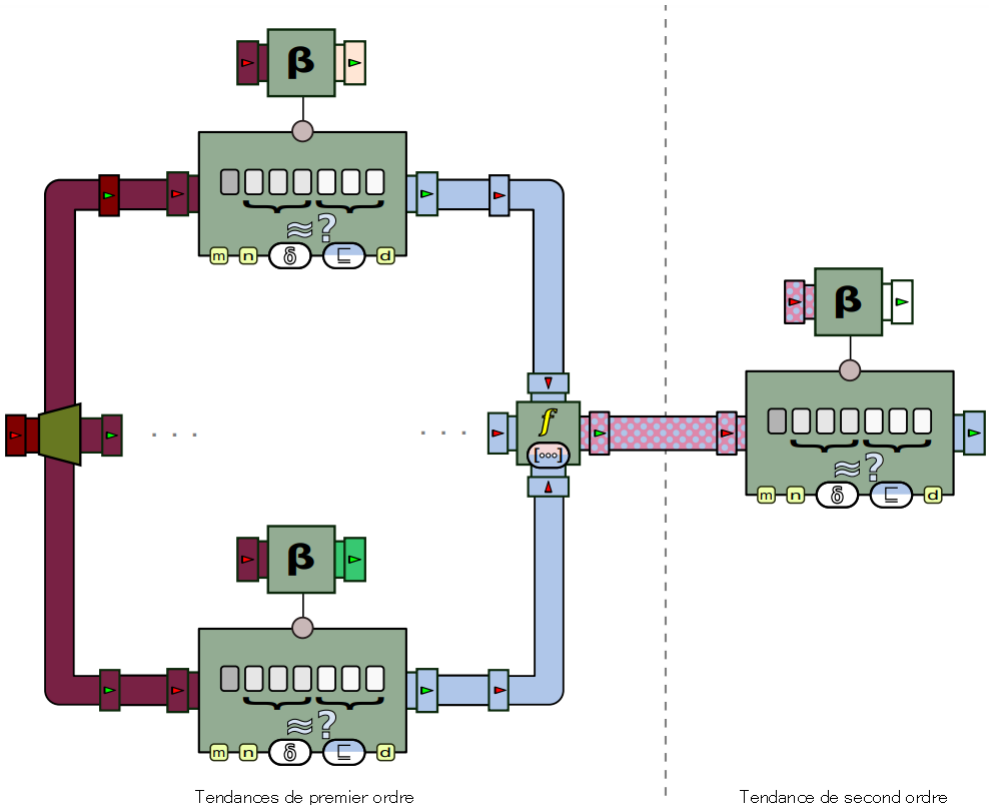


FIGURE 3.21 : Le modèle de distance de tendance de second ordre.

tendances du premier ordre. Ce niveau de second ordre est d'une nature particulière à un seul égard, car son processeur de tendance β fonctionne toujours sur des tableaux booléens. Bien que cette contrainte puisse paraître limitante, nous verrons, à travers quelques exemples, la souplesse dans l'expression de divers types de tendances de niveau supérieur.

3.7.1 DÉVIATION SOUTENUE

Comme premier exemple, nous nous concentrons sur un modèle de second ordre basé sur une tendance unique de premier ordre. Nous considérons un flux de valeurs numériques (c'est-à-dire $E \triangleq \mathbb{R}$), et prenons $\beta : R^m \to \mathbb{R}$ comme un processeur de tendance tel que :

$$\beta(x_1, \ldots, x_m) \triangleq \frac{1}{m} \sum_{i=1}^{m} x_i. \tag{3.9}$$

En d'autres termes, β calcule la moyenne mobile sur une fenêtre de largeur m. En utilisant la distance de Manhattan pour δ, le processeur de premier ordre génère donc $\top$ lorsque la moyenne sur une fenêtre de largeur m s'écarte trop de la moyenne de référence d.

Cependant, il existe des situations dans lesquelles un écart unique par rapport à la moyenne peut être considéré comme acceptable. Ce qui ne l'est pas, en revanche, est un écart prolongé par rapport à une telle moyenne ; une chose plus difficile à exprimer avec une tendance du premier ordre. Supposons qu'un écart prolongé soit défini comme étant plus de d' des m' dernières fenêtres dépassant le seuil moyen, pour certaines constantes d' et m'. Nous pouvons donc définir un modèle de distance de tendance de second ordre avec une fenêtre de largeur 1, où $E' = \mathbb{B}$. Le processeur $\beta' : \mathbb{B}^m \to \mathbb{N}$ est le processeur de tendance défini comme :

$$\beta'(\langle b_1, \ldots, b_{m'} \rangle) \triangleq \sum_{i=1}^{m'} \xi(b_i), \tag{3.10}$$

où $\xi : \mathbb{B} \to \{0, 1\}$ est défini comme :

$$\xi(b) \triangleq \begin{cases} 1 & \text{si } b = \top, \\ 0 & \text{sinon.} \end{cases} \tag{3.11}$$

Le processeur β compte donc combien des derniers événements d'entrée m' sont la valeur $\top$. Pour exprimer l'écart de tendance souhaité, il suffit d'utiliser $\leq$ comme fonction de comparaison et d' comme valeur seuil. Le résultat final du modèle de distance de tendance de second ordre est donc un flux de valeurs booléennes où la présence de $\top$ indique que d' des m' dernières fenêtres de largeur m avaient une moyenne ayant dépassé la moyenne de référence d. À noter que ces fenêtres ne doivent pas obligatoirement être *successives* pour que la déviation soit identifiée.

3.7.2 DÉVIATION MULTI FACTEURS

L'exemple précédent montrait comment le modèle de second ordre peut être utilisé pour «amortir» les résultats d'une tendance de premier ordre en exigeant qu'un écart soit présent un nombre minimum de fois dans un intervalle donné. Une deuxième utilisation possible du modèle consiste à atténuer le signalement des écarts en mettant en corrélation plusieurs modèles de premier ordre, et en exigeant qu'un certain nombre k renvoie $\top$ dans une fenêtre donnée.

Dans un tel cas, supposons qu'un nombre k de modèles de distance de tendance statique génère chacun un flux booléen. L'ensemble E d'événements d'entrée pour le modèle de second ordre devient donc $\mathbb{B}^k$. Pour une largeur de fenêtre donnée m', un processeur de tendance $\beta' : (\mathbb{B}^k)^{m'} \to \mathbb{N}$ peut être défini comme suit :

$$\beta'(\langle b_{1,1}, \ldots, b_{1,k}\rangle, \ldots, \langle b_{m',1}, \ldots, b_{m',k}\rangle) \triangleq \sum_{i=1}^{m'} \left(\xi \left(\bigvee_{j=1}^{k} b_{i,j} \right) \right). \tag{3.12}$$

Où ξ est défini comme dans l'exemple précédent. Ce processeur de tendances renvoie le nombre de tendances de premier ordre *distinctes* ayant affiché un écart au moins une fois au cours des m' fenêtres précédentes. Comme ci-dessus, une valeur de seuil $d' \in \{1, \ldots, k\}$ peut être utilisée pour imposer un nombre minimal de tendances de premier ordre.

De la même manière que les fenêtres incriminées n'ont pas besoin d'être successives dans le premier exemple, il n'est pas nécessaire que les tendances de premier ordre signalent un écart *simultanément*. Le modèle est également ouvert aux variantes de la définition du processeur β', telles que le calcul d'une somme pondérée attribuant une importance différente à la sortie de chaque modèle de distance de tendance de premier ordre.

3.8 CONCLUSION

Dans ce chapitre, nous avons présenté différents modèles permettant de détecter des tendances sur des flux avec plusieurs exemples d'application pour différentes situations. Dans un premier temps, nous avons vu la distance de tendance statique. Ensuite, on a défini ce qu'est un motif de référence multimodal, motif de référence dépendant du contexte et la distance de tendance auto corrélée. On a également montré comment utiliser les techniques d'extraction de tendance lorsqu'on traite des flux d'événements avec des modèles de référence multimodaux. Finalement, nous avons présenté la notion de tendance du deuxième ordre qui permet d'extraire la tendance du flux à différents niveaux afin d'aiguiser les résultats et qu'ils soient plus proche de la réalité du flux.

CHAPITRE IV
ANALYSE PRÉDICTIVE POUR LE TRAITEMENT DE FLUX D'ÉVÉNEMENTS

De nombreux outils et techniques permettent de calculer diverses fonctions prédictives sur les données. Cependant, la plupart d'entre eux ne fonctionnent pas en mode continu comme nous l'avons justifié précédemment, c'est-à-dire lorsque le système reçoit les événements d'entrée un par un, et une sortie ou une mise à jour devrait être produite en même temps. Par exemple, la méthodologie décrite dans [151] calcule un vecteur de caractéristiques basé sur l'exécution complète d'un processus, ce qui ne peut être fait que hors ligne. En outre, bon nombre des approches existantes sont également ad hoc : elles s'appliquent chacune à un type spécifique de fonction prédictive. Par exemple, effectuer une régression linéaire sur une fenêtre de valeurs numériques nécessite une configuration différente (et éventuellement un logiciel différent) de la construction d'un modèle de Markov qui devine l'événement suivant en fonction du précédent. Dans ce chapitre, nous passons à un second problème qui est de faire des prédictions sur des flux d'événements. Nous présentons un cadre générique permettant de calculer des prévisions sur un journal d'événements «en continu».

4.1 ANALYSE PRÉDICTIVE STATIQUE

Cette première section sera consacrée à la présentation des frameworks de traitement de flux d'événements. L'analyse prédictive présentée repose sur des éléments statiques, entre autres on utilise une fonction de prédiction prédéfinie. Pour mieux comprendre le cadre générique présenté, la section débutera par une mise en contexte.

Dans la suite, on suppose qu'un journal peut contenir des séquences d'événements entrelacées pour plusieurs instances de processus. Dans un tel cas, nous supposons que chaque événement contient un élément de données qui l'associe à l'instance correspondante

(généralement un identificateur de processus unique d'une forme quelconque). La sous-séquence d'événements appartenant à la même instance de processus s'appelle une tranche (slice) ; appliquer un traitement séparé à chacune de ces sous-séquences sera appelé découpage en tranches (*slicing*).

4.1.1 MODÈLE DE PRÉDICTION STATIQUE

La première (et la plus simple) forme de modèle d'analyse prédictive que nous introduisons s'appelle la *prédiction statique*. Dans ce modèle, une fenêtre glissante d'événements de largeur fixe est utilisée pour produire une prévision des événements à venir à partir du journal. Ceci est représenté graphiquement à la Figure 4.1.

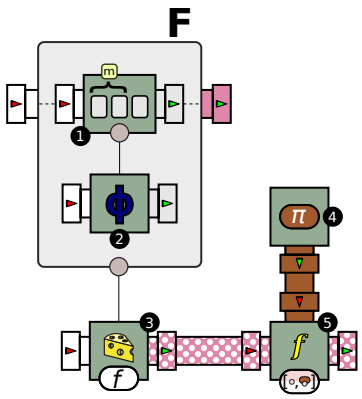


FIGURE 4.1 : Le modèle de prédiction statique.

Dans ce diagramme, une trace des événements entrants est d'abord envoyée dans une zone de découpage, représentée par la boîte 3. Cette boîte est paramétrée par une fonction f qui, à partir d'un événement, détermine la tranche à laquelle elle appartient. Par exemple, si chaque tranche est identifiée par un ID de processus, f peut être définie comme la fonction qui extrait cet ID de chaque événement. Cela a pour effet de séparer le flux d'événements entrant en plusieurs sous-flux pour autant qu'il existe de valeurs de retour possibles pour f.

La boîte de découpage reçoit également comme paramètre un workflow de transmission en continu, représenté par la boîte nommée F. Une instance de F est créée pour chaque sous-flux et est destinée à traiter le sous-flux correspondant à la tranche à laquelle elle est associée. Lorsqu'un événement entrant e est évalué par f, il est alors poussé vers l'instance de F associée à $f(e)$.

Bien que le flux de travail de flux F puisse être une chaîne arbitraire de processeurs, dans le cas présent, il est défini de manière précise. Premièrement, les événements entrants sont envoyés dans un processeur fenêtre (boîte 1) chargé de créer une fenêtre glissante des événements d'une largeur donnée m. On rappelle que si $e_1, e_2, \ldots$ est le flux d'événements, le processeur fenêtre produit d'abord une fenêtre constituée de $e_1, \ldots, e_m$, puis une autre fenêtre $e_2, \ldots, e_{m+1}$, et ainsi de suite. À son tour, chaque fenêtre est envoyée à un processeur de caractéristiques arbitraire (boîte 2). Une fonction ϕ est évaluée sur chaque fenêtre et la valeur de retour de cette fonction est ensuite envoyée à la sortie. Donc, si $e_1, e_2, \ldots$ est la séquence d'événements donnée à une seule instance de la boîte F, sa sortie sera le flux d'événements $\phi(e_1, \ldots, e_m), \phi(e_2, \ldots, e_{m+1}), \ldots$, etc.

La dernière étape de traitement de la boîte de découpage consiste à collecter les valeurs produites par chaque instance de F dans une table associative. Les clés de la table sont les identificateurs de tranche (c'est-à-dire chaque valeur possible de $f(e)$) et la valeur associée à chaque clé est le dernier événement produit par le flux de travail de flux correspondant F. La sortie de la boîte 3 est donc un flux de tables en évolution constante associant des identifiants de tranches à des valeurs arbitraires.

En résumé, le processeur de découpage divise le flux d'événements entrant en tranches, envoie chaque tranche à travers une instance distincte de F et rassemble la dernière valeur générée par chaque instance de F dans une table associative. Comme nous l'avons vu, les

valeurs de cette table résultent de l'évaluation d'une fonction ϕ sur la dernière fenêtre de m événements sur chaque tranche.

Ce flux de tables est ensuite envoyé à un processeur de *fonction*, représenté par la boîte 5. Comme son nom l'indique, cette étape a pour objectif d'effectuer une transformation sur chaque table entrante. Dans le cas présent, cela revient à appliquer une fonction π (fournie par la boîte 4) à chaque valeur de la table, en maintenant les clés intactes. Dans un tel contexte, π s'appelle une fonction prédictive. Intuitivement, les tables associatives produites à la sortie de la boîte 5 sont des «prédictions» constamment mises à jour et calculées séparément pour chaque tranche du journal d'origine.

Comme on peut le constater, ce workflow générique laisse plusieurs paramètres indéfinis. Formellement, soit Σ l'ensemble des événements du journal d'origine. Un modèle de prédiction statique concret est obtenu en fournissant la définition des quatre paramètres suivants :

— Une fonction $f : \Sigma \to S$, appelée fonction de découpage, qui associe chaque événement entrant à un identifiant de tranche $s \in S$

— Une largeur de fenêtre $m \in \mathbb{N}^+$

— Une fonction d'extraction de caractéristiques $\phi : \Sigma^m \to V$, qui prend une fenêtre de m événements successifs et calcule une valeur de caractéristique $v \in V$

— Une fonction prédictive $\pi : V \to P$, qui associe une valeur de caractéristique $v \in V$ à une prédiction $p \in P$.

4.1.2 EXEMPLES

Selon la manière dont ces quatre paramètres sont définis, le flux de travail générique peut représenter différents types de calculs prédictifs. Nous donnons ci-dessous quelques exemples.

PRÉVISION DE LA MOYENNE

Comme premier exemple, supposons que l'ensemble d'événements soit défini comme $\Sigma \triangleq \mathbb{Q}$, chaque événement étant un nombre rationnel (par exemple, le prix d'un stock ou le résultat d'un test effectué sur un patient). Dans cet exemple, le journal est constitué d'une seule tranche. Donc, on peut définir la fonction de découpage $f : \mathbb{Q} \to \{0\}$ en tant que fonction constante qui mappe tout événement à la valeur par défaut 0.

Soit m une largeur de fenêtre arbitraire et définissons la fonction d'extraction de caractéristiques $\phi : \mathbb{Q}^m \to \mathbb{Q}$ comme $\phi(q_1, \ldots q_m) \triangleq \frac{1}{m}\sum_1^m q_i$. Autrement dit, ϕ calcule la moyenne des m valeurs dans la fenêtre. Soit $\pi : \mathbb{Q} \to \mathbb{Q}$ la fonction identité $\pi(q) \triangleq q$.

L'instanciation du workflow de prédiction statique de cette manière donne un flux de sortie de tables associatives de la forme $\{0 \mapsto q_1\}, \{0 \mapsto q_2\}, \ldots$; la i-ième valeur de sortie, q_i, représente la moyenne des valeurs numériques des événements $i-11$ à $i-1$. Cela signifie que la prédiction pour la valeur suivante est simplement la moyenne des 10 valeurs qui la précèdent.

PRÉVISION MOYENNE AVEC PLUSIEURS INSTANCES

Le même exemple peut être complexifié en supposant que le journal d'entrée contienne plusieurs valeurs entrelacées appartenant à différentes instances de processus. À cette fin, supposons que $\Sigma \triangleq \mathbb{N} \times \mathbb{Q}$. Chaque événement (n, q) est constitué d'un identificateur de processus arbitraire $n \in \mathbb{N}$ et d'une valeur numérique $q \in \mathbb{Q}$. Chaque sous-flux peut être récupéré en définissant la fonction de découpage $f : \mathbb{N} \times \mathbb{Q} \to \mathbb{N}$ comme $f(n, q) \triangleq n$. Cela aura pour effet de regrouper les événements ayant le même identifiant dans le même sous-flux. Définissons ensuite la fonction caractéristique $\phi : (\mathbb{N} \times \mathbb{Q})^m \to \mathbb{Q}$ comme $\phi((n_1, q_1), \ldots (n_m, q_m)) \triangleq \frac{1}{m}\sum_1^m q_i$.

La fonction effectue le même calcul qu'auparavant, sauf qu'elle doit maintenant extraire la valeur numérique q de chaque tuple (n,q). La fonction prédictive π est alors définie comme précédemment. La sortie du workflow est à nouveau un flux de tables associatives, cette fois de la forme $\{n_1 \mapsto q_1, \ldots, n_k \mapsto q_k\}$. Pour chaque tranche n_i, la prédiction associée est la moyenne des m derniers événements de cette tranche.

Le lecteur devrait remarquer comment l'utilisation du découpage dans le motif produit une prédiction spécifique à chaque instance de processus. En effet, dans les cas d'utilisation les plus réalistes, le calcul d'une prédiction basée sur une fenêtre d'événements dans laquelle plusieurs instances de processus sont mélangées aurait peu de sens. Prenons le cas simple où chaque instance de processus représente des mesures numériques obtenues d'un seul client ou patient : dans un tel contexte, chaque instance est susceptible de contenir des valeurs numériques qui suivent leur propre distribution (moyenne et écart type), et nécessitent donc leur propre prévision.

RÉGRESSION LINÉAIRE

La prévision de la moyenne d'une fenêtre de valeurs numériques est sans doute un moyen grossier de faire une prévision. Si l'on ne s'attend pas à ce que ces valeurs oscillent autour d'une référence inconnue mais fixe, on peut obtenir une meilleure prédiction en utilisant des techniques simples de régression statistique [228].

Considérons un flux d'événements ayant la même structure que le dernier exemple, avec $\Sigma \triangleq \mathbb{N} \times \mathbb{Q}$, et la fonction de découpage f tel que $f((n,q)) = n$. Soit $\mathbb{F}$ l'ensemble de toutes les fonctions linéaires à une variable avec des coefficients rationnels en une variable (c'est-à-dire de la forme $f(x) = ax + b$). Définissons $\phi : (\mathbb{N} \times \mathbb{Q})^m \to \mathbb{F}$, la fonction qui, à partir d'un ensemble d'événements $(n,q_1),(n,q_2),\ldots,(n,q_m)$, renvoie l'équation de la droite de

régression qui correspond « le mieux » aux points $(0,q_1),(1,q_2),\ldots,(m-1,q_m)$ (par exemple, la droite des moindres carrés).

Par exemple, pour $m=3$, étant donné les événements d'entrée $(n,30)$, $(n,31)$, $(n,35)$ (pour un certain identifiant de tranche arbitraire n), la fonction ϕ renverrait la ligne de régression obtenue à partir des points $(0,30)$, $(1,31)$, $(2,35)$. En utilisant une technique classique telle que celle des moindres carrés, on pourrait obtenir une équation telle que $2,5x+29,5$.

Comme on peut le constater, le résultat de la boîte de découpage (boîte 3 de la Figure 4.1) est cette fois une table dont les valeurs sont des fonctions linéaires en une variable. Chacune de ces fonctions peut ensuite être utilisée pour faire une prévision sur la valeur suivante, en l'évaluant simplement à $f(m)$. À cette fin, on peut définir la fonction prédictive $\pi : \mathbb{F} \to \mathbb{Q}$, tel que $\pi(g) = g(m)$. Autrement dit, la fonction π prend en entrée une fonction linéaire $g(x) = ax+b$ et renvoie la valeur de $g(m)$, c'est-à-dire : $am+b$.

Ceci est illustré à la Figure 4.2. Le diagramme du haut illustre les événements de deux instances de processus («cercle» et «losange») entrelacés dans le même log. Le processeur de tranches recompose chaque sous-flux séparément, ce qui conduit aux diagrammes du bas. Une fenêtre de $m=3$ événements sur chaque tranche est utilisée pour calculer une ligne de régression, qui est ensuite étendue pour fournir une prévision pour la valeur suivante. Étant donné que, dans cet exemple, deux tranches différentes ont accumulé au moins trois événements, une prévision distincte peut être établie pour la valeur suivante des deux. Le cercle en pointillé et le losange représentent le prochain événement prévu dans le diagramme du haut, en fonction de la tranche à laquelle il appartiendra.

Le lecteur remarquera comment, en fournissant une définition différente de ϕ et de π, le même flux de travail de base peut calculer un type de prévision totalement différent. Notons également que ce schéma de régression de base suppose que les valeurs de chaque tranche

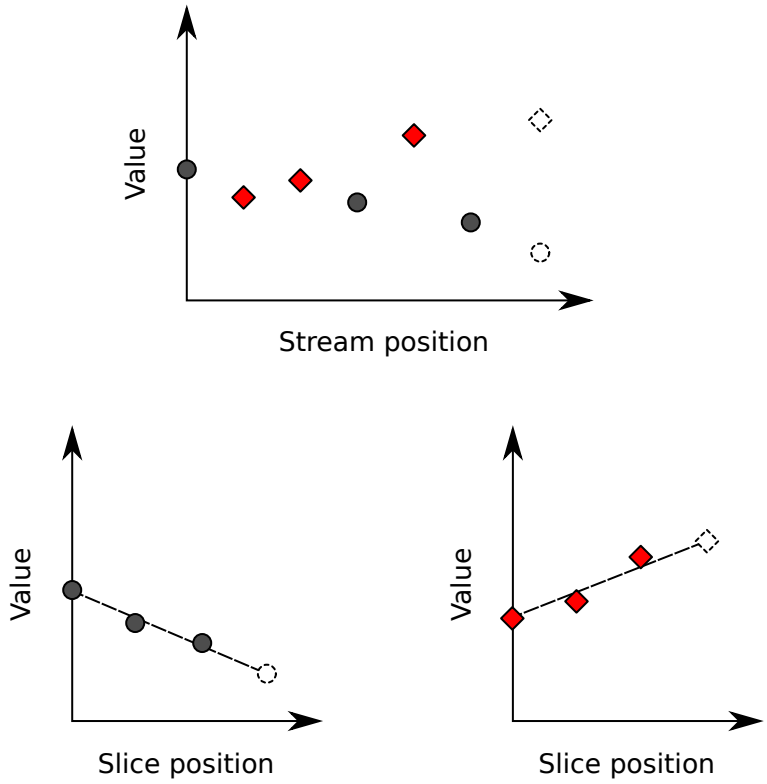


FIGURE 4.2 : Utilisation de la régression linéaire pour prédire la valeur suivante dans un flux de nombres.

sont équidistantes sur l'axe des x. Dans le cas où les événements contiennent un horodatage (tel qu'une heure universelle), ϕ pourrait être affinée de sorte que chaque événement soit placé horizontalement à sa position relative en fonction de sa valeur d'horodatage, conduisant ainsi à une prévision plus réaliste basée sur le temps écoulé.

4.2 APPRENTISSAGE D'UNE FONCTION PRÉDICTIVE

Jusqu'ici, les exemples que nous avons montrés supposent que les fonctions prédictives π sont statiques et connues à l'avance. De plus, ces fonctions sont également *stateless*, en ce sens que leur définition est indépendante de toute information externe qui aurait pu être collectée à partir, par exemple, d'autres journaux. Cependant, il existe de nombreuses situations dans lesquelles une prédiction sur le journal actuel n'a de sens que si elle est basée sur quelque

chose qui a été observé ou calculé (on pourrait dire «appris») sur les exécutions antérieures du même processus.

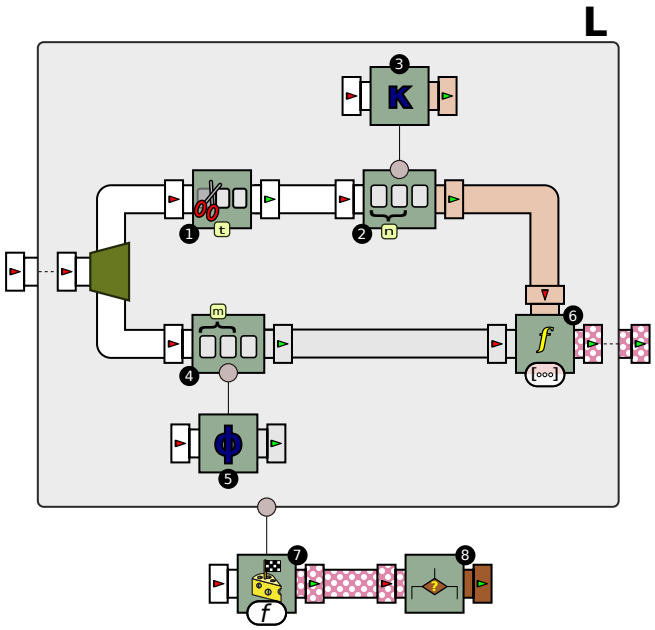


FIGURE 4.3 : Le modèle d'apprentissage prédictif.

4.2.1 LE MODÈLE D'APPRENTISSAGE PRÉDICTIF

C'est l'objet du modèle d'apprentissage prédictif, illustré à la Figure 4.3. Comme dans le modèle de prédiction statique, une bonne partie du processus est effectuée séparément pour chaque instance. C'est pourquoi le flux de travail recommence avec une boîte à tranches (#7). Le flux de travail appliqué à chaque tranche est toutefois différent. Tout d'abord, le sous-flux d'une tranche donnée est divisé en deux copies. Le chemin du bas est envoyé à une fenêtre de largeur m (boîte 4), et une fonction caractéristique ϕ est appliquée à chaque fenêtre comme précédemment (boîte 5). Le chemin du haut est coupé de ses premiers t événements (boîte 1), puis est envoyé à une boîte de fenêtre de largeur n (boîte 2). Sur cette fenêtre, une autre fonction κ est appliquée : nous l'appelons la *fonction de classification* et ses valeurs de sortie sont appelées *classes*. Les caractéristiques calculées par ϕ et les classes calculées par

κ sont ensuite combinées par paires dans un processeur de fonction (boîte 6), qui les place simplement dans un tuple de caractéristiques/classes.

Au lieu de regrouper ces n-uplets dans une table associative, il est demandé à la boîte de découpage de renvoyer directement le résultat de chaque tranche au fur et à mesure de sa production (c'est pourquoi la boîte 7 comporte un pictogramme légèrement différent de celui de la Figure 4.1). La dernière étape du processus consiste à diriger ce flux de n-uplets de classes/caractéristiques dans un algorithme de classification, représenté par la boîte 8. Étant donné un ensemble de tuples de la forme (v, c) (où v est une caractéristique et c est une classe), cet algorithme a pour but de générer une fonction π qui « apprend » l'association entre les valeurs de v et les valeurs de c. Une fois cette association connue, elle peut alors être utilisée comme fonction prédictive du flux de travail de prédiction statique décrit précédemment.

Une instance concrète de ce flux de travail générique est définie par sept paramètres :

— Une fonction de découpage $f : \Sigma \to S$, définie comme précédemment, qui sépare un journal en plusieurs sous-flux

— Deux largeurs de fenêtre $m, n \in \mathbb{N}^+$

— Une fonction d'extraction de caractéristiques $\phi : \Sigma^m \to V$, qui prend une fenêtre de m événements successifs et calcule une valeur de caractéristique $v \in V$

— Une fonction de classification $\kappa : \Sigma^n \to C$, qui prend une fenêtre de n événements successifs et calcule une classe $p \in P$

— Un décalage $t \in \mathbb{N}$ définissant la distance entre la fenêtre «caractéristique» et la fenêtre «classe»

— Une fonction d'apprentissage $\ell : 2^{V \times C} \to \Pi_P^V$, qui prend un ensemble de paires caractéristique/valeur et produit la fonction prédictive $\pi \in \Pi_P^V$, définie par $\pi : V \to P$. L'ensemble Π_P^V est défini comme l'ensemble de toutes les fonctions avec domaine V et avec image P.

Comme on peut le constater, le résultat final de ce workflow est une fonction π ayant la même signature que la fonction prédictive du workflow précédent.

4.2.2 EXEMPLES

Comme dans le cas précédent, ce flux de travail peut être instancié de plusieurs manières pour correspondre à différentes tâches «d'apprentissage». Nous donnons quelques exemples dans ce qui suit.

DURÉE MOYENNE

Considérons un journal dont l'ensemble d'événements est $\Sigma \triangleq \mathbb{N} \times A \times \mathbb{Q}$. Chaque événement est un triplet (n,a,q), où n est un identifiant de processus, a est un nom d'action arbitraire tiré d'un ensemble A et q est un horodatage. Avec $m = 1$, définissons la fonction caractéristique $\phi : \mathbb{N} \times A \times \mathbb{Q} \to A$ comme $\phi(n,a,q) \triangleq a$. Soit $t = 0$ et $n = 2$. On définit la fonction de classification $\kappa : (\mathbb{N} \times A \times \mathbb{Q})^2 \to \mathbb{Q}$ comme $\kappa((n,a,q),(n',a',q')) \triangleq q' - q$.

Intuitivement, ϕ renvoie le nom d'un événement, tandis que κ en calcule la durée (c'est-à-dire la différence d'horodatage entre cet événement et celui qui le suit). Nous rappelons au lecteur que ce traitement est spécifique à la tranche : la durée est définie comme le temps écoulé entre deux événements successifs d'une même instance. Cependant, en raison de la sémantique du processeur de tranchage, la sortie de la boîte 7 de la Figure 4.3 est un flux de n-uplets (a,d) rassemblés sur toutes les tranches. Par conséquent, l'entrée de la fonction d'apprentissage est un ensemble de tuples $S = \{(a_1,d_1),\ldots,(a_n,d_n)\}$, où a_i est un nom d'événement et d_i est sa durée calculée. Soit S_a l'ensemble défini comme $S_a \triangleq \{(a_i,d_i) \in S : a_i = a\}$. Une fonction d'apprentissage $\ell : 2^{A \times \mathbb{Q}} \to \Pi^A_{\mathbb{Q}}$ peut être définie comme $\ell(S) \triangleq \pi$, avec $\pi : A \to \mathbb{Q}$ défini

comme :

$$\pi(a) = \begin{cases} \frac{\sum_{(a,d)\in S_a} d}{|S_a|} & \text{si } S_a \neq \emptyset, \\ 0 & \text{sinon.} \end{cases}$$

En d'autres termes, $\pi(a)$ renvoie la moyenne de toutes les durées calculées pour l'événement a (sur toutes les tranches), ou 0 si l'événement a n'a pas encore été vu. Cela signifie que la durée de chaque événement est «apprise» via une agrégation de durées pour cet événement observé dans plusieurs instances passées du processus.

Comme nous l'avons dit précédemment, cette fonction apprise π peut ensuite être utilisée comme base pour prédire la durée d'événements futurs. On pourrait instancier le flux de travail de prédiction statique en réutilisant les mêmes fonctions π, ϕ et la même largeur de fenêtre $m = 1$. La prévision associée à chaque tranche serait la durée moyenne historique du dernier événement vu dans cette tranche.

PROCHAIN ÉVÉNEMENT LE PLUS PROBABLE

Les prédictions ne doivent pas nécessairement être numériques. Dans cet exemple, nous essayons de prédire le prochain événement à se produire dans chaque tranche, en fonction de la fenêtre de m événements qui le précède. À cette fin, considérons un flux d'événements de la forme $(n,a) \in \mathbb{N} \times A$, composé d'un identifiant de processus n et d'un nom d'action a. Nous définissons la fonction caractéristique $\phi : (\mathbb{N} \times A)^m \to A^m$ comme $\phi((n_1,a_1),\ldots,(n_m,a_m)) \triangleq (a_1,\ldots,a_m)$. La fonction de caractéristiques accumule simplement dans un vecteur la liste ordonnée des m derniers noms d'événements survenus dans une instance spécifique d'un processus. On a vu que c'est ce qu'on appelle un *m-gramme*. Soit $t = 1$, $n = 1$ et définissons la fonction de classification $\kappa : \mathbb{N} \times A \to A$ telle que $\kappa((n,a)) \triangleq a$. La classe associée à chaque m-gramme est simplement le nom de l'événement suivant.

Par conséquent, la fonction d'apprentissage ℓ prend en entrée un multi-ensemble de n-uplets $S = \{((a_{1,1},\ldots,a_{m,1}),a'_1),\ldots,((a_{1,k},\ldots,a_{m,1}),a'_k)\}$, chacun composé d'un m-gramme et d'un nom d'événement. La distinction que S est un multi-ensemble est importante, car le nombre d'occurrences de chaque tuple doit être préservé. À partir d'un tel multi-ensemble, définissons $\pi : A^m \to A$ la fonction telle que $\pi((a_1,\ldots,a_m)) = a$ si et seulement si a est le nom de l'action le plus souvent associé au m-gramme $(a_1,\ldots,a_m)$ dans S. Le résultat est une fonction qui «apprend» l'événement suivant d'un processus en l'associant aux m événements qui le précèdent. Comme auparavant, cette fonction apprise peut ensuite être réintégrée dans le flux de travail de la prédiction statique et peut être utilisée pour calculer des prévisions relatives au prochain événement possible d'un processus.

Toutes les prévisions que nous avons considérées jusqu'à présent sont unimodales, car elles consistent en un seul élément : le nom de l'événement suivant, la durée ou la valeur numérique. Toutefois, le flux de travail proposé prend également en compte les prévisions multimodales. Pour illustrer ce concept, complexifions l'exemple précédent de la manière suivante. Soit $\mu_S : A^m \to 2^A$, la fonction qui, pour un multi-ensemble S donné et un m-gramme $a_1,\ldots,a_m$, renvoie le multi-ensemble de tous les noms d'événements a_i tel que $((a_1,\ldots,a_m),a_i) \in S$. En d'autres termes, μ trouve le multi-ensemble $M \subseteq A$ de toutes les actions associées à un m-gramme donné. Enfin, définissons la fonction $\nu_M : A \to \mathbb{N}$ telle que $\nu_M(a) = d$ soit le nombre d'occurrences de a dans M.

À partir de ces définitions, on peut concevoir une fonction prédictive $\pi : A^m \to (A \to \mathbb{N})$, définie par $\pi((a_1,\ldots,a_m)) \triangleq g$, où $g : A \to \mathbb{N}$ est une fonction telle que :

$$g(a) \triangleq \frac{\nu_{\mu_S(A)}(a)}{|\mu_S(A)|.}$$

Cette fois, la sortie de π est elle-même une fonction ; pour un m-gramme et un nom d'événement a donné, cette fonction renvoie le pourcentage de nombre de fois où cet événement a suivi les événements $a_1, \ldots, a_m$. Cela peut être vu comme une distribution de probabilité tirée des associations entre une fenêtre de m événements et l'événement suivant.

UTILISATION DE L'APPRENTISSAGE MACHINE

Dans les exemples précédents, le terme apprentissage a été utilisé dans un sens large. Nos fonctions d'apprentissage simples dans la boîte 8 peuvent également être remplacées par des algorithmes complexes d'apprentissage automatique (ML) [229].

A titre d'exemple, considérons un ensemble d'événements $\Sigma \triangleq \mathbb{N} \times A \times \mathbb{Q}$ composé d'un identifiant de processus, d'un nom d'événement et d'un horodatage (comme défini précédemment). Supposons que les éléments de A soient totalement ordonnés par une relation arbitraire $\prec$. Définissons la fonction caractéristique $\phi :\triangleq (\mathbb{N} \times A \times \mathbb{Q})^m \to \mathbb{N}^m \times A$ comme $\phi((n_1, a_1, q_1), \ldots, (n_1, a_1, q_1)) \triangleq (c_1, \ldots, c_m, a_m)$. La sortie de ϕ est un tuple avec $m+1$ éléments ; les premiers m éléments sont des valeurs numériques, définies de telle sorte que c_i représente le nombre d'occurrences du i-ème élément de A dans la fenêtre des événements d'entrée (n_1, a_1, q_1), …, (n_m, a_m, q_m). Le dernier élément du tuple est l'étiquette du dernier événement de la fenêtre.

Par conséquent, si $A = \{a, b, c\}$ et $m = 5$ (en supposant que $a \prec b \prec c$), le quadruplet $(3, 2, 0, b)$ indiquerait que a a été vu trois fois au cours des cinq derniers événements, b deux fois et c n'a pas du tout été vu. Le dernier élément du quadruplet indique que le dernier événement de la fenêtre est un b. Le résultat de cette fonction peut être vu comme un vecteur de taille fixe, chaque élément représentant une « caractéristique » calculée sur une fenêtre d'événements. Dans la terminologie ML, on appelle cela un *vecteur de caractéristiques*. Notez

que les éléments d'un vecteur de caractéristiques ne doivent pas nécessairement être du même type.

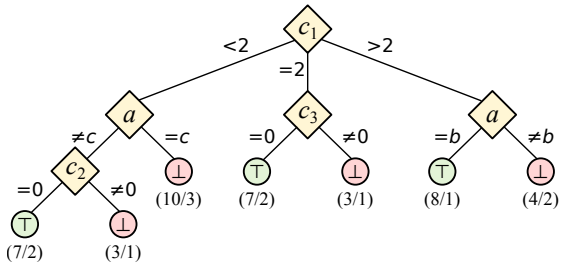


FIGURE 4.4 : Arbre de décision qui relie les valeurs de quatre éléments d'un vecteur (nommé c_1, c_2, c_3 et a) à une classe booléenne.

Pour continuer notre exemple, prenons $t = 0$, $m = 2$ et définissons $\kappa : (\mathbb{N} \times A \times \mathbb{Q})^2 \to \{\top, \bot\}$ comme $\kappa((n,a,q),(n',a',q')) \triangleq (q' - q > 10)$. Il s'agit d'une variation de la fonction durée utilisée dans le premier exemple. Cette fois, elle renvoie une seule valeur booléenne, avec «vrai» (True) indiquant que la durée est supérieure à 10. Avec de telles définitions, la boîte 7 de la Figure 4.3 génère un flux de n-uplets de la forme $((c_1, \ldots, c_m, a), b)$. Les c_i sont le nombre d'occurrences de chaque événement, a est le dernier événement de la fenêtre et b indique si cet événement a duré plus de 10 unités de temps.

Ces n-uplets constituent la base d'un problème typique d'apprentissage machine, qui consiste à découvrir la relation (le cas échéant) entre les valeurs du vecteur de caractéristiques et la classe qui lui est associée (ici, la classe est la valeur booléenne b). Comme il a déjà été mentionné, il existe une très grande variété d'algorithmes d'apprentissage et la description de chacun d'entre eux sort du cadre du présent document. Pour les besoins de cette thèse, il suffit de les considérer comme une «boîte noire» qui calcule en quelque sorte une règle reliant les caractéristiques aux classes.

Pour illustrer ce propos, citons simplement un de ces algorithmes, appelé ID3 [209]. Cet algorithme de classification produit en sortie une structure appelée arbre de décision, tel

que celui représenté à la Figure 4.4. Les nœuds de cet arbre représentent des attributs et les feuilles correspondent à des classes. Pour un vecteur de caractéristiques donné, la classe qui lui est associée par l'arbre de décision est déterminée en partant de la racine et en suivant les branches correspondant à la condition qui s'applique au vecteur. Par exemple, le vecteur de caractéristiques $3, 2, 0, b$ serait associé à la classe TRUE, car dans ce cas, $c_1 > 2$ et le dernier événement de la fenêtre est b.

Comme on peut le constater, un arbre de décision peut être utilisé comme fonction prédictive π du workflow de prédiction statique. Selon les définitions précédentes, associer la classe au vecteur $3, 2, 0, b$ revient à «prédire» que l'événement b dépassera 10 unités de temps. Notez cependant que la classification est rarement précise à 100%. Dans la Figure 4.4, les chiffres situés sous chaque feuille indiquent la proportion de toutes les instances correctement classées. Par exemple, 8/1 indique que, parmi tous les vecteurs de caractéristiques tels que $c_1 > 2$ et ayant b comme dernier événement, 8 d'entre eux sont correctement classés, tandis que 1 appartient à la classe opposée. En règle générale, l'objectif d'un algorithme d'apprentissage est de proposer une classification qui minimise la fraction d'instances mal classées.

Les exemples que nous avons montrés utilisent un décalage t égal à zéro ou à un ; par conséquent, la fenêtre de caractéristiques et la classe qui lui est associée sont très proches les unes des autres. En utilisant des valeurs plus grandes de t, le décalage entre le moment où les entités sont calculées et la classe que l'on souhaite leur associer devient plus grand. Ceci est illustré à la Figure 4.5. Par exemple, en définissant $t = 10$, la classe associée au processeur de caractéristiques serait calculée sur une fenêtre contenant 10 événements ultérieurement. Intuitivement, cela signifie que la fonction prédictive est formée pour prévoir une classe à l'avance.

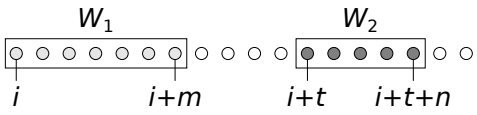


FIGURE 4.5 : Décalage entre la fenêtre utilisée pour apprendre la classification (W_1) et la fenêtre utilisée pour définir l'étiquette (W_2).

4.3 MODÈLE DE PRÉDICTION D'AUTO-APPRENTISSAGE

La dernière étape logique du processus consiste à combiner l'apprentissage d'associations entre des caractéristiques et des classes à la prédiction de classes en temps réel, sur la même instance de journal. C'est ce que nous appelons *l'auto-apprentissage*.

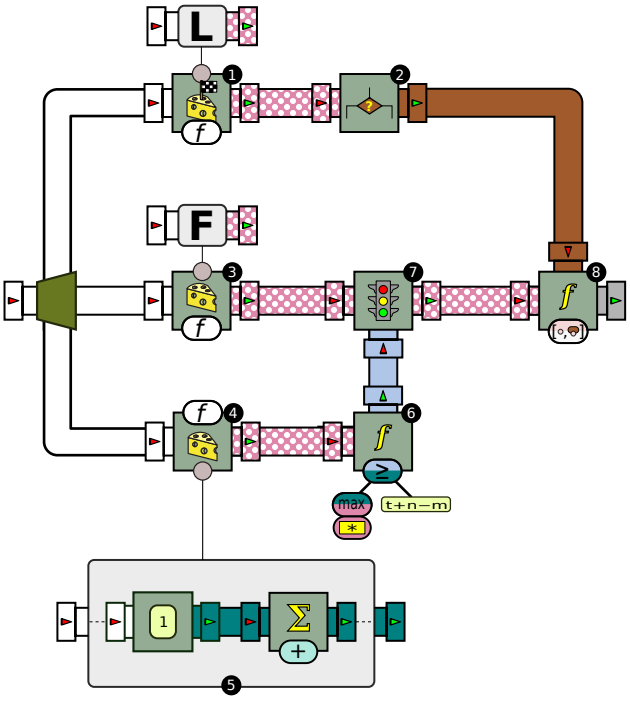


FIGURE 4.6 : Le modèle de prédiction d'auto-apprentissage.

Le processus peut être illustré à la Figure 4.6. Dans ce flux de travail, le flux d'entrée est divisé en trois copies. Dans la trajectoire la plus haute, la boîte L (de la Figure 4.3) crée les paires caractéristique / classe comme indiqué dans le modèle d'apprentissage prédictif ; la sortie est envoyée à une fonction d'apprentissage comme auparavant. Dans une deuxième

copie du flux (partie médiane), une entité est calculée sur une fenêtre des m événements passés, en utilisant la même boîte F que celle de la Figure 4.1. Par conséquent, ces deux premières parties du flux de travail produisent respectivement un flux de fonctions prédictives (boîte 2) et un flux de caractéristiques (boîte 3).

À la toute fin de la chaîne, la boîte 8 joint ces deux flux et applique un classificateur aux caractéristiques calculées pour chaque tranche. Cependant, en raison de la sémantique synchrone des processeurs, si on les associe directement, la fonction prédictive sera appliquée au même moment où elle a été calculée. En se référant à nouveau à la Figure 4.5, cela signifierait que le processeur d'apprentissage recevrait une paire caractéristique/classe calculée sur W_1 et W_2 et que la fonction prédictive résultante serait alors appliquée à nouveau sur W_1.

De toute évidence, le but de ce modèle n'est pas d'obtenir la classification d'une fenêtre d'événements ; il faut plutôt la deviner à l'avance. C'est pourquoi la partie apprentissage prédictif du flux de travail doit d'abord passer par une période d'échauffement au cours de laquelle un apprentissage a lieu, avant de commencer à appliquer la fonction prédictive résultante à de nouvelles fenêtres. Par conséquent, la troisième branche du flux d'entrée (partie inférieure) a précisément pour but de supprimer les premières caractéristiques calculées par la boîte F afin de décaler la fonction prédictive de la fenêtre d'événements sur laquelle elle est appliquée. Plus précisément, les fenêtres de caractéristiques doivent être ignorées jusqu'à ce que le processeur du classifieur produise une première fonction prédictive ; cela se produit exactement quand au moins une tranche a reçu $t + m - n$ événements. En effet, il s'agit du nombre d'événements requis pour produire une première paire caractéristique / classe pour cette tranche et, par conséquent, pour que le processeur de classificateur produise une première fonction prédictive (bien que très fragmentaire). Le décalage requis est obtenu en comptant la longueur de chaque tranche (boîte 5) et en filtrant tout événement provenant du processeur de

fonctions (boîte 7) jusqu'à ce que la plus grande tranche atteigne la longueur souhaitée (boîte 6).

Cette troisième branche du flux de travail, dont la présence est requise pour des considérations de «plomberie», rend le diagramme global un peu inélégant. Cependant, nous verrons dans la section suivante que, lorsque le flux de travail est utilisé comme une boîte noire, cette caractéristique devient transparente pour l'utilisateur. En effet, comme les deux flux de travail précédents, le flux de travail de prédiction à auto-apprentissage est complètement défini par six paramètres, qui se trouvent être les mêmes que pour le flux de travail à apprentissage prédictif.

4.3.1 EXEMPLE

Comme nous l'avons mentionné précédemment, ce workflow peut obtenu en combinant les deux workflows précédents. Selon la manière dont les paramètres sont définis, le flux de travail générique peut représenter différents types de calculs prédictifs. Nous donnons l'exemple ci-dessous.

DURÉE MOYENNE

Reprenons l'exemple du journal dont l'ensemble d'événements est $\Sigma \triangleq \mathbb{N} \times A \times \mathbb{Q}$. Chaque événement est un triplet (n,a,q), où n, a et q sont respectivement : un identifiant de processus, un nom d'action arbitraire et un horodatage. Dans ce flux de travail, l'entrée est dupliquée en trois copies. Dans la partie la plus haute, la boite L crée les paires action / durée renvoyées respectivement par la fonction ϕ renvoie le nom d'un événement, et la fonction κ calcule la durée de celui-ci. Reprenons également la définition de la fonction caractéristique $\phi : \mathbb{N} \times A \times \mathbb{Q} \to A$ comme $\phi(n,a,q) \triangleq a$ et la fonction de classification $\kappa : (\mathbb{N} \times A \times \mathbb{Q})^2 \to \mathbb{Q}$ comme $\kappa((n,a,q),(n',a',q')) \triangleq q' - q$. Avec $m = 1$, $t = 0$ et $n = 2$.

La sortie de la boîte 1 de la Figure 4.6 est un flux de n-uplets (a,d) rassemblés sur toutes les tranches. Par conséquent, l'entrée de la fonction d'apprentissage (boîte 2) est un ensemble de tuples $S = \{(a_1,d_1),\ldots,(a_n,d_n)\}$, où a_i est un nom d'événement et d_i est sa durée calculée. La sortie de la boîte 7 est une fonction d'apprentissage $\pi(a)$ qui renvoie la moyenne de toutes les durées calculées pour l'événement a (sur toutes les tranches), ou 0 si l'événement a n'a pas encore été vu. Cette fonction apprise π est utilisée pour prédire la durée d'événements futurs.

Une deuxième copie du journal $\Sigma \triangleq \mathbb{N} \times A \times \mathbb{Q}$ est traité par la boite F (dans la partie médiane). Définissons la fonction d'extraction de caractéristiques $\phi : \mathbb{N} \times A \times \mathbb{Q} \to A$ comme $\phi(n,a,q) \triangleq a$, avec $m = 1$. Autrement dit, ϕ renvoie le nom de l'événement dans la fenêtre m.

Finalement, la fonction de la boîte 8 renvoie une prédiction de la durée de l'action (caractéristique extraite dans la boîte F et sortie de la boîte 3) en appliquant le modèle π (appris et renvoyé par la boîte 2).

4.4 CONCLUSION

Ce chapitre a permis d'introduire un framework générique pour le calcul des prédictions dans un log de business process. Nous avons tout d'abord introduit le modèle d'analyse prédictive statique et quelques exemples explicatifs. Ensuite, on a fait de l'application d'algorithmes d'apprentissage en utilisant une fonction prédictive. Ce modèle permet d'utiliser des fonctions avancées tels que des algorithmes d'apprentissage machine pour faire des prévisions. Enfin, nous avons proposé le modèle de prédiction d'auto-apprentissage, où le modèle de prédiction est mis à jour avec chaque événement entrant dans le workflow.

CHAPITRE V
ÉTUDE EXPÉRIMENTALE

Le besoin d'extraire les tendances, à partir des flux d'événements, a été expliqué et justifié dans les chapitres précédents. Un cadre générique pour détecter les écarts de tendances a été proposé de même qu'un cadre pour prédire de futures tendances possibles. Ce chapitre sera consacré à l'implémentation des deux cadres dans un outil d'exploration de flux d'événements.

Dans les deux chapitres précédents, on a montré que de nombreux problèmes de détection de tendances et d'analyse prédictive peuvent être vus comme des cas particuliers d'un petit nombre de workflows génériques et paramétrables. Ces workflows ont été définis de manière théorique seulement. On ne sait pas s'ils peuvent être mis en œuvre concrètement, au moyen de quelle technologie. On ne sait pas non plus l'ordre de grandeur de la vitesse de traitement, ni leur scalabilité en fonction de leurs divers paramètres. L'objectif de ce chapitre est de faire le pont entre la théorie et la pratique en décrivant une implémentation et en évaluant expérimentalement sa performance.

La prochaine section servira à décrire d'une façon générale l'outil sur lequel se basera ce travail : BeepBeep. Nous verrons pourquoi le choix s'est arrêté sur cet outil, quelques notions de base et des exemples de requêtes auxquels peut répondre cet outil.

5.1 CHOIX TECHNOLOGIQUES

L'objectif de la présente section est de permettre à l'utilisateur de créer des instances des workflows des deux chapitres précédents en ne spécifiant que les paramètres, de telle sorte à ce que le tout soit en mode **streaming**.

Les outils de *runtime verification* sont centrés sur la logique, la plupart ne font pas d'agrégations, ni de traitement sur des fenêtres d'événements, et par-dessus tout, il est extrêmement difficile de créer une extension à ce type d'outils. De leur côté, les outils de *complex event processing* sont basés sur les tuples et obligent à tout écrire via leur langage de requête. Finalement, les outils statistiques ne sont pas streaming.

Le choix de l'outil d'application s'est arrêté sur BeepBeep. Ce n'est pas nécessairement le seul outil ou le meilleur, mais sa simplicité et son extensibilité en font une plateforme raisonnable pour une preuve de concept. En effet, BeepBeep est un système qui implémente directement les définitions introduites au chapitre 3 (processeurs, chaine de processeurs, etc.). Il est important de noter que tous les workflows des chapitres précédents sont représentés avec des pictogrammes de BeepBeep.

BeepBeep est un moteur de requêtes pour le traitement de flux d'événements développé au Laboratoire d'informatique formelle à l'Université du Québec à Chicoutimi [230]. Il est mis à la disposition du public sous licence open source. Sa version initiale permettait de faire du *runtime verification*, tolérait des événements complexes avec des structures imbriquées et permettait l'utilisation de la logique temporelle linéaire.

La version la plus récente de l'outil est le BeepBeep 3. Le manuel d'instructions de cette version est disponible sur son site web [13]. Un des avantages de BeepBeep est qu'il est développé au LIF, ce qui facilite beaucoup de choses au point de vue logistique. La force de BeepBeep 3 réside dans le fait qu'avec toutes les fonctionnalités qu'il offre, il reste léger, très intuitif et personnalisable.

13. https://liflab.github.io/beepbeep-3/

5.1.1 EXTENSIONS DU NOYAU

Il est possible d'étendre le noyau de BeepBeep pour ajouter de nouvelles fonctionnalités, et ce, à travers des palettes ou des fonctions ou processeurs implémentés sur mesure.

PALETTES

Afin de rendre le moteur BeepBeep plus léger, un système modulaire a été mis au point. Les fonctionnalités de l'outil ont été dispersées sur plusieurs bibliothèques supplémentaires (comme des fichiers .JAR) appelées *palettes*. Ces palettes contiennent les définitions de nouveaux processeurs ou fonctions à utiliser avec les éléments centraux de BeepBeep. Une palette est incluse dans le projet uniquement si on a besoin de son contenu. Les palettes ont plusieurs objectifs : lire des types de fichiers spéciaux, générer des traces, accéder à un réseau, etc.

Parmis les palettes existantes : Tuple, Java Widgets, Plots, Réseaux, Analyse de fichiers JSON et XML.

CRÉATION DE SES PROPRES FONCTIONS ET PROCESSEURS

BeepBeep est un petit noyau de processeurs et de fonctions intégrés. Le reste des fonctionnalités sont implémentées dans des palettes à utilisation optionnelle. C'est là que réside l'un des avantages de BeepBeep, son extensibilité.

Il se peut que l'on ait besoin d'un processeur non existant dans BeepBeep. Dans ce cas, BeepBeep permet de créer de nouveaux processeurs personnalisés qui peuvent ensuite être composés avec les processeurs existants. Il existe plusieurs façons pour la création d'un nou-

veau processeur. En héritant de l'objet `Processor`, ou bien en héritant de `GroupProcessor` en combinant des processeurs déjà existants.

Tout comme pour les processeurs, il est possible de personnaliser des fonctions. Deux méthodes sont disponibles pour la création de nouvelles fonctions. La première serait d'hériter de la classe `FunctionTree`, en composant des objets fonctions préexistants, ou bien en héritant de la classe `Function` ou un de ses multiples descendants.

5.2 MISE EN OEUVRE ET EXPÉRIENCES POUR LES DÉVIATIONS DE TENDANCES

Nous discutons maintenant de la manière dont les concepts décrits dans les chapitres précédents ont été implémentés dans un outil d'exploration de flux d'événements réel. À cette fin, une bibliothèque Java appelée Pat The Miner ou PTM a été développée. Dans cette section, nous expliquons d'abord comment la bibliothèque a été mise en œuvre et comment elle peut être utilisée. Nous passons ensuite à la description des mesures empiriques effectuées sur cette bibliothèque et donnons un aperçu de ses performances dans diverses conditions.

5.2.1 UNE PALETTE BEEPBEEP POUR L'EXPLORATION DE DONNÉES EN CONTINU

Pat The Miner a été développé en tant qu'une extension du moteur de traitement des événements BeepBeep, décrit dans la section 5.1. Plus précisément, il s'agit d'une palette BeepBeep qui définit les nouveaux objets `Processor` et `Function` spécifiques aux calculs analytiques sur les journaux d'événements. Elle fournit notamment deux nouveaux processeurs pour la détection des écarts de tendance, appelés processeurs `TrendDistance` et `SelfCorrelatedTrendDistance`. Ces deux processeurs fonctionnent exactement de la manière

décrite dans la section précédente. Par exemple, voici un extrait de code Java qui instancie un modèle de distance de tendance :

```
TrendDistance<Number,Number,Number> td = new TrendDistance(
  6, 200, new RunningAverage(),
  new FunctionTree(Numbers.absoluteValue,
       new FunctionTree(Numbers.subtraction,
           StreamVariable.X, StreamVariable.Y)),
  0.5, Numbers.isLessThan);
```

FIGURE 5.1 : Modèle de distance de tendance

Dans ce fragment de code, on peut reconnaître tous les paramètres définissant une instance spécifique du modèle de distance de tendance. Ici, `RunningAverage` est un objet qui calcule la moyenne mobile sur un flux de nombres. La valeur `200` est la largeur de la fenêtre glissante sur laquelle cette moyenne doit être calculée. Le nombre `6` est la tendance de référence ; ici, nous calculons la déviation du flux de nombres par rapport à une moyenne de référence de `6`. Le constructeur `FunctionTree` crée une fonction définissant la métrique de distance : c'est une fonction binaire qui reçoit deux valeurs, les soustrait et prend la valeur absolue de la différence (remarquez qu'il s'agit de la définition de la distance de Manhattan de dimension 1). La valeur `0.5` est le seuil de distance maximum et `Numbers.isLessThan` est une référence à la fonction utilisée pour comparer la distance calculée à ce seuil.

À partir de là, `td` est un objet `Processor` qui peut être utilisé comme tout autre processeur BeepBeep. Son entrée peut être connectée à n'importe quel flux de valeurs numériques et sa sortie produit un flux de valeurs booléennes. Le fait de surveiller l'apparition de la valeur `false` dans ce flux de sortie peut être utilisé pour détecter un écart de tendance dans le flux d'entrée. De plus, étant donné que tous les processeurs BeepBeep fonctionnent en mode continu, cela signifie que cet écart peut être détecté en temps réel, au fur et à mesure que le processeur progresse dans le flux d'entrée.

Ceci devrait être mis en contraste avec la manière dont le même écart de tendance pourrait être calculé en utilisant d'autres logiciels disponibles. Par exemple, la Figure 5.2 montre un moyen possible d'obtenir le même résultat en utilisant le langage de script du programme statistique R. Il est important de noté que R fonctionne en batch et non en continu. En effet, Hahsler. M et al. [231] montionne clairement que R n'est pas la platforme idéale pour le traitement des flux de données en temps réel. La comparaison est, donc, faite entre le traitement réalisé par R en batch et celui de BeepBeep en continu. Les quatre premières lignes définissent la fonction de distance : les lignes 5 à 9 calculent un vecteur de sortie avec la tendance sur une fenêtre glissante de 200 événements, tandis que les lignes 10 à 17 produisent un autre vecteur de sortie de valeurs booléennes équivalent à ce qui est produit par le workflow à distance de tendance statique. On peut faire valoir que ce code est moins intuitif quant à ce qui est exactement calculé, bien qu'une partie de cette fonctionnalité puisse être regroupée dans une fonction définie par l'utilisateur avec des paramètres. On peut également se demander si le code pourrait être rédigée de façon plus efficace.

Cependant, à part toutes ces considérations, une différence importante demeure : le résultat calculé par le code R ne fonctionne pas en mode continu. C'est-à-dire que le vecteur de tendances `vec1` est d'abord *complètement* calculé avant d'être passé à la seconde moitié du script, où le vecteur de Booléens `vec2` est à nouveau *complètement* calculé et retourné. Il n'est pas possible de recevoir les événements de sortie un par un, pendant que les entrées sont en cours de consommation. Ceci limite considérablement la possibilité d'utiliser un tel script pour détecter les écarts en temps réel. Enfin, chacun de ces vecteurs est créé et *reste* en mémoire pendant toute la durée du programme. Cela devrait être mis en contraste avec un mode d'opération en continu, dans lequel les événements d'entrée sont ignorés dès qu'ils ont été consommés et traités. En fait, nous verrons dans la section 5.3 que la consommation de mémoire de PTM est constante pendant l'exécution du programme.

```
dist <- function(x, y) {
  result <- abs(x-y)
}
spots <- seq(from = 1, to = (length(data) - 200 + 1), by = 1)
result <- vector(length = length(spots))
for(i in 1:length(spots)) {
    result[i] <- dist(data[spots[i]:(spots[i] + 200 - 1)])
}
vec2 = array(dim = length(result))
for (i in 1:length(vec1)) {
  if (abs(vec1[i] - 6) > 0.5) {
    vec2[i] = TRUE
  } else {
    vec2[i] = FALSE
  }
}
```

FIGURE 5.2 : Fragment de code écrit dans le langage de script du logiciel statistique R, qui effectue le même calcul sur un journal d'entrée que l'extrait BeepBeep de la Figure 5.1.

En revanche, d'autres outils tels que Esper [14] et Siddhi [126] peuvent effectuer un calcul en continu et fournissent déjà des fonctions statistiques courantes telles que la moyenne et l'écart type qui peuvent être évaluées sur une fenêtre glissante. Toutefois, ces calculs doivent être écrits dans des instructions *ad hoc* `select` dans le langage de requête de l'outil et ne sont pas encapsulés dans un modèle générique paramétrable, comme c'est le cas ici. Une comparaison détaillée de BeepBeep avec ces outils dépasse le cadre de cette thèse. Un rapport technique [6] traite d'une telle comparaison.

Conformément aux observations faites dans la section précédente, nous pouvons voir que la classe `TrendDistance` est générique. Les trois noms de classe `Number` apparaissant dans la déclaration de type de `td` font respectivement référence au type de la tendance calculée, au type des événements produits par le processeur de tendance et au type de valeur renvoyé par

14. http://www.espertech.com

la métrique de distance. Cela signifie que `TrendDistance` peut être instancié différemment, en utilisant d'autres objets pour le processeur de tendances, la métrique de distance et la tendance de référence. Par exemple, l'extrait de code suivant instancie un modèle de distance de tendance, en utilisant un vecteur des trois premiers moments statistiques comme tendance :

```
TrendDistance<DoublePoint,Number,Number> td = new TrendDistance(
  new DoublePoint(new double[]{0.3d, 0.1d, 0.6d}),
  200, new RunningMoments(3),
  new PointDistance(new EuclideanDistance()),
  2, Numbers.isLessThan);
```

FIGURE 5.3 : Exemple de distance de tendance, avec un vecteur des trois premiers moments statistiques comme tendance.

Cette fois, le motif de référence est un vecteur de valeurs numériques (`DoublePoint`), et la métrique de distance est la distance euclidienne sur ces vecteurs.

L'objet processeur pour la distance de tendance auto corrélée peut être créé d'une manière similaire ; les arguments de son constructeur correspondent aux paramètres que prend ce modèle particulier.

5.3 MISE EN OEUVRE ET EXPÉRIENCES DE DISTANCE DE TENDANCE

Afin d'évaluer la faisabilité de l'approche, nous avons procédé à diverses mesures empiriques sur la mise en œuvre concrète de Pat The Miner sur les tendances observées précédemment.

Les expériences ont été mises en œuvre en utilisant le framework de test LabPal [232], qui permet de regrouper tout le code, les bibliothèques et les données d'entrée nécessaires dans un seul fichier exécutable autonome, de sorte que n'importe qui puisse télécharger et reproduire

	Tendance	Métrique
C1	Moyenne mobile	Distance de Manhattan de la dimension 1
C2	Vecteur des 3 premiers moments statistiques	Distance euclidienne
C3	Distribution cumulative des symboles	Map distance
C4	Vecteur des fréquences des symboles	Distance euclidienne au centroïde du cluster le plus proche
C5	Ensemble de *N*-grammes	Index de Jaccard
C6	Durée moyenne d'une tranche	Distance de Manhattan de dimension 1

TABLEAU 5.1 : Processeurs de tendance et métriques de distance associées utilisés dans les expériences.©Massiva Roudjane

indépendamment les expériences. Une instance de laboratoire téléchargeable contenant toutes les expériences de ce document peut être obtenue en ligne [233].

Dans l'ensemble, nos mesures empiriques impliquent 118 expériences individuelles, qui ont généré au total 482 points de données distincts. Toutes les expériences ont été exécutées sur une Intel CORE i5-7200U 2.5 GHz fonctionnant sous Ubuntu 18.04, à l'intérieur d'une machine virtuelle Java 8 avec 1746 Mo Mo de mémoire.

Dans une première série d'expériences, nous avons mesuré les performances du modèle de distance de tendance sur différents calculs de tendance. Plus précisément, nous avons tenté de quantifier le débit auquel on peut s'attendre en essayant de détecter un écart pour une tendance de référence fixe en temps réel sur un flux d'événements donné. Les expériences ont été réalisées en générant à la volée un flux d'événements de nombres aléatoires ou de symboles discrets, selon la tendance à calculer sur le flux d'événements. Ce flux a ensuite été envoyé au processeur de modèle de distance de tendance. Les processeurs de tendance et les métriques de distance utilisés dans les expériences sont résumés dans le Tableau 5.1. Ces diverses configurations de workflow peuvent être vues comme des versions abstraites de certaines des requêtes introduites dans le chapitre 1.

Bien que les flux d'événements dans ces expériences soient *synthétiques* et ne proviennent pas d'un exemple du monde réel, nous soutenons que ceci n'a aucune incidence sur

les mesures de débit. Considérons par exemple la boîte 1 dans le workflow de la Figure 3.11, dans le contexte du scénario C1 : cette boîte calcule la somme sur une fenêtre de n nombres. Il n'y a aucune raison de penser que l'exécution de cette boîte prend un temps différent si ces nombres proviennent d'un log «réel» plutôt que d'un journal synthétique. À son tour, la boîte 3 calcule la soustraction d'un certain nombre P et la valeur de sortie de la boîte 1. Encore une fois, il n'y a aucune raison de penser que l'exécution de cette boîte prend un temps différent si ces deux nombres proviennent d'un journal «réel» plutôt que d'un journal généré. Le même raisonnement peut être appliqué, petit à petit, jusqu'à atteindre le résultat final.

De la même manière, puisque le but des expériences est de mesurer le débit qui peut être atteint par le processeur, le seuil de distance réel n'est pas pertinent. En effet, le processeur de distance de tendance renvoie un flux de valeurs booléennes, et si ces valeurs sont vraies (le flux d'entrée est suffisamment proche de la tendance de référence) ou fausses (le flux d'entrée est trop éloigné de la tendance de référence) n'a aucun impact sur le calcul. Ceci, à son tour, n'a aucun impact sur le nombre d'événements par seconde qui peuvent être traités pour un processeur de tendance donné. Par conséquent, dans chacune des expériences, le motif de référence et le seuil de distance ont été fixés à des valeurs «factices» arbitraires ; les valeurs réelles utilisées n'ont aucun impact pratique sur les mesures.

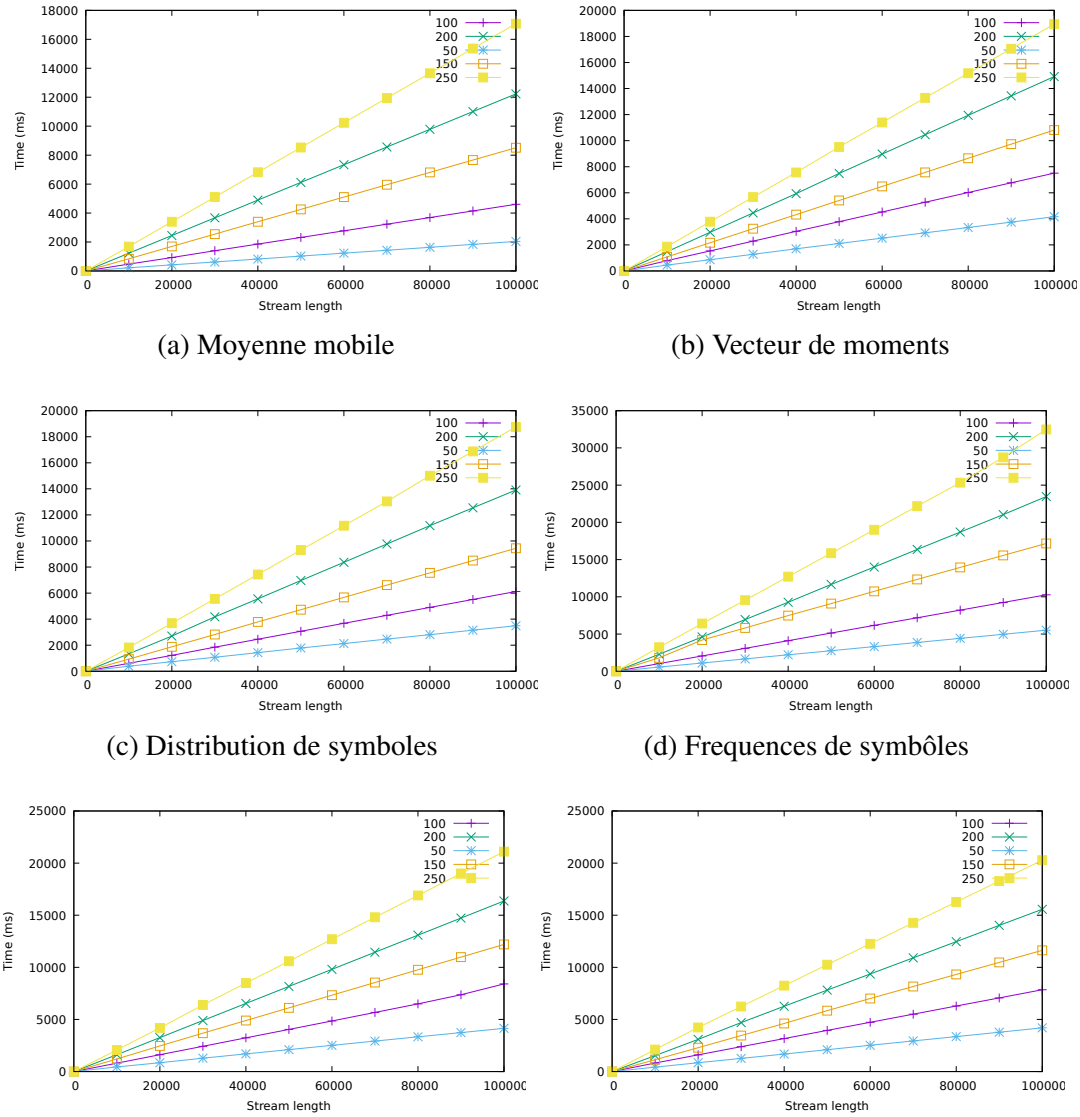
Enfin, il est utile de noter que les flux d'entrée synthétiques que nous générons contiennent le minimum d'informations dans chaque événement requis par chaque scénario spécifique. Par exemple, le scénario C1 calcule la moyenne mobile sur une fenêtre de valeurs numériques, donc le flux d'entrée donné à cette instance de workflow spécifique est un flux de nombres bruts. En d'autres termes, nous nous concentrons sur la partie déviation de tendance du processus, qui suppose que tout prétraitement requis sur le flux d'entrée pour extraire les valeurs d'intérêt a déjà été effectué. De toute évidence, dans des situations réelles, ces informations sont beaucoup plus susceptibles d'être contenues dans des événements avec une structure plus

riche (c'est-à-dire une plus grande «charge utile»), comme un fragment XML qui contient plusieurs attributs. La raison de l'omission est double. Premièrement, ce prétraitement peut dépendre de la taille de la charge utile des événements d'entrée, mais il entraîne un coût qui ne dépend pas des événements précédents sur les événements futurs. Son impact sur le débit global est donc au pire un facteur constant. Deuxièmement, une fois ce prétraitement effectué sur chaque événement, les performances du flux de travail en aval sont complètement indépendantes de la taille d'origine des événements en amont.

La Figure 5.4b montre le temps de calcul cumulé du processeur de distance de tendance, lorsque le motif à calculer est un vecteur des trois premiers moments statistiques (configuration C2 dans le Tableau 5.1). Le graphique illustre le temps de calcul pour différentes largeurs de fenêtre, allant de 50 à 200 événements. Il montre clairement une tendance linéaire. Cela signifie que le calcul ne «ralentit» pas à mesure que le processeur progresse dans le flux d'événements. Comme prévu, une fenêtre plus large implique un calcul plus important à faire pour chaque fenêtre, et donc un débit inférieur. Dans le cas du vecteur de moments, le débit moyen est d'environ 24000.0 Hz pour une fenêtre de 50 événements, et 5280.0 Hz pour une fenêtre de 250 événements.

Une remarque doit cependant être faite sur les valeurs de débit, qui s'applique à tous les autres résultats de ce type dans le reste de cette section. Un débit de 5280.0 représente le taux *maximum* auquel les événements peuvent être traités en temps réel. Cela ne signifie pas que les événements *sont* produits aussi rapidement dans un cas d'utilisation réel. Par exemple, un ensemble de journaux réels tirés d'un récent défi «Business Process Intelligence Challenge» contient 2,5 millions d'événements sur une période de trois ans, ce qui représente moins de 3 000 événements par *jour* [234]. D'autres scénarios d'analyse des journaux, tels que la surveillance de l'exécution des jeux vidéo, ne produisent pas plus de 160 événements

produits par seconde [42]. En fait, un débit de seulement 1200 Hz suffit pour traiter cent millions d'événements par jour.



(a) Moyenne mobile

(b) Vecteur de moments

(c) Distribution de symboles

(d) Frequences de symbôles

(e) Ensemble de N-grammes

(f) Durée moyenne d'une slice

FIGURE 5.4 : Temps de calcul cumulé sur un flux d'événements, pour différents processeurs de tendance et largeurs de fenêtre.©Massiva Roudjane

Les autres processeurs de tendance et les métriques de distance présentent un comportement très similaire : seule la plage de débit diffère entre les expériences. Par exemple, dans la

configuration C3 (illustrée dans la Figure 5.4c), le débit varie de 28600.0 Hz pour une fenêtre de 50 événements à 5330.0 Hz pour une fenêtre de 250. En fait, pour une largeur de fenêtre donnée, tous les processeurs de tendance que nous avons étudiés se comportent linéairement en fonction de la longueur du flux d'événements. Cela est dû au fait que la majeure partie du calcul dans le modèle de distance de tendance se fait sur une fenêtre de taille fixe, qui ne dépend pas du nombre d'événements traités depuis le début du flux. En d'autres termes, la charge de travail pour calculer la première fenêtre est semblable à la charge de travail nécessaire pour calculer la n-ième fenêtre.

Nous avons également mesuré l'impact de la largeur de la fenêtre sur les calculs de tendance. La Figure 5.5 montre le débit moyen des différents processeurs de tendance, en variant la largeur de la fenêtre dans chaque cas. Cette figure confirme l'intuition selon laquelle l'augmentation de la taille de la fenêtre a un impact négatif sur le débit. Si $f(n)$ est une fonction qui renvoie le temps de calcul (en secondes) pour une seule fenêtre de largeur n, alors $1/f(n)$ de ces fenêtres peuvent être traitées pendant une seconde. À mesure que $f(n)$ augmente avec n, le ratio $1/f(n)$ diminue, donnant la forme d'une fonction inverse. C'est en effet ce qui peut être observé pour tous les processeurs de tendance représentés sur la Figure 5.5.

Nos expériences ont également mesuré la consommation de mémoire de chacune des chaînes de processeurs. L'utilisation de la mémoire est notoirement difficile à calculer en Java, car la machine virtuelle Java (JVM) effectue le ramasse-miettes à des intervalles difficiles à prévoir. Par conséquent, la mesure de la taille de la machine virtuelle Java est un moyen très indirect de connaître la mémoire totale consommée par des objets «vivants» à un moment donné. Pour résoudre ce problème, nous avons utilisé l'outil `SizePrinter` fourni par la bibliothèque de sérialisation d'objets Azrael. Cet outil effectue une traversée récursive de tous les objets d'un programme et de leurs champs membres, et les décompose en un ensemble de

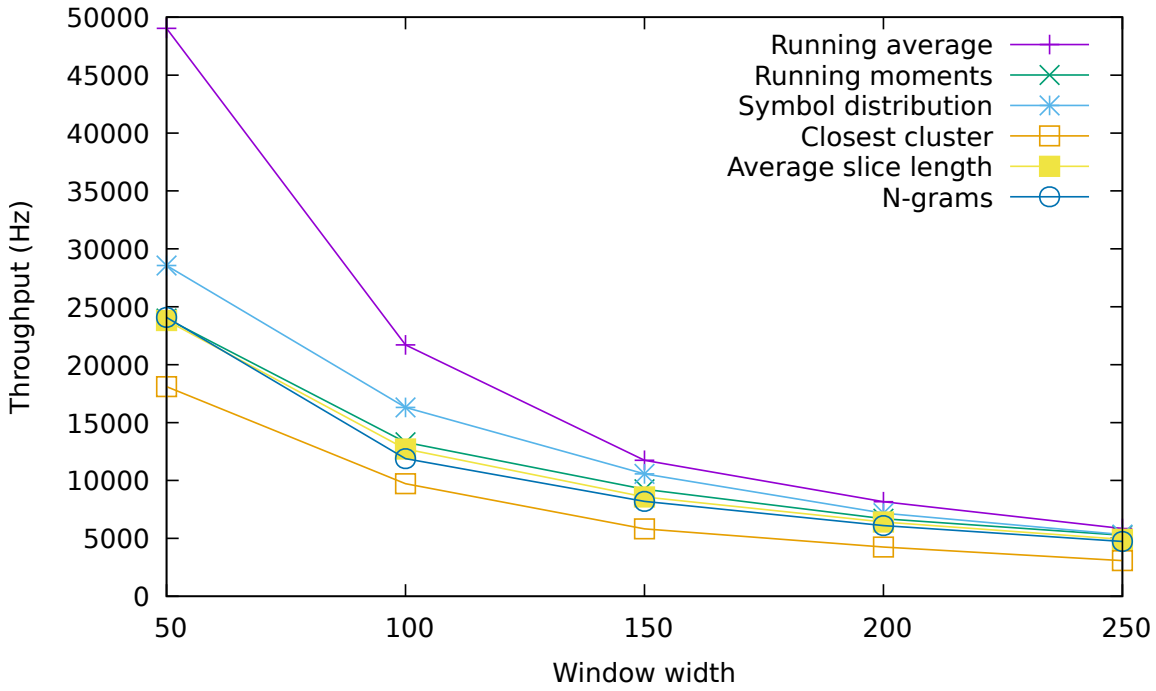


FIGURE 5.5 : Impact de la largeur de la fenêtre sur le débit, pour différents calculs de distance de tendance.©Massiva Roudjane

valeurs primitives dont la taille individuelle allouée est connue. Il peut donc être utilisé pour fournir une estimation relativement fidèle du nombre d'octets pris par tous les objets dans un scope donné.

La Figure 5.6 montre la taille de mémoire maximale consommée par chaque modèle de workflow, pour différentes largeurs de fenêtre. Il montre que la taille de la mémoire suit clairement une tendance linéaire par rapport à la largeur de la fenêtre, quelle que soit la tendance réelle calculée. Cela n'est pas surprenant : à part la mémoire consommée par les objets `Processor` et `Function` d'une instance spécifique du workflow (dont le nombre ne varie pas), le seul élément restant en mémoire est la fenêtre elle-même et les événements qu'elle contient. Bien que cela ne soit pas indiqué, des tendances similaires ont été observées pour les autres schémas de flux de travail.

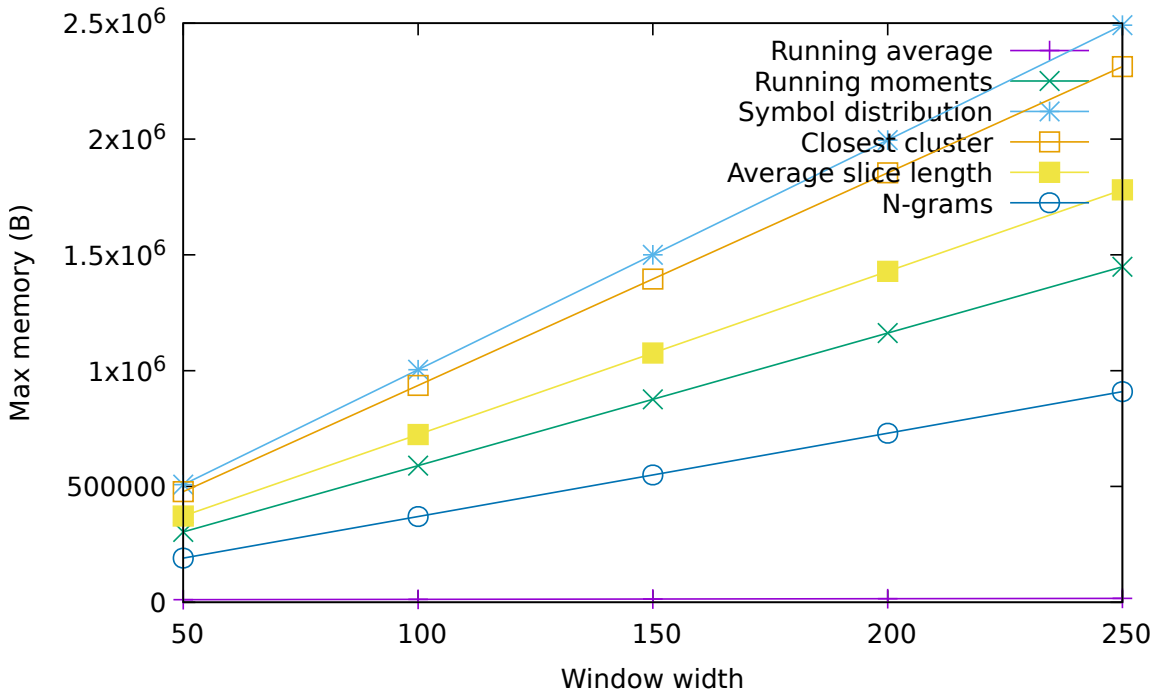


FIGURE 5.6 : Impact de la largeur de la fenêtre sur la consommation de mémoire, pour différents calculs de distance de tendance.©Massiva Roudjane

Enfin, nous avons également comparé les performances de la chaîne de processeurs BeepBeep avec des scripts écrits et exécutés par le logiciel statistique *R*. Les résultats sont résumés au Tableau 5.2. Les résultats montrent que BeepBeep a un débit qui est comparable, et dans certains cas plus élevé, que le fragment de code *R* équivalent. C'est particulièrement le cas pour le scénario C6 (longueur de tranche moyenne), pour lequel BeepBeep s'exécute plus rapidement par une certaine marge. Comme on peut le voir, les utilisateurs ne sont pas excessivement pénalisés par l'utilisation de notre architecture générique, et le fait que notre solution proposée offre des performances qui se comparent raisonnablement à un outil bien établi.

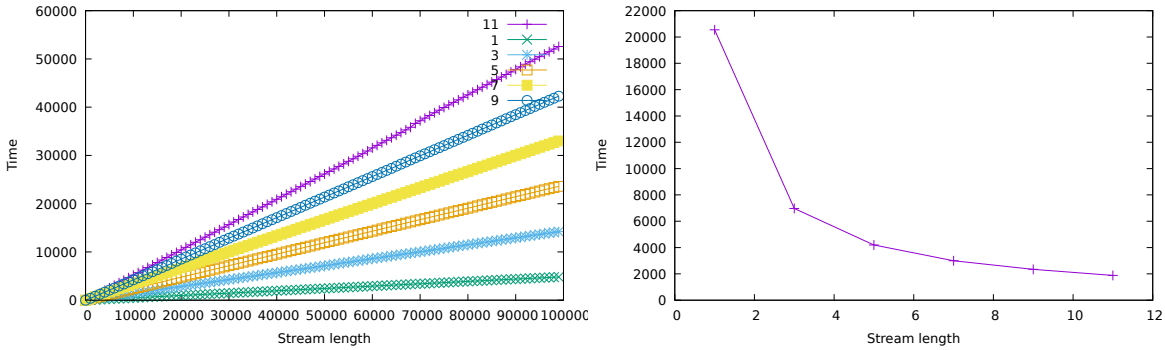
Fonction de tendance	R	BeepBeep
Longueur moyenne de tranche	4585.730	23815.432
Cluster le plus proche	19790.422	18103.006
N-grams	50378.336	24085.02
moyenne mobile	48100.527	49044.14
Moments courants	26525.465	23981.055
Distribution de symbôles	19197.734	28563.553

TABLEAU 5.2 : Comparaison de débit en événements par seconde, entre les scripts BeepBeep et R, pour une largeur de fenêtre de 50. ©Massiva Roudjane

5.3.1 EXPÉRIENCES DE SECOND ORDRE

Dans une deuxième série d'expériences, nous avons mesuré le débit du modèle de distance de tendance de second ordre. Étant donné que ce modèle est principalement constitué d'une combinaison de modèles de distance de tendance de premier ordre, nous avons utilisé un seul processeur de distance de tendance statique (moyenne mobile) et l'avons répété un nombre prédéfini de fois pour créer un modèle de second ordre composé d'un nombre variable de processeurs de premier ordre. Le processeur de distance de tendance du second ordre utilise la fonction de corrélation multifactorielle décrite dans la section 3.7.2. De cette façon, nous pouvons mesurer l'impact du nombre de processeurs de premier ordre sur le débit global ; ce débit lui-même serait mis à l'échelle en fonction du débit de base de chaque distance de tendance de premier ordre spécifique, comme cela a été mesuré dans les expériences précédentes.

Les résultats sont présentés dans la Figure 5.7. Dans la Figure 5.7a, le débit du modèle est indiqué par rapport à la longueur de trace, pour les tendances de premier ordre variant en nombre (k) de 1 à 11. Sans surprise, le temps d'exécution pour un k donné est linéaire en la longueur de la trace et correspond à peu près à k fois le temps d'exécution d'un de ces



(a) Débit

(b) Impact du nombre de tendances de premier ordre

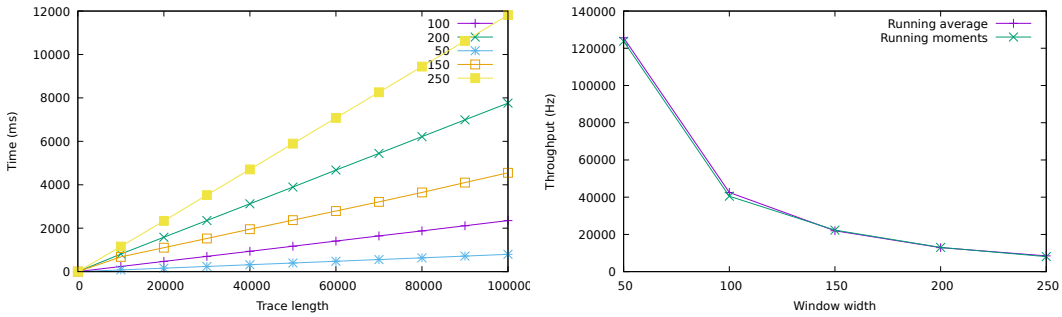
FIGURE 5.7 : Débit et impact du nombre de tendances de premier ordre pour le modèle de distance de tendance de second ordre.©Massiva Roudjane

processeurs. La Figure 5.7b montre l'impact de k sur le débit global du modèle. On peut voir que la courbe suggère une tendance qui suit l'inverse de k.

Des mesures similaires ont été effectuées sur le modèle de distance de tendance contextuelle et sont illustrées dans la Figure 5.8. Dans ces expériences, la source d'entrée des événements est un flux de tuples contenant un horodatage et une valeur numérique fictive. La moyenne mobile de la valeur numérique sur une fenêtre de largeur n est utilisée comme tendance, tandis que le jour de la semaine dans le dernier événement de la fenêtre est utilisé comme contexte. On ne voit pratiquement aucune différence entre ces durées de fonctionnement et les chiffres calculés pour le modèle de distance de tendance statique.

5.3.2 EXPÉRIENCES AUTO CORRÉLÉES

Dans un troisième lot d'expériences, nous avons mesuré le débit du processeur de distance de tendance auto corrélée. Comme nous l'avons vu précédemment, ce modèle diffère d'une simple distance de tendance par le fait que la tendance de référence n'est pas une valeur

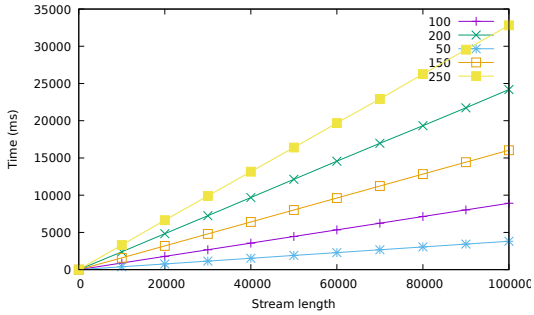


(a) Débit (b) Impact de la largeur de la fenêtre

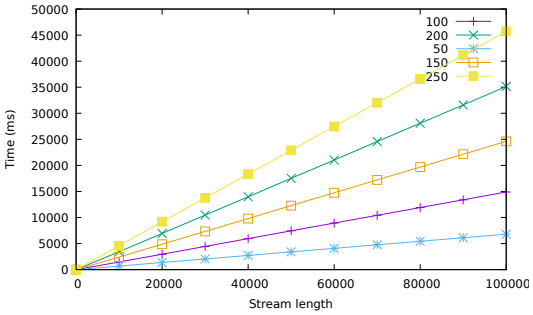
FIGURE 5.8 : Un extrait de résultats expérimentaux pour le flux de travail de la distance de tendance contextuelle. ©Massiva Roudjane

fixe, mais est plutôt calculée à la volée sur une fenêtre glissante d'événements passés. Nous nous attendons donc à ce qu'il soit plus lourd en termes de charge de calcul. Plus précisément, supposons que le coût de calcul de la fenêtre glissante domine le calcul global du modèle de distance de tendance. Étant donné que le modèle de distance de tendance auto corrélée implique deux de ces calculs de fenêtre, son débit doit être réduit de moitié par rapport au modèle de distance de tendance.

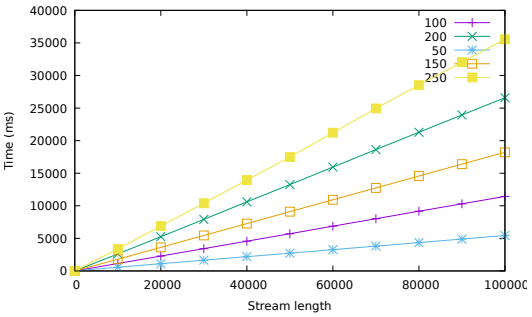
La Figure 5.9b montre le temps d'exécution du modèle de distance de tendance auto corrélé dans la configuration C2. Pour toutes les configurations testées, le temps d'exécution de ce flux de travail semble se comporter à nouveau de manière linéaire dans le nombre total d'événements traités. Cependant, comme prévu, le débit est réduit par rapport à l'utilisation d'un modèle fixe. Ceci est résumé dans le Tableau 5.3, qui compare le débit du modèle de distance de tendance par rapport au modèle de distance de tendance auto corrélé, pour la même taille de fenêtre. Globalement, sur toutes les mesures de tendance, nous avons observé que l'utilisation de l'autocorrélation entraîne un ralentissement d'au plus 46 % pour une largeur de fenêtre de 50 et d'au plus 58 % pour une largeur de fenêtre de 250. Cela correspond à peu près à la réduction de 50 % du débit prévue par notre estimation brute.



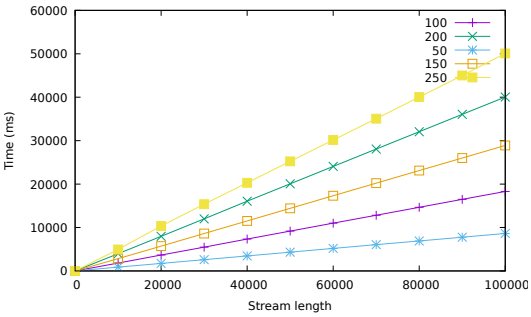
(a) moyenne mobile



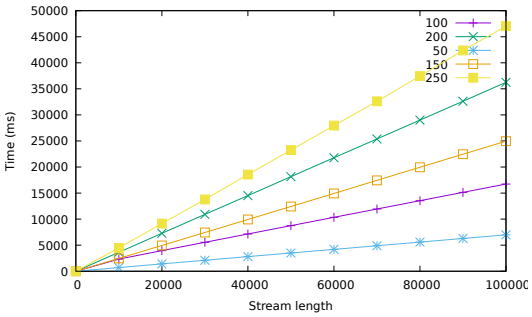
(b) Vecteur de moments



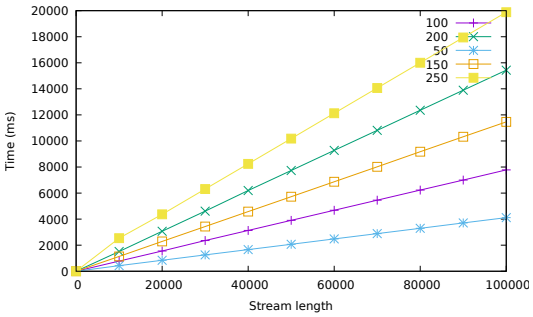
(c) Distribution de symbôles



(d) Frequences de symbôles



(e) Ensemble de N-gramme



(f) Durée moyenne d'une slice

FIGURE 5.9 : Temps de calcul cumulé sur un flux d'événements, pour différents processeurs de tendance et largeurs de fenêtre. ©Massiva Roudjane

Encore une fois, le débit global est influencé négativement par la largeur des fenêtres elles-mêmes, comme l'illustre la Figure 5.10. Nous observons la même tendance que pour le modèle de distance de tendance, bien qu'avec des débits globaux inférieurs.

Fonction de tendance	Distance de tendance	Distance de tendance auto corrélée
Cluster le plus proche	18103.006	11583.574
N-grams	24085.02	14314.486
moyenne mobile	49044.14	26219.455
Moments courants	23981.055	14706.029
Distribution de symbôles	28563.553	18399.447

TABLEAU 5.3 : Comparaison du débit (en Hz) pour la distance de tendance statique (STD) par rapport au flux de travail de distance de tendance auto corrélée (SCTD), pour chaque calcul de tendance. La largeur de la fenêtre pour toutes les expériences est de 200.©Massiva Roudjane

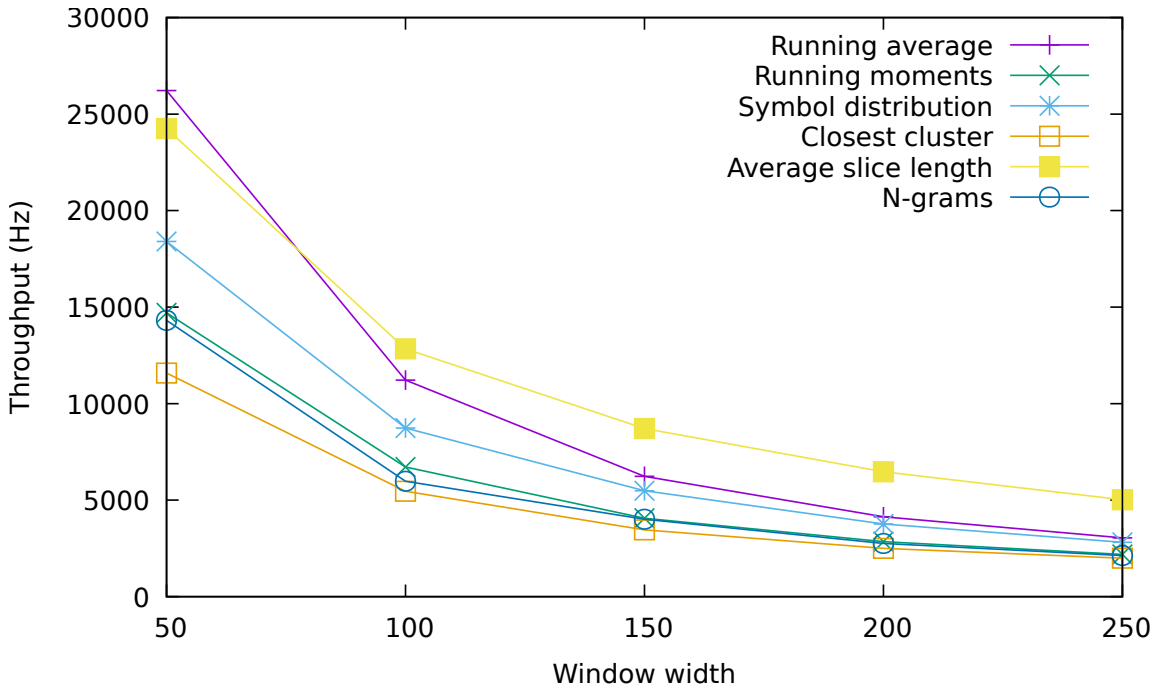


FIGURE 5.10 : Impact de la largeur de la fenêtre sur le débit, pour divers calculs de distance de tendance auto corrélés. ©Massiva Roudjane

5.3.3 EXPÉRIENCES D'EXTRACTION DE TENDANCE

Une dernière série d'expériences implique l'évaluation empirique du flux de travail d'extraction de tendance. Cette fois, pour un calcul de tendance donné (β) et un algorithme d'agrégation (α), les paramètres qui ont un impact sur le temps de calcul sont le nombre de journaux préenregistrés n_ℓ sur lesquels calculer une tendance, ainsi que la longueur ℓ de ces journaux. Le temps d'exécution de β doit être proportionnel à la fois à n_ℓ et ℓ. Cependant, étant donné que l'algorithme d'agrégation se voit attribuer un objet de " tendance " similaire pour chaque journal, son temps d'exécution ne devrait dépendre que de n_ℓ.

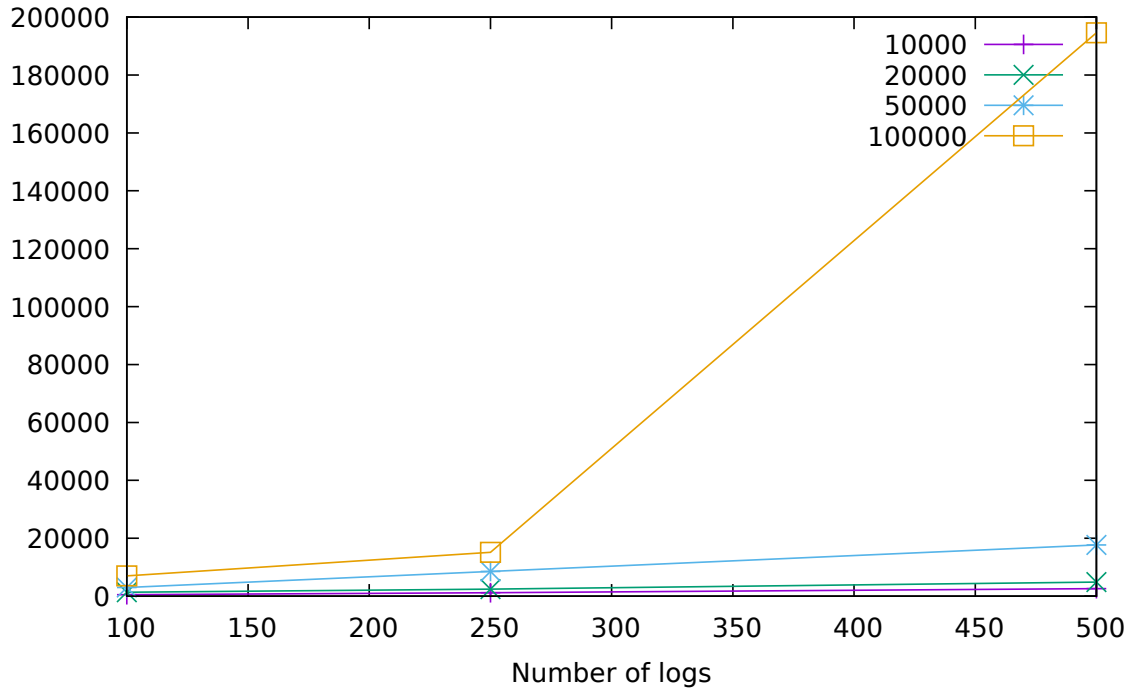


FIGURE 5.11 : Impact du nombre de journaux pour le workflow d'extraction de tendance, en utilisant la distribution des symboles comme calcul de tendance (β) et l'algorithme de clustering k-means comme calcul d'agrégation (α). Chaque série de données représente une longueur différente donnée à chaque journal de l'ensemble (de 10 000 événements à 50 000 événements par journal).©Massiva Roudjane

La Figure 5.11 montre comment le nombre de journaux affecte le temps d'exécution du workflow d'extraction de tendance, dans le cas où le calcul de tendance est la distribution de symboles et le calcul d'agrégation est l'algorithme de clustering k-means mentionné précédemment (fourni par la bibliothèque Apache Commons Math mentionnée précédemment). Comme on peut le voir, le nombre de journaux utilisés comme base de calcul et la longueur de chaque journal contribuent à la durée du processus d'extraction des tendances. Dans le pire des cas, 2.6 logs de 50 000 événements peuvent toujours être traités via ce flux de travail à chaque seconde. Bien que cela ne soit pas montré ici, il faut mentionner que le temps de calcul d'une tendance pour chaque log domine largement le temps alors pris par la fonction d'agrégation α. En d'autres termes, la majeure partie du travail consiste à extraire une tendance d'un grand ensemble de journaux et non à exécuter l'algorithme de clustering à partir de ces tendances par la suite.

5.4 MISE EN OEUVRE ET EXPÉRIENCES POUR L'ANALYSE PRÉDICTIVE

Les modèles de workflow et les exemples donnés jusqu'à présent ont été décrits au niveau théorique. Cependant, la question reste de savoir si ces modèles peuvent être utilisés dans la pratique. En particulier, les efforts nécessaires pour mettre en œuvre concrètement l'un de ces schémas restent à déterminer. Il en va de même pour la mesure dans laquelle ces modèles sont évolutifs sur des journaux de taille réaliste. Dans cette section, nous décrivons nos efforts pour mettre en œuvre de manière concrète les exemples présentés précédemment et pour mesurer expérimentalement leur efficacité dans diverses conditions.

5.4.1 MODIFICATIONS À PAT THE MINER

Afin de répondre aux questions susmentionnées, on a créé une nouvelle version de Pat The Miner (PTM) avec de nouvelles fonctionnalités. Nous avons implémenté les trois

modèles de flux de travail décrits dans le présent document sous forme d'objets génériques dans Pat The Miner. Pat The Miner utilise la bibliothèque Apache Commons Math pour la manipulation statistique de base et la classification et la bibliothèque Weka [25] pour l'apprentissage automatique.

Une caractéristique intéressante de cette version de Pat The Miner est que l'utilisateur n'a pas à gérer directement les objets de ces deux bibliothèques. Au contraire, ces concepts sont manipulés à un niveau d'abstraction plus élevé en fournissant simplement des paramètres pour instancier correctement les modèles de flux de travail déjà fournis par notre framework. Par exemple, la Figure 5.12 montre un fragment de code qui crée une instance du workflow d'apprentissage prédictif. Cet exemple suppose que les événements sont des quadruplets de nombres, c'est-à-dire $e = (q_0, q_1, q_2, q_3)$. La première ligne définit les valeurs des paramètres m, t et n du workflow. Les lignes 2 à 6 créent un tableau d'objets Weka `Attribute`; pour chaque attribut, son nom et ses valeurs possibles sont définis. Les lignes 7 à 10 créent le processeur de caractéristiques ϕ ; dans ce cas, le processeur prend simplement les éléments aux positions 1 et 2 dans l'événement d'entrée, et les fusionne en un tuple (q_1, q_2) : cela constitue le vecteur caractéristique. Les lignes 11 à 12 définissent de manière similaire le processeur de classe κ; sur un événement donné, la fonction retourne vrai ou faux, selon que $q_3 > 0$.

La fonction de découpage f est définie à la ligne 13. Le flux d'entrée des événements sera séparé en sous-flux en fonction de la valeur q_0 de chaque événement. Les lignes 14 à 15 définissent l'algorithme d'apprentissage à utiliser. Dans ce cas, l'algorithme J48 fourni par la bibliothèque Weka est utilisé (pratiquement tous les autres algorithmes de classification fournis par Weka pourraient être utilisés à la place de celui-ci). Enfin, les lignes 16 à 17 rassemblent tous ces éléments en créant l'instance du workflow d'apprentissage prédictif.

```
int m = 1, t = 1, n = 1;
Attribute[] attributes = new Attribute[] {
  createAttribute("A", 0, 1, 2),
  createAttribute("B", 0, 1, 2),
  createAttribute("C", "true", "false")
};
ApplyFunction phi = new ApplyFunction(new FunctionTree(
  new Bags.ToArray(Number.class, Number.class),
  new NthElement(1),
  new NthElement(2)));
ApplyFunction kappa = new ApplyFunction(new FunctionTree(
  Numbers.isGreaterThan, new NthElement(3), new Constant(0));
Function f = new NthElement(0);
UpdateClassifierFunction uc = new UpdateClassifierFunction(
  new J48(), "test", attributes);
PredictiveLearning ct = new PredictiveLearning(
  f, phi, m, t, kappa, n, uc);
```

FIGURE 5.12 : Définition d'une instance du workflow d'apprentissage prédictif à l'aide de code Java.

Comme on peut le voir, la création d'un flux de travail complet qui calcule un vecteur d'entités sur une fenêtre coulissante, calcule une classe sur une autre fenêtre, envoie ces paires d'entités / classes à un algorithme d'apprentissage automatique et génère la classification résultante, peut être instanciée en moins de dix lignes Java. Bien qu'ils ne soient pas représentés, les deux autres workflows (prédiction statique et prédiction d'autoapprentissage) sont créés de manière similaire.

Étant donné que l'objet `PredictiveLearning` est lui-même un processeur BeepBeep, une fois défini, il peut être connecté en amont et en aval à d'autres processeurs afin d'obtenir une chaîne de traitement de bout en bout. C'est ce qui est montré dans la Figure 5.13. Une chaîne de processeurs est créée, qui lit successivement les lignes de texte d'un fichier, convertit chaque ligne en un tableau de valeurs, envoie ces tableaux dans l'objet `PredictiveLearning` créé dans la Figure 5.12, et stocke le dernier événement produit par cette chaîne dans un évier. (L'objet `Pump` est simplement là pour extraire des lignes de texte du fichier et les pousser

dans la chaîne jusqu'à la fin du fichier.) L'objet `sink` peut alors être interrogé pour obtenir le classificateur qu'il contient, qui pourrait ensuite être enregistré ou réutilisé ailleurs (comme dans une instance du workflow `StaticPrediction`).

```
ReadLines rl = new ReadLines(
  new FileInputStream(new File("file.csv")));
ApplyFunction af = new ApplyFunction(
  Strings.SplitString.instance);
SinkLast sink = new SinkLast();
Pump pump = new Pump();
Connector.connect(rl, af, ct, pump, sink);
pump.run();
```

FIGURE 5.13 : Utilisation du processeur d'apprentissage prédictif pour générer un classificateur à partir d'un journal importé à partir d'un fichier CSV.

5.5 EXPÉRIENCES D'ANALYSE PRÉDICTIVE

Afin d'évaluer la faisabilité d'effectuer des analyses prédictives en temps réel, nous avons conçu un ensemble d'expériences destiné à mesurer le débit des workflows de flux d'événements proposés, pour diverses fonctions, largeurs de fenêtre et algorithmes d'apprentissage. Tout comme pour la section 5.3, toutes les expériences sont contenues dans une instance de l'environnement expérimental LabPal [232]. Tout comme dans le premier cas, une instance de laboratoire téléchargeable contenant toutes les expériences de ce document peut être obtenue en ligne [235].

Dans l'ensemble, nos mesures empiriques impliquent 122 des expériences individuelles, qui ont généré ensemble 355 points de données distincts. Toutes les expériences ont été exécutées sur une Intel CORE i5-7200U 2.5 GHz fonctionnant sous Ubuntu 18.04, à l'intérieur d'une machine virtuelle Java 8 avec 1754 Mo Mo de mémoire.

5.5.1 EXPÉRIENCE DE PRÉDICTION STATIQUE

Le premier ensemble d'expériences mesure le débit des modèles de prédiction statique donnés comme exemples dans le chapitre 4. Dans chaque cas, le flux d'entrée a été généré avec 100000 événements aléatoires, chacun composé d'un identificateur de tranche, d'un horodatage et d'un nom d'événement, et d'une valeur numérique aléatoire. Le nombre de tranches et le nombre d'événements générés pour chaque tranche sont tous deux configurables par l'utilisateur.

m	Régression linéaire	Prédiction moyenne
5	497517.4	393704.72
10	490200.97	271741.84
30	146200.3	105933.266
100	276245.84	25265.54

TABLEAU 5.4 : Débit de prédiction statique en événements par seconde, pour une largeur de fenêtre variable (m), avec 100 tranches. ©Massiva Roudjane

Dans une première série d'expériences, nous avons fait varier le nombre de tranches et la longueur de chaque tranche, pour une largeur de fenêtre fixe. Ces expériences ont révélé une absence d'impact sur le débit en fonction de ces paramètres. Cela peut s'expliquer par le fait que le modèle de prédiction statique entraîne un coût de calcul fixe pour chaque événement ; la tranche à laquelle appartient cet événement n'a pas d'importance. De plus, puisque chaque prédiction est calculée sur une fenêtre d'événement de taille fixe, la longueur de chaque tranche n'a pas d'importance non plus.

Cependant, le débit devrait être affecté négativement par l'augmentation de la taille de la fenêtre. Par conséquent, un autre ensemble d'expériences mesure le débit, cette fois en faisant varier ce paramètre. Le Tableau 5.4 montre le débit obtenu pour les deux exemples de prédiction statique présentés précédemment (régression linéaire et prédiction moyenne). Seules les valeurs agrégées sont affichées. Le lecteur est référé au bundle LabPal pour des

résultats individuels détaillés et des tableaux et graphiques supplémentaires. Dans les deux cas, le nombre de tranches entrelacées dans le journal a été défini à 100, ce qui signifie que chaque tranche contient 1 000 événements. Chaque ligne du tableau représente le débit en calculant la ligne de régression ou la moyenne sur différentes largeurs de fenêtre m.

Comme on peut le constater, la largeur de la fenêtre a une influence sur le débit, qui diminue progressivement à mesure que la largeur de la fenêtre augmente. Néanmoins, pour ces deux fonctions prédictives, on peut encore s'attendre à un débit de l'ordre de dizaines de milliers d'événements par seconde.

5.5.2 EXPÉRIENCES DE L'APPRENTISSAGE PRÉDICTIF

Notre deuxième série d'expériences mesure le débit du modèle d'apprentissage prédictif décrit dans le chapitre 4. Encore une fois, ce débit est mesuré pour chacun des exemples présentés dans cette section, comme le montre le Tableau 5.5. Dans le cas du problème d'apprentissage de classificateur, un vecteur aléatoire de trois entités numériques est calculé sur chaque fenêtre glissante. Le processeur d'apprentissage est configuré pour former un arbre de décision, en utilisant l'algorithme J48 fourni par la bibliothèque Weka.

m	Durée moyenne	Prochain événement le plus probable	Apprentissage du classificateur
1	52938.594	156251.56	4922.0356
2	264552.9	86059.38	5040.8813
3	719431.6	62461.586	4995.554
4	529105.8	61958.49	5043.6777

TABLEAU 5.5 : Débit d'apprentissage prédictif en événements par seconde, pour une largeur de fenêtre variable (m). ©Massiva Roudjane

Cette fois encore, la largeur de la fenêtre de fonction est utilisée comme variable indépendante qui est modifiée. Observons comment des fenêtres plus petites sont utilisées dans cet ensemble d'expériences. Cela est dû au fait que les fonctions prédictives apprises sont plus complexes, et surtout plus sensibles à la largeur de la fenêtre que les simples présentées dans la

section précédente. En particulier, le problème d'événement suivant le plus probable accumule des *m*-grammes de noms d'événements successifs. Le nombre de *m*-grammes possibles croît exponentiellement avec *m*, qui est précisément la largeur de la fenêtre.

Ces expériences révèlent sans surprise que le débit est affecté négativement par la largeur de la fenêtre. Ils montrent également le coût de calcul plus élevé qu'entraîne l'introduction d'un algorithme d'apprentissage automatique à la place des fonctions plus simples utilisées par les autres problèmes. Il convient de noter, cependant, que dans la mise en œuvre actuelle du modèle d'apprentissage prédictif, une nouvelle fonction prédictive est calculée à chaque nouvel événement d'entrée. Si son objectif est simplement de calculer une fonction prédictive hors ligne, à partir d'un ensemble de journaux préexistants, alors seule la dernière sortie de ce type est nécessaire : cela correspond à la fonction prédictive qui prend en compte l'ensemble complet des événements d'entrée. Dans un tel cas, les paires d'entités / classes pourraient être accumulées en mémoire et la fonction prédictive ne serait calculée qu'une seule fois sur la base de l'ensemble d'apprentissage. Bien qu'il ne soit pas implémenté, ce mode de traitement hors ligne entraînerait un débit beaucoup plus élevé, en particulier pour les classificateurs d'apprentissage automatique.

5.5.3 EXPÉRIENCES D'APPRENTISSAGE AUTO CORRÉLÉ

La dernière série d'expériences mesure le débit du modèle d'auto-apprentissage. Étant donné que la composante « apprentissage » de ce modèle est identique à l'apprentissage prédictif, les mêmes problèmes et paramètres d'entrée peuvent être utilisés pour mesurer le débit. Ceci est indiqué dans le Tableau 5.6. L'observation globale qui peut être extraite est que le débit suit une tendance similaire à celle du workflow d'apprentissage prédictif et est quelque peu plus lent. Cela s'explique par le fait qu'un travail supplémentaire est effectué dans ce flux de travail, par rapport au précédent.

m	Durée moyenne	Prochain événement le plus probable	Apprentissage du classificateur
1	275484.84	36697.613	4883.8154
2	202841.78	37750.473	4620.4775
3	238097.62	26205.713	4911.158
4	263854.88	25163.814	4675.1284

TABLEAU 5.6 : Débit d'auto-apprentissage, pour une largeur de fenêtre variable (m). ©Massiva Roudjane

5.6 CONCLUSION

Des dizaines d'expériences, mesurant les performances des flux de travail pour des scénarios et des paramètres divers, ont été effectuées et présentées . On tire de ces résultats les conclusions suivants :

— BeepBeep est une plateforme appropriée pour mettre en œuvre tous les workflows proposés. Avec les classes développées, instancier un de ces workflows peut se faire en une dizaine de lignes de code.

— L'augmentation de la taille de la fenêtre a un impact négatif sur le débit dans les fonctions de distance de tendance et du modèle d'apprentissage auto corrélé. La taille de la mémoire est linéairement influencée par la largeur de la fenêtre étant donné qu'elle contient principalement les processeurs et la fenêtre elle-même et les événements qu'elle contient.

— Les expériences du second ordre ont montré que le nombre de processeurs de premier ordre influence négativement le débit avec une tendance qui suit l'inverse du nombre de tendances de premier ordre utilisées.

— Les résultats de l'évaluation du flux de travail d'extraction de tendance ont montré que le nombre de journaux préenregistrés, ainsi que la longueur de ces journaux sont les paramètres qui ont plus influencé le débit.

— Le modèle de prédiction statique est influencé par la largeur de la fenêtre. Le débit diminue, dans ces expériences, progressivement à mesure que la largeur de la fenêtre augmente.

— Les expériences de l'apprentissage prédictif révèlent que le débit est affecté négativement par la largeur de la fenêtre. Le coût de calcul est aussi plus élevé avec l'introduction d'un algorithme d'apprentissage. On a également vu que les fonctions prédictives apprises sont plus complexes et plus sensibles à la largeur de la fenêtre. Le modèle auto corrélé a une tendance similaire à celle du workflow d'apprentissage prédictif et est un peu plus lent dû au traitement additionnel effectué dans ce flux.

On a également montré que BeepBeep offre des performances comparables à celles de R, un outil statistique stable et largement utilisé. Les expériences ont délibérément été réalisées sur un ordinateur ayant des caractéristiques standards, afin de démontrer que même avec un ordinateur personnel, avec les paramètres testés, on peut s'attendre à des traitements de l'ordre du millier d'événements.

CONCLUSION

Les journaux d'événements sont en expansion ; les traiter adéquatement est une nécessité pour avoir une vision plus claire des processus sous-jacents. On a vu que la problématique de l'extraction de l'information des logs n'est pas totalement résolue. D'une part, on a des outils à notre disposition pour faire des traitements avancés sur différents types de données, mais pas sur des logs d'événements. D'autre part, on possède plusieurs outils pour traiter ces logs. Toutefois, les traitements qu'ils peuvent fournir sont très limités.

Ce travail se situe donc à l'intersection de ces deux domaines. Tout au long de cette thèse, on a essayé de répondre aux deux questions suivantes : comment peut-on extraire de l'information des logs d'événements qui arrivent en temps réel en utilisant des algorithmes de data mining ? Est-il possible de prédire les tendances d'un flux d'événements en temps réel ?

EXTRACTION DE TENDANCES ET ANALYSE PRÉDICTIVE

Dans [67, 236], nous avons étudié le concept de *tendance* dans un journal d'événements et nous nous sommes concentrés sur la tâche de détecter les *déviations* de tendance dans un journal en temps réel. À cette fin, nous avons introduit trois flux de travail de traitement génériques : les flux de travail de distance de tendance statiques et auto corrélés, et le flux de travail d'extraction de tendance. Tous les trois décrivent un processus général, étape par étape, pour transformer les flux d'événements. Des types spécifiques de calculs sont obtenus en donnant différentes définitions aux divers paramètres acceptés par ces flux de travail.

Nous avons vu, à travers des exemples, comment diverses tâches courantes de traitement du journal deviennent des cas particuliers d'un de ces flux de travail. Celles-ci incluent de nombreuses mesures statistiques descriptives, des distributions et même certaines opérations d'exploration de données telles que le clustering. Grâce au caractère générique de ces flux de travail, tout calcul peut en principe être utilisé pour le calcul de tendance. De plus, nous

avons vu comment une implémentation des workflows en tant qu'extension du moteur de flux d'événements BeepBeep permet de calculer les tendances et de détecter les violations de manière continue. Les résultats expérimentaux montrent que, pour certains types de tendances, on peut s'attendre à une vitesse de traitement de l'ordre de milliers d'événements par seconde avec du matériel standard.

Dans la deuxième partie de ce travail, nous avons présenté un cadre générique permettant de calculer différents types de prévisions en fonction d'un journal d'événements de processus métier. Dans ce contexte, chaque journal peut contenir plusieurs instances d'un processus dont les événements sont entrelacés dans un seul flux global. Nous avons d'abord introduit le modèle du workflow de prédiction statique, qui sépare le journal en un sous-flux pour chaque instance de processus et calcule une prédiction en fonction du contenu d'une fenêtre glissante d'événements. Ensuite, le workflow d'apprentissage prédictif a montré comment une fonction peut être calculée (apprise) à partir d'instances précédentes du processus, puis être utilisée comme prédicteur pour les instances futures. Cela peut être fait, entre autres, en exécutant un algorithme d'apprentissage automatique sur un vecteur de caractéristiques calculé sur une fenêtre glissante d'événements. Enfin, la prédiction auto corrélée associe les deux modèles précédents et tente de calculer une prédiction sur les événements futurs, en fonction de ce qui a été appris sur les événements passés du même journal.

Comme nous l'avons vu, tous ces modèles sont génériques et peuvent correspondre à différents types de prévisions en fonction des paramètres précis qui leur sont attribués. De plus, en les mettant en implémentant en tant que chaînes de processeurs BeepBeep, tous ces modèles peuvent fonctionner sur des journaux préenregistrés, ainsi qu'en temps réel. Les résultats expérimentaux montrent que, dans des conditions raisonnables, ces modèles peuvent être exécutés sur des journaux de taille réaliste et traiter des milliers d'événements par seconde.

LIMITES ET TRAVAUX FUTURS

Les idées présentées dans ce travail ouvrent la voie à de multiples extensions et soulèvent également de nombreuses questions de recherche intéressantes. Premièrement, sur le plan technique, nous étudions la possibilité d'intégrer Pat The Miner en tant que plug-in pour la plate-forme ProM. De plus, puisque la bibliothèque est écrite en Java, elle pourrait facilement incorporer des fonctionnalités d'exploration de données de bibliothèques Java existantes, telles que WEKA.

Deuxièmement, les calculs de tendance utilisant moins de fonctions «traditionnelles» que les vecteurs de caractéristiques doivent être explorés. Grâce au grand nombre de plugins existants disponibles pour BeepBeep, on peut imaginer des calculs de tendance impliquant un mélange de statistiques, de méthodes formelles et de traitement du signal. Troisièmement, comme nous l'avons déjà mentionné, les calculs de tendance présentés dans ce travail peuvent être considérés comme un parent éloigné des techniques et outils d'extraction de processus existants. Il serait très intéressant d'examiner les conséquences de la création d'un modèle de processus, reconstruit à partir d'un ensemble de journaux, en tant que tendance de référence sur laquelle les mesures de distance pourraient être définies.

Plusieurs variantes des workflows de tendances elles-mêmes pourraient également être proposées, par exemple en faisant varier l'emplacement relatif des fenêtres par rapport au flux, en comparant les tendances calculées sur deux flux en temps réel les unes par rapport aux autres, ou en permettant le calcul de tendances multimodales à l'intérieur le flux de travail d'autocorrélation. Une piste de recherche intéressante pourrait également impliquer le concept d'*analyse prédictive*. Dans un tel paramètre, les fonctionnalités d'une fenêtre d'événements sont utilisées pour entraîner une fonction de classificateur afin d'apprendre une étiquette de catégorie calculée sur une autre fenêtre placée plus loin dans le futur. Ce classificateur peut ensuite être utilisé pour prédire une étiquette en l'exécutant sur une fenêtre d'événements en

cours. Un tel flux de travail pourrait facilement être mis en œuvre en tant que chaîne générique de processeurs et est en cours de développement pour une future publication.

L'utilisation des flux de travail de tendance définis dans le présent document soulève également un certain nombre de problèmes techniques. Premièrement, nous avons vu que l'impact de la largeur de la fenêtre sur le débit des calculs de tendance peut être problématique pour les très grandes fenêtres. Une première possibilité consisterait à étudier les fenêtres dites «à sauts», qui glisseraient à travers un flux à une étape plus large que chaque événement. De plus, le coût de calcul des fenêtres pourrait également être atténué par le recours au parallélisme. Il s'avère que le moteur de traitement de flux d'événements BeepBeep permet l'exécution de certains types de calcul en parallèle lorsque la machine hôte dispose d'un processeur multithread [237]. Cependant, ce processus nécessite beaucoup de peaufinage pour être réellement bénéfique. L'insertion du *multi-threading* dans les processeurs de distance de tendance et l'étude de son impact sur le débit global sont planifiées comme travaux futurs.

Plusieurs questions restent également sans réponse concernant les flux de travail eux-mêmes. Par exemple, l'autocorrélation peut souffrir de la «vision tunnel», dans laquelle un changement de tendance très progressif et prolongé peut passer inaperçu si les fenêtres comparées sont trop petites et n'est donc jamais suffisamment différent pour déclencher une alarme. De plus, le cas où un journal est séparé en plusieurs sections, chacune suivant une tendance différente, n'est pas correctement pris en compte par nos modèles de flux de travail. Des questions de l'ordre de *correctness* et *precision* du modèle se posent également. Dans l'ensemble, les premiers résultats obtenus en utilisant ces flux de travail sont néanmoins prometteurs et pourraient s'avérer utiles dans un large éventail de cas d'utilisation impliquant des journaux de différentes sortes.

Enfin, bien que les modèles de flux de travail proposés soient génériques, ils peuvent toujours représenter un ensemble limité de toutes les tâches de détection de tendance possibles.

Par exemple, le modèle de flux de travail de distance de tendance statique peut fonctionner sur des types d'événements arbitraires, et le processeur de tendance β peut, en théorie, être composé de n'importe quel morceau de code Turing-complet : le flux de travail ne fait aucune hypothèse sur le calcul réellement effectué dans cette boîte. Pourtant, ce calcul doit encore se faire sur une fenêtre glissante constituée d'un nombre fixe d'événements. Cela exclut les fonctionnalités informatiques qui nécessiteraient une connaissance de l'ensemble du journal, tel que la saisonnalité dans un modèle de série chronologique. À cet égard, la mise en œuvre d'une nouvelle technique appelée le profil matriciel [238], qui prend en charge les mises à jour incrémentielles de son modèle, pourrait être un moyen d'introduire l'analyse des séries chronologiques dans notre modèle proposé.

Dans l'ensemble, les premiers résultats obtenus à l'aide de ces flux de travail sont néanmoins prometteurs et pourraient s'avérer utiles dans un large éventail de cas d'utilisation impliquant des journaux de différents types.

Plusieurs questions et développements ultérieurs de ce cadre de base doivent être abordés. Le premier est la question des métadonnées. Nous appelons métadonnées toute structure arbitraire associée à la tranche dans son ensemble, mais n'appartenant à aucun événement particulier. Par exemple, dans un journal d'un business process impliquant plusieurs clients, de telles métadonnées pourraient fournir des informations sur les clients : leur âge, leur revenu ou toute autre donnée externe qui ne se produit pas à un moment donné et ne peut donc pas être considéré comme un «événement». Actuellement, les modèles de workflow que nous fournissons ne gèrent pas les métadonnées de manière élégante.

La solution proposée laisse également en suspens plusieurs questions typiques de tout problème d'apprentissage automatique : quelles sont les caractéristiques pertinentes à calculer dans une fenêtre d'événements ? Quelle est la classe que nous essayons de prédire et comment devrait-elle être calculée ? Quelle est la largeur à donner aux différentes fenêtres du workflow ?

Quel algorithme d'apprentissage faut-il utiliser et avec quels paramètres ? . Une réponse appropriée à ces questions nécessite une bonne connaissance du cas d'utilisation sous-jacent et une certaine dose d'intuition. Sans surprise, il n'existe pas de solution unique. Cependant, nous proposons dans le présent travail une solution et en recadrant le problème de prévision en tant que workflow de base avec une poignée de paramètres à définir, le présent document et la mise en œuvre qu'il décrit facilitent, selon nous, l'exploration de différentes approches par les utilisateurs.

IMPACT ET RETOMBÉES

Depuis la publication des articles résultant des travaux de cette thèse, d'autres chercheurs ont développé des systèmes suivant les mêmes principes.

La plateforme Palisade [239] s'est inspirée des résultats exposés dans l'article *Real-Time Data Mining for Event Streams* [67] pour faire de la détection d'anomalies. Les résultats présentés dans cet article démontrent la performance de l'approche proposée dans cette thèse et la facilité de répliquer le workflow générique proposé.

La plateforme de Truong et al. [240] propose une architecture pour doter les plateformes de données en streaming avec des modèles d'apprentissage prédictif et de modèles de traitement d'événements complexes. Ils suggèrent l'utilisation de machine learning dans un pipeline de streams. Les auteurs ont cité l'article *Predictive Analytics for Event Stream Processing* [68] comme étant le premier travail ayant été réalisé dans ce sens.

L'état de l'art sur Automatisation de la découverte de l'architecture d'entreprise basée sur l'exploration d'événements à partir des journaux API Gateway [241] a cité les papiers *Real-Time Data Mining for Event Streams* [67] et *Predictive Analytics for Event Stream Processing* [68].

Ces citations récentes des publications tirées de cette thèse révèlent l'intérêt à développer des modèles génériques pour la prédiction et la détection d'anomalie dans les flux d'événements.

BIBLIOGRAPHIE

[1] C. Gunther et H. Verbeek, *XES - standard definition*, ser. BPM reports. BPMcenter. org, 2014.

[2] U. L. Richard Sharpe, Ed Warnicke, "Wireshark user's guide," 2004. [En ligne]. Repéré à : https://www.wireshark.org/docs/wsug_html/

[3] S. Varvaressos, K. Lavoie, A. B. Massé, S. Gaboury, et S. Hallé, "Automated bug finding in video games : A case study for runtime monitoring," dans *Seventh IEEE International Conference on Software Testing, Verification and Validation, ICST 2014, March 31 2014-April 4, 2014, Cleveland, Ohio, USA*, 2014, pp. 143–152. [En ligne]. Repéré à : https://doi.org/10.1109/ICST.2014.27

[4] W. M. P. van der Aalst, *Process Mining*. Springer, 2011.

[5] F. Chen, M. d'Amorim, et G. Rosu, "A formal monitoring-based framework for software development and analysis," dans *Formal Methods and Software Engineering, 6th International Conference on Formal Engineering Methods, ICFEM 2004, Seattle, WA, USA, November 8-12, 2004, Proceedings*, 2004, pp. 357–372. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-540-30482-1_31

[6] S. Hallé, "From complex event processing to simple event processing," *CoRR*, vol. abs/1702.08051, 2017. [En ligne]. Repéré à : http://arxiv.org/abs/1702.08051

[7] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, et S. B. Zdonik, "The design of the borealis stream processing engine," dans *CIDR 2005, Second Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2005, Online Proceedings*, 2005, pp. 277–289. [En ligne]. Repéré à : http://cidrdb.org/cidr2005/papers/P23.pdf

[8] F. Michel, "Flickr-statistics," https://github.com/frmichel/flickr-statistics, (consulté le 2 févr. 2020).

[9] J. Clement, "Number of monthly active instagram users 2013-2018," https://www.statista.com/statistics/253577/number-of-monthly-active-instagram-users/, (consulté le 2 févr. 2020).

[10] ——, “Most popular social networks worldwide as of october 2019, ranked by number of active users,” https://www.statista.com/statistics/272014/global-social-networka-ranked-by-number-of-users/, (consulté le 2 févr. 2020).

[11] IDC, “The digital universe of opportunities : Rich data and the increasing value of the internet of things,” https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm, (consulté le 2 avril 2020).

[12] SolarWinds, “Amazing Facts and Figures About the Evolution of Hard Disk Drives,” https://royal.pingdom.com/amazing-facts-and-figures-about-the-evolution-of-hard-disk-drives/, (consulté le 2 févr. 2020).

[13] J. Silva, J. R. Borré, A. P. P. Castillo, L. Castro, et N. Varela, “Integration of data mining classification techniques and ensemble learning for predicting the export potential of a company,” dans *The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops, April 29 - May 2, 2019, Leuven, Belgium*, ser. Procedia Computer Science, E. M. Shakshuki et A. Yasar, éds., vol. 151. Elsevier, 2019, pp. 1194–1200. [En ligne]. Repéré à : https://doi.org/10.1016/j.procs.2019.04.171

[14] B. Radhakrishnan, G. Shineraj, et K. Anver Muhammed, “Application of data mining in marketing,” *CoRR*, vol. abs/1310.8462, 2013. [En ligne]. Repéré à : http://arxiv.org/abs/1310.8462

[15] F. Sun, Y. Pan, J. White, et A. Dubey, “Real-time and predictive analytics for smart public transportation decision support system,” dans *2016 IEEE International Conference on Smart Computing, SMARTCOMP 2016, St Louis, MO, USA, May 18-20, 2016*. IEEE Computer Society, 2016, pp. 1–8. [En ligne]. Repéré à : https://doi.org/10.1109/SMARTCOMP.2016.7501714

[16] A. Berry et Z. Milosevic, “Real-time analytics for legacy data streams in health : Monitoring health data quality,” dans *17th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2013, Vancouver, BC, Canada, September 9-13, 2013*, 2013, pp. 91–100. [En ligne]. Repéré à : https://doi.org/10.1109/EDOC.2013.19

[17] M. S. Amin, Y. K. Chiam, et K. D. Varathan, “Identification of significant features and data mining techniques in predicting heart disease,” *Telematics and Informatics*, vol. 36, pp. 82–93, 2019. [En ligne]. Repéré à : https://doi.org/10.1016/j.tele.2018.11.007

[18] M. S. Gerber, “Predicting crime using twitter and kernel density estimation,” *Decision Support Systems*, vol. 61, pp. 115–125, 2014. [En ligne]. Repéré à : https://doi.org/10.1016/j.dss.2014.02.003

[19] S. Suh, *Practical applications of data mining*. Jones & Bartlett Publishers, 2012.

[20] C. C. Aggarwal, *Data mining : the textbook*. Springer, 2015.

[21] M. Bramer, *Principles of data mining*. Springer, 2007, vol. 180.

[22] B. Liu, *Web data mining : exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

[23] RapidMiner, https://rapidminer.com/products/studio/, (consulté le 2 févr. 2020).

[24] P. B. Cerrito, *Introduction to data mining using SAS Enterprise Miner*. SAS Institute, 2006.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, et I. H. Witten, “The WEKA data mining software : an update,” *ACM SIGKDD explorations newsletter*, vol. 11, n^{o} 1, pp. 10–18, 2009.

[26] S. Europe, “Statistica software,” https://www.statsoft.de/en/statistica/statistica-software, (consulté le 2 févr. 2020).

[27] Project R, https://www.r-project.org/, (consulté le 26 mai 2019).

[28] Staffware, *Workflow Patterns according to Staffware*. Staffware, 2018, http://www.workflowpatterns.com/vendors/documentation/vc_staffware.pdf.

[29] S. K. Sarin, “Workflow and data management in inconcert,” dans *ICDE 1996*, 1996, pp. 497–499. [En ligne]. Repéré à : https://doi.org/10.1109/ICDE.1996.492199

[30] H. M. W. E. Verbeek, A. Hirnschall, et W. M. P. van der Aalst, “XRL/Flower : Supporting inter-organizational workflows using XML/Petri-net technology,” dans *WES 2002*, 2002,

pp. 93–108. [En ligne]. Repéré à : https://doi.org/10.1007/3-540-36189-8_8

[31] P. M. Catt, “Research note : The theory and practice of sap’s ERP forecasting functionality,” *J. Enterprise Inf. Management*, vol. 21, n° 5, pp. 512–524, 2008. [En ligne]. Repéré à : https://doi.org/10.1108/17410390810904265

[32] P. Kanhere et H. K. Khanuja, “A methodology for outlier detection in audit logs for financial transactions,” dans *2015 International Conference on Computing Communication Control and Automation*, 2015, pp. 837–840.

[33] Apache, “Apache, http server project,” https://httpd.apache.org/, (consulté le 5 févr. 2020).

[34] H. Walch, *Practical IIS : Internet Information Services in the Data Center and the Cloud*. Apress, 2014.

[35] J. Olivain et J. Goubault-Larrecq, “The orchids intrusion detection tool,” dans *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, ser. Lecture Notes in Computer Science, K. Etessami et S. K. Rajamani, éds., vol. 3576. Springer, 2005, pp. 286–290. [En ligne]. Repéré à : https://doi.org/10.1007/11513988_28

[36] M. Roesch, “Snort : Lightweight intrusion detection for networks,” dans *Proceedings of the 13th Conference on Systems Administration (LISA-99), Seattle, WA, USA, November 7-12, 1999*, D. W. Parter, éd. USENIX, 1999, pp. 229–238. [En ligne]. Repéré à : http://www.usenix.org/publications/library/proceedings/lisa99/roesch.html

[37] M. Desnoyers et M. R. Dagenais, “The lttng tracer : A low impact performance and behavior monitor for gnu/linux,” dans *OLS (Ottawa Linux Symposium)*, vol. 2006. Citeseer, 2006, pp. 209–224.

[38] B. Desmond, J. Richards, R. Allen, et A. G. Lowe-Norris, *Active Directory : Designing, Deploying, and Running Active Directory*. O’Reilly, 2013.

[39] A. B. Cheikh, Y. Blein, S. Chehida, G. Vega, Y. Ledru, et L. du Bousquet, “An environment for the ParTraP trace property language (tool demonstration),” dans *Runtime Verification - 18th International Conference, RV 2018, Limassol, Cyprus, November 10-13, 2018, Proceedings*, ser. Lecture Notes in Computer Science, C. Colombo et

M. Leucker, éds., vol. 11237. Springer, 2018, pp. 437–446. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-030-03769-7_26

[40] H. Nguyen, B. Acharya, R. Ivanov, A. Haeberlen, L. T. X. Phan, O. Sokolsky, J. Walker, J. Weimer, W. H. III, et I. Lee, "Cloud-based secure logger for medical devices," dans *Proceedings of the First IEEE International Conference on Connected Health : Applications, Systems and Engineering Technologies, CHASE 2016, Washington, DC, USA, June 27-29, 2016*. IEEE Computer Society, 2016, pp. 89–94. [En ligne]. Repéré à : https://doi.org/10.1109/CHASE.2016.48

[41] E. Lee, Y. Jang, D. Yoon, J. Jeon, S. Yang, S. Lee, D. Kim, P. P. Chen, A. Guitart, P. Bertens, Á. Periáñez, F. Hadiji, M. Müller, Y. Joo, J. Lee, I. Hwang, et K. Kim, "Game data mining competition on churn prediction and survival analysis using commercial game log data," *IEEE Trans. Games*, vol. 11, n° 3, pp. 215–226, 2019. [En ligne]. Repéré à : https://doi.org/10.1109/TG.2018.2888863

[42] S. Varvaressos, K. Lavoie, S. Gaboury, et S. Hallé, "Automated bug finding in video games : A case study for runtime monitoring," *ACM Computers in Entertainment*, vol. 15, n° 1, 2017.

[43] H. Wu, B. Salzberg, et D. Zhang, "Online event-driven subsequence matching over financial data streams," dans *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, G. Weikum, A. C. König, et S. Deßloch, éds. ACM, 2004, pp. 23–34. [En ligne]. Repéré à : https://doi.org/10.1145/1007568.1007574

[44] S. S. Sabry, N. M. Kaittan, et I. Majeed, "The road to the blockchain technology : Concept and types," *Periodicals of Engineering and Natural Sciences*, vol. 7, n° 4, pp. 1821–1832, 2019.

[45] A. M. Leadbetter, D. Smyth, R. Fuller, E. O'Grady, et A. Shepherd, "Where big data meets linked data : Applying standard data models to environmental data streams," dans *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, J. Joshi, G. Karypis, L. Liu, X. Hu, R. Ak, Y. Xia, W. Xu, A. Sato, S. Rachuri, L. H. Ungar, P. S. Yu, R. Govindaraju, et T. Suzumura, éds. IEEE Computer Society, 2016, pp. 2929–2937. [En ligne]. Repéré à : https://doi.org/10.1109/BigData.2016.7840943

[46] S. Ayhan, J. Pesce, P. Comitz, D. Sweet, S. Bliesner, et G. Gerberick, “Predictive analytics with aviation big data,” dans *2013 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE, 2013, pp. 1–13.

[47] N. Panagiotou, I. Katakis, et D. Gunopulos, “Detecting events in online social networks : Definitions, trends and challenges,” dans *Solving Large Scale Learning Tasks. Challenges and Algorithms - Essays Dedicated to Katharina Morik on the Occasion of Her 60th Birthday*, ser. Lecture Notes in Computer Science, S. Michaelis, N. Piatkowski, et M. Stolpe, éds., vol. 9580. Springer, 2016, pp. 42–84. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-319-41706-6_2

[48] W. M. P. van der Aalst, T. Weijters, et L. Maruster, “Workflow mining : Discovering process models from event logs,” *IEEE Trans. Knowl. Data Eng.*, vol. 16, n° 9, pp. 1128–1142, 2004. [En ligne]. Repéré à : https://doi.org/10.1109/TKDE.2004.47

[49] R. Agrawal, C. M. Johnson, J. Kiernan, et F. Leymann, “Taming compliance with sarbanes-oxley internal controls using database technology,” dans *ICDE 2006*, L. Liu, A. Reuter, K. Whang, et J. Zhang, éds. IEEE Computer Society, 2006, p. 92. [En ligne]. Repéré à : https://doi.org/10.1109/ICDE.2006.155

[50] D. Zong, G. Mao, et X. Wu, “An intrusion detection model based on mining data streams,” dans *DMIN 2008*, 2008, pp. 398–403.

[51] C. Sun, H. Zhang, J.-G. Lou, H. Zhang, Q. Wang, D. Zhang, et S.-C. Khoo, “Querying sequential software engineering data,” dans *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*. Hong Kong, China : ACM Press, 2014, pp. 700–710. [En ligne]. Repéré à : http://dl.acm.org/citation.cfm?doid=2635868.2635902

[52] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, et S. B. Zdonik, “Monitoring streams - A new class of data management applications,” dans *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*. Morgan Kaufmann, 2002, pp. 215–226. [En ligne]. Repéré à : http://www.vldb.org/conf/2002/S07P02.pdf

[53] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, et J. Widom, “Stream : The stanford data stream management system,” Stanford InfoLab, Technical Report 2004-20, 2004. [En ligne]. Repéré à : http://ilpubs.stanford.edu:8090/641/

[54] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, et M. A. Shah, "TelegraphCQ : Continuous dataflow processing for an uncertain world," dans *CIDR*, 2003.

[55] E. Wu, Y. Diao, et S. Rizvi, "High-performance complex event processing over streams," dans *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, S. Chaudhuri, V. Hristidis, et N. Polyzotis, éds. ACM, 2006, pp. 407–418. [En ligne]. Repéré à : http://doi.acm.org/10.1145/1142473.1142520

[56] G. Reger, H. C. Cruz, et D. E. Rydeheard, "Marq : Monitoring at runtime with QEA," dans *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, ser. Lecture Notes in Computer Science, C. Baier et C. Tinelli, éds., vol. 9035. Springer, 2015, pp. 596–610. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-662-46681-0_55

[57] B. D'Angelo, S. Sankaranarayanan, C. Sánchez, W. Robinson, B. Finkbeiner, H. B. Sipma, S. Mehrotra, et Z. Manna, "LOLA : runtime monitoring of synchronous systems," dans *12th International Symposium on Temporal Representation and Reasoning (TIME 2005), 23-25 June 2005, Burlington, Vermont, USA*. IEEE Computer Society, 2005, pp. 166–174. [En ligne]. Repéré à : http://dx.doi.org/10.1109/TIME.2005.26

[58] C. Colombo, G. J. Pace, et G. Schneider, "LARVA — safer monitoring of real-time java programs (tool paper)," dans *Seventh IEEE International Conference on Software Engineering and Formal Methods, SEFM 2009, Hanoi, Vietnam, 23-27 November 2009*, 2009, pp. 33–37. [En ligne]. Repéré à : https://doi.org/10.1109/SEFM.2009.13

[59] H. Barringer, D. E. Rydeheard, et K. Havelund, "Rule systems for run-time monitoring : from eagle to ruler," *J. Log. Comput.*, vol. 20, n° 3, pp. 675–706, 2010. [En ligne]. Repéré à : https://doi.org/10.1093/logcom/exn076

[60] J. Claiborne et A. Gupta, "Machine learning classifiers for predicting transit fraud," dans *24th Americas Conference on Information Systems, AMCIS 2018, New Orleans, LA, USA, August 16-18, 2018*. Association for Information Systems, 2018. [En ligne]. Repéré à : https://aisel.aisnet.org/amcis2018/DataScience/Presentations/37

[61] I. Ullah, B. Raza, A. K. Malik, M. Imran, S. ul Islam, et S. W. Kim, "A churn prediction

model using random forest : Analysis of machine learning techniques for churn prediction and factor identification in telecom sector," *IEEE Access*, vol. 7, pp. 60 134–60 149, 2019. [En ligne]. Repéré à : https://doi.org/10.1109/ACCESS.2019.2914999

[62] A. Bhowmick, T. Abdou, et A. Bener, "Predictive analytics in healthcare epileptic seizure recognition," dans *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, CASCON 2018, Markham, Ontario, Canada, October 29-31, 2018.*, I. Onut, A. Jaramillo, G. Jourdan, D. C. Petriu, et W. Chen, éds. ACM, 2018, pp. 323–330. [En ligne]. Repéré à : https://dl.acm.org/citation.cfm?id=3291327

[63] Y. Lu, R. Krüger, D. Thom, F. Wang, S. Koch, T. Ertl, et R. Maciejewski, "Integrating predictive analytics and social media," dans *2014 IEEE Conference on Visual Analytics Science and Technology, VAST 2014, Paris, France, October 25-31, 2014*, M. Chen, D. S. Ebert, et C. North, éds. IEEE Computer Society, 2014, pp. 193–202. [En ligne]. Repéré à : https://doi.org/10.1109/VAST.2014.7042495

[64] S. M. Idrees, M. A. Alam, et P. Agarwal, "A prediction approach for stock market volatility based on time series data," *IEEE Access*, vol. 7, pp. 17 287–17 298, 2019. [En ligne]. Repéré à : https://doi.org/10.1109/ACCESS.2019.2895252

[65] Y. Liang, Y. Zhang, H. Xiong, et R. K. Sahoo, "Failure prediction in IBM bluegene/l event logs," dans *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*. IEEE Computer Society, 2007, pp. 583–588. [En ligne]. Repéré à : https://doi.org/10.1109/ICDM.2007.46

[66] S. Hallé, *Event Stream Processing with BeepBeep 3 : Log Crunching and Analysis Made Easy*. Presses de l'Université du Québec, 2018.

[67] M. Roudjane, D. Rebaïne, R. Khoury, et S. Hallé, "Real-time data mining for event streams," dans *22nd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2018, Stockholm, Sweden, October 16-19, 2018*. IEEE Computer Society, 2018, pp. 123–134. [En ligne]. Repéré à : https://doi.org/10.1109/EDOC.2018.00025

[68] ——, "Predictive analytics for event stream processing," dans *23rd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2019, Paris, France, October 28-31, 2019*. IEEE, 2019, pp. 171–182. [En ligne]. Repéré à : https://doi.org/10.1109/EDOC.2019.00029

[69] M. Roudjane, D. Rebaïne, R. Khoury, et S. Hallé, “Detecting trend deviations with generic stream processing patterns,” *Inf. Syst.*, vol. 101, p. 101446, 2021. [En ligne]. Repéré à : https://doi.org/10.1016/j.is.2019.101446

[70] P. A. Laplante, *Dictionary of computer science, engineering and technology*. CRC Press, 2000.

[71] B. Alpern et F. B. Schneider, “Recognizing safety and liveness,” *Distributed Computing*, vol. 2, nº 3, pp. 117–126, 1987. [En ligne]. Repéré à : https://doi.org/10.1007/BF01782772

[72] M. Leucker et C. Schallhart, “A brief account of runtime verification,” *J. Log. Algebr. Program.*, vol. 78, nº 5, pp. 293–303, 2009. [En ligne]. Repéré à : https://doi.org/10.1016/j.jlap.2008.08.004

[73] R. H. Dolin, L. Alschuler, C. Beebe, P. V. Biron, S. L. Boyer, D. J. Essin, E. Kimber, T. Lincoln, et J. E. Mattison, “Review : The HL7 clinical document architecture,” *JAMIA*, vol. 8, nº 6, pp. 552–569, 2001. [En ligne]. Repéré à : https://doi.org/10.1136/jamia.2001.0080552

[74] T. Bray et S. J. DeRose, “Extensible markup language (XML) part 2 : Linking,” *World Wide Web Journal*, vol. 2, nº 4, pp. 67–82, 1997. [En ligne]. Repéré à : http://www.w3.org/TR/WD-xml-link-970731

[75] T. Bray, J. Paoli, et C. M. Sperberg-McQueen, “Extensible markup language (XML),” *World Wide Web Journal*, vol. 2, nº 4, pp. 27–66, 1997. [En ligne]. Repéré à : http://www.w3.org/TR/WD-xml-970807

[76] E. Verbeek et W. van der Aalst, “Extensible event stream,” http://www.xes-standard.org/.

[77] X. W. Group *et al.*, “Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams,” *IEEE Std*, vol. 1849, pp. 1–50, 2016.

[78] G. Acampora, A. Vitiello, B. Di Stefano, W. van der Aalst, C. Gunther, et E. Verbeek, “Ieee 1849 : The xes standard : The second ieee standard sponsored by ieee computational intelligence society [society briefs],” *IEEE Computational Intelligence Magazine*, vol. 12, nº 2, pp. 4–8, 2017.

[79] H. Valdivieso, W. L. J. Lee, J. Munoz-Gama, et M. Sepúlveda, “Opyenxes : A complete python library for the extensible event stream standard,” dans *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018*, ser. CEUR Workshop Proceedings, W. M. P. van der Aalst, F. Casati, R. Conforti, M. de Leoni, M. Dumas, A. Kumar, J. Mendling, S. Nepal, B. T. Pentland, et B. Weber, éds., vol. 2196. CEUR-WS.org, 2018, pp. 71–75. [En ligne]. Repéré à : http://ceur-ws.org/Vol-2196/BPM_2018_paper_15.pdf

[80] T. Bray, “The javascript object notation (JSON) data interchange format,” *RFC*, vol. 8259, pp. 1–16, 2017. [En ligne]. Repéré à : https://doi.org/10.17487/RFC8259

[81] A. G. Lowe-Norris et R. Denn, *Windows 2000 active directory*. O’Reilly & Associates, Inc., 2000.

[82] R. Gerhards, “The syslog protocol,” =https ://tools.ietf.org/html/rfc5424 .

[83] ——, “The syslog protocol.”

[84] A. Orebaugh, G. Ramirez, et J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier, 2006.

[85] T. AbuHmed, A. Mohaisen, et D. Nyang, “A survey on deep packet inspection for intrusion detection systems,” *CoRR*, vol. abs/0803.0037, 2008. [En ligne]. Repéré à : http://arxiv.org/abs/0803.0037

[86] W. M. P. van der Aalst, A. H. M. ter Hofstede, et M. Weske, “Business process management : A survey,” dans *Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*, 2003, pp. 1–12. [En ligne]. Repéré à : https://doi.org/10.1007/3-540-44895-0_1

[87] R. Dijkman, J. Hofstetter, et J. Koehler, *Business Process Model and Notation*. Springer, 2011.

[88] S. White, “Using bpmn to model a bpel process,” *BPTrends*, vol. 3, n° 3, pp. 1–18, 2005.

[89] G. Cooper, “Dtrace : dynamic tracing in oracle solaris, mac OS x, and free BSD by brendan gregg and jim mauro,” *ACM SIGSOFT Software Engineering Notes*, vol. 37, nº 1, p. 34, 2012. [En ligne]. Repéré à : http://doi.acm.org/10.1145/2088883.2088902

[90] G. Kiczales, “Aspect-oriented programming,” *ACM Comput. Surv.*, vol. 28, nº 4es, p. 154, 1996. [En ligne]. Repéré à : http://doi.acm.org/10.1145/242224.242420

[91] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J. Loingtier, et J. Irwin, “Aspect-oriented programming,” dans *ECOOP’97 - Object-Oriented Programming, 11th European Conference, Jyväskylä, Finland, June 9-13, 1997, Proceedings*, ser. Lecture Notes in Computer Science, M. Aksit et S. Matsuoka, éds., vol. 1241. Springer, 1997, pp. 220–242. [En ligne]. Repéré à : https://doi.org/10.1007/BFb0053381

[92] A. Kane, “Runtime monitoring for safety-critical embedded systems,” Thèse de doctorat, Carnegie Mellon University, 2015.

[93] C. Artho, H. Barringer, A. Goldberg, K. Havelund, S. Khurshid, M. R. Lowry, C. S. Pasareanu, G. Rosu, K. Sen, W. Visser, et R. Washington, “Combining test case generation and runtime verification,” *Theor. Comput. Sci.*, vol. 336, nº 2-3, pp. 209–234, 2005. [En ligne]. Repéré à : https://doi.org/10.1016/j.tcs.2004.11.007

[94] J. Cumin, G. Lefebvre, F. Ramparany, et J. L. Crowley, “A dataset of routine daily activities in an instrumented home,” dans *Ubiquitous Computing and Ambient Intelligence - 11th International Conference, UCAmI 2017, Philadelphia, PA, USA, November 7-10, 2017, Proceedings*, ser. Lecture Notes in Computer Science, S. F. Ochoa, P. Singh, et J. Bravo, éds., vol. 10586. Springer, 2017, pp. 413–425. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-319-67585-5_43

[95] R. Caldas, M. Mundt, W. Potthast, F. Buarque de Lima Neto, et B. Markert, “A systematic review of gait analysis methods based on inertial sensors and adaptive algorithms,” *Gait & Posture*, vol. 57, pp. 204–210, 2017. [En ligne]. Repéré à : https://www.sciencedirect.com/science/article/pii/S0966636217302424

[96] J. Muangprathub, N. Boonnam, S. Kajornkasirat, N. Lekbangpong, A. Wanichsombat, et P. Nillaor, “Iot and agriculture data analysis for smart farm,” *Comput. Electron. Agric.*, vol. 156, pp. 467–474, 2019. [En ligne]. Repéré à : https://doi.org/10.1016/j.compag.2018.12.011

[97] A. Awad, G. Decker, et M. Weske, “Efficient compliance checking using BPMN-Q and temporal logic,” dans *Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings*, 2008, pp. 326–341. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-540-85758-7_24

[98] M. El Kharbili, A. K. A. de Medeiros, S. Stein, et W. M. P. van der Aalst, “Business process compliance checking : Current state and future challenges,” dans *Modellierung betrieblicher Informationssysteme - Modellierung zwischen SOA und Compliance Management - 27.-28. November 2008 Saarbrücken, Germany*, 2008, pp. 107–113. [En ligne]. Repéré à : http://subs.emis.de/LNI/Proceedings/Proceedings141/article2310.html

[99] G. Governatori, Z. Milosevic, et S. W. Sadiq, “Compliance checking between business processes and business contracts,” dans *Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), 16-20 October 2006, Hong Kong, China*, 2006, pp. 221–232. [En ligne]. Repéré à : https://doi.org/10.1109/EDOC.2006.22

[100] D. Knuplesch et M. Reichert, “A visual language for modeling multiple perspectives of business process compliance rules,” *Software and System Modeling*, vol. 16, n° 3, pp. 715–736, 2017. [En ligne]. Repéré à : https://doi.org/10.1007/s10270-016-0526-0

[101] S. Rinderle-Ma et S. Kabicher-Fuchs, “An indexing technique for compliance checking and maintenance in large process and rule repositories,” *Enterprise Modelling and Information Systems Architectures*, vol. 11, pp. 2 :1–2 :24, 2016. [En ligne]. Repéré à : https://doi.org/10.18417/emisa.11.2

[102] J. Rodrigues, *Health Information Systems : Concepts, Methodologies, Tools, and Applications, Volume 1*. IGI Global, 2010.

[103] J. Horcas, M. Pinto, L. Fuentes, W. Mallouli, et E. M. de Oca, “An approach for deploying and monitoring dynamic security policies,” *Computers & Security*, vol. 58, pp. 20–38, 2016. [En ligne]. Repéré à : https://doi.org/10.1016/j.cose.2015.11.007

[104] “Snare,” https://www.intersectalliance.com/, (consulté le 12 mars 2019).

[105] ManageEngine, “EventLog analyzer,” https://www.manageengine.com/products/eventlog, (consulté le 12 mars 2019).

[106] "Splunk," https://splunk.com, (consulté le 12 mars 2019).

[107] S. Proctor, "Snare," https://github.com/stevenproctor/lumberjack, (consulté le 12 mars 2019).

[108] F. Chen et G. Rosu, "Towards monitoring-oriented programming : A paradigm combining specification and implementation," *Electr. Notes Theor. Comput. Sci.*, vol. 89, nº 2, pp. 108–127, 2003. [En ligne]. Repéré à : https://doi.org/10.1016/S1571-0661(04)81045-4

[109] ——, "Mop : an efficient and generic runtime verification framework," dans *Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2007, October 21-25, 2007, Montreal, Quebec, Canada*, R. P. Gabriel, D. F. Bacon, C. V. Lopes, et G. L. S. Jr., éds. ACM, 2007, pp. 569–588. [En ligne]. Repéré à : http://doi.acm.org/10.1145/1297027.1297069

[110] P. O. Meredith, D. Jin, D. Griffith, F. Chen, et G. Rosu, "An overview of the MOP runtime verification framework," *STTT*, vol. 14, nº 3, pp. 249–289, 2012. [En ligne]. Repéré à : https://doi.org/10.1007/s10009-011-0198-6

[111] D. A. Basin, M. Harvan, F. Klaedtke, et E. Zalinescu, "MONPOLY : monitoring usage-control policies," dans *Runtime Verification - Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers*, ser. Lecture Notes in Computer Science, S. Khurshid et K. Sen, éds., vol. 7186. Springer, 2011, pp. 360–364. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-642-29860-8_27

[112] F. Chen et G. Rosu, "Java-mop : A monitoring oriented programming environment for java," dans *Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, TACAS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, 2005, pp. 546–550. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-540-31980-1_36

[113] A. P. Sistla et E. M. Clarke, "The complexity of propositional linear temporal logics," dans *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, H. R. Lewis, B. B. Simons, W. A. Burkhard, et L. H. Landweber, éds. ACM, 1982, pp. 159–168. [En ligne]. Repéré à : https://doi.org/10.1145/800070.802189

[114] C. Allan, P. Avgustinov, A. S. Christensen, L. J. Hendren, S. Kuzins, O. Lhoták,

O. de Moor, D. Sereni, G. Sittampalam, et J. Tibble, “Adding trace matching with free variables to aspectj,” dans *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2005, October 16-20, 2005, San Diego, CA, USA*, R. E. Johnson et R. P. Gabriel, éds. ACM, 2005, pp. 345–364. [En ligne]. Repéré à : http://doi.acm.org/10.1145/1094811.1094839

[115] E. Bodden, L. J. Hendren, P. Lam, O. Lhoták, et N. A. Naeem, “Collaborative runtime verification with tracematches,” *J. Log. Comput.*, vol. 20, n° 3, pp. 707–723, 2010. [En ligne]. Repéré à : https://doi.org/10.1093/logcom/exn077

[116] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, et W. G. Griswold, “Getting started with ASPECTJ,” *Commun. ACM*, vol. 44, n° 10, pp. 59–65, 2001. [En ligne]. Repéré à : http://doi.acm.org/10.1145/383845.383858

[117] H. Barringer, A. Goldberg, K. Havelund, et K. Sen, “Rule-based runtime verification,” dans *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, Italy, January 11-13, 2004, Proceedings*, ser. Lecture Notes in Computer Science, B. Steffen et G. Levi, éds., vol. 2937. Springer, 2004, pp. 44–57. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-540-24622-0_5

[118] M. A. Köhl, H. Hermanns, et S. Biewer, “Efficient monitoring of real driving emissions,” dans *Runtime Verification - 18th International Conference, RV 2018, Limassol, Cyprus, November 10-13, 2018, Proceedings*, ser. Lecture Notes in Computer Science, C. Colombo et M. Leucker, éds., vol. 11237. Springer, 2018, pp. 299–315. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-030-03769-7_17

[119] F. Chen et G. Rosu, “Parametric trace slicing and monitoring,” dans *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, ser. Lecture Notes in Computer Science, S. Kowalewski et A. Philippou, éds., vol. 5505. Springer, 2009, pp. 246–261. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-642-00768-2_23

[120] D. C. Luckham, *The power of events – An introduction to complex event processing in distributed enterprise systems*. ACM, 2005.

[121] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, et M. A. Shah,

"Telegraphcq : Continuous dataflow processing for an uncertain world," dans *CIDR 2003, First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 5-8, 2003, Online Proceedings*. www.cidrdb.org, 2003. [En ligne]. Repéré à : http://www-db.cs.wisc.edu/cidr/cidr2003/program/p24.pdf

[122] S. Madden, M. A. Shah, J. M. Hellerstein, et V. Raman, "Continuously adaptive continuous queries over streams," dans *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, 2002, pp. 49–60. [En ligne]. Repéré à : http://doi.acm.org/10.1145/564691.564698

[123] S. Chandrasekaran et M. J. Franklin, "Streaming queries over streaming data," dans *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*. Morgan Kaufmann, 2002, pp. 203–214. [En ligne]. Repéré à : http://www.vldb.org/conf/2002/S07P01.pdf

[124] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, et J. Widom, "STREAM : the stanford data stream management system," dans *Data Stream Management - Processing High-Speed Data Streams*, 2016, pp. 317–336. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-540-28608-0_16

[125] A. Arasu, S. Babu, et J. Widom, "An abstract semantics and concrete language for continuous queries over streams and relations," Stanford, Rapport Technique, 2002.

[126] S. Suhothayan, K. Gajasinghe, I. L. Narangoda, S. Chaturanga, S. Perera, et V. Nanayakkara, "Siddhi : a second look at complex event processing architectures," dans *Proceedings of the 2011 ACM SC Workshop on Gateway Computing Environments, GCE 2011, Seattle, WA, USA, November 18, 2011*, R. Dooley, S. Fiore, M. L. Green, C. Kiddle, S. Marru, M. E. Pierce, M. Thomas, et N. Wilkins-Diehr, éds. ACM, 2011, pp. 43–50. [En ligne]. Repéré à : http://doi.acm.org/10.1145/2110486.2110493

[127] L. Brenna, J. Gehrke, M. Hong, et D. Johansen, "Distributed event stream processing with non-deterministic finite automata," dans *DEBS*, A. S. Gokhale et D. C. Schmidt, éds. ACM, 2009.

[128] M. Stonebraker et A. Weisberg, "The voltdb main memory DBMS," *IEEE Data Eng. Bull.*, vol. 36, n° 2, pp. 21–27, 2013. [En ligne]. Repéré à : http://sites.computer.org/debull/A13june/VoltDB1.pdf

[129] G. G. Koch, B. Koldehofe, et K. Rothermel, “Cordies : expressive event correlation in distributed systems,” dans *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS 2010, Cambridge, United Kingdom, July 12-15, 2010*, J. Bacon, P. R. Pietzuch, J. Sventek, et U. Çetintemel, éds. ACM, 2010, pp. 26–37. [En ligne]. Repéré à : http://doi.acm.org/10.1145/1827418.1827424

[130] A. Alexandrov, R. Bergmann, S. Ewen, J. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, et D. Warneke, “The stratosphere platform for big data analytics,” *VLDB J.*, vol. 23, n° 6, pp. 939–964, 2014. [En ligne]. Repéré à : http://dx.doi.org/10.1007/s00778-014-0357-y

[131] J. Cao, X. Wei, Y. Liu, D. Mao, et Q. Cai, “LogCEP — complex event processing based on pushdown automaton,” *Int. Journal of Hybrid Information Technology*, pp. 71–82, 2014.

[132] K. Mamouras, M. Raghothaman, R. Alur, Z. G. Ives, et S. Khanna, “StreamQRE : modular specification and efficient evaluation of quantitative queries over streaming data,” dans *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, Barcelona, Spain, June 18-23, 2017*, 2017, pp. 693–708. [En ligne]. Repéré à : https://doi.org/10.1145/3062341.3062369

[133] G. Cugola et A. Margara, “TESLA : a formally defined event specification language,” dans *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS 2010, Cambridge, United Kingdom, July 12-15, 2010*, J. Bacon, P. R. Pietzuch, J. Sventek, et U. Çetintemel, éds. ACM, 2010, pp. 50–61. [En ligne]. Repéré à : http://doi.acm.org/10.1145/1827418.1827427

[134] R. M. Dijkman, S. P. Peters, et A. M. ter Hofstede, “A toolkit for streaming process data analysis,” dans *EDOC*, 2016, pp. 304–312.

[135] N. Narkhede, G. Shapira, et T. Palino, *Kafka : The Definitive Guide, Real Time Data and Stream Processing at Scale*. O’Reilly, 2017.

[136] Apache Foundation, “Apache Flume,” https://flume.apache.org/, (consulté le 12 mars 2019).

[137] “Apache Samza,” http://samza.apache.org, retrieved February 14th, 2017.

[138] L. Neumeyer, B. Robbins, A. Nair, et A. Kesari, “S4 : Distributed stream computing platform,” dans *ICDMW 2010, The 10th IEEE International Conference on Data Mining Workshops, Sydney, Australia, 13 December 2010*, W. Fan, W. Hsu, G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, et X. Wu, éds. IEEE Computer Society, 2010, pp. 170–177. [En ligne]. Repéré à : http://dx.doi.org/10.1109/ICDMW.2010.172

[139] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, et I. Stoica, “Apache spark : a unified engine for big data processing,” *Commun. ACM*, vol. 59, nº 11, pp. 56–65, 2016. [En ligne]. Repéré à : http://doi.acm.org/10.1145/2934664

[140] “Apache Storm,” http://storm.apache.org, retrieved February 14th, 2017.

[141] “ZeroMQ,” https://zeromq.org/, (consulté le 12 mars 2019).

[142] G. Frankowski, M. Jerzak, M. Miłostan, T. Nowak, et M. Pawłowski, “Application of the complex event processing system for anomaly detection and network monitoring,” *Computer Science*, vol. 16, nº 4, p. 351, 2015. [En ligne]. Repéré à : https://journals.agh.edu.pl/csci/article/view/1278

[143] A. Koliousis, M. Weidlich, R. C. Fernandez, P. Costa, A. L. Wolf, et P. Pietzuch, “SABER : Window-based hybrid stream processing for heterogeneous architectures,” dans *ACM SIGMOD 2016*. ACM - Association for Computing Machinery, June 2016. [En ligne]. Repéré à : https://www.microsoft.com/en-us/research/publication/saber-window-based-hybrid-stream-processing-for-heterogeneous-architectures/

[144] T. D. Matteis et G. Mencagli, “Parallel patterns for window-based stateful operators on data streams : An algorithmic skeleton approach,” *International Journal of Parallel Programming*, vol. 45, nº 2, pp. 382–401, 2017. [En ligne]. Repéré à : https://doi.org/10.1007/s10766-016-0413-x

[145] Amazon, “Kinesis data streams,” https://aws.amazon.com/kinesis/data-streams/, (consulté le 12 mars 2019).

[146] “StreamBase SQL,” http://streambase.com, (consulté le 12 mars 2019).

[147] R. Krishnan, J. Goldstein, et A. Raizman, “A hitchhiker’s guide to StreamInsight queries,

version 2.1," 2012. [En ligne]. Repéré à : http://support.sas.com/documentation/onlinedoc/dfdmstudio/2.4/dfU_ELRG.pdf

[148] S. Hallé, "When RV meets CEP," dans *RV 2016*, ser. Lecture Notes in Computer Science, Y. Falcone et C. Sánchez, éds., vol. 10012. Springer, 2016, pp. 68–91. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-319-46982-9_6

[149] H. Verbeek, R. J. C. Bose, et J. Chandra, "Prom 6 tutorial," *ProM.(nd). Extraído el*, vol. 27, 2010, http://www.promtools.org/prom6/downloads/prom-6.0-tutorial.pdf.

[150] B. F. van Dongen, A. Karla A. de Medeiros, H. M. W. Verbeek, A. Weijters, et W. M. P. Aalst, "The ProM framework : A new era in process mining tool support," dans *Lecture Notes in Computer Science*, vol. 3536, 06 2005, pp. 444–454.

[151] W. M. P. van der Aalst, A. Bolt, et S. J. van Zelst, "Rapidprom : Mine your processes and not just your data," *CoRR*, vol. abs/1703.03740, 2017. [En ligne]. Repéré à : http://arxiv.org/abs/1703.03740

[152] M. Hofmann et R. Klinkenberg, *RapidMiner : Data mining use cases and business analytics applications*. CRC Press, 2013.

[153] F. Mannhardt, M. de Leoni, et H. A. Reijers, "Heuristic mining revamped : an interactive, data-aware, and conformance-aware miner," *BPM 2017 Demos*, vol. 1920, 2017.

[154] F. Mannhardt, M. De Leoni, et H. A. Reijers, "The multi-perspective process explorer." *BPM (Demos)*, vol. 1418, pp. 130–134, 2015.

[155] G. Janssenswillen et B. Depaire, "bupar : Business process analysis in R," dans *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, September 13, 2017*, 2017. [En ligne]. Repéré à : http://ceur-ws.org/Vol-1920/BPM_2017_paper_193.pdf

[156] M. La Rosa, H. A. Reijers, W. M. Van Der Aalst, R. M. Dijkman, J. Mendling, M. Dumas, et L. García-Bañuelos, "Apromore : An advanced process model repository," *Expert Systems with Applications*, vol. 38, n° 6, pp. 7029–7040, 2011.

[157] D. M. Hawkins, *Identification of outliers*. Springer, 1980, vol. 11.

[158] Y. Yi, W. Zhou, Y. Shi, et J. Dai, "Speedup Two-Class Supervised Outlier Detection," *IEEE Access*, vol. 6, pp. 63 923–63 933, 2018. [En ligne]. Repéré à : https://ieeexplore.ieee.org/document/8502871/

[159] A. El-Kilany, N. E. Tazi, et E. Ezzat, "Semi-supervised outlier detection via bipartite graph clustering," dans *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*. Agadir, Morocco : IEEE, novembre 2016, pp. 1–6. [En ligne]. Repéré à : http://ieeexplore.ieee.org/document/7945629/

[160] J. Gao, H. Cheng, et P.-N. Tan, "Semi-supervised outlier detection," dans *Proceedings of the 2006 ACM symposium on Applied computing - SAC '06*. Dijon, France : ACM Press, 2006, p. 635. [En ligne]. Repéré à : http://portal.acm.org/citation.cfm?doid=1141277.1141421

[161] Z. Ferdousi et A. Maeda, "Unsupervised Outlier Detection in Time Series Data," dans *22nd International Conference on Data Engineering Workshops (ICDEW'06)*. Atlanta, GA, USA : IEEE, 2006, pp. x121–x121. [En ligne]. Repéré à : http://ieeexplore.ieee.org/document/1623916/

[162] S. Zanero et S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system," dans *Proceedings of the 2004 ACM symposium on Applied computing - SAC '04*. Nicosia, Cyprus : ACM Press, 2004, p. 412. [En ligne]. Repéré à : http://portal.acm.org/citation.cfm?doid=967900.967988

[163] Y. Zhou, H. Zou, R. Arghandeh, W. Gu, et C. J. Spanos, "Non-parametric outliers detection in multiple time series A case study : Power grid data analysis," dans *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, S. A. McIlraith et K. Q. Weinberger, éds. AAAI Press, 2018, pp. 4605–4612. [En ligne]. Repéré à : https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16315

[164] C. C. Aggarwal et P. S. Yu, "Outlier detection for high dimensional data," dans *Proceedings of the 2001 ACM SIGMOD international conference on Management of data - SIGMOD '01*. Santa Barbara, California, United States : ACM Press, 2001, pp. 37–46. [En ligne]. Repéré à : http://portal.acm.org/citation.cfm?doid=375663.375668

[165] C.-T. Lu, D. Chen, et Y. Kou, “Algorithms for spatial outlier detection,” dans *Third IEEE International Conference on Data Mining*. Melbourne, FL, USA : IEEE Comput. Soc, 2003, pp. 597–600. [En ligne]. Repéré à : http://ieeexplore.ieee.org/document/1250986/

[166] A. K. Singh et S. Lalitha, “A novel spatial outlier detection technique,” *Communications in Statistics - Theory and Methods*, vol. 47, n° 1, pp. 247–257, janvier 2018. [En ligne]. Repéré à : https://www.tandfonline.com/doi/full/10.1080/03610926.2017.1301477

[167] C. Cortes, K. Fisher, D. Pregibon, et A. Rogers, “Hancock : a language for extracting signatures from data streams,” dans *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, R. Ramakrishnan, S. J. Stolfo, R. J. Bayardo, et I. Parsa, éds. ACM, 2000, pp. 9–17. [En ligne]. Repéré à : https://doi.org/10.1145/347090.347094

[168] J. Feigenbaum, S. Kannan, M. Strauss, et M. Viswanathan, “Testing and spot-checking of data streams (extended abstract),” dans *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, D. B. Shmoys, éd. ACM/SIAM, 2000, pp. 165–174. [En ligne]. Repéré à : http://dl.acm.org/citation.cfm?id=338219.338248

[169] J. H. Fong et M. Strauss, “An approximate lp difference algorithm for massive data streams,” *Discrete Mathematics & Theoretical Computer Science*, vol. 4, n° 2, pp. 301–322, 2001. [En ligne]. Repéré à : http://dmtcs.episciences.org/290

[170] S. Guha, N. Mishra, R. Motwani, et L. O’Callaghan, “Clustering data streams,” dans *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*. IEEE Computer Society, 2000, pp. 359–366. [En ligne]. Repéré à : https://doi.org/10.1109/SFCS.2000.892124

[171] C. Ordonez, “Clustering binary data streams with k-means,” dans *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ser. DMKD ’03. New York, NY, USA : ACM, 2003, pp. 12–19. [En ligne]. Repéré à : http://doi.acm.org/10.1145/882082.882087

[172] B. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, et A. Biliris, “Online data mining for co-evolving time sequences,” dans *Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 - March 3, 2000*, D. B. Lomet et G. Weikum, éds. IEEE Computer Society, 2000, pp. 13–22. [En

ligne]. Repéré à : https://doi.org/10.1109/ICDE.2000.839383

[173] C. C. Aggarwal, “On abnormality detection in spuriously populated data streams,” dans *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*, H. Kargupta, J. Srivastava, C. Kamath, et A. Goodman, éds. SIAM, 2005, pp. 80–91. [En ligne]. Repéré à : https://doi.org/10.1137/1.9781611972757.8

[174] F. Angiulli et F. Fassetti, “Detecting distance-based outliers in streams of data,” dans *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, M. J. Silva, A. H. F. Laender, R. A. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, et A. O. Falcão, éds. ACM, 2007, pp. 811–820. [En ligne]. Repéré à : https://doi.org/10.1145/1321440.1321552

[175] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, et D. Gunopulos, “Online outlier detection in sensor data using non-parametric models,” dans *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, U. Dayal, K. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, et Y. Kim, éds. ACM, 2006, pp. 187–198. [En ligne]. Repéré à : http://dl.acm.org/citation.cfm?id=1164145

[176] K. Yamanishi, J. Takeuchi, G. J. Williams, et P. Milne, “On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms,” *Data Min. Knowl. Discov.*, vol. 8, n° 3, pp. 275–300, 2004. [En ligne]. Repéré à : https://doi.org/10.1023/B:DAMI.0000023676.72185.7c

[177] I. Assent, P. Kranen, C. Baldauf, et T. Seidl, “AnyOut : Anytime outlier detection on streaming data,” dans *Database Systems for Advanced Applications - 17th International Conference, DASFAA 2012, Busan, South Korea, April 15-19, 2012, Proceedings, Part I*, ser. Lecture Notes in Computer Science, S. Lee, Z. Peng, X. Zhou, Y. Moon, R. Unland, et J. Yoo, éds., vol. 7238. Springer, 2012, pp. 228–242. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-642-29038-1_18

[178] P. J. Rousseeuw et M. Hubert, “Robust statistics for outlier detection,” *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, vol. 1, n° 1, pp. 73–79, janvier 2011. [En ligne]. Repéré à : http://doi.wiley.com/10.1002/widm.2

[179] T. Toliopoulos, A. Gounaris, K. Tsichlas, A. Papadopoulos, et S. Sampaio, “Parallel

Continuous Outlier Mining in Streaming Data," dans *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. Turin, Italy : IEEE, octobre 2018, pp. 227–236. [En ligne]. Repéré à : https://ieeexplore.ieee.org/document/8631496/

[180] M. M. Breunig, H. Kriegel, R. T. Ng, et J. Sander, "LOF : identifying density-based local outliers," dans *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, W. Chen, J. F. Naughton, et P. A. Bernstein, éds. ACM, 2000, pp. 93–104. [En ligne]. Repéré à : https://doi.org/10.1145/342009.335388

[181] F. T. Liu, K. M. Ting, et Z. Zhou, "Isolation forest," dans *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. IEEE Computer Society, 2008, pp. 413–422. [En ligne]. Repéré à : https://doi.org/10.1109/ICDM.2008.17

[182] R. K. Sevakula et N. K. Verma, "Clustering based outlier detection in fuzzy SVM," dans *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Beijing, China : IEEE, juillet 2014, pp. 1172–1177. [En ligne]. Repéré à : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6891600

[183] G. Baillargeon, *Statistique Appliquée Pour les Sciences de la Gestion et les Sciences économiques : Traitement de Données Avec Excel, SPSS et MINITAB*. Editions SMG, 2009. [En ligne]. Repéré à : https://books.google.ca/books?id=us5zJgAACAAJ

[184] T. Chen, Q. Mao, Y. Yang, M. Lv, et J. Zhu, "Tinydroid : A lightweight and efficient model for android malware detection and classification," *Mobile Information Systems*, vol. 2018, pp. 1–9, 10 2018.

[185] S. A. Hofmeyr, S. Forrest, et A. Somayaji, "Intrusion detection using sequences of system calls," *J. Comput. Secur.*, vol. 6, n° 3, pp. 151–180, 1998. [En ligne]. Repéré à : http://content.iospress.com/articles/journal-of-computer-security/jcs109

[186] I. Guyon, S. Gunn, M. Nikravesh, et L. A. Zadeh, *Feature extraction : foundations and applications*. Springer, 2008, vol. 207.

[187] M. Yang, K. Kpalma, et J. Ronsin, "A survey of shape feature extraction techniques," 2008.

[188] S. Ding, H. Zhu, W. Jia, et C. Su, "A survey on feature extraction for pattern recognition," *Artificial Intelligence Review*, vol. 37, nº 3, pp. 169–180, 2012.

[189] M. Kantardzic, *Data mining : concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[190] M. Panda, S. Dehuri, et M. R. Patra, *Modern Approaches of Data Mining : Theory and Practice*. Alpha Science International, Ltd, 2015.

[191] Y. Rubner, C. Tomasi, et L. J. Guibas, "The Earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, nº 2, pp. 99–121, 2000. [En ligne]. Repéré à : https://doi.org/10.1023/A:1026543900054

[192] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," dans *Soviet physics doklady*, vol. 10, nº 8. Soviet Union, 1966, pp. 707–710.

[193] R. W. Hamming, "Error detecting and error correcting codes," *The Bell system technical journal*, vol. 29, nº 2, pp. 147–160, 1950.

[194] X. Gao, B. Xiao, D. Tao, et X. Li, "A survey of graph edit distance," *Pattern Anal. Appl.*, vol. 13, nº 1, pp. 113–129, 2010. [En ligne]. Repéré à : https://doi.org/10.1007/s10044-008-0141-y

[195] D. Nguyen, N. A. Smith, et C. P. Rosé, "Author age prediction from text using linear regression," dans *Proceedings of the 5th ACL Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities, LaTeCH@ACL 2011, 24 June, 2011, Portland, Oregon, USA*, 2011, pp. 115–123. [En ligne]. Repéré à : http://aclweb.org/anthology/W/W11/W11-1515.pdf

[196] G. A. Seber et A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012, vol. 329.

[197] D. C. Montgomery, E. A. Peck, et G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2012, vol. 821.

[198] R. E. Kass, "Nonlinear regression analysis and its applications," *Journal of the American Statistical Association*, vol. 85, nº 410, pp. 594–596, 1990.

[199] D. A. Ratkowsky, “Principles of nonlinear regression modeling,” *Journal of Industrial Microbiology*, vol. 12, nº 3-5, pp. 195–199, 1993.

[200] A. Ruckstuhl, “Introduction to nonlinear regression,” *IDP Institut fur Datenanalyse und Prozessdesign, Zurcher Hochschule fur Angewandte Wissenschaften. See : http ://www. idp. zhaw. ch.(Cited on p. 365.)*, 2010.

[201] P. J. Huber et al., “Robust regression : asymptotics, conjectures and monte carlo,” *The Annals of Statistics*, vol. 1, nº 5, pp. 799–821, 1973.

[202] P. J. Rousseeuw et A. M. Leroy, *Robust regression and outlier detection*. John wiley & sons, 2005, vol. 589.

[203] S. Hussain et P. Sprent, “Non-parametric regression,” *Journal of the Royal Statistical Society. Series A (General)*, pp. 182–191, 1983.

[204] W. Hardle, E. Mammen *et al.*, “Comparing nonparametric versus parametric regression fits,” *The Annals of Statistics*, vol. 21, nº 4, pp. 1926–1947, 1993.

[205] A. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, et O. K. Hussain, “Support vector regression with chaos-based firefly algorithm for stock market price forecasting,” *Appl. Soft Comput.*, vol. 13, nº 2, pp. 947–958, 2013. [En ligne]. Repéré à : https://doi.org/10.1016/j.asoc.2012.09.024

[206] E. Altay et M. H. Satman, “Stock market forecasting : artificial neural network and linear regression comparison in an emerging market,” *Journal of Financial Management & Analysis*, vol. 18, nº 2, p. 18, 2005.

[207] D. Baur, M. Saisana, et N. Schulze, “Modelling the effects of meteorological variables on ozone concentration—a quantile regression approach,” *Atmospheric Environment*, vol. 38, nº 28, pp. 4689–4699, 2004.

[208] J. B. Bremnes, “Probabilistic wind power forecasts using local quantile regression,” *Wind Energy : An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 7, nº 1, pp. 47–54, 2004.

[209] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, n° 1, pp. 81–106, 1986. [En ligne]. Repéré à : https://doi.org/10.1023/A:1022643204877

[210] ——, *C4. 5 : programs for machine learning*. Elsevier, 2014.

[211] B. W. Silverman et M. C. Jones, “E. fix and j.l. hodges (1951) : An important contribution to nonparametric discriminant analysis and density estimation : Commentary on fix and hodges (1951),” *International Statistical Review / Revue Internationale de Statistique*, vol. 57, n° 3, pp. 233–238, 1989. [En ligne]. Repéré à : http://www.jstor.org/stable/1403796

[212] M. H. Dunham, *Data mining : Introductory and advanced topics*. Pearson Education India, 2006.

[213] J. R. Quinlan, “Simplifying decision trees,” *International Journal of Man-Machine Studies*, vol. 27, n° 3, pp. 221–234, 1987. [En ligne]. Repéré à : https://doi.org/10.1016/S0020-7373(87)80053-6

[214] P. M. Domingos et G. Hulten, “Mining high-speed data streams,” dans *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, R. Ramakrishnan, S. J. Stolfo, R. J. Bayardo, et I. Parsa, éds. ACM, 2000, pp. 71–80. [En ligne]. Repéré à : https://doi.org/10.1145/347090.347107

[215] A. Bifet, R. Gavalda, G. Holmes, et B. Pfahringer, *Machine learning for data streams : with practical examples in MOA*. MIT press, 2018.

[216] Z. Xing, J. Pei, et P. S. Yu, “Early classification on time series,” *Knowl. Inf. Syst.*, vol. 31, n° 1, pp. 105–127, 2012. [En ligne]. Repéré à : https://doi.org/10.1007/s10115-011-0400-x

[217] H. M. Koduvely, *Learning Bayesian Models with R*. Packt Publishing Ltd, 2015.

[218] B. Mehlig, “Machine learning with neural networks,” *arXiv preprint arXiv :1901.05639*, 2019.

[219] I. Steinwart et A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.

[220] L. Breiman, “Random forests,” *Machine learning*, vol. 45, n° 1, pp. 5–32, 2001.

[221] P. Gil, H. Martins, et F. Januário, “Outliers detection methods in wireless sensor networks,” *Artificial Intelligence Review*, Feb 2018. [En ligne]. Repéré à : https://doi.org/10.1007/s10462-018-9618-2

[222] E. Kranakis, E. M. Belding, et E. Modiano, éds., *Proceedings of the 8th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2007, Montreal, Quebec, Canada, September 9-14, 2007*. ACM, 2007.

[223] A. Ayadi, O. Ghorbel, A. M. Obeid, et M. Abid, “Outlier detection approaches for wireless sensor networks : A survey,” *Computer Networks*, vol. 129, pp. 319–333, 2017. [En ligne]. Repéré à : https://doi.org/10.1016/j.comnet.2017.10.007

[224] A. Bédard et S. Hallé, “Model checking of stream processing pipelines,” dans *28th International Symposium on Temporal Representation and Reasoning, TIME 2021, September 27-29, 2021, Klagenfurt, Austria*, ser. LIPIcs, C. Combi, J. Eder, et M. Reynolds, éds., vol. 206. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 5 :1–5 :17. [En ligne]. Repéré à : https://doi.org/10.4230/LIPIcs.TIME.2021.5

[225] R. Hamid, A. Y. Johnson, S. Batta, A. F. Bobick, C. L. Isbell, et G. Coleman, “Detection and explanation of anomalous activities : Representing activities as bags of event *n*-grams,” dans *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. IEEE Computer Society, 2005, pp. 1031–1038. [En ligne]. Repéré à : https://doi.org/10.1109/CVPR.2005.127

[226] H. Inoue et A. Somayaji, “Lookahead pairs and full sequences : A tale of two anomaly detection methods,” dans *Proceedings of the 2nd Annual Symposium on Information Assurance*, 2007, pp. 9–19. [En ligne]. Repéré à : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.295.8158

[227] M. Grill, T. Pevný, et M. Rehák, “Reducing false positives of network anomaly detection by local adaptive multivariate smoothing,” *J. Comput. Syst. Sci.*, vol. 83, n° 1, pp. 43–57, 2017. [En ligne]. Repéré à : https://doi.org/10.1016/j.jcss.2016.03.007

[228] J. Cohen, P. Cohen, S. G. West, et L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences, 3rd Edition*. Routledge, 2003.

[229] T. M. Mitchell, *Machine Learning*. McGraw Hill, 2017.

[230] S. Hallé et R. Villemaire, “Runtime monitoring of message-based workflows with data,” dans *12th International IEEE Enterprise Distributed Object Computing Conference, ECOC 2008, 15-19 September 2008, Munich, Germany*. IEEE Computer Society, 2008, pp. 63–72. [En ligne]. Repéré à : https://doi.org/10.1109/EDOC.2008.32

[231] M. Hahsler, M. Bolanos, et J. Forrest, “Introduction to stream : An extensible framework for data stream clustering research with r,” *Journal of Statistical Software*, vol. 76, nº 14, p. 1–50, 2017. [En ligne]. Repéré à : https://www.jstatsoft.org/index.php/jss/article/view/v076i14

[232] S. Hallé, R. Khoury, et M. Awesso, “Streamlining the inclusion of computer experiments in a research paper,” *IEEE Computer*, vol. 51, nº 11, pp. 78–89, 2018. [En ligne]. Repéré à : https://doi.org/10.1109/MC.2018.2876075

[233] M. Roudjane et S. Hallé, “Benchmark for a beepbeep data mining library,” mai 2018. [En ligne]. Repéré à : https://doi.org/10.5281/zenodo.1252497

[234] B. van Dongen et F. Borchert, “BPI challenge 2018,” Eindhoven University of Technology, Rapport Technique, 2018.

[235] M. Roudjane et S. Hallé, “Benchmark for a beepbeep data mining library,” octobre 2019. [En ligne]. Repéré à : https://doi.org/10.5281/zenodo.3478564

[236] M. Roudjane, D. Rebaïne, R. Khoury, et S. Hallé, “Detecting trend deviations with generic stream processing patterns,” *Information Systems*, p. 101446, 2019.

[237] S. Hallé, R. Khoury, et S. Gaboury, “Event stream processing with multiple threads,” dans *RV 2017*, ser. Lecture Notes in Computer Science, S. K. Lahiri et G. Reger, éds., vol. 10548. Springer, 2017, pp. 359–369. [En ligne]. Repéré à : https://doi.org/10.1007/978-3-319-67531-2_22

[238] C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, et E. J. Keogh, “Matrix profile I : all pairs similarity joins for time series : A unifying view that includes motifs, discords and shapelets,” dans *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, F. Bonchi,

J. Domingo-Ferrer, R. A. Baeza-Yates, Z. Zhou, et X. Wu, éds. IEEE Computer Society, 2016, pp. 1317–1322. [En ligne]. Repéré à : https://doi.org/10.1109/ICDM.2016.0179

[239] S. Kauffman, M. Dunne, G. Gracioli, W. Khan, N. Benann, et S. Fischmeister, "Palisade : A framework for anomaly detection in embedded systems," *J. Syst. Archit.*, vol. 113, p. 101876, 2021. [En ligne]. Repéré à : https://doi.org/10.1016/j.sysarc.2020.101876

[240] N. N. T. Luong, Z. Milosevic, A. Berry, et F. A. Rabhi, "An open architecture for complex event processing with machine learning," dans *24th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2020, Eindhoven, The Netherlands, October 5-8, 2020*. IEEE, 2020, pp. 51–56. [En ligne]. Repéré à : https://doi.org/10.1109/EDOC49727.2020.00016

[241] C. R. Pinheiro, S. Guerreiro, et H. S. Mamede, "Automation of enterprise architecture discovery based on event mining from API gateway logs : State of the art," dans *23rd Conference on Business Informatics, CBI 2021, Bolzano, Italy, September 1-3, 2021. Volume 2*, J. P. A. Almeida, D. Bork, G. Guizzardi, M. Montali, H. A. Proper, et T. P. Sales, éds. IEEE, 2021, pp. 117–124. [En ligne]. Repéré à : https://doi.org/10.1109/CBI52690.2021.10062

[242] *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*. Morgan Kaufmann, 2002.

[243] J. Bacon, P. R. Pietzuch, J. Sventek, et U. Çetintemel, éds., *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS 2010, Cambridge, United Kingdom, July 12-15, 2010*. ACM, 2010.