# Automation of anomaly detections in real time by combining numeric and semantic processing

Badre Belabbess

UNIVERSITÉ PARIS-EST

THÈSE DE DOCTORAT

---

# Automatisation de détections d'anomalies en temps réel par combinaison de traitements numériques et sémantiques

---


Auteur:
Badre BELABBESS

Encadrant:
Dr. Olivier CURÉ


Thèse soumise en accord avec les exigences requises
pour l'obtention du Diplôme de Doctorat

Ecole Doctorale
Mathématiques et Sciences et Technologies de l'Information et de la Communication

**Rapporteurs**: Pr Frédérique LAFOREST – *Telecom Saint-Etienne* Pr Albert BIFFET – *Telecom Paris Tech*

**Examinateur** : Pr Hubert NAACKE – *Université Pierre-Marie Curie*

**Président**: Pr Frédérique LAFOREST – *Telecom Saint-Etienne*

23 Septembre, 2018

*"La victoire sur soi est la plus grande des victoires."*

Platon

UNIVERSITÉ PARIS-EST

# *Abstract*

Ecole Doctorale

Mathématiques et Sciences et Technologies de l'Information et de la Communication

Diplôme de Doctorat

**Automatisation de détections d'anomalies en temps réel par combinaison de traitements numériques et sémantiques**

by Badre BELABBESS

Les systèmes informatiques impliquant la détection d'anomalies émergent aussi bien dans le domaine de la recherche que dans l'industrie. Ainsi, des domaines aussi variés que la médecine (identification de tumeurs malignes), la finance (détection de transactions frauduleuses), les technologies de l'information (détéction d'intrusion réseau) et l'environnement (détection de situation de pollution) sont largement impactés.

L'apprentissage automatique propose un ensemble puissant d'approches qui peuvent aider à résoudre ces cas d'utilisation de manière efficace. Cependant, il représente un processus lourd avec des règles strictes qui supposent une longue liste de tâches telles que l'analyse et le nettoyage de données, la réduction des dimensions, l'échantillonnage, la sélection des algorithmes, l'optimisation des hyperparamètres, etc. Il implique également plusieurs experts qui travailleront ensemble pour trouver les bonnes approches.

De plus, les possibilités ouvertes aujourd'hui par le monde du sémantique montrent qu'il est possible de tirer parti des technologies du web afin de raisonner intelligemment sur les données brutes pour en extraire de l'information à forte valeur ajoutée. L'absence de systèmes combinant les approches numériques d'apprentissage automatique et les techniques sémantiques du web des données constitue la motivation principale derrière les différents travaux proposés dans cette thèse.

Enfin, les anomalies détectées ne signifient pas nécessairement des situations de réalité anormales. En effet, la présence d'informations externes pourrait aider à la prise de décision en contextualisant l'environnement dans sa globalité. Exploiter le domaine spatial et les réseaux sociaux permet de construire des contextes enrichis sur les données des capteurs. Ces contextes spatio-temporels deviennent ainsi une partie intégrante de la détection des anomalies et doivent être traités en utilisant une approche Big Data.

Dans cette thèse, nous présentons trois systèmes aux architectures variées, chacun ayant porté sur un élément essentiel des écosystèmes big data, temps-réel, web sémantique et apprentissage automatique :

- **WAVES** : Plateforme Big Data d'analyse en temps réel des flux de données RDF capturées à partir de réseaux denses de capteurs IoT. Son originalité tient dans sa capacité à raisonner intelligemment sur des données brutes afin d'inférer des informations implicites à partir d'informations explicites et d'aider dans la prise de décision. Cette plateforme a été développée dans le cadre d'un projet FUI dont le principal cas d'usage est la détection d'anomalies dans un réseau d'eau potable.
- **RAMSSES** : Système hybride d'apprentissage automatique dont l'originalité est de combiner des approches numériques avancées ainsi que des techniques sémantiques éprouvées. Il a été spécifiquement conçu pour supprimer

le lourd fardeau de l'apprentissage automatique qui est chronophage, complexe, source d'erreurs et impose souvent de disposer d'une équipe pluridisciplinaire.

- **SCOUTER** : Système intelligent de "scrapping web" permettant la contextualisation des singularités liées à l'Internet des Objets en exploitant aussi bien des informations spatiales que le web des données.

# *Acknowledgements*

Je tiens à remercier le Laboratoire d'Innovation d'Atos de m'avoir permis de mener mon projet de doctorat dans les meilleures conditions, et de m'avoir fourni toute l'aide nécessaire tout au long de ces 3 années. Mes plus sincères remerciements vont à mon directeur de thèse et mentor **M. Olivier Curé** pour son soutien continu, sa patience et ses conseils avisés.

Je suis également très reconnaissant envers mon manager **M. Gabriel Kepeklian** pour son énérgie, son écoute et la transmission de son savoir. Je suis également reconnaissant envers tous les membres de l'équipe du laboratoire qui m'ont réservé le meilleurs accueil dès le premier jour et qui ont contribué au développement de ce projet. De plus, mes sincères remerciements vont à **l'Ecole Doctorale de l'Université Paris-Est** pour m'avoir donné l'opportunité de réaliser mes ambitions en effectuant ce diplôme.

Au cours de ces 3 années, j'ai rencontré des gens extraordinaires qui ont contribué à développer mes qualités humaines ainsi que mes compétences professionnelles, chaque moment passé aux côtés de mes collègues restera un souvenir mémorable et une expérience enrichissante. Ma gratitude à mes collègues et amis **Tendry Randriamalala, Xiang Nan Ren, Jeremy Lhez**, et **Zakaria Khattabi** pour ces moments incroyables et leur amitié.

Aucun mot ne peut exprimer combien je suis reconnaissant envers ma famille pour leur amour et leur soutien continus malgré les difficultés. Merci à eux de m'avoir élevé et appris à être à la recherche continue de défis. Merci à ma mère pour son amour et ses soins sans fin. Merci à mes frères et tous les membres de ma famille pour m'avoir aidé à réaliser mes rêves.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **ESP** | **E**vent **S**tream**P**rocessing |
| **TEC** | **T**raitement d'**E**vènements**C**complexes |
| **IoT** | **I**netrent **o**f **T**hings |
| **RDF** | **R**esource **D**escription **F**ramework |
| **OWL** | **W**eb **O**ntology **L**anguage |
| **XML** | **E**Xtensible **M**arkup **L**anguage |
| **RDD** | **R**esilient **D**istributed **D**atasets |
| **JVM** | **J**ava **V**irtual **M**achine |
| **ACP** | **A**nalyse en **C**omposante **P**rincipale |
| **KO** | **K**ilo **O**ctets |
| **GO** | **G**iga **O**ctets |
| **MO** | **M**éga **O**ctets |
| **MS** | **M**illi **S**econdes |
| **RSP** | **R**df **S**tream **P**rocessing |
| **TAL** | **T**raitement **A**utomatique du **L**angage Naturel |
| **TLN** | **T**raitement du **L**angage **N**aturel |

*A mes parents, mes frères et mes amis.*

# Chapter 1

# Introduction

## 1.1 Contexte

Au cours des dernières années, des solutions innovantes ont été proposées pour surmonter les défis posés par l'émergence de villes durables et respectueuses de l'environnement. Le développement du concept de l'Internet des objets (IoT) a créé un nouveau champs de possibilités, notamment les réseaux intelligents, les bâtiments intelligents et les systèmes de transport intelligents. La plupart de ces systèmes exploitent des réseaux de capteurs générant des quantités massives de données et nécessitant un traitement spatio-temporel des flux dans le cadre des écosystèmes Big Data.

De plus, chaque jour, des centaines de millions de personnes prennent des photos, des vidéos, envoient des messages et effectuent des transactions en ligne. Selon un récent rapport d'IDC [ Matt Zwolenski, 2014], la quantité de données générées en 2017 était de 4,4 zettaoctets (milliards de gigaoctets), ce nombre augmente de façon exponentielle et atteindra 44 zettaoctets d'ici 2020. Il en résulte une énorme quantité de données hétérogènes provenant de différents types de sources.

Avoir une telle quantité de données générées en continu nécessite des moteurs puissants capables de les traiter efficacement afin de répondre aux besoins actuels dans un laps de temps très court. Ces défis ont conduit au développement de solutions de stockage Big Data et de systèmes de traitement tolérants aux pannes pouvant fonctionner très rapidement et avec une grande précision. En raison des limitations matérielles physiques, tout en ayant une charge de données élevée, l'exécution rapide nécessite une distribution, et donc tous les frameworks de traitement de flux Big Data ont été développés pour fournir un traitement de données distribué et évolutif pour les besoins en temps réel.

Cependant, les plateformes de traitement de flux existantes ne disposent pas toujours d'informations sur les métadonnées et sont souvent insuffisantes pour fournir un contexte raisonnable pour décrire les données. Pour surmonter cette lacune, les technologies sémantiques peuvent être utilisées pour convertir des données en un

ensemble de graphes connexes, avec la possibilité de manipuler des schémas qui conduisent à inférer des informations implicites.

## 1.2 Motivation

L'examen intelligent des données collectées par les capteurs permet d'améliorer la prise de décision en particulier dans le cadre de l'identification de singularités pouvant conduire à des risques environnementaux ou économiques. Dans le cadre de cette thèse, le cas d'usage visé concerne un grand réseau d'eau potable géré par un leader national expert dans le domaine de l'eau. La découverte de telles irrégularités dans le réseau d'eau est une préoccupation critique tant sur le plan écologique que financier. En effet, le volume réel d'eau perdue dans le monde a généré une perte de 32 milliards de m3/an (soit 14 milliards d'euros par an) dont 90 % reste difficilement identifiable en raison de la nature souterraine du réseau. Le réseau d'eau français de notre partenaire compte environ 100 000 km de canalisations équipées de plus de 3 000 capteurs et fournit de l'eau potable à environ 12 millions de clients. Sur la base de recherches approfondies menées par les experts, les anomalies peuvent être identifiées en utilisant des mesures de pression et de débit envoyées par des capteurs spécifiques dispersés sur tout le réseau de canalisations.

L'apprentissage automatique contient un ensemble puissant d'approches qui peut aider à détecter des anomalies de manière efficace. Cependant, il représente un processus lourd avec des règles strictes et une multitude de tâches telles que l'analyse et le nettoyage des données, la réduction de dimension, l'échantillonnage, la sélection d'algorithmes appropriés, le réglage précis des hyper-paramètres, etc. Il implique également plusieurs experts dont un data scientist, un statisticien, un mathématicien, un développeur et un testeur pour trouver les approches appropriées. Le système proposé dans cette thèse vise à simplifier ce processus lourd et accélérer le déploiement d'une solution en peu de temps afin d'identifier les anomalies et les contextualiser.

Enfin, dans de nombreux cas d'utilisation où des solutions de traitement de flux peuvent être appliquées, les données dirigées vers des unités de calcul sont alimentées par différentes sources hétérogènes dont le web. La quantité de données publiques disponibles sur le Web aussi bien sur les réseaux sociaux, les flux RSS que l'open data peut être extraite et analysée intelligemment pour fournir un contexte précis et riche en informations nouvelles. Il y a donc un fort potentiel que ces données puissent être connectées et qu'elles conduisent à inférer de nouvelles informations implicites pouvant apporter un éclairage nouveau à la décision des opérateurs sur les anomalies détectées.

## 1.3 Objectifs et Contributions

L'objectif de cette thèse est de proposer une approche originale de détection d'anomalies combinant des techniques numériques avancées et des outils sémantiques poussés. Le but est d'aller plus loin qu'une simple détection précise mais aussi de fournir une contextualisation large issue de sources hétérogènes pouvant influencer la prise de décision finale. Le système permettant de réaliser ces tâches est double car il comprends deux principaux composants:

- **RAMSSES** étant le module qui automatise le processus d'apprentissage en utilisant un générateur de réquêtes sémantiques afin d'apporter une visibilité interne.
- **SCOUTER** qui lui est spécialisé dans l'ingestion et l'analyse de données externes afin d'apporter un contexte aux anomalies détectées par le premier module.

Le premier module est un système de détection d'anomalies basé à la fois sur l'apprentissage automatique et la génération de requêtes sémantiques, utilisant également un mélange d'approches de traitement par lots (batch) et par flux (stream), pour détecter les anomalies dans les flux RDF. La sortie finale générée étant une liste d'anomalies détectées dans ce flux. Ceci est un domaine de recherche innovant, puisque d'autres approches, mises en évidence plus tard dans la section de travail connexe, utilisent soit l'approche d'apprentissage automatique soit l'approche de raisonnement sémantique.

Le deuxième module, Scouter, est un système de "scrapping" intelligent basé sur une puissante unité d'analyse TAL. Ce système recueille des informations sur les réseaux sociaux (Twitter et Facebook), les fournisseurs de données publiques, les flux RSS des journaux, les données météorologiques mais aussi les points d'intérêts dans une zone donnée pour en extraire un profil spécifique. Les données collectées sont analysées et stockées pour fournir une source d'information externe pouvant expliquer certaines fausses anomalies et fournir un contexte global à l'ensemble des événements recueillis.

En combinant ces deux outils, la plateforme proposée dans cette thèse permet non seulement de résoudre le problème d'automatisation des processus d'apprentissage mais aussi d'offrir un accès aux données externes issues du web pour améliorer la précision de la détection d'anomalies. Suivant le concept d'aide à la décision et de business intelligence, cette thèse se propose de soutenir les décisions à fort enjeu économique ou environnemental en mélangeant de manière originale des approches numériques et des techniques sémantiques au sein d'un système Big Data performant dans un environnement temps-réel exigeant.

# Chapter 2

# Etat de l'Art

Ce chapitre fournit les informations sur l'état de l'art actuel pour mieux comprendre les concepts traités dans cette thèse. Il commence par introduire le concept de détection d'anomalies et les techniques utilisées jusqu'à récemment. Ensuite, nous évoquons l'apprentissage automatique et ses différents modèles. Puis nous nous focalisons sur l'aspect technologique en expliquant les notions de Big Data et de traitement des flux où nous mentionnons différents cas d'utilisation ainsi que les frameworks disponibles actuellement. Enfin, nous détaillons les principes du Web Sémantique pour expliquer l'exploitation du Web des données ainsi que les principales techniques de TLN.

## 2.1 Détéction d'anomalies

Les anomalies sont appelées outliers [Hodge and Austin, 2004], exceptions [Suzuki et al., 2003] ou aberrations [Chandola, Banerjee, and Kumar, 2009] dans différents domaines d'application. La détection des anomalies est l'identification des échantillons qui ne sont pas conformes au comportement attendu. En réalité, les échantillons normaux ont généralement des distributions similaires, alors que les échantillons anormaux ont des distributions différentes. La détection d'anomalies a été appliquée dans de nombreux domaines, notamment la détection de la fraude [Bolton and David, 2002], d'intrusion [Lazarevic et al., 2003] et les soins de santé [Çinar and Acir, 2017]. Elle peut être généralisée en construisant un modèle à partir des données d'apprentissage et en prédisant l'état des données futures.

### 2.1.1 Méthodes à Seuils Fixes

La méthode à seuils est la plus simple, la plus couramment utilisée dans les applications industrielles. Elle nécessite en général une configuration manuelle et impose un ajustement manuel au fur et à mesure que les données changent avec le temps. Le détecteur d'anomalie marque les données comme étant anormales lorsqu'elles se

trouvent à l'intérieur ou à l'extérieur d'une plage spécifique. Les inconvénients de ces méthodes sont :

- Les paramètres sont généralement définis par l'utilisateur, plutôt que appris. Cette étape est souvent chronophage et s'avére être un processus difficile lors de la surveillance de nombreux flux de données. De plus, ces paramètres doivent être ajustés en fonction de l'environnement changeant et on besoin d'une maintenance manuelle.
- Les méthodes à seuils peuvent générer beaucoup de faux positifs. Par exemple, si un opérateur informatique augmente la mémoire sur un serveur, le pourcentage de mémoire utilisée peut tomber en dessous d'un seuil. Le seuil donnera alors continuellement de faux positifs jusqu'à modification manuelle car il n'apprend pas automatiquement le nouveau comportement.
- Puisque cette méthode ne tient pas compte des séquences temporelles, les seuils simples ne peuvent pas identifier l'origine des changements qui ont lieu dans les résultats.

### 2.1.2 Méthodes à Seuils Dynamiques

Ces techniques sont généralement basées sur des écarts-types, des moyennes mobiles, une comparaison avec les moindres carrés et une analyse d'histogrammes. Les inconvénients de ces méthodes sont :

- Une méthode statistique spécifique cible un comportement très spécifique dans les données. Donc, une seule méthode ne détectera pas un large éventail de types d'anomalies. Cependant, elles sont très rapides à calculer et ont de très bonnes performances pour les anomalies "connues".
- Elles n'apprennent pas les modèles et et ne recherchent pas ensuite les variations de ces modèles. Au lieu de cela, elles prédéterminent les types d'anomalies à rechercher, d'où leur incapacité à trouver de nouveaux types d'anomalies.
- Elles sont inefficaces face aux modèles temporels. Si vous brouillez les derniers points de données, les mesures statistiques telles que la moyenne et l'écart-type restent inchangées. Ainsi, aucune anomalie ne sera détectée même si le comportement semble complètement différent. Or, dans le cadre d'un vrai projet industriel, il y a un grand nombre de changements comportementaux significatifs qui n'entraînent pas de changements statistiques significatifs.

### 2.1.3 Méthodes basées sur l'apprentissage automatique

Différents algorithmes ont été proposés et peuvent être regroupés en trois classes en fonction des caractéristiques des données d'apprentissage [Chandola, Banerjee, and Kumar, 2009]:

- **Approches supervisées** : Des échantillons normaux et anormaux existent dans l'ensemble de données d'apprentissage, et ils sont utilisés conjointement pour former le modèle de détection. Le modèle formé identifie les échantillons de test comme normaux ou anormaux.
- **Approches semi-supervisées** : Seuls les échantillons normaux sont disponibles dans l'ensemble d'apprentissage; c'est-à-dire que l'utilisateur ne peut pas obtenir d'informations sur les anomalies. Les échantillons inconnus sont classés comme aberrants lorsque leur comportement est loin de celui des échantillons normaux connus.
- **Approches non supervisées** : l'information de classe de tous les échantillons dans les données d'apprentissage est inconnue des chercheurs; c'est-à-dire que les échantillons de l'ensemble d'apprentissage peuvent contenir à la fois des échantillons normaux et anormaux, mais la classification de chaque échantillon est inconnue.

Une grande quantité de données d'entraînement étiquetées est requise pour les approches supervisées. Aussi, la collecte d'échantillons positifs et négatifs est difficile et s'avère souvent très chronophage. En outre, la détection de nouveaux modèles aberrants avec un modèle entraîné sur des valeurs aberrantes connues est difficile. Les approches non supervisées ne nécessitent pas d'informations sur les étiquettes pour les données d'apprentissage, mais souffrent souvent de taux élevés de fausses alarmes et de faibles taux de détection [Xue, Shang, and Feng, 2010]. Dans de nombreuses applications, les échantillons normaux sont faciles à obtenir, tandis que les échantillons anormaux sont coûteux à collecter; nous nous concentrons donc sur la détection d'anomalies semi-supervisées.

La plupart des approches de détection d'anomalies actuelles sont conçues pour des ensembles de données de faible dimension et font face à des défis à mesure que les dimensions augmentent. L'application directe de ces approches à des ensembles de données de grandes dimensions peut produire de mauvais résultats [Singh, Kushwaha, and Vyas, 2013]. Une méthode largement utilisée pour relever ce défi est la cartographie des données de grande dimension dans un sous-espace de dimension inférieure et le traitement des nouvelles données avec des algorithmes de détection conventionnels. Différentes approches de réduction de dimension ont été proposées, telles que l'ensachement de caractéristiques [Lazarevic and Kumar,

2006], l'analyse en composantes principales [Shyu et al., 2008], l'algorithmie génétique [Zhang, Chung, and Lo, 2007], l'analyse discriminante linéaire [Liu et al., 2015 ] et l'apprentissage automatique [Erfani et al., 2016].

Le réglage des paramètres est une autre tâche difficile pour la détection paramétrique semi-supervisée des anomalies. Les critères classiques de mesure des performances d'un modèle entraîné ne peuvent pas être utilisés en raison du manque de mesures pour les valeurs aberrantes. De plus, le biais entre fausse alarme et fausse acceptation est difficile à contrôler. Certains chercheurs ont généré des valeurs aberrantes artificielles dans l'ensemble de validation pour ajuster les paramètres des modèles formés [3, 13, 17], mais les valeurs aberrantes artificielles peuvent ne pas refléter la distribution des valeurs aberrantes réelles. Un algorithme de détection adaptative non paramétrique a été proposé [18] et permet d'estimer un score d'anomalie pour chaque échantillon de requête via une technique du plus proche voisin. L'échantillon de requête a été classé comme anomalie lorsque le score est tombé en dessous du niveau de fausse alarme souhaité. Cependant, la valeur du plus proche voisin a été calculée dans l'espace complet, de sorte qu'elle pourrait souffrir de la malédiction de la dimensionnalité dans le cas du big data.

## 2.2 Apprentissage en Profondeur

Le concept d'apprentissage en profondeur est apparu pour la première fois en 2006 comme un nouveau champ de recherche dans le domaine de l'apprentissage automatique. Il a d'abord été connu comme un apprentissage hiérarchique et impliquait généralement de nombreux domaines de recherche liés à la reconnaissance des formes.

L'apprentissage en profondeur considère principalement deux facteurs clés: le traitement non linéaire en plusieurs couches ou étapes et l'apprentissage supervisé ou non supervisé [Bengio, 2009]. Le traitement non linéaire dans plusieurs couches fait référence à un algorithme dans lequel la couche courante prend la sortie de la couche précédente comme entrée. La hiérarchie est établie entre les couches pour organiser l'importance des données à considérer comme utiles ou non. D'un autre côté, l'apprentissage supervisé et non supervisé est lié à l'étiquette cible de la classe, sa disponibilité signifie un système supervisé, alors que son absence signifie un système non supervisé.

Le tableau ci-dessous [Vargas, Mosavi, and Ruiz, 2017] résume plusieurs applications réalisées au cours des années précédentes relatives à l'apprentissage en profondeur. En général, la reconnaissance vocale et le traitement de l'image restent les applications phares. Cet examen ne considère que quelques-unes de la grande liste d'applications.

| Author | Application | Method/algorithm | Year |
|---|---|---|---|
| Tai Sing Lee, David Mumford | Hierarchical Bayesian inference in the visual cortex | Particle filtering and Bayesian - belief propagation | 2003 |
| Hinton, Geoffrey E., Simon Osindero, Yee-WhyeTeh. | Digit Classification | Complementary Priors on Belief networks | 2006 |
| Mohamed, Abdel-rahman, George Dahl, Geoffrey Hinton | Deep Belief Networks for phone recognition | Back propagation and associative memory architecture | 2009 |
| Abdel-Hamid Ossama, Mohamed Abdel-rahman,Jiang Hui, Penn Gerald | Multi-speaker speech recognition | Local filtering and max-pooling infrequency domain | 2012 |
| Kiran B. Raja, R. Raghavendra, Vinay Krishna Venuri, Christoph Busch | Iris Recognition by using smartphones' cameras | Deep sparse filtering | 2015 |
| Silver David, et al | Mastering the Game of Go with Deep Neural Networks and Tree Search | Supervised learning and reinforcement learning | 2016 |
| Francesco Marra, Giovanni Poggi, Carlo Sansone, | Iris sensor model identification | Convolutional neural networks | 2017 |

FIGURE 2.1: Cas d'Usages d'Apprentissage en Profondeur 2003-2017

L'apprentissage en profondeur implique une analyse de couches abstraites et des méthodes hiérarchiques. Cependant, il peut être utilisé dans de nombreuses applications de la vie réelle. À titre d'exemple, dans le traitement d'images numériques; une image à l'échelle de gris coloriage à partir d'une image qui était faite manuellement par les utilisateurs qui devaient choisir chaque couleur en fonction de leur propre jugement. En appliquant un algorithme d'apprentissage en profondeur, la coloration peut être effectuée automatiquement par un ordinateur. De même, le son peut être ajouté à une vidéo en utilisant les réseaux neuronaux récurrents (RNN) dans le cadre des méthodes d'apprentissage en profondeur [Marra et al., 2017].

L'apprentissage en profondeur peut être compris comme une méthode pour améliorer les résultats et optimiser les temps de traitement dans plusieurs processus informatiques. Dans le domaine du traitement du langage naturel, des méthodes d'apprentissage en profondeur ont été appliquées pour la génération des légendes d'images [Mohamed, Dahl, and Hinton, 2009] et la génération d'écriture manuscrite [Deng and Yu, 2014].

## 2.3 Big Data

Le Big Data est un concept représentant des données qui sont si volumineuses, véloces et complexes que les technologies de stockage et de traitement traditionnelles ne peuvent pas les gérer efficacement. Durant cette dernière décennie, le terme a été principalement utilisé pour décrire l'acquisition de grandes masses de données afin d'extraire des informations à valeur ajoutée à partir des données brutes. L'analyse de ces données permet de trouver des corrélations pour repérer par exemple des tendances commerciales, prédire certaines maladies ou encore fournir des explications à des singularités. En 2001 [**3ddata**], un rapport de recherche définissait les défis et les opportunités de croissance des données sur quatre axes, à savoir:

- **Le Volume** : Le même rapport estime que chaque jour 2,4 trillions de gigabits de données sont générées principalement du fait de la croissance exponentielle du réseau de téléphonie mobile. S'agissant de la notion de Volume, on parle de big data au-delà de 100 To (données non structurées). Ainsi, puisque six des sept milliards de la population mondiale possède actuellement un téléphone portable, les messages SMS et WhatsApp, les photos, les vidéos et les nombreuses applications contribuent à faire augmenter significativement le volume total à traiter. Sans oublier également l'émérgence de l'IoT et les centaines de milliers de capteurs déployés à travers le monde pour récupérer des quantités de données importantes liées à divers cas d'usage tels que l'agriculture, l'énérgie, les transports ou l'environnement.

- **La vélocité** : Elle correspond à la vitesse de circulation des flux de données provenant de sources telles que les processus métier, les machines, les réseaux et les interactions humaines avec les réseaux sociaux, les appareils mobiles, etc. Ces données en temps réel peuvent aider les pouvoirs publics et les entreprises à prendre des décisions utiles qui offrent des avantages concurrentiels stratégiques et un retour sur investissement.

- **La variété** : Ce concept fait référence aux sources de plus en plus diversifiées et aux types de données nécessitant une gestion innovante et une analyse plus poussée. Traditionnellement, le stockage des données concernait des sources telles que des tableurs (e.g., Microsoft Excel), des documents internes spécifiques (fichiers au format doc, pdf, etc) ainsi que des bases de données (principalement SQL). Aujourd'hui, les données se présentent sous forme de courriels, photos, vidéos, dispositifs de surveillance, audio, et bien d'autres formats atypiques. Il est donc devenu nécessaire d'intégrer ces types de données complexes et multiples - qu'elles soient structurées, semi-structurées ou non structurées - à partir d'un ensemble de systèmes bien définis et de sources identifiées. Cependant, la variété des données non structurées ayant explosé ces dernières années, cela a créé des problèmes de stockage, d'extraction et d'analyse des données auxquels les ingénieurs Big Data essayent de remédier.

- **La véracité** : Cette dimension fait référence aux biais, aux bruits et aux différentes anomalies présentes dans les jeux de données générés. Aujourd'hui, alors que le volume continue à augmenter en pétaoctets et que les coûts continuent à diminuer, les problèmes de qualité de l'information sont devenus plus importants que jamais (Hall, 2013). IBM estime que la mauvaise qualité des données coûte environ 3,1 milliards de dollars par an aux consommateurs américains et qu'environ 27% des répondants d'une enquête ne savaient pas à quel point leurs données étaient inexactes (2013). Cependant, ce n'est que récemment que l'importance de la qualité de l'information a été reconnue, avec des appels à caractériser les données volumineuses selon l'incertitude qui y est présente.

Beaucoup d'efforts ont été déployés pour développer des solutions de stockage de données massives structurées et non structurées sous forme de systèmes de fichiers distribués. En 2011, LexisNexis a développé un framework de partage de fichiers distribués pour le stockage et le requêtage. Plus tard, elle a acquis Seisint & ChoicePoint Inc, et a fusionné ses plates-formes de traitements parallèles à haute vitesse [Middleton, 2011] en un seul logiciel libre d'utilisation sous licence Apache. En 2004, Google a publié un article de recherche sur un paradigme devenu très populaire nommé MapReduce [Jeffrey Dean, 2004], dans lequel les données sont distribuées sur une grappe de machines et les programmes traités en parallèle. Les résultats sont ensuite rassemblés et livrés à l'utilisateur final. Le principe de ce framework a été adopté par un très célèbre projet Open Source Apache nommé Hadoop [1]. En 2011, le système de fichiers Quantcast (QFS) [Michael Ovsiannikov and Sutter, 2011], qui est une alternative au Hadoop Distributed File System, a été publié avec une compatibilité plug-in pour MapReduce.

D'une manière générale, et selon un récent rapport du célèbre cabinet de stratégie McKinsey [**mckinsey**], un écosystème ou des composants big data souhaitant percer sur le marché des hautes technologies doivent être caractérisés par des techniques avancées d'analyse de données telles que l'apprentissage automatique ou le traitement du langage naturel et disposer d'outils de visualisations efficaces offrant à l'utilisateur final une expérience intuitive.

## 2.4 Traitement de flux de données

Le traitement de flux de données constitue un champ de recherche dédié à la recherche de solutions pour gérer efficacement des flux important de données durant des laps de temps très courts (de l'ordre de la milliseconde)[Hirzel et al., 2014]. Le traitement d'événements complexes se base sur l'agrégation d'événements au

[1] Hadoop Eco-System: http://hadoop.apache.org/

sein de fenêtres temporelles définies (fixes ou glissantes). Ces événements peuvent provenir de sources variées visant à se compléter. Les moteurs TEC sont optimisés pour traiter des événements discrets ou continus en appliquant des décisions, des règles ou des raisonnements intelligents aux modèles d'événements. Plusieurs types de traitement d'événements ont émérgé ces dernières années afin d'analyser les données de streaming en temps réel telles que les requêtes SQL continues [Zou et al., 2010] ou des fenêtres tamponnées. L'analyse en continu, qui est la capacité de calculer continuellement des mesures mathématiques ou statistiques à la volée dans les flux de données arrivant, constitue une partie importante de ce domaine de recherche. Les solutions de traitement de flux sont conçues pour gérer un débit élevé en temps réel avec une architecture évolutive, hautement disponible et tolérante aux pannes [Chen et al., 2016].

Classiquement, un moteur de traitement de flux doit répondre aux exigences suivantes:

- Capacité à traiter une quantité massive d'événements arrivant sous forme de flux. Le traitement peut correspondre au filtrage, l'agrégation, l'application de règles définies au préalable, l'automatisation de processus, la prédiction de singularités, la surveillance ou l'alerte.
- Réactivité en temps réel aux conditions changeantes du marché. L'évolutivité de la solution à mesure que les volumes de données augmentent en taille et en complexité.
- Facilité d'intégration avec l'infrastructure existante et les nouvelles sources de données disponibles.
- La tolérance aux pannes particulièrement en environnement distribué sur un cluster industriel de plusieurs centaines de machines interconnectées.

Le traitement des flux de données a trouvé ses premiers usages dans le secteur de la finance, les marchés boursiers étant passés de la négociation directe au commerce électronique. Aujourd'hui, ce domaine de recherche est appliqué dans quasiment toutes les industries, en particulier partout où des données massives sont générées à travers les activités humaines, les machines installées en production ou les différents capteurs IoT. Parmi les cas d'usage phares où le traitement de flux peut apporter des solutions efficaces, on peut citer la surveillance réseau, la gestion des risques, le e-commerce, la détection de fraude ou d'anomalies, le routage intelligent des commandes, l'analyse des coûts de transaction, la tarification et analyse client, la gestion des données de marché et le trading algorithmique.

## 2.5 Web Sémantique

Dans cette section, nous allons expliquer brièvement ce qu'est la sémantique et pourquoi elle est utilisée par vagues. De plus, nous discuterons du traitement des flux RDF.

### 2.5.1 Pourquoi la sémantique?

La sémantique est le support permettant de fournir du sens aux éléments présents dans un langage donné. Dans le contexte de nos travaux, l'interprétation de données sémantisées permet de déduire de nouvelles connaissances. La sémantique est un symbole qui peut se rapporter à des choses ou à des concepts, cela aide à répondre à des questions et à déduire de nouvelles connaissances.

Prenons un exemple: Bob aime les chiens, les chiens font peur à Kate. Ces deux phrases sont de la forme "sujet-verbe-objet", les deux représentent une information. Bob et Kate se réfèrent à des personnes spécifiques; les chiens représentent un type d'animaux; aime et effraie représentent la relation entre les entités mentionnés avec la connaissance de base des humains qui leur permettent de reconnaître ce que sont les entités. Ils sont équipés de nouvelles connaissances et pourraient répondre à la question "qui aime les animaux ?".

La sémantique est le processus de communication avec suffisamment de sens pour aboutir à une action. Quand il s'agit de machines, la technologie sémantique, et sa vitrine la plus active, le Web Sémantique, consiste à utiliser la sémantique pour représenter, combiner et partager les connaissances entre les communautés de machines.

En informatique, les données sémantiques sont représentées sous la forme de triplets (sujet, prédicat et objet) et représentent un énoncé linguistique. Le sujet est une chose (entité) dont une classe conceptuelle comme les gens et les lieux. Les prédicats sont une propriété de l'entité à laquelle ils sont attachés. Le nom ou la date de naissance d'une personne ou le symbole boursier ou l'adresse postale d'une entreprise sont tous des exemples de prédicats. Les objets appartiennent à deux classes: les entités qui peuvent être le sujet dans d'autres triplets, et les valeurs littérales telles que les chaînes ou les nombres. De manière générale, différents triplets peuvent être reliés entre eux en utilisant les objets d'autres triplets, aboutissant à une chaîne de relations qui forme un graphe orienté.

Une ontologie est un modèle de domaine de connaissances, décrivant les concepts au sein de ce domaine et les relations entre les concepts dans le domaine. La figure 2.2 ci-dessous montre un exemple d'une ontologie simple. Ce n'est pas toujours le cas d'avoir une ontologie simple, les ontologies peuvent devenir des monstres de complexité. Les bonnes pratiques nous ont appris qu'il est préférable d'avoir

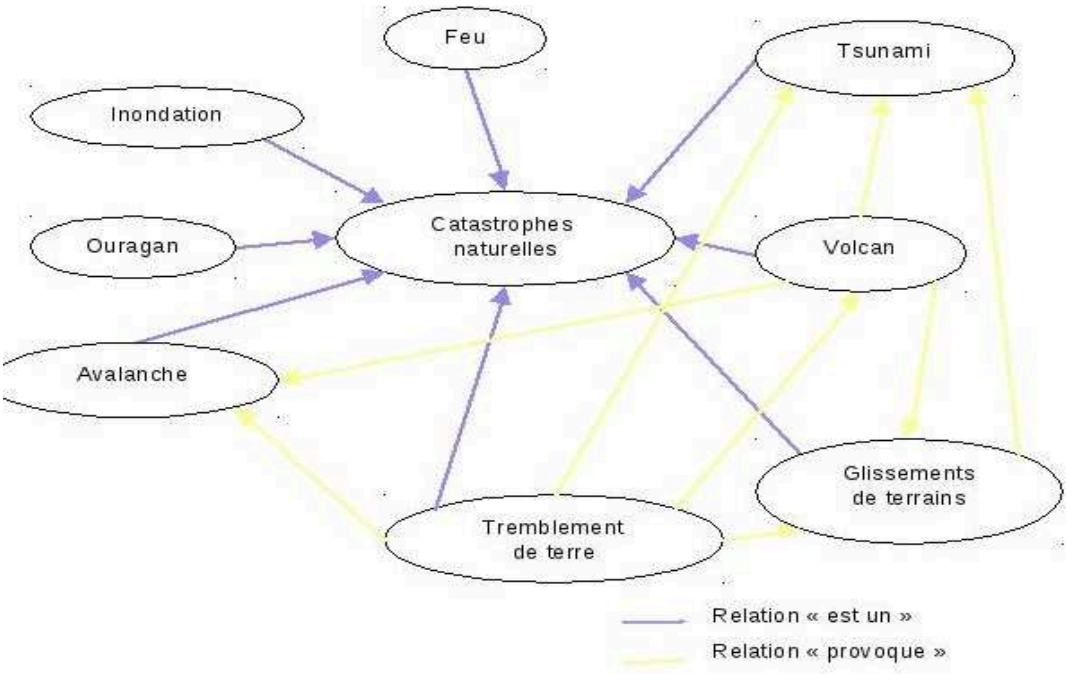


FIGURE 2.2: Exemple d'Ontologie

des ontologies précises et ciblées. Par conséquent, deux catégorisations principales d'ontologies ont été proposées [Guarino, 1998]. La première catégorisation proposée inclut les ontologies de représentation, génériques, de tâche et d'application. La deuxième approche proposée inclut des ontologies de premier niveau, de domaine, de demande et d'application. Comme les graphes orientés sont des structures de données bien connues en informatique et en mathématiques, c'est un choix très pratique pour représenter des données sémantiques.

### 2.5.2 Technologies du Web sémantique

#### Concepts

Le Web sémantique est destiné à fournir des données liées par des propriétés spécifiques, son concept de base est l'intégration de données provenant de différentes sources. Pour atteindre cet objectif, un modèle d'échange unifié adapté à toutes les sources doit exister.

RDF est un modèle standard pour l'échange de données sur le web, il dispose de fonctionnalités qui facilitent la fusion des données même si les schémas sous-jacents diffèrent, et supporte spécifiquement l'évolution des schémas dans le temps. Il est utilisé pour coder les informations et définit les propriétés ainsi que les classes, son but est de faciliter l'utilisation des données avec différents codages sans avoir besoin de convertir les données. Par exemple, pas besoin de changer un nom de propriété dans un ensemble de données pour correspondre au nom de propriété sémantiquement identique utilisé dans un autre ensemble de données. Au lieu de cela, on peut

ajouter une instruction RDF qui indique que les deux propriétés ont la même signification.

Les données RDF étaient initialement codées en XML et destinées au traitement automatisé, mais plusieurs formats de sérialisation communs sont actuellement utilisés notamment: Turtle [2], N- Triples [3], N-Quads [4], Json-LD [5] et N3 [6].

Comme mentionné précédemment, en utilisant des triplets liés par des propriétés un graphe orienté de nœuds peut être formé. Les noeuds d'un graphe RDF sont les sujets et les objets des énoncés qui composent le graphe. Il existe deux types de nœuds: les ressources et les littéraux. Les littéraux représentent des valeurs de données concrètes comme des nombres ou des chaînes et ne peuvent pas faire l'objet de déclarations, mais uniquement les objets. Les ressources, en revanche, représentent tout le reste, et peuvent être soit des sujets, soit des objets. Les arêtes du graphe sont les prédicats, également appelés propriétés, représentent les connexions entre les ressources. Cependant, les prédicats sont eux-mêmes des ressources, et les déclarations RDF peuvent être faites à propos des prédicats tout comme ils le peuvent pour n'importe quelle autre ressource.

L'utilisation de RDF lors de l'utilisation de l'échange de données offre les avantages suivants:

- **Standaridsation** : RDF et RDFS (le langage de schéma RDF) sont des standards, tout comme le langage d'ontologie Web (OWL) qui est construit sur RDF et RDFS. OWL offre une modélisation et une inférence de classe et de propriété plus riches. Le langage de requête SPARQL est un standard qui partage des mots clés avec SQL. Cependant, il opére sur des graphes et non sur des relations.

- **Flexibilité** : Définir les propriétés utilisées avec les classes est similaire à la définition des colonnes dans une table de base de données relationnelle. Cependant, nul besoin de définir des propriétés pour chaque instance d'une classe. Ceci est analogue à une table de base de données qui peut manquer des colonnes pour les lignes qui n'ont pas de valeurs pour ces colonnes (une représentation de données fragmentée). En outre, il est possible d'effectuer des instructions RDF ad hoc sur n'importe quelle ressource sans avoir besoin de mettre à jour les schémas globaux. Les requêtes SPARQL peuvent contenir des clauses de correspondance facultatives qui fonctionnent bien avec les représentations de données fragmentées.

- **Enrichissement** : Les ontologies partagées facilitent la fusion de données provenant de différentes sources. Cela est possible grâce à la présence de

[2]Tutle: https://www.w3.org/TR/turtle/
[3]N-Triples: https://www.w3.org/TR/n-triples/
[4]N-Quads: https://www.w3.org/TR/ n-quads /
[5]Json-LD: http://json-ld.org/
[6]N3: https://www.w3.org/TeamSubmission/n3/

standards forts bien implantés aussi bien dans les communautés scientifiques qu'industrielles. De plus, RDF est basé sur des protocoles Internet éprouvés comme HTTP et supporte naturellement la mise à l'échelle du web.

- **Inférence** : RDFS ou OWL crée automatiquement de nouvelles informations sur des éléments tels que l'appartenance à une classe. L'inférence est supportée par plusieurs logiques différentes. L'inférence prend en charge la fusion de données définies à l'aide de différentes ontologies ou schémas en faisant des déclarations sur l'équivalence des classes et des propriétés.
- **Interrogation** : Le langage SPARQL définit la sémantique d'interrogation standard et un protocole d'accès aux données à utiliser avec le modèle de données RDF (Resource Description Framework). SPARQL fonctionne pour n'importe quelle source de données pouvant être mappée à RDF. Une requête SPARQL est de la forme H <- B, où B, le corps de la requête, est un modèle de graphe RDF complexe composé de motifs graphiques de base avec différents opérateurs algébriques tels que UNION, OPTIONAL, etc .; et H, la tête de la requête, est une expression qui indique comment construire la réponse à la requête.

#### Langage de Requêtes

La nature de l'ESP est de traiter un ensemble d'opérations sur des flux de données illimités. Un tel traitement est généralement effectué dans une fenêtre qui découpe les flux de données infinis entrants en blocs finis. Ces fenêtres peuvent être définies sur certaines contraintes temporelles, par exemple prendre les 3 dernières minutes de données entrantes, ou par-dessus des paramètres non temporels, par exemple basé sur une session.

Les opérations exécutées dans une fenêtre peuvent être définies en utilisant un langage de programmation, par exemple Scala, ou décrites dans un langage de requête qui est ensuite désigné comme un langage de requête continu. Dans ce travail, nous nous concentrons sur les langages de requêtes continus basés sur SPARQL tels que C-SPARQL. Dans [García et al., 2014], les auteurs présentent C-SPARQL, un moteur de traitement de flux RDF, qui intègre une technologie relativement mature permettant aux utilisateurs d'enregistrer des requêtes et de les exécuter en continu sur des fenêtres.

Le noyau de C-SPARQL est basé sur une architecture simple de boîte noire. Il est composé de deux composants principaux:

- **Moteur Continu** : Le premier composant concerne la gestion de flux et est chargé d'enregistrer les requêtes dans le système et d'appliquer la fenêtre spécifiée sur les flux de données.

| System | Report policy | Streaming operator | Window operator | Sequencing operator | Consumption policy |
|---|---|---|---|---|---|
| CQELS | Content change | Istream | Sliding | SEQ | $\mathcal{P}^u$ |
| $SPARQL_{stream}$ | Window close | Rstream, Dstream, Istream | Sliding | - | $\mathcal{P}^u$ |
| C-SPARQL | Window close | Rstream | Sliding | $SEQ^n$ (timestamp function) | $\mathcal{P}^u$ |
| EP-SPARQL (unrestricted) | Content change | Rstream | Landmark | SEQ | $\mathcal{P}^u$ |
| EP-SPARQL (chronological) | Content change | Rstream | Landmark | $SEQ^c$ | $\mathcal{P}^n$ |
| EP-SPARQL (recent) | Content change | Rstream | Landmark | $SEQ^r$ | $\mathcal{P}^n$ |

FIGURE 2.3: Comparaison des Langages de Requêtes sur Flux Sémantiques

- **Moteur SPARQL** : Lorsque les données dans la fenêtre ont été extraites, la partie SPARQL de la requête est exécutée à l'aide du moteur SPARQL. Techniquement, C-SPARQL est construit sur des frameworks bien connus tels que ESPER [7], un moteur de traitement d'événements largement utilisé pour la gestion de flux, et Jena [8], un framework Web sémantique populaire pour le langage de programmation Java.

C-SPARQL propose ses propres extensions de langage pour interroger périodiquement des triplets annotés dans le temps. Ces extensions consistent en un ensemble de mots-clés tels que `RANGE` qui contrôle le mécanisme de la fenêtre en fonction de l'heure ou du nombre de triplets, et `STEP` qui indique la fréquence à laquelle la requête doit être exécutée. Les opérateurs SPARQL 1.1 standard peuvent être utilisés dans les fenêtres telles que l'agrégation, la commande et la comparaison. C-SPARQL produit la même sortie que les requêtes SPARQL mais mises à jour dans chaque exécution de requête. Il prend en charge le chaînage de requêtes où C-SPARQL peut produire de nouveaux flux RDF en utilisant les requêtes CONSTRUCT et DESCRIBE, qui à leur tour peuvent être l'entrée de la requête suivante.

[7] http://esper.codehaus.org/
[8] https://jena.apache.org/

## 2.6 Systèmes de Traitement de Flux Numériques

De nombreux efforts ont été déployés pour développer de puissants moteurs de traitement de flux, qui répondent aux problèmes de faible latence, de tolérance aux pannes et d'évolutivité. Nous décrirons ensuite les moteurs de traitement de flux en temps réel (RSP) les plus efficaces et les plus populaires.

### 2.6.1 Apache Storm

Apache Storm [Toshniwal et al., 2014a], développé par BackType (société rachetée par Twitte), est un framework sous licence libre qui fournit une collection d'outils pour distribuer massivement des calculs et des opérations en parallèle sur un cluster industriel de plusieurs dizaines de machines. Il contient des composants performants tels que ZooKeeper [9] pour la gestion de cluster, ZeroMQ [10] pour la gestion de messages massifs en temps réel, et Kafka [11] pour l'organisation des données en file d'attente.

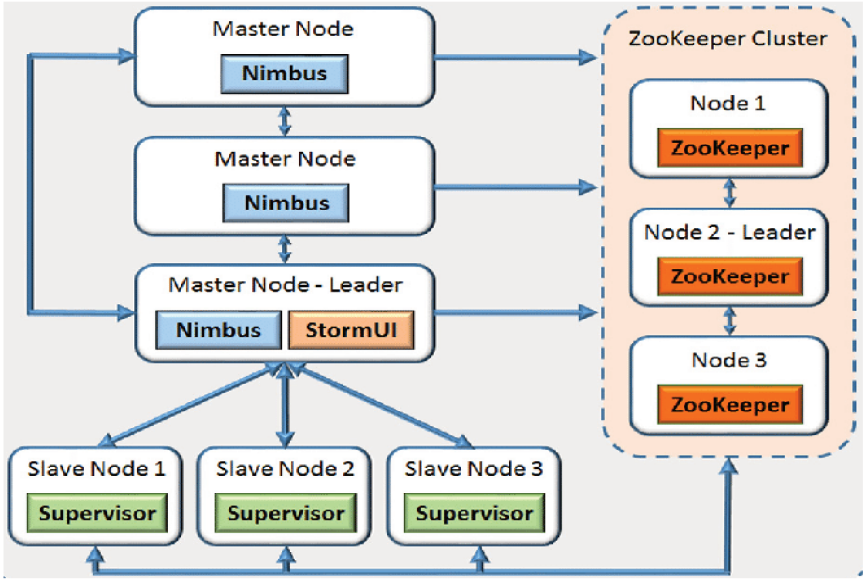


FIGURE 2.4: Architecture Storm

Storm traite le flux de tuples circulant dans des topologies. une topologie est un graphe orienté contenant des sommets, chacun d'entre eux est un nœud de calcul qui représente le flux de données entre les sommets. Les sommets sont divisés en deux ensembles, les spouts et les bolts. Les spouts sont des sources de tuples pour la topologie et les bolts permettent le traitement des tuples entrants. Storm s'exécute sur l'architecture abstraite, la topologie est transmise à un noeud maître central, puis les tâches sont distribuées sur des nœuds de travail dans un mode de traitement de

[9] ZooKeeper: https://zookeeper.apache.org/
[10] ZeroMQ: http://zeromq.org/
[11] Apache Kafka: https://kafka.apache.org/

cluster. Les données sont organisées à partir du spout producteur qui déversé les informations vers un bolt consommateur [**storm**].

Chaque noeud se compose de plusieurs threads d'exécution, un superviseur communique avec le noeud principal et l'état de cluster entier est stocké dans le serveur Zookeeper cité plus haut. Afin de créer une topologie Storm en utilisant n'importe quel langage de programmation, on utilise un Nimbus (noeud maître) qui définit une topologie contenant l'architecture en spouts et en bolts. Storm fournit deux types de garanties sémantiques:

- **Au moins une fois** : En augmentant la topologie avec un bolt de reconnaissance des opérations (ACKer) qui suit le graphe acyclique des données pour chaque tuple émis par un spout.
- **Au plus une fois** : En désactivant le mécanisme de reconnaissance dans le système.

### 2.6.2 Apache Spark

MapReduce et ses variantes ont rencontré le succès dans la mise en œuvre d'applications à grande échelle de données. Néanmoins, étant donné qu'il est construit autour d'un modèle de flux de données acyclique, il n'est pas populaire pour d'autres applications. Spark [Zaharia et al., 2010] a été créé pour les applications qui réutilisent un ensemble de données fonctionnant sur plusieurs opérations parallèles.

Spark fournit deux abstractions principales de la programmation parallèle: RDD et opérations parallèles. RDD [Zaharia et al., 2012] est une collection d'objets en lecture seule, partitionnées sur un ensemble de machines qui peuvent être reconstruites si une partition est perdue, un bloc de RDD contient suffisamment d'informations pour calculer l'ensemble du RDD.

Dans Spark, chaque RDD est représenté comme un objet Scala et peut être construit de l'une des quatre façons suivantes: **(1)** à partir d'un fichier, **(2)** paralléliser la collection Scala, **(3)** transférer un RDD existant, **(4)** modifier la persistance d'un RDD existant. Les opérations parallèles sont des opérations qui peuvent être effectuées sur les RDD, et comprennent les fonctions reduce, collect et foreach. Spark a deux variables partagées: Les variables de diffusion qui sont des copies pour chacun des workers, et les accumulateurs qui sont des variables que les travailleurs peuvent seulement "ajouter" à l'aide d'une opération associative.

Spark a été construit au-dessus de Mesos pour permettre à Spark de fonctionner parallèlement aux systèmes de cluster existants et de partager des données avec eux. Un RDD est stocké en tant que chaîne d'objets, où chaque objet contient un pointeur

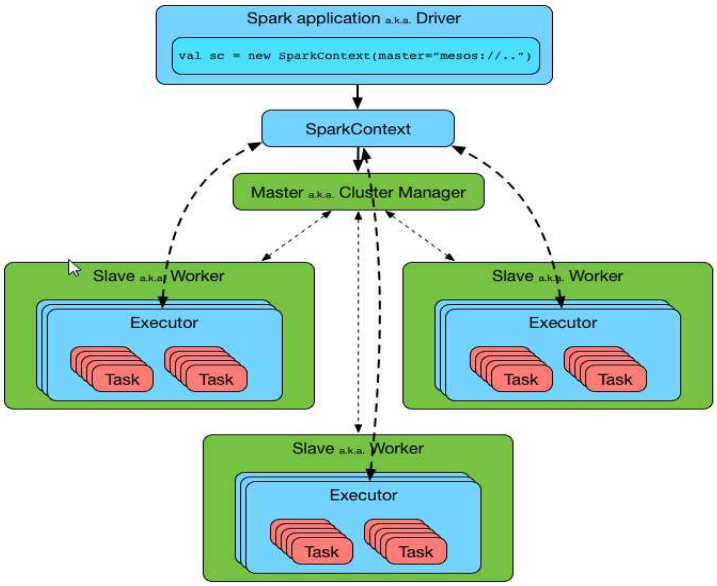


FIGURE 2.5: Architecture Spark

vers son parent et comment le parent a été transformé. De plus, chacun de ces objets implémente une interface simple qui consiste en trois opérations: getPartition, getIterator et getPreferredLocations.

Lorsqu'une opération parallèle est appelée, Spark crée une tâche pour traiter chaque partition de l'ensemble de données, chaque tâche est envoyée à son emplacement préféré à l'aide d'un mécanisme appelé planification de délai. Une fois lancé, getPartition est utilisé pour lire une partition. Une telle conception rend les fautes faciles à manipuler, surtout avec la mise en cache existante. Pour expédier des fermetures aux travailleurs, Spark s'appuie sur le fait que les objets Scala sont compatibles avec les objets Java, et peut être sérialisé en utilisant la sérialisation Java.

Les deux types de variables partagées sont implémentés à l'aide de classes avec des formats de sérialisation personnalisés. Lors de la création d'une variable de diffusion, il est enregistré dans un fichier dans un système de fichiers partagé, le formulaire sérialisé sera le chemin d'accès de ce fichier. Chaque fois qu'un worker demande cet objet de diffusion, Spark vérifie s'il est disponible dans le cache et le lit, sinon il le lit dans le système de fichiers. Le stockage HDFS est utilisé initialement pour les variables de diffusion, mais récemment, il est stocké en mémoire.

La variable partagée Accumulator est implémentée en lui donnant un identifiant unique lorsqu'elle est créée, la forme sérialisée contenant l'ID. Une copie séparée de l'accumulateur pour chaque thread exécute une tâche en utilisant des variables locales thread. Après chaque exécution de la tâche, le worker envoie un message au programme contenant les mises à jour effectuées sur les différents accumulateurs. Le programme applique les mises à jour de chaque partition de chaque opération une seule fois pour éviter le double comptage lorsque les tâches sont ré-exécutées en raison d'échecs.

### 2.6.3 Spark streaming

Comme indiqué précédemment, Spark permet de traiter des données orignellement figées à un instant T. A l'opposé,Spark Streaming s'occupe des opérations sur des flux de données continues c'est à dire au fur et à mesure de leur arrivée.

Au départ, un contexte de base est introduit avec une durée, ensuite le framework va rassembler des tuples pendant cette durée afin d'en fabriquer un RDD. Ce protocole d'accumulation et de production de RDD va se faire tant que le programme est lancé. Par conséquent, Spark Streaming est habituellement nommé framework de micro-batches contrairement à un mode de traitement des évènements de manière unitaire.


FIGURE 2.6: Workflow Spark Streaming

Si Storm permet un traitement en temps-réel des évènements, Spark Streaming en revanche ajoute un délai entre l'arrivée d'un message et son traitement, d'où l'appelation de micro-batch. Ainsi, il garantit un traitement des tuples en mode "exactly once" (chaque message est délivré une et une seule fois au programme, sans perte de messages), et at least once en conditions dégradées (un message peut être délivré plusieurs fois, mais toujours sans pertes). Storm permet, lui, de régler le niveau de garantie mais, pour optimiser les performances, le mode at most once (chaque message est délivré au maximum une fois mais des pertes sont possibles) doit être utilisé.

Enfin, l'API de Spark Streaming est identique à l'API classique de Spark. Il est ainsi possible de manipuler des flux de données de la même manière que l'on manipule des données figées. Spark Streaming étant destiné à traiter des données qui arrivent en continu, il est nécessaire de choisir une source de données adaptée. Il est donc préféreable de choisir des sources ouvrant une socket réseau et restant en écoute. Il est également possible d'implémenter une source de données sur mesure en étendant la classe Receiver.

### 2.6.4 Apache Flink

Apache Flink [Carbone et al., 2015] suit un paradigme qui englobe le traitement des flux de données en tant que modèle unifiant pour l'analyse en temps réel, les flux continus et le traitement par lots dans le modèle de programmation et dans le moteur d'exécution. Originaire du projet Stratosphere, Flink est un projet de haut

niveau de la Fondation Apache Software qui est développé et soutenu par une communauté grande et vivante (composée de plus de 180 contributeurs open-source au moment de la rédaction de ce document). Il est actuellement utilisé en production dans plusieurs entreprises d'envergure internationale.

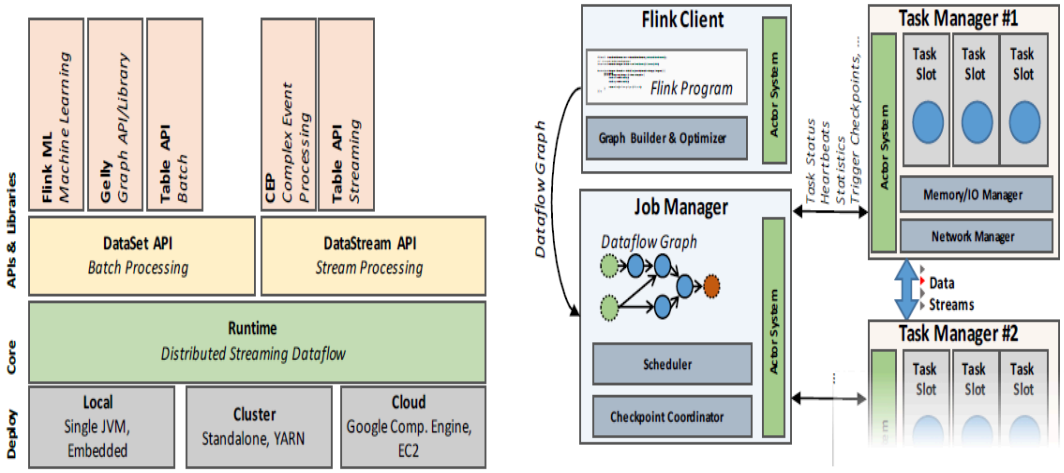


FIGURE 2.7: Architecture Apache Flink

En combinaison avec des files d'attente de messages qui autorisent une réexécution quasi-arbitraire de flux de données (comme Apache Kafka ou Amazon Kinesis), les programmes de traitement de flux ne font aucune différence entre le traitement des derniers événements en temps réel. Au lieu de cela, ces différents types de calculs commencent simplement leur traitement à différents points dans le flux continu, et maintiennent différentes formes d'état pendant le calcul. Grâce à un mécanisme de fenêtrage très flexible, Flink permet de calculer des résultats précoces et approximatifs, ainsi que des résultats précis et retardés dans la même opération, évitant ainsi de devoir combiner différents systèmes pour les deux cas d'utilisation.

Flink prend en charge différentes notions de temps (événement-temps, ingestion-temps, temps de traitement) afin de donner aux programmeurs une grande flexibilité dans la définition de la manière dont les événements doivent être corrélés. Dans le même temps, Flink reconnaît qu'il y a, et qu'il y aura, un besoin de traitement par lots dédié (traitant des ensembles de données statiques). Les requêtes complexes sur les données statiques sont toujours une bonne correspondance pour une abstraction de traitement par lots. En outre, le traitement par lots est toujours nécessaire à la fois pour les implémentations héritées de l'utilisation de la diffusion en continu.

Pour prendre en charge les cas d'utilisation par lots, Flink dispose d'une API spécialisée pour traiter les ensembles de données statiques. Il utilise des structures de données et des algorithmes spécialisés pour les versions batch des opérateurs, comme Join ou Grouping. Le résultat est que Flink se présente comme un framework de traitement par lots à part entière et efficace au-dessus d'un environnement d'exécution en continu, y compris des bibliothèques pour l'analyse graphique et l'apprentissage automatique.

### 2.6.5 Google Cloud Dataflow

Cloud dataflow[Tyler Akidau, 2013a] a été développé chez Google pour surmonter les limitations des moteurs existants. Par exemple, les systèmes de traitement par lots souffrent de latence lors de la collecte de toutes les données d'entrée dans un lot avant le traitement. Pour beaucoup d'autres systèmes de diffusion en continu, y compris Aurora, TelegraphCQ, Niagara et Esper, on ne sait toujours pas comment la tolérance aux pannes sera garantie à grande échelle.

Les systèmes qui offrent l'évolutivité et la tolérance aux pannes sont insuffisants pour les opérations de correction car ils ne peuvent fournir qu'une seule sémantique (Storm, Samza, Pulser). Les autres systèmes n'ont pas les primitives temporelles nécessaires pour le fenêtrage ou fournissent un fenêtrage limité à un tuple, ou des fenêtres basées sur le traitement (Spark, Sonora, Trident). Les systèmes de streaming MillWheel et Spark Streaming manquent de modèles de programmation de haut niveau.

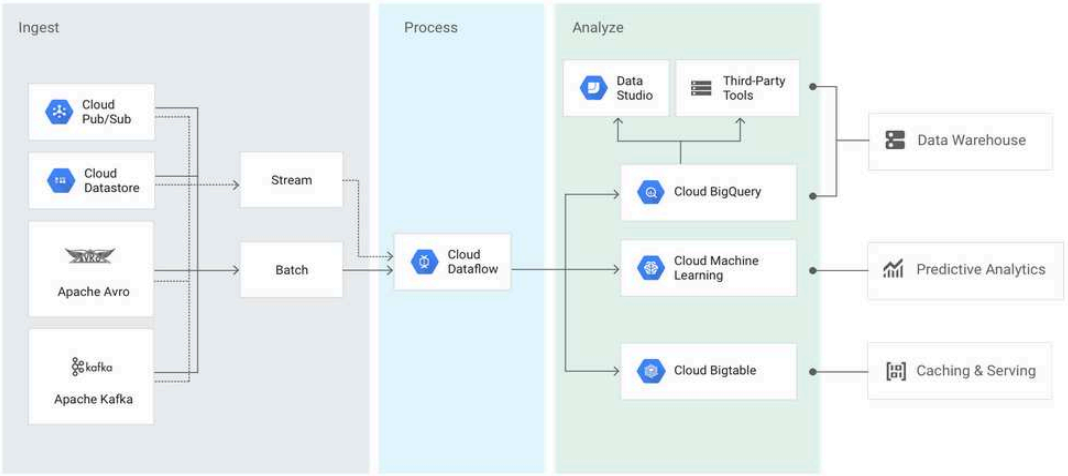


FIGURE 2.8: Architecture Google Cloud Dataflow

La contribution principale de Cloud Dataflow se situe dans le calcul des résultats ordonnés par l'événement, les fenêtres temporelles, les caractéristiques des données elles-mêmes avec la correction, la latence et le coût ajustable sur un large spectre de combinaisons. Ce framework permet de décomposer la mise en œuvre du workflow à travers quatre dimensions connexes et séparer la notion logique du processus de l'implémentation sous-jacente.

Cloud Dataflow fournit deux transformations de base qui fonctionnent sur les paires (clé, valeur): ParDo et GroupByKey. L'opération ParDo fonctionne par élément sur chaque élément d'entrée et se traduit donc naturellement par des données illimitées. L'opération GroupByKey, d'autre part, recueille toutes les données pour une clé donnée avant de les envoyer en aval pour la réduction. Si la source d'entrée est illimitée, la solution commune à ce problème, qui est le fait que l'exhaustivité des données ne sera jamais connue, est de fenêtrer les données.

Cloud Dataflow fournit également des déclencheurs, qui sont des mécanismes pour stimuler la production de résultats en réponse à des signaux internes ou externes. Le modèle de flux de données prend en charge des implémentations de déclencheurs prédéfinies pour le déclenchement aux estimations d'achèvement, il prend également en charge la composition des déclencheurs en combinaisons logiques. De plus, les utilisateurs peuvent définir leurs propres déclencheurs. L'implémentation du modèle a été faite en interne dans JavaFlume [Craig Chambers, 2013] avec MillWheel [Tyler Akidau, 2013b], où MillWheel est utilisé comme moteur d'exécution sous-jacent pour le mode streaming. Le code de base du fenêtrage et du déclenchement est écrit pour être assez général, et une partie importante est partagée entre les implémentations par lots et en streaming.

### 2.6.6 Autres moteurs de traitement de flux

InfoSphere Streams est le produit phare d'IBM pour le traitement de flux. Il offre un serveur d'événements hautement évolutif, des fonctionnalités d'intégration et d'autres fonctionnalités nécessaires à la mise en œuvre de cas d'utilisation de traitement de flux.

TIBCO StreamBase est un système hautes performances pour la création rapide d'applications qui analysent et agissent sur des données de streaming en temps réel. L'objectif de StreamBase est d'offrir un produit qui aide les développeurs à créer rapidement des systèmes en temps réel et à les déployer facilement. La base StreamBase LiveView est un entrepôt de données en continu qui consomme des données provenant de sources de données en temps réel, crée un entrepôt de données en mémoire et fournit des résultats de requête et des alertes aux utilisateurs finaux.

Apache Samza est un processeur d'infrastructure de traitement de flux distribués, récemment soutenu par LinkedIn. Esper est un framework majeur pour Java et .NET. AWS Kinesis est un service de cloud géré d'Amazon pour le traitement en temps réel des données en continu. Il est profondément intégré avec d'autres services cloud AWS tels que S3, Redshift ou DynamoDB. Enfin, DataTorrent est une plateforme de streaming en temps réel fonctionnant nativement sur Hadoop.

## 2.7 Systèmes de Traitement de flux sémantiques

Au cours des dernières années, les flux de données dynamiques ont suscité un intérêt considérable au sein de la communauté Web sémantique. Le traitement de ces flux a récemment fait l'objet de systèmes RSP basés principalement sur l'exécution centralisée. Reconnaissant les limitations de scalabilité des systèmes mono-machine, les efforts ont reposé sur des frameworks de traitement de flux génériques pour distribuer des requêtes sur un cluster de machines.

Par exemple, CQELS-Cloud [Phuoc et al., 2013], se compose d'un coordinateur d'exécution et de plusieurs conteneurs d'opérateurs. Le coordinateur d'exécution est utilisé pour distribuer les tâches de traitement et chaque conteneur d'opérateur est utilisé pour exécuter une seule opération telle que l'union, l'agrégation, etc. Bien que l'approche semble intéressante, le système n'est pas assez générique et ne permet pas aux utilisateurs finaux de définir des requêtes personnalisées. En effet, à notre connaissance, CQELS Cloud n'est pas open source ce qui entrave son application dans de nombreux cas d'utilisation.

En outre, selon le lien [12] fourni par l'équipe CQELS, les utilisateurs doivent modifier le code source afin de définir leurs propres requêtes et données d'entrée. Ils ont également besoin de modifier plusieurs paramètres (par exemple, nombre d'exécuteurs pour chaque tâche, type d'agrégation, nombre de tampons de sortie) ce qui n'est pas pratique pour les applications industrielles. Comme nous ne pouvons pas accéder au code source ou exécuter nos requêtes, nous n'avons pas considéré ce système dans nos expériences.

Un autre moteur RSP appelé KATTS [Fischer, Scharrenbach, and Bernstein, 2013] a été proposé sur la base d'un algorithme de partitionnement de graphes pour optimiser la bande passante du réseau et distribuer les requêtes. Toutefois, il ne prend pas en charge la syntaxe SPARQL et les utilisateurs sont obligés de transformer une requête en un format d'arborescence XML. En outre, plusieurs opérateurs SPARQL ne sont pas supportés comme `OPTIONAL`, `LIMIT`, `SUM` pour n'en nommer que quelques-uns. L'interrogation de données statiques n'est également pas possible.

[12] https://code.google.com/archive/p/cqels/wikis/CQELSCloud.wiki

# Chapter 3

# Plateforme WAVES

WAVES [1] est un moteur RSP distribué construit sur des frameworks big data populaires, il supporte les requêtes continues sur les données de streaming basées sur les événements. Il a été développé pour pallier à certaines insuffisances de systèmes existants présentant des problèmes d'évolutivité, de latence et l'absence du support de requêtes SPARQL.

## 3.1 Architecture : Modulaire, Tolérante aux pannes, Temps réel

En entrée, le système traite des données dynamiques basées sur les événements atomiques arrivant en continu mais supporte également l'enrichissement par des données statiques. Il s'appuye sur des composants distribués et robustes bien installés sur le marché des hautes technologies dont:

- Une base de données mémoire à écriture/lecture très rapide [ *Redis, Dec 2015*],
- Un broker de messagerie en file d'attente supportant le mode distribué et ne perdant pas les messages en entrée en cas de panne [ Wang et al., 2015],
- Un framework de distribution des calculs sur un cluster industriel [ Toshniwal et al., 2014b].

La figure 3.1 représente une vue d'ensemble de l'architecture WAVES.

Un scénario typique de notre système débute par l'obtention des mesures capturées sur un réseau de capteurs donné. Les événements de ces flux sont ensuite nettoyés et filtrés, puis transformés en sérialisation RDF compressée avec la possibilité d'être échantillonnés avant la transformation. Les flux résultants sont soumis au broker de messages Kafka pour être remis aux composants principaux de calcul/raisonnement. À l'autre extrémité du broker Kafka, l'unité de calcul/raisonnement distribuée (nommée Smart'Op sur le schéma d'architecture 3.1) lit les événements RDF

[1] Détails sur le projet WAVES disponible sur `https://waves-rsp.org/`

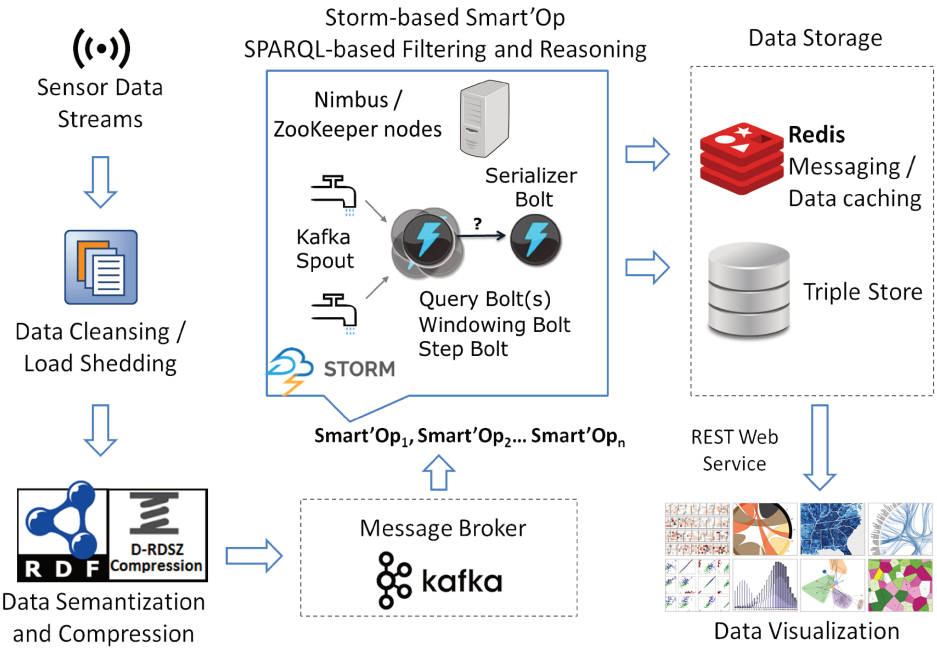


FIGURE 3.1: Architecture Logique WAVES

compressés à partir des évènements Kafka. Le résultat est la création d'un ensemble de données traitées par fenêtres temporelles selon l'ordre d'arrivée. Le but du fenêtrage est de pouvoir organiser les messages atomiques par horodatage afin de réaliser un traitement temps-réel et de garantir des résultats cohérents pour le cas d'usage.

Une fois qu'un point d'exécution est atteint, la Smart'Op évalue la requête SPARQL configurée par rapport aux ensembles de données. Un exemple de requête SPARQL est détaillé sur la figure 3.2, d'autres requêtes sont disponibles en **Annexe A** . Le but de la requête qui suit est de détecter le flux dont la valeur d'observation est supérieure à la moyenne des observations passées. Elle permet ainsi de calculer la moyenne des valeurs d'observation des jours passés pour les capteurs de flux et renvoie l'observation qui dépasse cette valeur spécifique.

Le résultat est la génération de nouveaux événements RDF compressés qui sont envoyés à Kafka pour être traités par une autre Smart'Ops, archivés ou affichés à l'utilisateur final à l'aide d'un module de visualisation dont le but est de faciliter l'interprétation des flux analysés à travers différents types de visualisations dynamiques.

L'étape liée au nettoyage des données a lieu avant la la sérialisation RDF et immédiatement après la réception des capteurs. Le module qui aborde cette partie est facultatif et repose sur des modèles statistiques tels que l'analyse des valeurs aberrantes basée sur des seuils min-max. WAVES est livré avec une API Java [2] pour aider les développeurs à réaliser les extensions désirées. Enfin, l'utilisateur final

[2]WAVES Java API disponible sur https://waves-rsp.org/api/index.html

```
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX qudt:<http://data.nasa.gov/qudt/owl/qudt#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cuahsi: <http://his.cuahsi.org/ontology/cuahsi#>

CONSTRUCT{
    ?event ssn:isProducedBy ?sensor.
    ?event ssn:startTime ?time.
    ?observation qudt:numericValue ?value.
    ?sensor ssn:observes ?flow.
}
WHERE {
    ?event ssn:isProducedBy ?sensor;
           ssn:hasValue ?observation;
           ssn:startTime ?time.
    ?observation qudt:numericValue ?value.
    ?sensor ssn:observes ?flow.
}
GROUP BY ?flow
HAVING ( ?value > AVG(?value) )
```

FIGURE 3.2: Exemple de Requête SPARQL dans WAVES

peut facilement définir les paramètres du système en fonction d'un fichier de configuration RDF qui inclut des paramètres tels que la taille de la fenêtre, le pas, la requête continue, etc. Une partie de ce fichier de configuration est détaillée sur la figure 3.3, l'ensemble du fichier est disponible en **Annexe B**.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix waves: <http://www.waves-rsp.org/configuration#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://localhost:9091/waves/versailles/> .

<http://localhost:9091/waves/first-tests> {
    <cleaner/3> a waves:Cleaner ;
        waves:consumesStream <rawStream/2> ;
        waves:id "3"^^xsd:int ;
        waves:installation <installation> ;
        waves:producesStream <rawStream/4> .

    <converter/7> a waves:Converter ;
        waves:consumesStream <rawStream/6> ;
        waves:id "7"^^xsd:int ;
        waves:installation <installation> ;
        waves:model """@prefix ssn:<http://purl.oclc.org/NET/ssnx/ssn#> . \r
@prefix qudt:<http://data.nasa.gov/qudt/owl/qudt#> . \r
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> . \r
@prefix waves:<http://example.org/waves#> .\r
$eventId \r
    a ssn:SensorOutput;\r
    ssn:isProducedBy $uri(?id);\r
    ssn:hasValue $bnode(?obsId);\r
    ssn:startTime $timestamp .\r
```

FIGURE 3.3: Extrait du Fichier de Configuration sous format .trig

### 3.1.1 Modélisation Evenementielle et Temporelle

Suivant la philosophie des moteurs RSP, WAVES modélise les flux de données comme un flux illimité d'événements. En effet,chacun de ces événements est considéré comme un ensemble de triplets RDF associés à un horodatage, et la structure d'un événement représente une observation enregistrée par le capteur et associée à un horodatage. Contrairement à de nombreux moteurs RSP, WAVES ne considère pas les événements comme un ensemble de triplets RDF horodatés indépendants, mais comme un graphe d'évènements atomiques qui ne peut pas être divisé. Par conséquent, le système évalue la requête continue par rapport à l'ensemble des événements dans une fenêtre donnée. Cette stratégie permet de faire notamment aux problèmes de débit rencontrés par de nombreux moteurs RSP Kolchin et al., 2016.

Étant conçu comme un moteur de traitement de flux en temps réel, consommant des mesures de capteur en cours de production, de nombreux cas d'utilisation nécessitent le traitement d'anciennes données enregistrées sur des fichiers, pour lesquels WAVES utilise sa propre référence temporelle. Cette référence temporelle est une horloge distribuée, synchronisée, décalée et accélérée:

- décalée pour alignerl'horloge de WAVES sur l'horodatage des événements lus,
- accélérée pour être en mesure de traiter des mois de données héritées en quelques minutes.

Les données d'horodatage provenant de nombreux fichiers d'entrée sont distribuées sur un cluster et traitées en même temps que les données statiques à l'aide des références temporelles.

Apache Kafka est de plus en plus populaire et largement utilisé dans les moteurs de traitement de flux. Il peut être connecté à la plupart des frameworks de traitement de flux open-source, y compris Apache Spark Streaming et Apache Storm. C'est un composant de gestion de messagerie distribué basé sur l'approche de publication-abonnement. Les événements sont récupérés à partir de Kafka par un composant Smart'Op basé sur Storm, lui-même composé d'un ensemble de nœuds distribués mis en œuvre par une topologie spécifique de Bolts et de Spouts.

En effet, le Spout est une source des flux qui peut lire des données provenant d'une source externe comme le broker Kafka. Quant au Bolt, il s'agit d'une unité de traitement logique qui effectue tous types de calculs tels que le filtrage, la jointure, l'agrégation ou l'interaction avec des entrepôts de données externes. Les deux Bolt et Spout s'exécutent en tant que tâches réparties sur un cluster Storm et chaque tâche correspond à un thread d'exécution. Les topologies sont exécutées sur un ou plusieurs workers, un worker étant un processus physique s'exécutant sur une JVM qui exécute un sous-ensemble de tâches pour la topologie.

La plate-forme est conçue pour être tolérante aux pannes, ce qui permet un fonctionnement continu même en cas de défaillance de composants potentiels. Pour ce faire, WAVES s'appuie sur Storm comme moteur de traitement principal qui garantit un fonctionnement infini du processus, avec un redémarrage automatique des workers en cas de panne, grâce au mécanisme de reconnaissance (*"ACKing"*). Le cœur de Storm vérifie efficacement si un tuple source est entièrement traité ou pas, ainsi il relit les tuples ayant rencontré un échec lors du traitement dans un délai configurable de topologie. Par conséquent, Storm offre une garantie de traitement *"at-least once"*, c'est à dire qu'un message quelconque peut être délivré plusieurs fois, mais toujours sans pertes d'informations lors du passage entre spout et bolts ainsi qu'au cours du traitement par le bolt.

Chaque topic Kafka représente un flux d'événements unique auquel chaque canal d'une topologie WAVES s'abonne. En d'autres termes, pour évaluer une requête, le nombre de canaux devant exister est le même que le nombre de flux d'entrée. Pour chacun de ces flux, un verrou de fenêtrage est chargé de stocker les événements reçus jusqu'à ce qu'un point d'exécution (étape) soit atteint. En raison du fait que les topologies WAVES sont distribuées, un besoin de stockage partagé est apparu. Pour répondre à ce besoin, le stockage en mémoire Redis est utilisé, il supporte nativement les structures de données riches (telles que les ensembles triés), les requêtes à distance, la réplication, et la persistance périodique sur disque.

Une fois qu'une étape de traitement est atteinte, le bolt Storm exécutant la requête SPARQL rassemble les événements dans la fenêtre courante pour chacun des flux d'entrée de l'entrepôt. Ensuite, il le décompresse pour enrichir un triple store RDF en mémoire, et déclenche l'exécution de la requête SPARQL. Il génére ainsi un événement pour l'étape en cours à partir des résultats de la requête. Enfin, après le codage et la compression RDF, le Serializer écrit cet événement dans le broker Kafka associé à un flux de sortie de la topologie.

## 3.2 Cas d'usage : Jeux de données, Modèle Conceptuel et ontologie

WAVES se concentre sur un cas d'utilisation spécifique basé sur le traitement de données dans des capteurs interconnectés à grande échelle. Des capteurs sont déployés dans un système de gestion de l'eau pour surveiller différents phénomènes physiques ou propriétés chimiques. Par conséquent, non seulement nous avons des flux de données hétérogènes, mais la quantité de données transmises augmente proportionnellement à la taille du réseau de capteurs et à la fréquence d'émission.

### 3.2.1 Jeux de Données

WAVES utilise des ensembles de données du monde réel décrivant différentes mesures d'eau capturées par des capteurs. Les valeurs de débit, de pression et de chlore sont des exemples de ces mesures. Chaque mesure est composée de la valeur de l'observation, de l'unité, de l'échelle, de l'horodatage et de l'identité du capteur. En outre, les fichiers RDF qui contiennent toutes les informations statiques décrivant la topologie du système de distribution d'eau sont disponibles et potentiellement utilisables.

Les réseaux de capteurs sont déployés dans des régions que nous appelons des zones de consommation. Chaque capteur surveille une condition spécifique et a une position fixe. Cependant, il peut être rattaché à plusieurs zones de consommation, en particulier celles qui sont placées sur des frontières.

### 3.2.2 Modèle conceptuel et ontologie

Pour traiter de telles données hétérogènes, nous devons assurer l'unification syntaxique en utilisant un modèle de données spécifique (ici RDF) et un vocabulaire commun pour assurer l'interopérabilité. WAVES utilise les ontologies suivantes pour construire la représentation sémantique des données du réseau de capteurs:

- SSN [3] (Semantic Sensor Network): Cette ontologie décrit des capteurs, des observations et des concepts associés. Son but est de modéliser des capteurs, des systèmes, des processus et des observations. Cependant, SSN ne décrit pas les concepts de domaine, le temps, les emplacements, etc. Ils sont destinés à être inclus à partir d'autres ontologies.
- CUAHSI [4] (Consortium des universités pour l'avancement des sciences hydrologiques, Inc): WAVES utilise les concepts définis par cette ontologie hydrologique pour les phénomènes physiques ou propriétés chimiques observés par les capteurs.
- QUDT [5] (Quantités, Unités, Dimensions et Types de données Ontologies): QUDT est utilisé pour définir différentes unités de mesure.
- WGS84 [6] (World Geodetic System 1984): Les capteurs sont identifiés par une URI unique et ont des coordonnées géographiques. Ils appartiennent à des plates-formes qui correspondent à des zones de consommation. Nous utilisons WGS84 pour gérer les informations géographiques des capteurs.

La structure des événements est représentée dans la figure **??**.

[3] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[4] http://his.cuahsi.org/ontologyfiles.html/
[5] http://www.qudt.org/
[6] https://gis.icao.int/eganp/webpdf/REF08-Doc9674.pdf

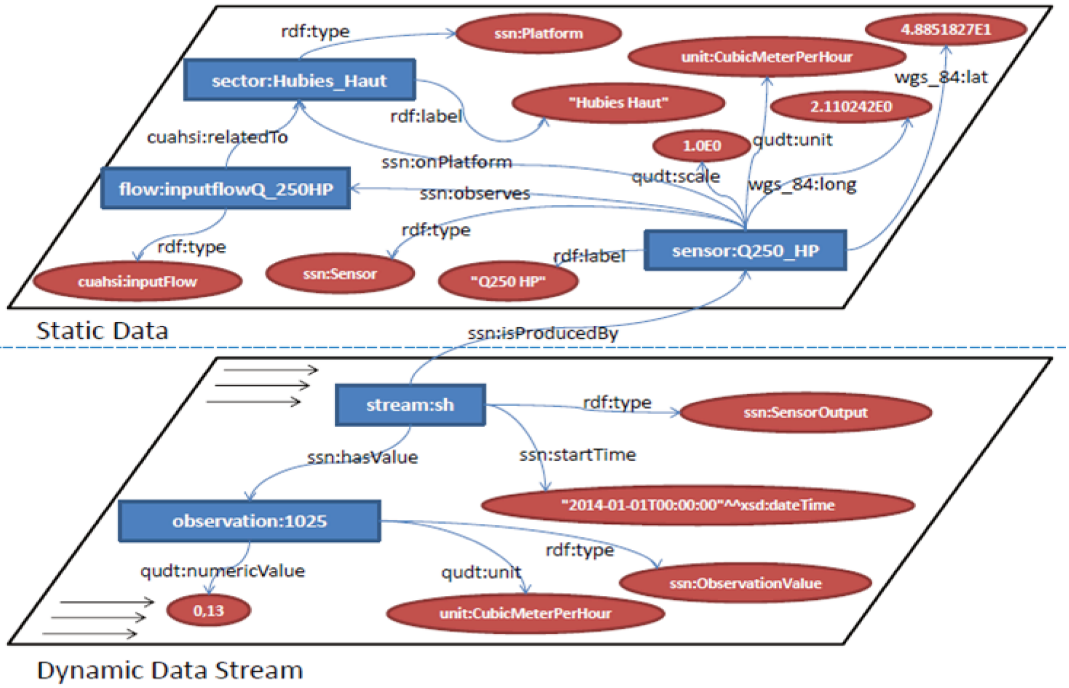


FIGURE 3.4: Modèle d'Evènement

## 3.3 Détection d'anomalies en mode distribué

Dans cette section, nous nous situons dans un contexte d'apprentissage non supervisé sur des séries temporelles. Une série chronologique est une collection d'observations d'éléments de données bien définis obtenus par des mesures répétées au fil du temps.

A partir du flux de données défini au préalable, nous avons utilisé une requête spécifique pour obtenir des séries temporelles pour la consommation sectorielle en agrégeant les valeurs. En parlant de séries chronologiques, nous devrions tenir compte de ses différentes composantes et modèles: tendance, saisonnalité, cycles particuliers. Une tendance est définie comme le mouvement à long terme dans une série chronologique sans effets liés au saisons. Cela peut être une longue augmentation ou une longue diminution.

### 3.3.1 Méthodologie

Le clustering est un processus de partitionnement d'un ensemble de données en sous-ensembles significatifs appelés clusters. C'est la méthode d'apprentissage non supervisée la plus courante. Dans notre cas, le regroupement a pour but de découvrir des modèles de distribution sur deux jours différents en regroupant les événements fermés au sein des clusters.

Tout d'abord, nous devons choisir un algorithme de clustering. Ensuite, nous essayons d'optimiser son paramètre pour avoir des partitions optimales. Après cela, chaque cluster est analysé pour déterminer si ses événements internes sont anormaux ou non. Enfin, les événements anormaux sont agrégés en anomalies.

Après avoir déterminé la méthode que nous utiliserons, l'étape suivante consiste à choisir le modèle le mieux adapté à nos besoins. Nous avons expérimenté différents types d'algorithmes de classification existants, en utilisant une recherche de grille pour trouver les bons paramètres qui conviennent à chaque modèle.

### 3.3.2 Hypothèses

Comme nous l'avons dit précédemment, nous supposons qu'en raison de la périodicité des séries chronologiques, les singularités peuvent être détectées sous forme de valeurs irrégulières. Ainsi, sur la base des données historiques de consommation pour un secteur, nous pouvons prédire la consommation actuelle et la comparer aux valeurs réelles. Au-delà d'un certain seuil, nous pouvons découvrir des anomalies.

Nous avons chercher à améliorer cette méthode traditionnelle en utilisant les connaissances de terrain. L'approche consiste à effectuer une comparaison de deux jours entre chaque secteur et les autres secteurs comparables. Au lieu d'utiliser la variation de consommation d'un secteur en deux jours (le jour en cours et la veille) pour trouver une éventuelle anomalie, nous exploitons la co-variation des consommations de deux secteurs entre le jour en cours et un jour de référence.

Nous avons pris des données d'une région (MACAO) composée de cinq secteurs. Pour chaque secteur, nous avons calculé la consommation des événements pour le jour en cours et le jour de référence. On peut donc associer la consommation pour obtenir les points comme dans la figure ref fig: secteurs-analyse.

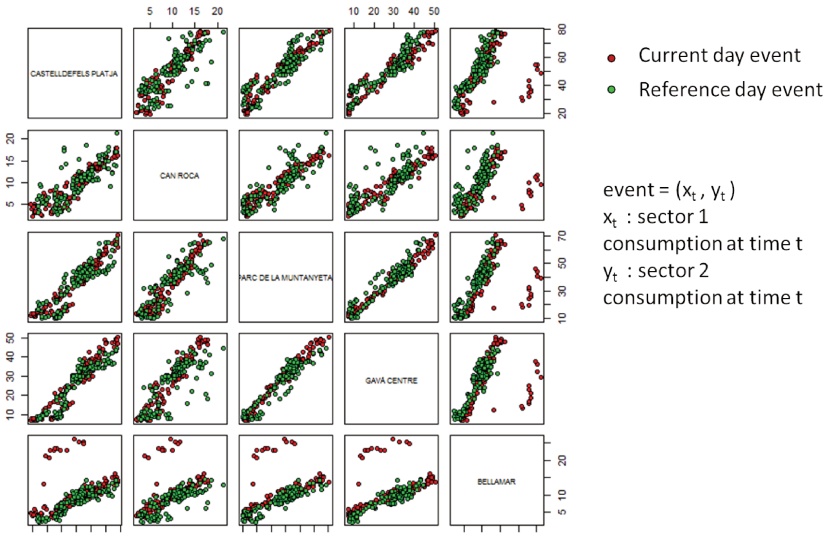


FIGURE 3.5: Exemple de fuite d'eau dans le secteur Bellamar

### 3.3.3 Implémentation

Dans cette section, nous décrivons comment la mise en œuvre du module de raisonnement a été effectuée.

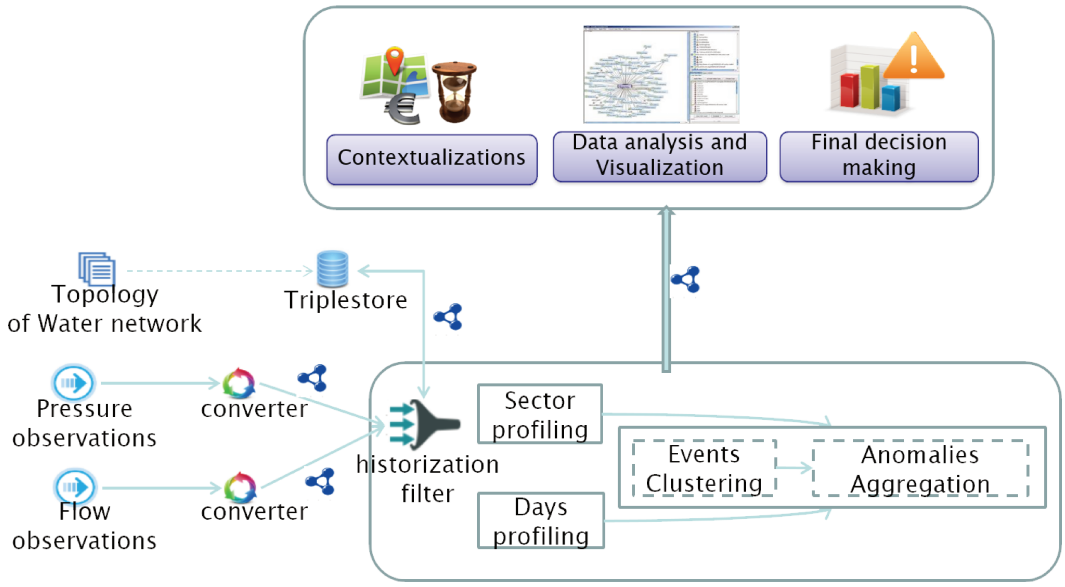


FIGURE 3.6: Topologie de Raisonnement WAVES

Notre module de détection des anomalies est divisé en deux parties:

- 1ère Etape: Analyser les données et générer des anomalies.
- 2nde Etape: Contextualiser et visualiser les anomalies potentielles pour prendre une décision finale.

Pour créer une topologie, le système utilise des propriétés sémantisées, publiées dans un triplestore externe, qui contient toute la configuration requise. Pour générer cette configuration, une interface graphique a été ajoutée ultérieurement.

Le module Convertisseur utilise un modèle d'événement et crée des données dynamiques contenant les observations. Ensuite, le filtre d'historisation consomme le flux, accède aux données statiques si nécessaire, génère des événements Waves. Enfin, la sortie est stockée dans un triplestore.

Les tâches sont effectuées périodiquement. Pour ce faire, nous utilisons une classe Java: TimerTask. Nous avons trois tâches. Ce ne sont que les différentes étapes du raisonnement expliquées précédemment: profilage sectoriel, profilage des jours et regroupement.

Tous les calculs sont réalisés dans les classes Python, car ce langage offre un ensemble complet de bibliothèques et des outils simples et efficaces pour l'exploration de données et l'analyse de données.

Pour interconnecter Java et Python, nous utilisons une bibliothèque appelée JEP [7].

[7] https://pypi.python.org/pypi/jep

Jep intègre CPython en Java via JNI et peut être utilisé en toute sécurité dans un environnement fortement threadé. En d'autres termes, JEP crée un interpréteur Python dans la JVM (Java Virtual Machine). Ce faisant, il est facile de passer des variables avec un type de données primitif entre les deux environnements. Nous avons essayé d'utiliser d'autres classes d'interpréteurs Python telles que Jython [8]. Cependant, nous avons rencontré quelques difficultés lors de l'utilisation de bibliothèques Python non natives.

Pour l'implémentation de modèles de clustering, nous avons utilisé deux bibliothèques de python différentes: Scikit learn [9] ets pyclustering [10]. Pour communiquer avec les triplestores, un package appelé RDFlib [11].

### 3.3.4 Modèles Utilisés

#### Kmeans

Kmeans est un regroupement basé sur un centroïde. L'algorithme fonctionne comme suit:
(1) Premièrement, en utilisant un nombre fixe et déterminé de groupes k, le centre des groupes est initialisé.
(2) Chaque point de données est attribué au groupe le plus proche
(3) Nous définissons la position de chaque groupe à la moyenne de tous les points de données appartenant à ce groupe
(4) Répétez les étapes 2-3 jusqu'à la convergence

Étant donné que k-means est un algorithme heuristique, les centres de cluster trouvés peuvent ne pas être optimaux. C'est-à-dire que l'algorithme est garanti pour converger sur un optimum local, mais pas nécessairement global. Les choix des centres initiaux affectent la qualité des résultats. Le paramètre à optimiser est le **nombre de clusters**. La faiblesse de cette méthode pour notre cas d'utilisation est que les clusters trouvés sont pairs.

#### DBScan

Le regroupement spatial basé sur la densité d'applications avec bruit Sander et al., 1998 est un algorithme de regroupement basé sur la densité. Son algorithme considère les grappes comme des zones de haute densité séparées par des zones de faible densité. Il y a deux parametres à l'algorithme, *min_sample* et *eps*, qui définissent formellement ce que nous entendons par dense. Tout cluster contient au moins

[8] http://www.jython.org/
[9] http://scikit-learn.org/stable/
[10] https://pypi.python.org/pypi/pyclusterings
[11] https://pypi.python.org/pypi/rdflib

*min_samples* points. eps décrit la distance maximale (rayon) à prendre en compte. Plus haut *min_samples* ou *eps* inférieurs indiquent une densité plus élevée nécessaire pour former un cluster. Par conséquent, les tailles de cluster sont inégales mais ont tendance à avoir la même densité.

#### OPTICS

Points de classement pour identifier la structure de classification Ankerst et al., 1999 est un algorithme permettant de trouver des groupes de densité dans des données spatiales. Il s'agit d'une amélioration de DBSCAN car elle répond à la principale faiblesse de DBSCAN: détecter les clusters significatifs dans des données de densité variable. Les paramètres à optimiser sont les mêmes que DBSCAN: *eps* et *minSamples*

#### Clustering agglomératif

Le regroupement agglomératif effectue un regroupement hiérarchique en utilisant une approche ascendante: chaque observation commence dans sa propre grappe et les grappes sont fusionnées successivement. Nous avons une hiérarchie de grappes représentée sous la forme d'un arbre (ou d'un dendrogramme). La classification hiérarchique [12] (également appelée analyse hiérarchique par grappes ou HCA) est une méthode d'analyse par grappes qui cherche à créer une hiérarchie de grappes.

#### ROCK clustering

Clustering RObust à l'aide de linKs **1999: RRC** est un clustering hiérarchique robuste qui utilise des liens et non des distances lors de la fusion de clusters. Définissons $link(p_i, p_j)$ le nombre de voisins communs entre $p_i$ et $p_j$. Si $link(p_i, p_j)$ est grand, il est probable que $p_i$ et $p_j$ soient dans le même cluster. La distance radiale pour déterminer le voisinage est le seul paramètre à varier

## 3.4 Evaluation

Dans le cadre d'un projet de recherche, l'évaluation fait partie intégrante du travail. L'évaluation est un processus qui examine de manière critique un programme. Dans ce chapitre, nous décrivons ce qui est évalué, comment réalisons-nous ce processus et enfin les résultats obtenus.

[12] https://en.wikipedia.org/wiki/Hierarchical_clustering

### 3.4.1 Critères d'évaluation

Il existe déjà un module de métriques pour tester les performances du système en termes d'exécution du temps, de consommation de mémoire, etc. Pour ce travail, nous avons effectué différentes expériences d'évaluation. Nous allons mesurer la précision et le rappel. La précision est la fraction d'instances récupérées pertinentes. Le rappel est la sensibilité du système. Il mesure la fraction des instances pertinentes qui sont extraites, la complétude si les résultats.

Dans tous les cas, une précision accrue diminuera le rappel.

- Précision et rappel sur les résultats de la requête sur le système Waves: Nous voulons mesurer l'exactitude des données dans chaque fenêtre temporelle. Cela signifie que chaque événement doit être dans la fenêtre de temps prévue. Nous devons garder à l'esprit que l'horodatage de l'événement (l'heure dans l'événement) est différent de l'horodatage du système (l'heure à laquelle le système a reçu l'événement).

  Lors de l'exécution du module de raisonnement, nous avons constaté qu'il manquait encore des données. Pour certains secteurs, nous n'avons obtenu qu'environ 85% des données sur les valeurs de consommation. Cela soulève une question sur la complétude des données. Nous devons tester les performances du moteur de requêtes continues.

- Précision et rappel sur la détection des anomalies pour évaluer le modèle. Ici, la performance du modèle sur la détection de l'anomalie est estimée. Les résultats ici peuvent également aider à trouver le modèle le plus approprié pour notre cas.

### 3.4.2 Méthodologie

Pour évaluer le moteur d'interrogation continu du système, nous utilisons Oracle YaBench. Son utilisation principale est principalement pour évaluer C-SPARQL ou CQELS.L'oracle a besoin de:

- Les données en streaming matérialisées dans un fichier au format très spécifique
- Le taux de pas et la taille de la fenêtre
- La requête
- Les résultats matérialisés formatés du moteur de requête continu

Pour cela, nous matérialisons les résultats de la requête par fenêtre au format correct. Tout d'abord, l'oracle exécutera la requête de manière statique. De cette façon, les

| | | (a) Scenario1 | | (b) Scenario 2 | | (c) Scenario 3 | |
|---|---|---|---|---|---|---|---|
| | | WAVES | C-SPARQL | WAVES | C-SPARQL | WAVES | C-SPARQL |
| Precision | Q1-2s/2s | **100%** | **100%** | **100%** | 94% | **98%** | 80% |
| | Q1-4s/1s | **100%** | **100%** | **100%** | 88% | **84%** | 78% |
| Recall | Q2-2s/2s | **100%** | 93% | **97%** | 95% | **79%** | 56% |
| | Q2-4s/1s | **100%** | 91% | **94%** | 84% | **72%** | 43% |

TABLE 3.1: Precision/recall results for WAVES and C-SPARQL for (a) s= 10 sensors, (b) s= 50 sensors and (c) s= 100 sensors.

résultats obtenus seront la référence. Nous utilisons cette référence pour évaluer les résultats matérialisés que nous avons obtenus auparavant.

Pour le scénario, nous avons essayé de souligner le système. Nous avons utilisé quatre requêtes différentes, devenant plus complexes en terme de complexité. Ensuite, nous créons un générateur de flux pour augmenter la charge de données dans chaque fenêtre temporelle.

Les résultats que nous présentons ici sont extraits de la comparaison entre les ondes et d'autres systèmes tels que C-SPARQL. Nous utilisons des données provenant progressivement de 10, 50 et 100 capteurs.

Concernant les requêtes, nous avons utilisé deux requêtes différentes avec une complexité croissante:

- *Requête 1 :* Renvoie la liste des capteurs ayant des mesures comprises entre 5 et 12, ainsi que l'horodatage d'observation.
- *Requête 2 :* Il correspond à la consommation globale représentée par la somme des flux d'entrée regroupés par l'horodatage d'observation.

Nous pouvons voir que dans la plupart des cas, WAVES a de bons résultats en termes de précision et de recall. Dans quelques cas, nous n'atteignons pas l'objectif de 100% sur l'ensemble des données. Maintenant, nous avons la confirmation qu'il peut y avoir une valeur manquante parmi les séries de consommation.

### 3.4.3 Résultats du raisonnement

Nous prenons les données de janvier et février 2015. Nous avons plus d'une centaine de capteurs répartis dans 11 secteurs. Chaque capteur envoie 4 événements par heure. Dans cette expérimentation, nous accélérons le temps pour que 2 mois de données soient effectués en quelques minutes.

TABLE 3.2: SUMMARY OF THE EVALUATION OF THE MODELS

| Models | KMeans | Agglomerative Clustering | OPTICS | DBSCAN | ROCK |
|---|---|---|---|---|---|
| Precision | 0.90 | 0.87 | 0.97 | 0.84 | 0.78 |
| Recall | 0.65 | 0.70 | 0.85 | 0.69 | 0.57 |
| $\alpha$-threshold | 0.60 | 0.80 | 0.75 | 0.80 | 0.78 |
| $\beta$-threshold | 0.45 | 0.54 | 0.32 | 0.44 | 0.78 |
| distance used | Euclidean | Cityblocks | Euclidean | Manhattan | - |

## 3.5 Synthèse

# Chapter 4

# RAMSSES

L'émergence de l'Internet des Objets et du traitement des flux force les organisations à grande échelle à considérer la détection des anomalies comme un élément clé de leur activité. L'utilisation de l'apprentissage automatique pour résoudre de tels cas d'utilisation complexes est généralement un processus lourd, coûteux, fastidieux et sujet aux erreurs. Il implique de nombreuses tâches allant du nettoyage des données à la réduction des dimensions, en passant par la sélection d'algorithmes et le réglage précis des paramètres. Il nécessite également la participation de divers experts tels que les statisticiens, les programmeurs et les testeurs.

Avec RAMSSES, nous supprimons le fardeau de ce pipeline et démontrons que ces tâches peuvent être automatisées. Notre système s'appuie sur une architecture Lambda Nathan Marz, 2015 basée sur Apache Spark pour analyser les données historiques, effectuer le nettoyage et gérer la malédiction de la dimension. Ensuite, il identifie les attributs les plus intéressants et utilise un générateur de requêtes sémantique continu exécuté sur des flux. Les données échantillonnées sont traitées par des méthodes d'apprentissage automatique sélectionnées pour détecter les anomalies, un processus itératif utilisant les annotations de l'utilisateur final améliore considérablement la précision du système.

Après une description des principaux composants de RAMSSES, les performances et la pertinence du système sont démontrées par une évaluation approfondie des ensembles de données réelles et synthétiques.

## 4.1 Concepts et principes

RAMSSESS a été conçu pour traiter à la fois des données massives dynamiques et statiques en utilisant une architecture distribuée tolérante aux pannes qui peut facilement évoluer. L'objectif principal est de faire face à de lourdes charges de données en temps réel et de calculer des modèles intensifs à des fins d'apprentissage automatique.

Pour répondre aux besoins d'un système robuste, tolérant aux pannes, capable de servir un large éventail de charges de travail et de cas d'utilisation, et dans lequel une faible latence des lectures et écritures est requise, nous avons basé notre système sur une architecture Lambda. Ce type d'architecture big data, dont un exemple est présent sur la figure 4.1, résout le problème des lourdes fonctions de calcul sur les données en temps réel en décomposant le problème en trois couches: batch, speed et serving.

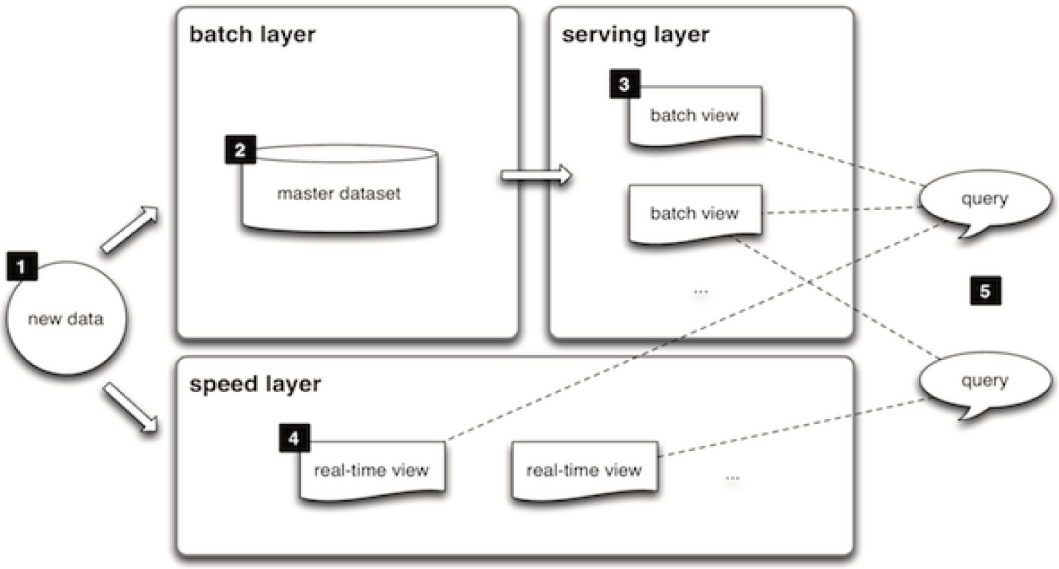


FIGURE 4.1: Exemple d'Architecture Lambda

En effet, l'architecture Lambda dispose des atouts suivants:

- Le traitement des données est précis et sans perte d'informations par la couche temps réel.
- L'introduction d'une nouvelle couche est compensée par la réduction des besoins en stockage.
- La possibilité de changer de données à des intervalles prédéfinis avec versionnement et conservation de l'historique via une écriture par lots.
- La tolérance aux erreurs manuelles dûe à la présence d'un réservoir de données brutes.
- L'amélioration du processus d'extraction des données et de lancement des algorithmes d'apprentissage automatique.
- Forte immuabilité et possibilité de recalcul lors du déroulement des différentes étapes du workflow.
- Une architecture efficace dans laquelle le traitement par lots et par flux fonctionnent ensemble pour répondre à plusieurs cas d'utilisation.

- L'exécution ad hoc de requêtes sur n'importe quel type de données afin d'en extraire des informations efficacement.

## 4.2 Architecture

Un scénario de bout en bout pour RAMSSES, comme décrit sur la figure 5.1 commence par le stockage de données chronologiques historiques d'un cas d'utilisation à des fins de pré-analyse. La mise en cache de lourdes charges de données nécessite un système de fichiers distribué robuste pour stocker et récupérer des fichiers en un temps pertinent. En raison de sa capacité et de sa fiabilité, Hadoop Distributed File System (HDFS) [Shvachko et al., 2010] est un système de stockage parfaitement adapté aux besoins de Big Data. En combinaison avec YARN [Vavilapalli et al., 2013], ce système augmente les capacités de gestion de données du cluster Hadoop et permet donc de traiter efficacement des pétaoctets de données.

RAMSSES tire parti d'un cluster Hadoop distribué pour prendre en charge la partie de traitement par lots lors de cette première étape. Cependant, dans la plupart des cas, les données brutes contiennent des enregistrements erronés ou incomplets qui doivent être nettoyés pour augmenter la précision des modules de traitement principaux. Deux étapes se produisent ici, le système déduira des données manquantes basées sur des techniques d'interpolation et de maximisation des attentes. De plus, des techniques de réduction de dimension telles que l'ACP [Boutsidis, Mahoney, and Drineas, 2008] seront tentées sur l'ensemble de données pour faciliter la diminution de la complexité de l'ensemble de données.

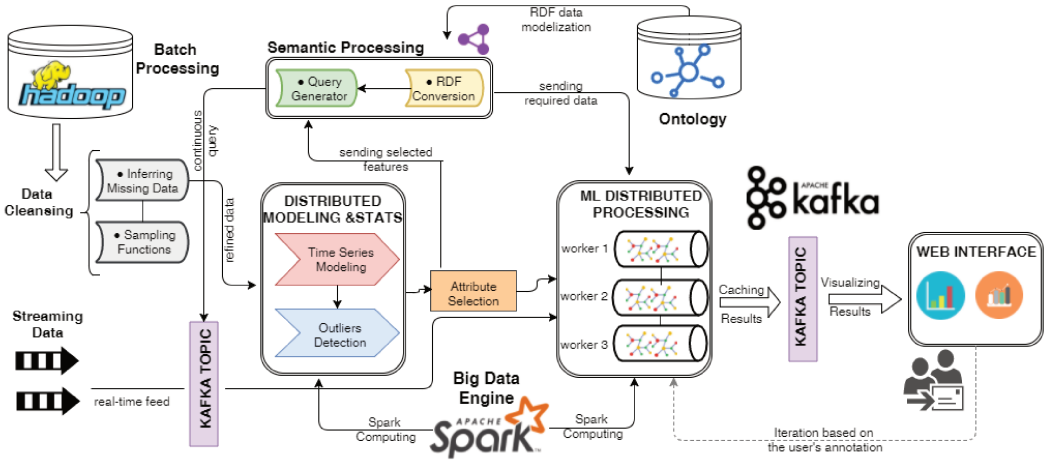


FIGURE 4.2: RAMSSES Architecture

Ensuite, une unité de modélisation distribuée appliquera plusieurs modèles de séries chronologiques sur les données nettoyées/échantillonnées pour les placer dans la bonne structure, en l'occurence la séparation entre données statiques et dynamiques ainsi que la génération des données manquantes. Ces unités visent à trouver des valeurs aberrantes spécifiques basées sur des techniques mathématiques

et statistiques qui s'appuieront sur la structure des séries chronologiques générées. Les valeurs aberrantes trouvées seront utilisées pour classer les caractéristiques de sorte que le système applique des modèles d'apprentissage automatique uniquement aux attributs requis, réduisant ainsi considérablement le temps de traitement. Cette méthode applique une allocation dynamique de données en optimisant la taille de chaque paquet de données transféré entre le moteur HDFS et le moteur Spark.

L'extraction des données minimales requises pour le traitement d'apprentissage automatique est gérée par l'unité sémantique. Après avoir converti les données réduites en RDF, en fonction d'une ontologie soigneusement conçue pour s'adapter au cas d'utilisation, un générateur de requête sélectionne le graphe de taille minimale en utilisant les fonctionnalités les mieux classées en fonction d'une requête SPARQL. Ce sous-graphe sera envoyé à l'unité de traitement principale qui utilise Spark pour exécuter plusieurs algorithmes de détection d'anomalies de manière distribuée.

Afin de sélectionner le bon algorithme convenant à la signature des flux, RAMSSES utilise un ensemble complexe de règles basées sur des statistiques telles que la dépendance des variables ou la distribution des données (c'est-à-dire la distribution normale ou gaussienne). Après avoir sélectionné l'algorithme approprié, nous l'appliquons sur la sortie générée par l'exécution de la requête SPARQL mentionnée précédemment sur les flux RDF, ce qui entraîne la découverte d'anomalies dans les flux. Enfin, les résultats trouvés seront écrits sur un système de messagerie, à savoir Apache Kafka, qui mettra en file d'attente les messages d'une manière ordonnée pour être utilisé par un outil de visualisation.

De nombreux cas d'utilisation sont liés à un système de production en exploitation déployé dans des environnements industriels. Concernant le cas d'utilisation des fuites d'eau, l'intervention d'un utilisateur final dans les locaux de la société Suez qui utilisera le résultat de l'anomalie détectée est hautement probable. Par conséquent, le système s'appuie sur les annotations de l'utilisateur final (c'est-à-dire des drapeaux d'anomalie vrai ou faux) qui créeront une nouvelle boucle d'itération responsable du stockage des signatures de chaque drapeau. Ces annotations seront aussi utilisées lors de la prochaine exécution pour valider ou invalider les anomalies nouvellement trouvées.

## 4.3 Détection de valeurs aberrantes

Les données traitées par RAMSSES sont associées à des horodatages, ce qui en fait un ensemble de valeurs chronologiques. Pour identifier les valeurs aberrantes avec précision, nous proposons un pipeline simple mais puissant basé sur une combinaison de modèles de modélisation des séries chronologiques et de détection des valeurs aberrantes. Les modèles ont été choisis sur la base de plusieurs documents de recherche [ITISE Conference, 2016]. Les principaux critères retenus ici

sont la généricité et l'adaptabilité étant donné que nous visons à résoudre divers cas d'utilisation. RAMSSES construit un modèle spécifique utilisé pour calculer une valeur attendue au temps $T$, puis un nombre d'erreurs $E$ sont calculées en comparant la valeur attendue avec la valeur réelle au temps $T$. Le système déterminera automatiquement les seuils sur $E$ et produira les anomalies les plus probables.

### 4.3.1 Modèles de Séries Temporelles

Nous avons sélectionné et implémenté les modèles les plus efficaces et les mieux établis au sein de la communauté de prévision statistique [Aggarwal and Reddy, 2013] pour répondre à la majorité des cas d'utilisation. Nous pouvons mentionner:

**Modèle de lissage exponentiel simple**: La plus simple des méthodes de lissage exponentielle. Elle convient à la prévision de données ne contenant pas de phénomène de saisonnalité (changement brusque de tendance).

**Modèle de lissage exponentiel double** (alias le lissage exponentiel de Holt): Un raffinement du modèle de lissage exponentiel simple en ajoutant un autre composant qui prend en compte la saisonnalité dans les données.

**Modèle exponentiel de lissage triple** (méthode de Winters aka): Un raffinement du modèle de lissage exponentiel double , mais ajoute un autre composant qui prend en compte la périodicité dans les données (absence de certaines données par exemple).

**Modèle de moyennes mobiles**: Basé sur une série temporelle construite artificiellement dans laquelle la valeur d'une période donnée est remplacée par la moyenne de cette valeur et les valeurs d'un certain nombre de périodes précédentes et successives.

**Modèle de régression linéaire multiple**: Il tente de modéliser la relation entre deux ou plusieurs variables explicatives et une variable de réponse en ajustant une équation linéaire aux données observées.

**Modèle de prévision naïve**: Un cas particulier du modèle de prévision de moyennes mobiles où le nombre de périodes utilisées pour le lissage est 1.

**Modèle olympique**: Génère des prévisions en utilisant une moyenne de données de référence sur une période. Par exemple, si la distance de la fenêtre temporelle est d'une semaine et que l'utilisateur demande les 4 dernières fenêtres passées, une prédiction à l'instant T sera générée à partir de la moyenne de (T-1) à (T-4) semaines.

**Modèle de régression polynomiale**: Un cas particulier de régression linéaire multiple dans lequel la relation entre les variables indépendantes x et y est modélisée comme un polynôme de nième degré dans x.

**Lissage spectral**: Implémente la technique de lissage basée sur la décomposition de la valeur singulière (SVD) de la série temporelle d'entrée.

**Modèle de moyennes mobiles pondérées**: Basé sur une série chronologique artificielle dans laquelle la valeur d'une période donnée est remplacée par la moyenne pondérée de cette valeur et les valeurs d'un certain nombre de périodes précédentes.

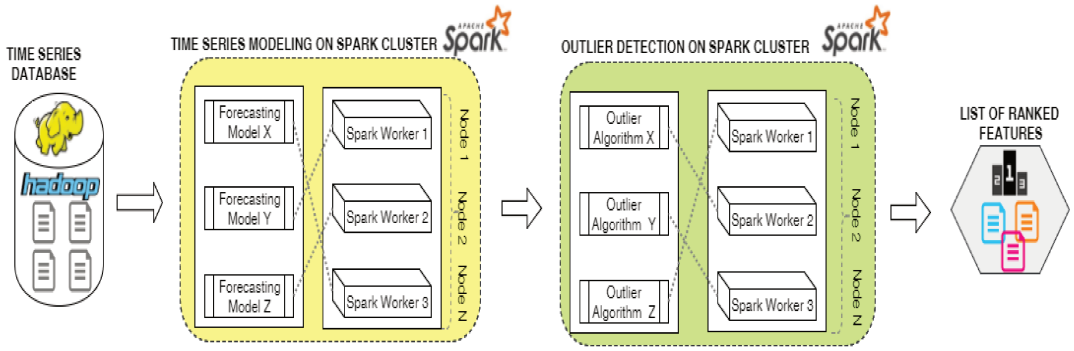


FIGURE 4.3: Processus de Détection de Valeurs Aberrantes

### 4.3.2 Modèles de Valeurs Aberrantes

Après la construction de la série temporelle, RAMSSES utilise les algorithmes suivants pour la détection des valeurs aberrantes. Ces modèles ont été sélectionnés à partir de diverses bibliothèques notamment le framework EGADS (Extensible Generic Anomaly Detection System) de Yahoo [Laptev, Amizadeh, and Flint, 2015]:

**Modèle Simple à Seuil**: Modèle de seuil simple qui renvoie une anomalie si elle est supérieure ou inférieure à un seuil.

**Détecteur Adaptatif à Changement de Densité**: Un algorithme basé sur la densité pour la détection des points de changement. L'algorithme fait glisser deux fenêtres côte-à-côte et calcule la Divergence Kullback-Leibler (KL) entre les distributions des résidus dans deux fenêtres pour chaque index temporel. La distribution des résidus dans chaque fenêtre est calculée de manière non paramétrique en utilisant l'estimation de la densité du noyau avec des noyaux gaussiens et des bandes passantes adaptatives.

**DBScan Model**: Algorithme de clustering basé sur la densité qui, avec un ensemble de points dans un espace, regroupe des points étroitement liés (points avec de nombreux voisins proches), marquant des points aberrants qui se trouvent seuls à faible densité.

**Modèle de Faible Densité Extrême**: Cette approche est non-paramétrique et utile dans les cas où la métrique de déviation n'est pas normalement distribuée. L'idée est de trouver des régions à faible densité de la distribution métrique de déviation, et d'utiliser le concept de densité locale où la localité est donnée par les voisins les plus proches.

**Modèle K-Sigma**: Cette approche est paramétrique et suppose que les données sont normalement distribuées avec une moyenne et une norme bien définie de déviation. En s'appuyant sur la distribution gaussienne, nous pouvons appliquer un outil statistique bien connu appelé la «règle des trois sigmas» qui stipule que 99,73% de tous les échantillons se situent dans les trois écarts-types de la moyenne. Par conséquent, en fonction de la valeur de K, on peut être sûr de la probabilité d'observer un échantillon au temps $t$.

**Modèle naïf** Basé sur des probabilités conditionnelles, ce modèle calcule une probabilité en comptant la fréquence des valeurs et des combinaisons de valeurs dans les données historiques. Le but est de trouver la probabilité qu'un événement se produise compte tenu de la probabilité d'un autre événement déjà survenu.

## 4.4 Sélection d'Attributs

Après avoir construit la nouvelle structure de séries temporelles basée sur les modèles disponibles dans notre système, tous les algorithmes du module de détection des valeurs aberrantes sont exécutés afin d'obtenir une liste des caractéristiques classées, c'est-à-dire les colonnes ayant la plus forte proportion de valeurs aberrantes. Par conséquent, nous générons un nouvel objet abstrait incluant son index, sa valeur et son horodatage. Enfin, nous comptons le nombre de valeurs aberrantes communes obtenues en exécutant toutes les combinaisons d'algorithmes sur chaque modèle de séries chronologiques.

Le résultat de ce processus est un nombre de valeurs aberrantes pour chacune des fonctionnalités. Les fonctionnalités sont classées en fonction de ces valeurs de comptage. Les résultats sont ensuite agrégés dans un fichier et transmis au composant générateur de requête sémantique qui l'utilisera comme base dans la clause *WHERE* de la requête SPARQL. L'algorithme 1 illustre comment les fonctionnalités sont classées :

## 4.5 Générateur de Requêtes Continues

Le processus de génération de requêtes crée des requêtes SPARQL pertinentes et précises sans aucune intervention de l'utilisateur final. Ces requêtes sont exécutées en continu sur notre moteur de traitement de flux RDF pour détecter automatiquement les anomalies. Les entrées de ce générateur sont:

- Un fichier texte contenant des entités classées.
- Les flux RDF qui correspondent aux graphes. Ce sera sur de tels éléments que la requête sera exécutée.

**Algorithm 1** Feature Ranking

**Input:** Tabular dataset D
**Output:** HashMap ranked features

```
1: Initialize HashMap out.
2: Let COL be a column from the dataset D
3: for all COL do
4:    let anomsMap be a HashMap to hold anomalies index and count
5:    let TSM be one of the time series building models.
6:    for all TSM do
7:       build COL time series using TSM
8:       let AD be one of the outliers detection models.
9:       for all AD do
10:         List < anomalies >list= AD.run(TSM) { run AD against TSM}
11:         let anomaly be anomaly in list
12:         for all anomaly do
13:            anomsMap.put(anomaly.getIndex(), anomsMap.get(anomaly.getIndex())+1)
14:         end for
15:      end for
16:   end for
17:   maxCount = anomsMap.getCommonAnomaliesMaxCount()
18:   out.put(COL.getName,maxCount)
19: end for
20: order out ASCENDING based on count value
21: Export out to CSV
```

- La base de connaissances statique du projet, c'est-à-dire la connaissance du domaine. Puisque les flux sont généralement très compacts, les connaissances redondantes (*e.g.* Coordonnées géographiques) sont stockées de manière statique.
- La terminologie (Tbox) utilisée par les graphes statiques et dynamiques.

La figure 4.4 présente un aperçu du système de génération de requêtes. La partie principale de la requête, la plus difficile à construire, est la clause WHERE. Nous l'initialisons en utilisant le graphe dynamique issu du flux, et nous remplaçons tous les noeuds vides par des variables. C'est une opération facile, qui nous donne une clause valide, mais beaucoup trop imprécise; nous devons le spécifier beaucoup plus. Notre prochaine étape consiste à identifier quelles sont les variables de la clause SELECT de la requête.

Pour ce faire, nous analysons le fichier des caractéristiques classées et identifions les informations pouvant être associées à notre URI graphique. La correspondance recherche à la fois les propriétés et les objets: la première représente les liens pertinents entre les concepts, la seconde les variables à extraire par la partie SELECT de la requête. Pour les objets identifiés, nous remplaçons le littéral dans le graphe par une variable, et l'ajoutons à la clause SELECT (il s'agit toujours d'un littéral, puisqu'il est stocké dans une colonne du fichier d'entrée).

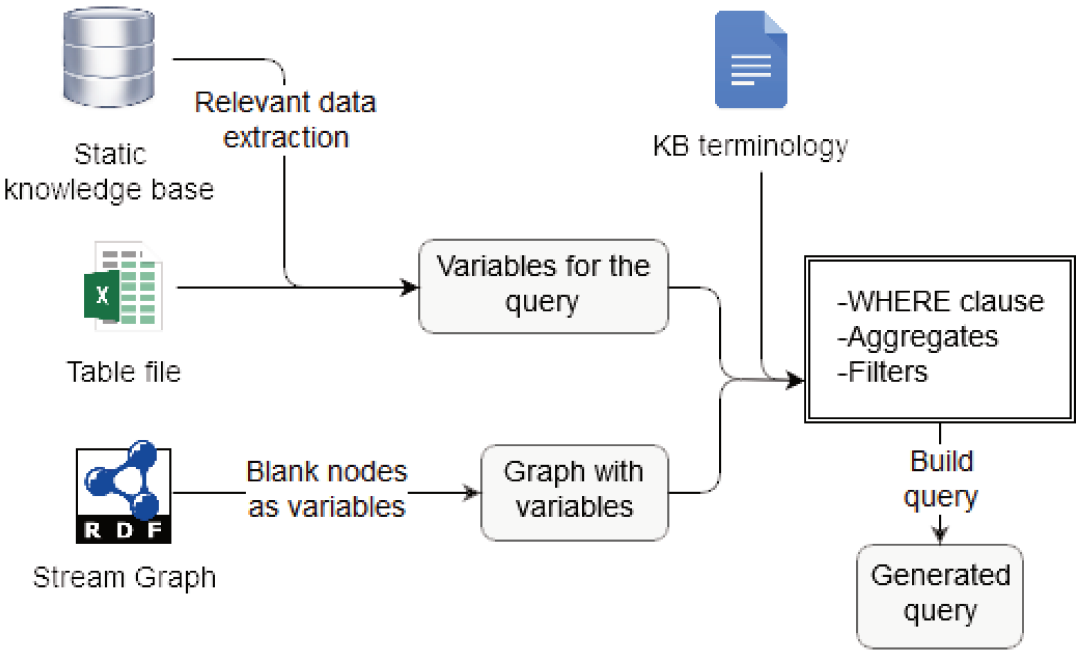


FIGURE 4.4: Processus de Génération de Requêtes Continues

À ce stade, nous savons qu'il existe un sous-graphe de notre clause *WHERE* réelle, composée des triplets liés entre eux et contenant l'URI des éléments correspondant au fichier des caractéristiques classées, ce qui est pertinent. Cependant, nous devons encore décider quels triplets restants ne sont pas pertinents. Pour chacun des sous-graphes, nous identifions trois possibilités:

- L'objet du triplet est une feuille et possède une propriété de type de données
- L'objet du triplet est une feuille et possède une propriété d'objet
- L'objet du triplet n'est pas une feuille (elle conduit à un graphique et possède une propriété d'objet).

Dans les deux premiers cas, nous devons identifier si l'objet du triple est statique ou dynamique, en interrogeant la base de données RDF statique. Si l'objet est stocké dans la base statique, nous avons une information redondante et elle n'est pas requise dans la requête. Sinon, il est pertinent pour la requête. Lorsque le triplet conduit à un autre graphe, nous devons savoir s'il est pertinent pour la requête, ou indépendant. En utilisant la Tbox du projet, nous pouvons vérifier si les propriétés initiales (à la racine du sous-graphe) sont *disjoint* ou *inverse*. Dans de tels cas, le sous-graphe entier peut être supprimé car il n'est pas pertinent pour la requête. Dans le cas où la Tbox est incomplète (c'est-à-dire des concepts, propriétés ou individus manquants), une telle méthode peut être problématique. De plus, si les propriétés ne sont pas disjointes, nous devons traiter tout le sous-graphe, et vérifier chaque triple opération coûteuse.

Pour spécifier encore plus la requête, il est possible d'utiliser des méthodes mathématiques pour identifier d'éventuels schémas dans le fichier de caractéristiques

classé. Dans certains cas, il permet de construire une clause *FILTER* sur certaines variables, en utilisant des valeurs numériques du document d'entrée. Enfin, l'opérateur lui-même peut améliorer les résultats en interagissant avec le système: la première requête peut être inexacte, ou retourner des paramètres non nécessaires. En ajoutant certains paramètres à la requête, ou en supprimant des valeurs retournées inutiles, l'utilisateur peut spécifier la requête étape par étape, jusqu'à ce qu'elle corresponde à ses besoins. Le générateur continuera à ajuster le graphique jusqu'à ce que plus aucun changement ne soit nécessaire. Des agrégats simples pourraient être facilement construits dans la requête, en utilisant des entrées provenant de l'utilisateur final; des problèmes apparaissent lorsque d'autres opérateurs doivent être ajoutés, tels que *GROUP BY* ou *HAVING*.

## 4.6 Apprentissage Automatique

RAMSSES vise à identifier les comportements inhabituels en surveillant les flux de données des capteurs. Les événements anormaux se réfèrent à des événements qui s'écartent de ceux considérés comme normaux sur la base de modèles historiques. Dans ce contexte non supervisé, c'est-à-dire des données sans catégories prédéfinies, RAMSSES doit détecter les anomalies qui n'ont pas été rencontrées précédemment. Le processus est basé sur l'estimation d'un modèle de comportements typiques à partir d'observations passées et la comparaison des observations actuelles par rapport à ce modèle.

### 4.6.1 Algorithmes de Clustering

Le système est destiné à fournir une liste d'algorithmes efficaces et génériques pouvant s'adapter à divers cas d'utilisation. Puisque nous cherchons à découvrir des signatures anormales sans étiquetage préalable, nous nous concentrons d'abord sur les approches de classification ci-dessous. D'autres méthodes seront testées et implémentées dans une future version de notre système.

- **K-means:** Cet algorithme est utilisé pour trouver des groupes dans des données basées sur une variable k. Il attribue itérativement des points de données à l'un des k groupes en fonction des fonctionnalités fournies, et les points de données sont regroupés en fonction des similarités des caractéristiques. Le résultat est une liste de centroïdes qui peuvent être utilisés pour classer de nouvelles données et les étiqueter. Chaque centroïde d'un cluster est une collection de valeurs de caractéristiques qui définissent les groupes résultants. L'examen des poids des entités centroïdes peut être utilisé pour interpréter qualitativement le type de groupe représenté par chaque groupe. Dans

les cas d'utilisation de détection d'anomalie, après avoir construit le modèle K-means en utilisant soit des données étiquetées avec des enregistrements d'anomalie connus étiquetés comme anomalies, soit en construisant le modèle avec des données propres sans anomalie, le modèle est utilisé pour regrouper de nouvelles données basées sur la distance euclidienne entre le nouveau point de données et les centroïdes des groupes de modèles. Si cette distance est supérieure à un certain seuil pour toutes les grappes disponibles, nous considérons cette valeur comme une anomalie.

- **Bisecting K-means:** C'est une combinaison de k-means clustering et de clustering hiérarchique. Il divise un cluster en deux sous-clusters à chaque étape en utilisant K-means, au lieu de partitionner les données en clusters dans chaque itération, jusqu'à ce que k clusters soient obtenus. Cet algorithme est basé sur K-means mais a l'avantage principal d'être plus efficace lorsque k est grand. De plus, les K-means de Bisecting produisent des grappes de taille similaire, alors que K-means est connu pour produire des grappes de tailles très différentes. Pour détecter une anomalie, la distance entre les points de données entrants et les centroïdes est calculée, et un point de données est considéré comme une anomalie si cette distance dépassait un certain seuil.

- **Modèle de mélange gaussien (GMM):** C'est un modèle probabiliste qui suppose que tous les points de données sont générés à partir d'un mélange d'un nombre fini de distributions gaussiennes avec des paramètres inconnus. On peut considérer les modèles de mélange comme un regroupement de K-means généralisant pour incorporer des informations sur la structure de covariance des données ainsi que sur les centres du Gaussien latent. Dans les cas d'utilisation de détection d'anomalie, cet algorithme est utilisé pour déterminer la probabilité d'appartenance d'un point de données entrant à l'un des clusters de modèles, où le modèle a été construit en utilisant des données propres sans anomalies ou données d'anomalie étiquetées. Si la probabilité résultante est inférieure à 0,05% pour l'ensemble des clusters du modèle, le point de données est considéré comme une anomalie, car il n'y a pas assez de confiance que ce point de données suivra l'un des clusters.

Pour proposer un système générique qui limite l'intervention des data scientists pour l'affinement des paramètres, RAMSSES résout deux problèmes majeurs rencontrés lors de la construction de modèles de clustering:

*Sélection du bon K*: Choisir la bonne valeur pour $k$ est crucial lors de la construction d'un modèle de clustering. Cela peut grandement affecter les résultats du regroupement et l'exactitude des résultats d'anomalies. Il existe différentes méthodes pour sélectionner une valeur de $k$ [D.T. Pham and Nguyen, 2005], nous avons implémenté la méthode *Elbow* car nous avons obtenu les meilleurs résultats avec cette technique. L'idée est d'exécuter le clustering K-means sur l'ensemble de données

pour différentes valeurs de $k$, par exemple de 1 à 25, pour chaque $k$, nous calculons la somme des erreurs quadratiques (SSE). Cette méthode propose que la valeur de $k$ qui a causé l'effet de coude soit la meilleure valeur. Par conséquent, notre objectif est de choisir une petite valeur de $k$ qui a encore un faible SSE, le coude représente généralement le point où nous commençons à avoir des rendements décroissants en augmentant $k$.

*Fixation du Seuil de Distance:* Pour obtenir des résultats très précis, il est crucial de définir avec soin la valeur du seuil de distance pour les deux algorithmes basés sur les K-means. Cependant, il s'agit d'une tâche complexe, surtout lorsque nous ne disposons pas d'informations détaillées sur les données traitées car, dans la plupart des cas, un tel seuil est défini par essais et erreurs. Pour résoudre ce problème, nous proposons une approche basée sur des quantiles de statistiques où l'on se base sur la règle .99-Quantile pour définir ce seuil, c'est-à-dire que tous les points de cette gamme .99-Quantile seraient considérés comme des anomalies. Pour fournir une flexibilité maximale à l'utilisateur, nous exposons cette valeur dans le fichier de configuration pour lui permettre d'entrer dans une plage de quantiles différente.

### 4.6.2 Séléction d'Algorithmes et Automatisation

Nous nous appuyons sur les données de lots historiques pour sélectionner l'un des algorithmes susmentionnés et l'appliquer aux données de transmission en continu pour la récupération des anomalies. Pour atteindre ce résultat, plusieurs analyses de données statistiques sont nécessaires pour choisir automatiquement l'algorithme approprié sans intervention humaine. Le processus est organisé en 4 étapes principales:

- **Standardisation des Données:** Ce concept survient lorsque des variables indépendantes continues sont mesurées à différentes échelles, ce qui signifie que ces variables ne contribuent pas de manière égale à l'analyse. L'idée est de redimensionner la variable d'origine pour avoir une plage et / ou une variance égale. Il est important de standardiser les variables avant de lancer des algorithmes d'apprentissage automatique car les techniques d'analyse de cluster dépendent du concept de mesure de la distance entre les différentes observations que nous essayons de regrouper. Si une variable est mesurée à une échelle plus élevée que les autres variables, toute mesure que nous utiliserons sera trop influencée par cette variable. Il existe différentes techniques pour normaliser les données [P.Trebuna and M.Fil'o, 2014], notre système utilise la méthode *Z-Score* car c'est l'une des méthodes les plus populaires parmi les data scientists. Dans ce cas, nous remettons à l'échelle une variable originale pour avoir une moyenne de zéro et un écart-type de un, suivant l'équation:

$$z = \frac{x - mean}{std.dev} \tag{4.1}$$

En règle générale, les données ne doivent pas être toujours normalisées. En effet, dans certains cas, les données ne présentent pas un grand écart entre les caractéristiques en termes d'échelle, dans ces cas, aucune mise à l'échelle n'est nécessaire. Afin de vérifier si ces données ont besoin d'une standardisation ou non, nous parcourons chacune des colonnes en divisant la moyenne et la variance de chacune d'elles, si l'un des résultats était supérieur au seuil prédéfini (3 pour la moyenne et 5 pour la variance dans notre cas d'utilisation), nous savons que la normalisation est nécessaire. L'algorithme 2 décrit ce processus.

**Algorithm 2** Standardization Check

**Input:** Tabular dataset D
**Output:** Boolean flag

```
Let col be a column from the dateset D
for all col do
    let colMean = col.getMean()
    let colVariance = col.getVariance()
    let anotherCol be another column from the dataset
    for all anotherCol do
        let mean = anotherCol.getMean()
        let variance = anotherCol.getVariance()
        if colMean ≥ mean ∗ treshold then
            return true
        end if
        if colVariance ≥ variance ∗ treshold then
            return true
        end if
    end for
end for
return false
```

- **Type de distribution:** La distribution de données est un facteur très important pour décider quel algorithme de clustering doit être utilisé. Par exemple, il existe quelques algorithmes de classification qui fonctionnent très bien avec une distribution normale (par exemple, modèle de mélange gaussien) où d'autres fonctionnent très bien pour des données distribuées multinomiales (par exemple *Allocation Latente de Dirichlet*). Pour décider quel algorithme utiliser, un test de distribution est effectué sur l'ensemble de données historique pour vérifier si les données suivent une distribution normale ou non. Il existe de nombreuses méthodes pour effectuer ce test [Yazici and Asma, 2007], celui qui

a été choisi est le test K-carré de D'Agostino[1] pour sa mesure d'ajustement efficace. Il est basé sur les transformations de l'échantillon utilisant la kurtosis et l'asymétrie. Pour effectuer ce test, une statistique D est calculée en suivant l'équation ci-dessous, la valeur D calculée est ensuite comparée à la table des valeurs de référence. Si la valeur calculée se trouve dans une plage spécifique trouvée de la table, nous supposons que cette donnée suit une distribution normale. Le test K2 de D'Agostino suit l'équation suivante:

$$K^2 = Z_1(g_1)^2 + Z_2(g_2)^2 \tag{4.2}$$

- **Dépendance des Variables:** Trouver la corrélation est une autre étape importante pour décider quel algorithme de cluster devrait être utilisé sur certaines données. Certains modèles fonctionnent bien avec des variables dépendantes où d'autres fonctionnent mieux avec des variables indépendantes. Plusieurs méthodes peuvent être utilisées pour déterminer la dépendance des variables dans un ensemble de données. S'il s'agit d'une variable catégorielle, le *test du Chi-carré* peut être utilisé, s'il s'agit d'une valeur quantitative, la corrélation simple de Pearson peut être utilisée. Comme ce travail se concentre sur la détection quantitative des anomalies, nous nous concentrerons uniquement sur la méthode de corrélation de Pearson. La corrélation de Pearson est une mesure linéaire entre deux variables $X$ et $Y$. Il en résulte une valeur entre *+1* et *-1*, où *1* est une corrélation linéaire positive totale, *0* n'est pas une corrélation linéaire et *-1* est une corrélation linéaire négative totale. L'équation suivante montre la formule pour calculer la corrélation entre deux variables où $n$ est la taille de l'échantillon, *xi/yi* représentent les échantillons simples indexés avec $i$ et *x/y* correspondent à la moyenne d'échantillon pour $x$ et $y$ :

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.3}$$

Si le résultat de la corrélation est supérieur à 0,8, les 2 variables sont annotées comme *FORTEMENT DÉPENDANTES*. Si la valeur est supérieure à 0,5, les 2 variables sont annotées comme *FAIBLEMENT DÉPENDANTES*. Dans tous les autres cas, les variables sont annotées comme *NON DÉPENDANTES*. Enfin, la corrélation moyenne pour l'ensemble des données est calculée pour définir approximativement la dépendance des variables.

[1] http://geai.univ-brest.fr/carpentier/2011-2012/Documents-R/Dago-test.pdf

- **Complexité des ensembles de données:** Une statistique très utile pour choisir l'algorithme de clustering à utiliser est de comprendre la complexité des données, y compris la taille des données et le nombre d'espaces des fonctionnalités. Certains algorithmes prennent beaucoup de temps en raison des mathématiques complexes utilisées et ne fonctionnent pas bien avec de très grands ensembles de données. Pour sélectionner l'algorithme approprié, nous suivons une approche directe. Considérant que le modèle de mélange gaussien échouera pour les données de haute dimensionnalité, si l'ensemble de données a des dimensions élevées (plus de 50 dimensions) et plus de 50 000 enregistrements en termes de taille, des algorithmes basés sur K-means seront suggérés. A ce stade, compte tenu des trois algorithmes mentionnés qui seront utilisés dans RAMSSES, les règles à suivre n'utilisent pas toutes les statistiques collectées sur les données. La raison en est que les paramètres restants seront utilisés dans le futur lorsque d'autres algorithmes de cluster seront ajoutés au système.

### 4.6.3 Apprentissage Iteratif

L'offre d'un système d'apprentissage automatique générique implique de prendre pour prémisse un environnement non supervisé où aucune étiquette n'est fournie par les experts du domaine. Dans RAMSSES, nous supposons que l'utilisateur final a un aperçu des données traitées. Par conséquent, nous pouvons tirer parti de son expertise pour améliorer nos estimations sur la probabilité d'une anomalie détectée.

Après avoir mis en cache les résultats du cluster Spark sur une file d'attente Kafka, nous exposons les anomalies découvertes dans un tableau de bord en temps réel. Un outil d'annotation permet aux utilisateurs finaux de confirmer ou de désapprouver une anomalie avec un certain degré de confiance. Le drapeau et le degré de confiance classent les anomalies confirmées avec la plus grande confiance en créant trois niveaux distincts: confiance très forte, moyenne et faible). Tous les nouveaux événements traités par le cluster Spark auront leur signature par rapport à ceux fortement confirmés par l'utilisateur. Si les signatures correspondent dans une certaine mesure (qui peut être réglée dans le fichier de configuration), le système marque l'événement comme une anomalie avec un niveau de probabilité élevé.

## 4.7 Evaluation

Dans cette section, nous décrivons la méthodologie utilisée pour évaluer les approches automatisées RAMSSES pour les parties numériques et sémantiques, nous discutons les paramètres d'évaluation des ensembles de données et les résultats obtenus. Nous évaluons et comparons également les performances du système dans

deux environnements différents: exécution locale sur une seule machine et exécution répartie sur un véritable cluster de plusieurs machines.

### 4.7.1 Méthodologie

Afin d'évaluer RAMSSES, des jeux de données étiquetés ont été utilisés. Pour chacun d'entre eux, un processus de nettoyage a été effectué en supprimant les enregistrements étiquetés comme des anomalies, l'ensemble de données résultant a été utilisé pour estimer le nombre de clusters. Ensuite, le modèle est construit en utilisant les données nettoyées avec les paramètres suivants: *k* comme étant le nombre estimé de clusters, *1000* comme étant le nombre maximum d'itérations, et *1e-9* la distance seuil au-dessus duquel les centroïdes des clusters convergent.

Les autres paramètres sont les paramètres par défaut de la Spark Machine Learning Library. Finalement, l'ensemble de données original incluant les anomalies a été comparé au modèle et le nombre d'anomalies détectées a été enregistré. Cette approche d'évaluation a été appliquée sur une machine locale et sur un cluster afin d'évaluer si le système est évolutif et fonctionne plus rapidement lorsqu'il est distribué.

### 4.7.2 Jeux de Données

Tous les jeux de données utilisés dans l'évaluation sont des jeux de données réels, sauf un qui est généré. Tous sont étiquetés et peuvent être trouvés dans des benchmarks bien connus [ Goldstein, 2015] à l'exception de l'ensemble de données synthétiques généré automatiquement et de l'ensemble de données de trafic de production de Yahoo qui peuvent être trouvés dans des documents de recherche spécifiques [ YAHOO, 2015]. Par exemple, nous pouvons mentionner les jeux de données suivants:

**ALOI** : C'est une collection d'images couleur de 1000 petits objets, enregistrée à des fins scientifiques.

**Shuttle** : Il décrit les positions des radiateurs dans une navette spatiale de la NASA.

**Speech** : Il contient des données du monde réel provenant de la langue anglaise enregistrée.

**Pen** : Il est basé sur la reconnaissance de texte manuscrit (global), cet ensemble de données UCI contient les chiffres manuscrits 0-9 de 45 auteurs différents.

**KDD** : Il contient du trafic normal et d'attaque simulé sur un niveau IP dans un environnement de réseau informatique afin de tester les systèmes de détection d'intrusion.

**Thyroid** : C'est est un ensemble de données du référentiel d'apprentissage automatique UCI dans le domaine médical.

**YahooS5** : C'est un jeu de données basé sur le trafic de production réel vers certains des comptes Yahoo! Propriétés. Il contient une série chronologique représentant les métriques de l'un des services Yahoo.

**Synthetic** : Cet ensemble de données est généré artificiellement pour avoir 3 distributions gaussiennes et s'adapter au modèle de mélange gaussien.

La table ci-dessous donne un aperçu clair de la structure et des spécificités de chaque ensemble de données, comme le nombre d'enregistrements ou le pourcentage d'anomalies.

| Name | Size | Dimensions | Anomalies Percentage |
|---|---|---|---|
| ALOI | 50,000 | 27 | 3.02% |
| Shuttle | 46,464 | 9 | 1.89% |
| Speech | 3,686 | 400 | 1.65% |
| Pen | 809 | 16 | 11.1% |
| KDD | 620,089 | 38 | 0.17% |
| Thyroid | 6,916 | 21 | 3.61% |
| YahooS5 | 1,441 | 1 | 7.49% |
| Synthetic | 4,000 | 2 | 3% |

TABLE 4.1: Liste des Jeux de Données

### 4.7.3 Evaluation du Clustering

Dans cette évaluation, tous les ensembles de données sont des mesures du monde réel provenant de différentes sources, chacune étant étiquetée pour indiquer la qualité du regroupement. Avant de discuter des résultats, nous allons expliquer les paramètres clés. Nous adoptons la sémantique standard des normes pour *Vrai Positif*, *Faux Négatif*, *Vrai Négatif* et *Faux Positif*, resp. TP, FN, TN et FP. En fonction du nombre de ces valeurs, nous définissons les mesures suivantes pour évaluer notre système dans différentes dimensions:

**Erreur:** Proportion de toutes les prédictions incorrectes, c'est une mesure de la gravité d'un modèle.

$$Erreur = \frac{Previsionsincorrectes}{touteslespredictions} = \frac{FP + FN}{FP + FN + TP + TN}$$

**Précision:** Proportion de toutes les prédictions correctes, c'est une mesure de la qualité d'un modèle.

$$Precision = \frac{Predictionscorrectes}{touteslesprdictions} = \frac{TP + TN}{FP + FN + TP + TN}$$

**Exactitude:** Proportion de toutes les prédictions positives qui sont correctes, c'est une mesure du nombre de prédictions positives qui étaient des observations positives réelles.

$$Precision = \frac{positivePredictedCorrectly}{allPositivePredictions} = \frac{TP}{TP + FP}$$

**Rappel:** Proportion de toutes les observations positives réelles qui sont correctes.

$$Rappel = \frac{Correctementprditpositif}{Touslesvraispositifs} = \frac{TP}{TP + FN}$$

**F-Measure:** Moyenne de la précision et du rappel, nous utilisons la moyenne harmonique pour ses ratios de précision à moyenne.

$$F - Measure = \frac{2 * precision * rappel}{precision + rappel}$$

Nous avons commencé l'évaluation avec l'ensemble de données YahooS5, la figure suivante montre les mesures de trafic en fonction du temps codé en millisecondes, les points bleu foncé sont étiquetés ici comme des anomalies.

Combiné avec la 2ème figure ci-dessus, on peut dire que les données ne suivent pas une distribution gaussienne, donc l'algorithme suggéré par le système pour détecter les anomalies est *K-Means*.

D'autre part, nous pouvons noter à partir du graphique suivant le profil de l'ensemble de données.

En combinant les 2 figures ci-dessous, on peut dire que l'ensemble de données suit une distribution gaussienne, donc le *Modèle de Mélange Gaussien* sera utilisé pour détecter les anomalies.

Nous avons évalué RAMSSES par rapport à tous les ensembles de données. Pour chacun, le système a suggéré un algorithme à utiliser, mais pour des raisons de comparaison critique des résultats, nous avons inclus les résultats lorsque nous avons utilisé d'autres algorithmes pour voir si l'algorithme suggéré était le meilleur choix ou non. La table suivante liste les résultats obtenus, au sein de chaque groupe, les entrées en gras représentent les résultats avec l'algorithme suggéré par le système.

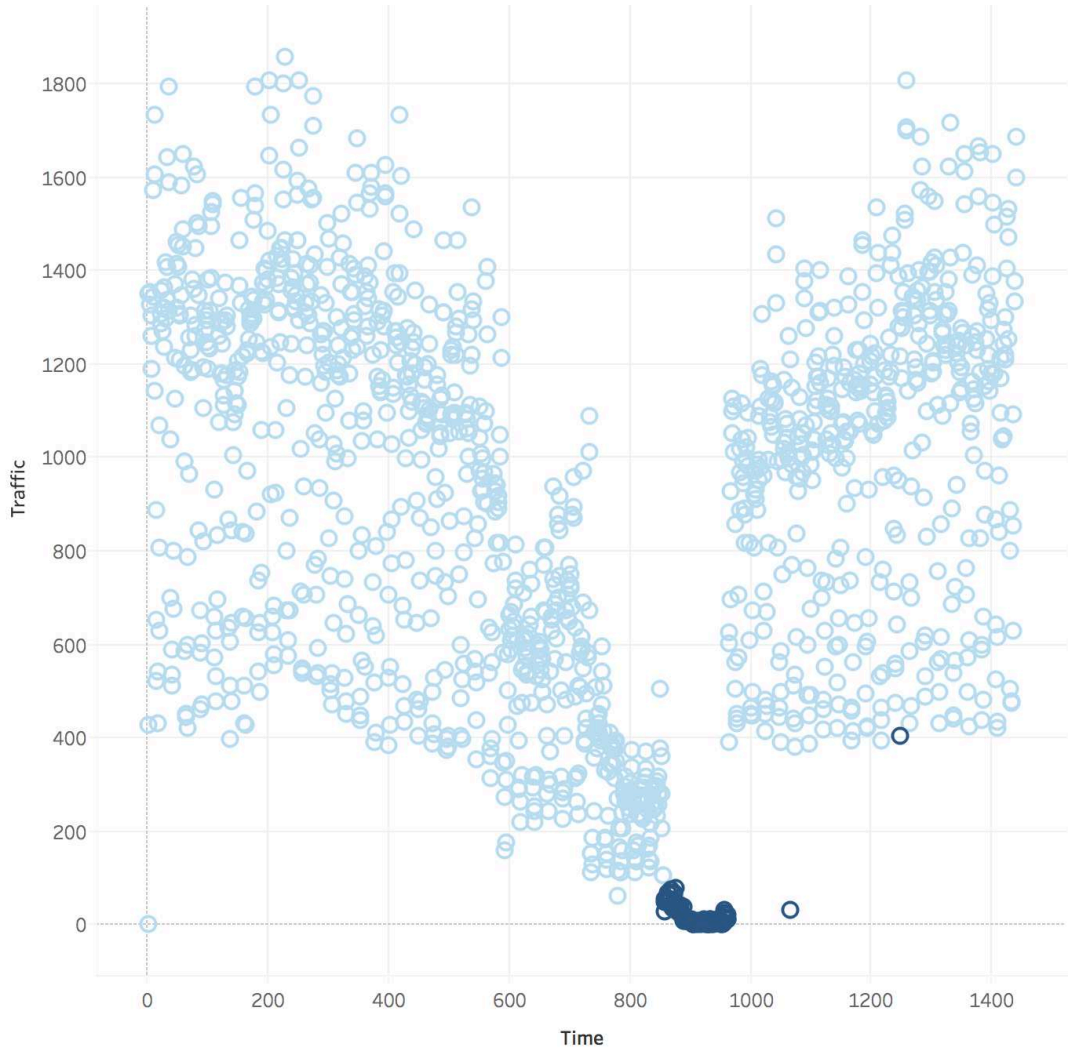


FIGURE 4.5: Mesures de Trafic Yahoo

Comme nous pouvons le relever, certains des ensembles de données ont eu d'excellents résultats à la fois en terme d'exactitude et de précision en utilisant l'algorithme suggéré par RAMSSES. Les jeux de données *Shuttle, Yahoo S5 et Pen* ont enregistré de très bons résultats pour la tâche de clustering, où la plupart des anomalies réelles sont détectées. Très peu d'enregistrements signalés comme des anomalies alors qu'en réalité, ils ne le sont pas, et très peu d'enregistrements qui sont des anomalies dans la réalité ne sont pas détectés.

Nous pouvons également noter que dans les jeux de données où l'algorithme suggéré est *Gaussian Mixture*, comme *Thyroid dataset*, les résultats étaient raisonnablement bons quand on considère l'exactitude et la précision. Pour ce cas, si on utilise d'autres algorithmes , par exemple K-Means, nous avons un excellent résultat de précision mais une très mauvaise précision. Cependant, le meilleur des résultats pour ces jeux de données a été atteint avec l'algorithme suggéré. La même chose s'applique sur *Synthetic dataset* mais avec moins de qualité de clustering.

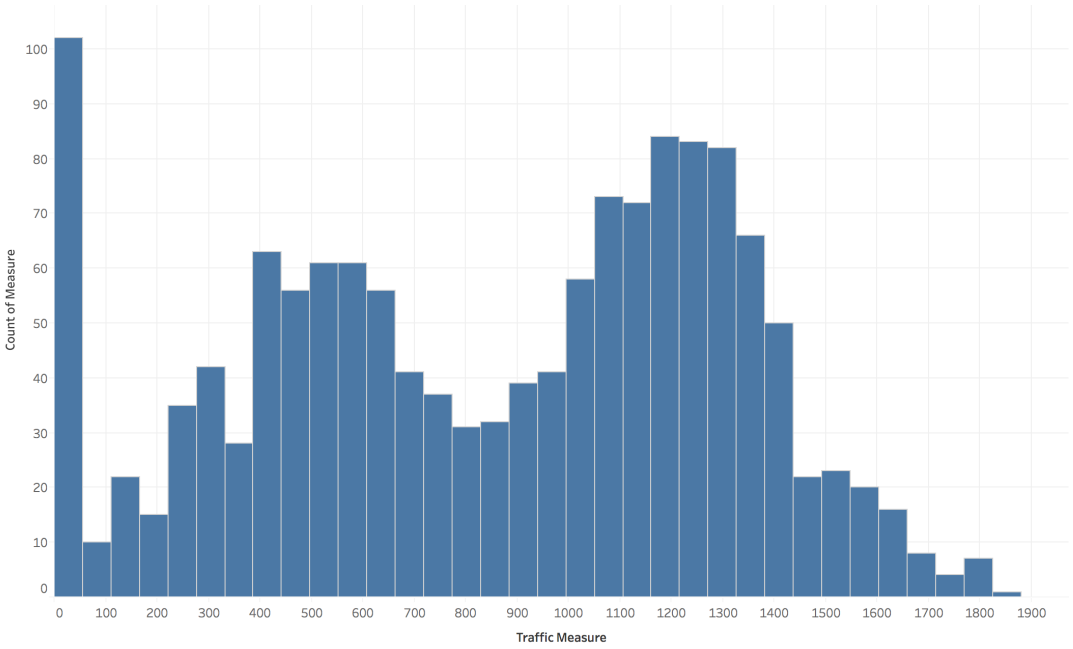


FIGURE 4.6: Histogramme des données de Trafic Yahoo

Concernant le dataset *ALOI et Speech datasets*, nous avons eu de très mauvais résultats en termes de précision pour tous les algorithmes utilisés. Spark a échoué lors de l'exécution de l'ensemble de données *Speech*, expliquant l'absence de résultats sur l'algorithme de mélange gaussien. En effet, cet algorithme peut mal fonctionner sur des données de grande dimension, ceci en raison de son inefficacité durant la tâche de regroupement basée sur des techniques statistiques.

Il est utile également de mentionner que le clustering est une méthode qui dépend fortement des connaissances préalables sur les caractéristiques des données pour pouvoir définir les paramètres appropriés tels que *k*, *nombre d'itérations* et *epsilon*. Nous avons fait de notre mieux pour donner une approche générale qui peut bien fonctionner avec différents ensembles de données. Nous pouvons conclure que RAMSSES peut suggérer l'algorithme à utiliser avec les données qui pourraient donner les meilleurs résultats.

### 4.7.4 Evaluation des Performances

Dans cette section, nous allons montrer les résultats de performances de notre approche et discuter de la durée d'utilisation des algorithmes de clustering Spark MLLib. Nous fonctionnons d'abord sur une machine locale avec un parallélisme maximum. Ensuite, nous passons à un cluster Amazon Services pour vérifier si les performances de RAMSSES augmentent considérablment en mode distribué sur un cluster industriel.

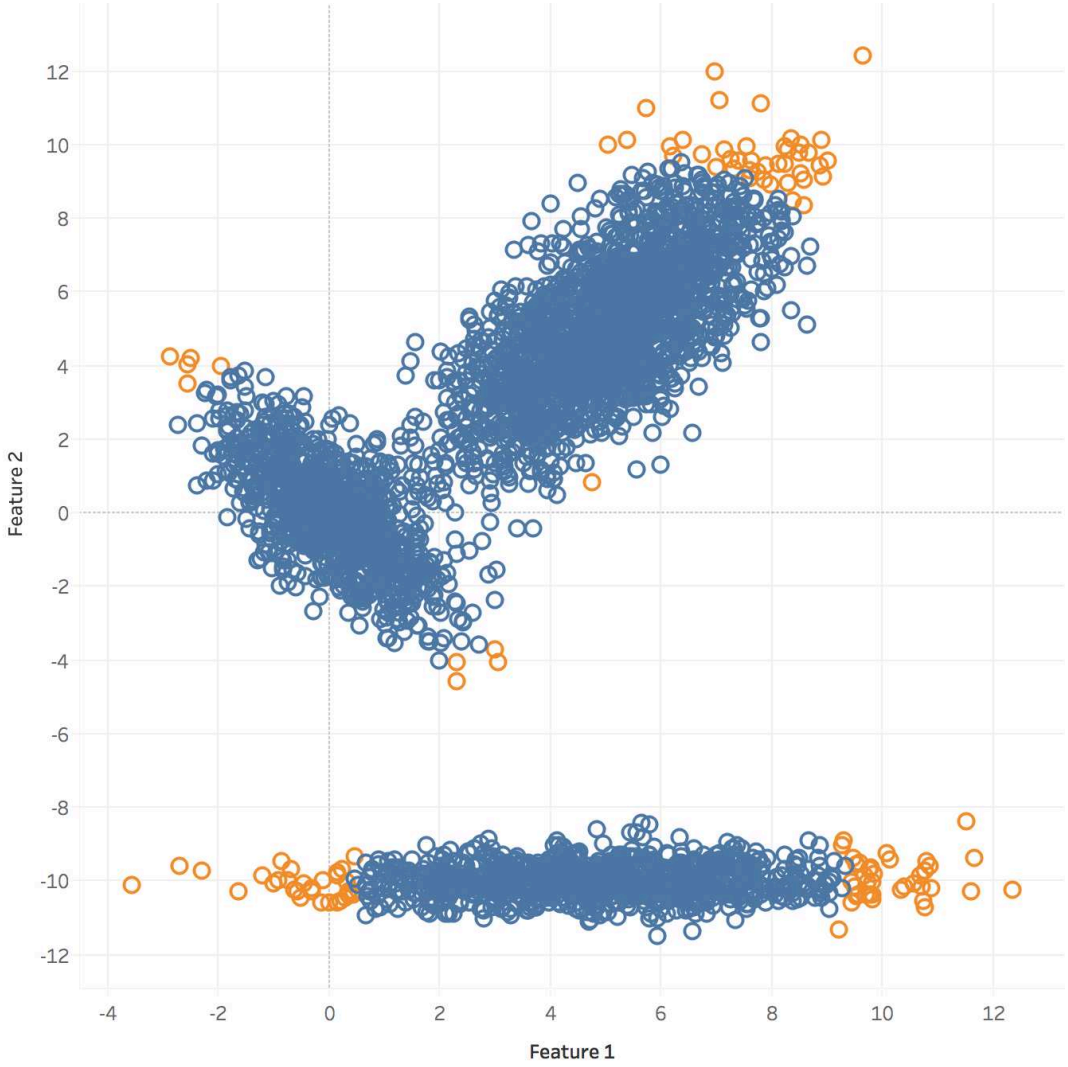


FIGURE 4.7: Synthetic Dataset(Feature 1 vs Feature 2)

**Evaluation Locale**

Toutes les tâches de clustering utilisant les trois algorithmes de clustering ont d'abord été exécutées sur une machine locale, les spécifications de cette machine sont:

- MacBook Pro exécutant macOS Sierra;
- Intel Core i7 CPU 2,5 GHz avec 4 cœurs;
- RAM 16 Go DDR3, cache L2 (par cœur): 256 Ko, cache L3: 6 Mo;
- Disque SSD de 500 Go;
- Spark 2.0.0 avec Java 8.

La figure suivante montre le temps d'exécution pour différents ensembles de données utilisant les algorithmes détaillés plus haut. Notez que pour l'ensemble des données *KDD*, lors de l'exécution de l'algorithme *Gaussian Mixture*, le temps

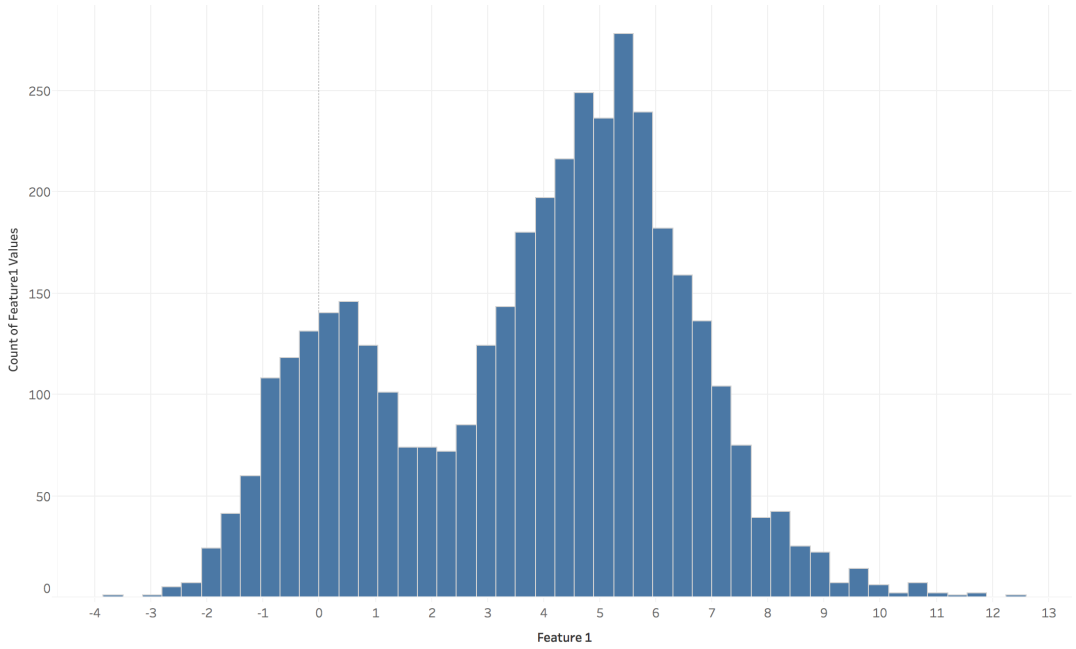


FIGURE 4.8: Synthetic Dataset Feature 1 Histogram

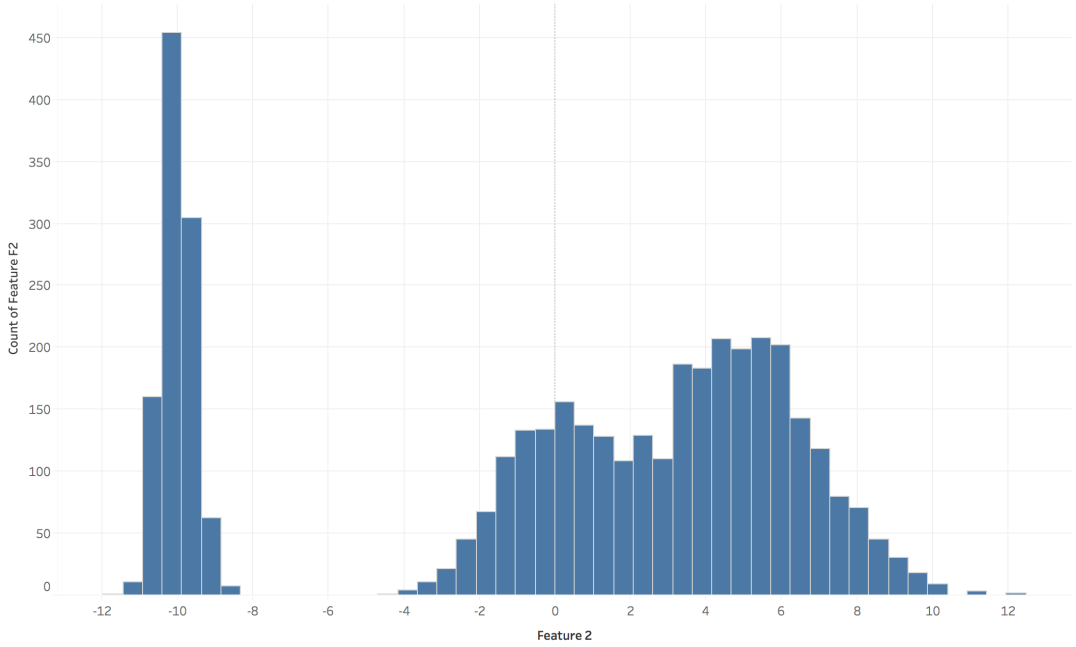


FIGURE 4.9: Synthetic Dataset Feature 2 Histogram

d'exécution était de *13146.025* secondes, soit environ *3.65* heures, donc il a été exclu du graphique cat il rendra les autres valeurs illisibles en raison de l'échelle graphique.

D'une manière générale, *Bisecting K-means* prend plus de temps que *K-means*. En effet, quand le nombre de dimensions de l'ensemble de données augmente, les performances du modèle *Gaussian Mixture* sont fortement impactées devenant l'algorithme le plus lent de notre évaluation.

| Dataset | Est K | Algorithm | Accuracy | Precision | Error | Recall | F-Measure |
|---|---|---|---|---|---|---|---|
| ALOI | 10 | **K-Means** | **0.961** | **0.043** | **0.039** | **0.117** | **0.063** |
| | 10 | **Bi.K-Means** | **0.961** | **0.032** | **0.039** | **0.093** | **0.047** |
| | 10 | Gaussian Mixture | 0.969 | 0.003 | 0.030 | 0.285 | 0.008 |
| YahooS5 | 15 | **K-Means** | **0.990** | **0.990** | **0.009** | **0.892** | **0.939** |
| | 15 | **Bi.K-Means** | **0.965** | **0.660** | **0.034** | **0.847** | **0.742** |
| | 15 | Gaussian Mixture | 0.212 | 0.990 | 0.782 | 0.087 | 0.160 |
| Thyroid | 5 | **Gaussian Mixture** | **0.605** | **0.632** | **0.394** | **0.056** | **0.104** |
| | 5 | K-Means | 0.954 | 0.008 | 0.045 | 0.029 | 0.012 |
| | 5 | Bi.K-Means | 0.955 | 0.004 | 0.044 | 0.016 | 0.006 |
| Shuttle | 5 | **K-Means** | **0.989** | **0.980** | **0.011** | **0.654** | **0.784** |
| | 5 | **Bi.K-Means** | **0.989** | **0.981** | **0.010** | **0.655** | **0.786** |
| | 5 | Gaussian Mixture | 0.981 | 0.019 | 0.782 | 0.5 | 0.013 |
| Speech | 3 | **K-Means** | **0.973** | **0.016** | **0.026** | **0.027** | **0.020** |
| | 3 | **Bi.K-Means** | **0.974** | **0.016** | **0.026** | **0.027** | **0.020** |
| | NA | NA | NA | NA | NA | NA | NA |
| Synthetic | 3 | **Gaussian Mixture** | **0.504** | **0.650** | **0.496** | **0.038** | **0.073** |
| | 3 | K-Means | 0.982 | 0.725 | 0.018 | 0.696 | 0.710 |
| | 3 | Bi.K-Means | 0.975 | 0.5 | 0.024 | 0.612 | 0.550 |
| Pen | 5 | **K-Means** | **0.954** | **0.667** | **0.045** | **0.895** | **0.764** |
| | 5 | **Bi.K-Means** | **0.961** | **0.733** | **0.038** | **0.904** | **0.810** |
| | 5 | Gaussian Mixture | 0.111 | 0.99 | 0.888 | 0.111 | 0.200 |
| KDD | 10 | **K-Means** | **0.989** | **0.476** | **0.011** | **0.075** | **0.129** |
| | 10 | **Bi.K-Means** | **0.989** | **0.476** | **0.011** | **0.075** | **0.129** |
| | 10 | Gaussian Mixture | 0.957 | 0.99 | 0.043 | 0.038 | 0.074 |

TABLE 4.2: Clustering Results

**Evaluation en mode Distribué**

Dans cette section, nous évaluerons si RAMSSES fonctionne vraiment mieux en termes de temps d'exécution lorsqu'il est distribué. Nous avons utilisé l'ensemble de données *KDD* car il s'agit du plus grand ensemble de données que nous ayons, et nous pouvons remarquer une différence de temps d'exécution lors de la distribution du processus de clustering. Nous comptons le comparer avec les performances locales en utilisant le MacBook Pro mentionné précédemment avec l'utilisation complète de ses 4 cœurs.

Le service Amazon AWS a été utilisé pour créer un cluster avec un nœud maître et quatre machines (workers), toutes de type *EMR m3.xlarge*, avec les spécifications suivantes:

- Processeurs Intel Xeon E5-2670 2,6 GHz, 4 cœurs;
- 15GB de RAM;
- 2 x SSD de 40 Go;

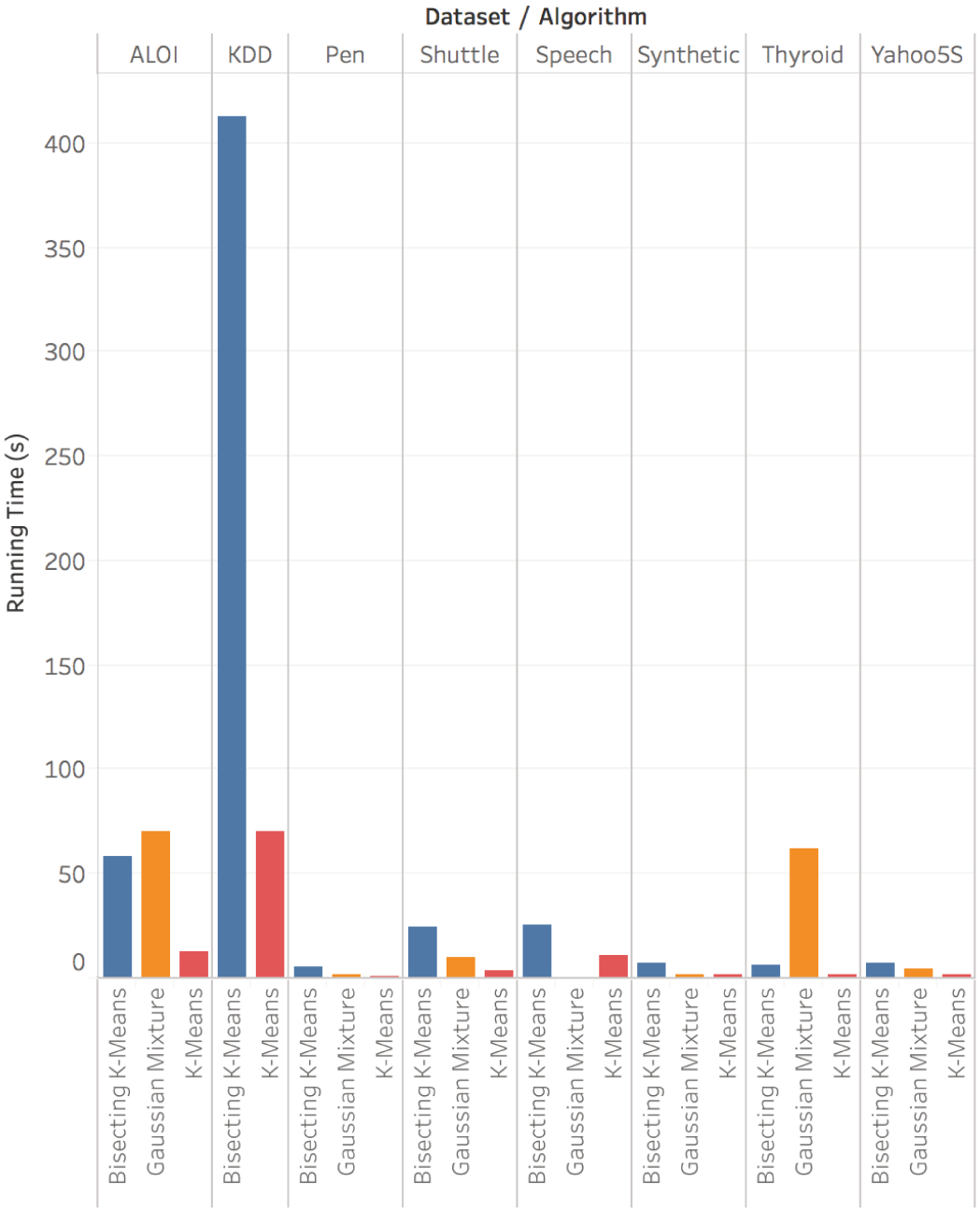


FIGURE 4.10: Running Time For Datasets Using Different Algorithms

- Spark 2.0.0 avec Java 8.

Comme le montre la figure suivante, nous avions respectivement 41,4%, 37,75% et 19,76% d'améliorations de performance pour *K-Means*, *Bisecting K-Means* et *GMM*. Nous pouvons prouver à partir de ce test que RAMSSES est capable d'évoluer et de mieux fonctionner lorsqu'il est distribué.

Cependant, nous ne prétendons pas que cela pourrait être la meilleure performance qui peut être obtenue par la distribution pour les raisons suivantes:

(i) Au cours de l'évaluation du cluster, les paramètres Spark ont été laissés avec des valeurs par défaut. Il est bien connu que le réglage est un aspect important

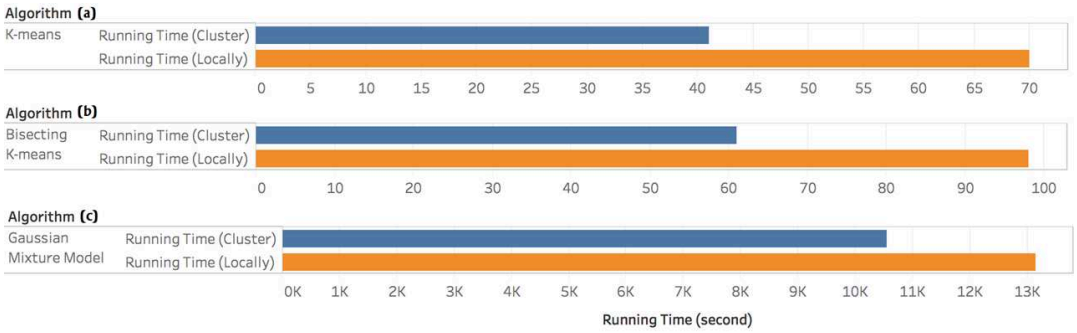


FIGURE 4.11: Local vs Cluster Running Time for ML Algorithm

de la performance de Spark.

(ii) Une forte amélioration des performances peut être constatée lors de la manipulation de jeux de données beaucoup plus volumineux tels que *KDD*. Nous pensons que le système devrait fonctionner mieux avec des ensembles de données beaucoup plus grands que l'ensemble de données utilisé.

### 4.7.5 Evaluation du Générateur de Requêtes Continues

Nous avons évalué notre générateur de requêtes en utilisant des mesures de capteurs archivées par notre partenaire. En raison de leur nature exclusive, les données ne sont pas rendues publiques. Elles représentent une année de mesures sur 3 zones géographiques différentes (2 françaises et 1 internationale).

La validation du générateur de requêtes a été effectuée à l'aide de panels de 5 experts du domaine. Intuitivement, nous leur avons présenté un ensemble de requêtes sur les 3 zones et leur demandons si les requêtes ont effectivement récupéré des informations qui pourraient aider à détecter les anomalies. Pour les requêtes les plus simples, la sortie a été jugée extrêmement satisfaisante (95% des requêtes ont été jugées pertinentes).

Avec des requêtes impliquant plus de paramètres, certaines imprécisions peuvent survenir, mais après discussion, nous avons découvert que l'ajustement des paramètres pertinents dans le fichier d'entrée résout la plupart des problèmes. Le souci principal réside dans les requêtes plus complètes, où certaines spécifications telles que les agrégats et les paramètres supplémentaires doivent être ajoutés manuellement.

L'évaluation suivante considère les gains de performance implicites de notre générateur de requêtes. Une méthode de vérification consiste à comparer la différence des données de flux entrant avec et sans nos requêtes.

Nous considérons maintenant un exemple de flux WAVES dans le monde réel. Les entrées provenant du flux (partie *a* de la figure) sont composées de triplets RDF provenant de chaque capteur. Nous avons également un fichier listant les éléments

**a. Input Stream (extract)**

```
_:1 type ssn:Sensor
_:1 measures _:2
_:2 hasFlow _:3
_:2 hasUnit qudt.cubicMeterPerHour
_:3 hasValue "4.4"8sd.double
_:3 timestamp "07:15:00"8sd:dateTime
```

**b. Generated Query**

```
SELECT ?value ?timestamp
WHERE {
?x1 hasFlow ?x2
?x2 hasValue ?value
?x2 timestamp ?timestamp
}
```

FIGURE 4.12: Query generation based on a stream

pertinents qui doivent être conservés pour construire la requête. Nous utilisons ce fichier pour identifier les triplets qui sont essentiels pour la construction de la requête, générant une clause WHERE avec un sous-graphe incluant tous les trois (partie b de la figure).

Cette liste d'éléments peut être construite en utilisant des fichiers tabulaires dans notre cas d'utilisation. Nous pouvons associer l'étiquette des colonnes à certaines propriétés de notre ontologie (en rouge sur la figure) et associer les valeurs des colonnes aux objets (en bleu). Ces valeurs représenteront les variables dans notre requête, et les étiquettes ne seront que des triplets pour spécifier la clause *WHERE*. Dans notre exemple, nous avons trois triplets extraits qui sont liés, et nous avons donc réduit le graphique de moitié par rapport à sa taille d'origine.

Dans cet exemple, il ne représente que trois triplets, mais avec le nombre de capteurs, les différents types de mesures et la fréquence des mesures, il peut apporter une amélioration significative. D'autres cas d'utilisation peuvent obtenir des gains différents en fonction des triplets de flux et de la liste des éléments pertinents.

| Zones | measure | Triples | Size |
|---|---|---|---|
| Zone 1 | 75% | 50% | 29% |
| Zone 2 | 66% | 52% | 32% |
| Zone 3 | 72% | 46% | 27% |

TABLE 4.3: Amélioration de Performances via la Génération de Requêtes

Pour chaque zone, nous avons des enregistrements pour 15 types de mesure différents, stockés dans des fichiers par année d'émission. Au total, un million de chaque type de mesure est archivé chaque année, pour une taille sur disque dur de 3,4Go. Le tableau 4.3 présente les gains pour chaque zone avec des requêtes générées sur les archives: la première colonne donne le pourcentage de types de mesures supprimées, la colonne 'Triples' fournit le nombre de triplets du flux conservé dans la requête. Enfin, la dernière colonne représente le gain de taille des flux récupérés de notre requête (par exemple dans la zone 1, 29 % des 1 millions de triplets en streaming ont été traités par notre requête).

La validation de la requête générée a été réalisée avec l'aide d'experts du domaine

pour notre cas d'utilisation. Nous avons généré et exécuté notre requête en utilisant les données fournies, et présenté nos résultats à leur appréciation. Les résultats sont présentés dans le tableau 4.4, pour trois requêtes différentes. Le premier était l'exemple présenté précédemment, et était très satisfaisant. La deuxième était une requête plus spécifique, qui est également très satisfaisante: Un expert s'est plaint d'un paramètre manquant, mais il pourrait être ajouté au résultat en adaptant les données d'entrée. La troisième requête était la plus délicate et visait à obtenir un résultat très précis. Certains évaluateurs ont jugé la précision satisfaisante, d'autres l'ont jugée insuffisante. Comme nous l'avons mentionné, la principale difficulté pour notre système de génération de requêtes est de construire des agrégats en utilisant les données d'entrée.

| Evaluator | Queries | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | X | X | × |
| 2 | X | X | X |
| 3 | X | × | × |
| 4 | X | X | × |
| 5 | X | X | X |

TABLE 4.4: Evaluation des Experts du Domaine

## 4.8 Synthèse

### 4.8.1 Récapitulatif

Dans ce chapitre, nous avons présenté RAMSSES, un système évolutif permettant la détection d'anomalies sur des flux temps réel en utilisant un mélange de techniques d'apprentissage automatique et une approche sémantique.

S'appuyant sur les caractéristiques statistiques des données historiques, le système s'appuie sur un ensemble complexe de règles pour sélectionner le meilleur algorithme qui peut être appliqué pour trouver des singularités de données. Les fonctionnalités identifiées comme support potentiel pour la détection d'anomalies doivent ensuite être extraites des flux de données. Les requêtes qui récupèrent ces informations sont générées automatiquement sous la forme de requêtes *SPARQL* puisque les flux sont modélisés comme des graphes *RD*. Les ensembles de réponses sont ensuite traités par des méthodes d'apprentissage automatique pour détecter les anomalies en temps quasi-réel.

À notre connaissance, RAMSSES est le premier système visant à automatiser le processus complet de réduction de dimension, de sélection de caractéristiques, de génération de requêtes et de détection d'anomalies. Une tâche qui est généralement effectuée manuellement par les utilisateurs finaux ayant besoin d'une certaine expertise dans le domaine d'application et de l'apprentissage automatique. Calculant sur

un environnement distribué pour gérer des données massives, RAMSSES propose l'algorithme qui pourrait donner les meilleurs résultats, donc nous avons également évalué ses performances et son évolutivité à la fois localement et sur un vrai cluster. Enfin, RAMSSES est utilisé en production chez l'un des partenaires de WAVES.

### 4.8.2 Pistes d'Amélioration

Comme il gère les flux RDF, et afin de réduire la complexité des graphes RDF qui pourraient arriver au système, un processus de générateur de requête est appliqué sur le flux avant d'essayer de détecter les anomalies. RAMSSES est incomplet dans ses fonctionnalités et pas totalement optimisé en termes de performances. Les prochaines étapes pour améliorer le système comprennent:

- L'enrichissement de la liste des algorithmes utilisés pour la détection des anomalies dans RAMSSES, et l'utilisation de bibliothèques tierces pour prendre en charge les algorithmes de clustering qui ne sont pas encore pris en charge nativement par Spark.
- Pour l'instant, RAMSSES ne traite que des enregistrements numériques, il ne fonctionne pas avec les données textuelles, nous prévoyons de prendre en charge des algorithmes capables de gérer des données textuelles telles que LDA.
- RAMSSES a été évalué en mode distribué en utilisant un seul ensemble de données, une évaluation plus poussée utilisant des ensembles de données plus importants est prévue.
- L'Utilisation des caractéristique statistique des données historiques dans le processus de sélection de l'algorithme, telles que la dépendance des variables.
- L'Intégration du composant de flux au moteur développé dans RAMSSES et l'évaluation de la capacité du système à gérer des flux de données très véloces.

# Chapter 5

# Scouter

Dans le chapitre précédent, nous avons souligné que pour garantir des détections de haute qualité, des métadonnées fournissant des contextes spatio-temporels sur les mesures des capteurs sont nécessaires. Ces métadonnées deviennent donc une partie intégrante du calcul de détection d'anomalie et doivent être traitées en utilisant une approche en continu. Dans ce chapitre, nous présentons Scouter, un outil générique qui aide à capturer, analyser, marquer et stocker les informations contextuelles d'un domaine d'application donné. Le traitement dépend d'une approche sémantique qui exploite les ontologies pour évaluer la pertinence des informations contextuelles. Ce chapitre fournit des détails sur l'architecture de Scouter, ses différentes fonctionnalités et l'évaluation de ses performances.

## 5.1 Concepts et principes

Au cours de plusieurs expériences menées sur le réseau français, constitué de 100.000 km de pipelines équipés de plus de 3.000 capteurs distribuant de l'eau potable à plus de 12 millions de clients, les experts terrain ont découvert que pour garantir une haute précision de détection des anomalies, une contextualisation des mesures est nécessaire surtout quand une singularité apparaît. Par exemple, une pression anormale ou des signatures de débit particulières pourraient indiquer une fuite d'eau.

Cependant, dans de nombreux cas de manifestations sportives ou culturelles populaires, ces particularités pourraient facilement s'expliquer à cause de certaines dispositions prises par la mairie pour fournir des fontaines d'eau. Les mêmes singularités peuvent également expliquer les conditions de temps chaud impliquant un arrosage de jardin dans une banlieue ou un feu de forêt obligeant les pompiers à utiliser des quantités importantes d'eau. Ainsi, une approche efficace de la contextualisation de la singularité doit intégrer une dimension spatio-temporelle sur les mesures analysées.

Les tâches liées à la capture, à l'analyse, à la notation et au stockage des métadonnées contextuelles sont exigeantes, car elles nécessitent des interactions entre les composants logiciels qui peuvent être difficiles à mettre en place, en particulier dans un environnement distribué. Ces composants gèrent généralement le traitement de flux, de langage naturel et d'ontologies, ainsi que le stockage de structures de données complexes. Conçu comme un système générique, Scouter vise à simplifier ces tâches en proposant des implémentations des principales fonctionnalités et en prenant en charge les aspects d'installation et de configuration.

## 5.2 Architecture

Scouter a été développé pour être un système générique capable de gérer les événements statiques et en streaming et de les analyser en utilisant un ensemble puissant de fonctions NLP et de méthodes sémantiques. Entièrement configurable, l'objectif principal de Scouter est d'extraire efficacement des données de différentes sources Web, de les traiter en peu de temps afin de noter chaque événement concernant sa capacité à expliquer les anomalies détectées par la plateforme. Comme détaillé par la figure ci-dessous, les principaux composants de notre système sont: un ensemble de connecteurs de données Web, une unité d'analyse multimédia, une unité de géolocalisation, un ordinateur central de stockage, un courtier de messagerie et un fournisseur de services Web .

Les connecteurs Web consomment des données provenant de différentes sources de données à une certaine fréquence en fonction de configurations prédéfinies qui peuvent être spécifiées à l'aide d'une interface Web. Ces sources comprennent

- Les réseaux sociaux à savoir Twitter et Facebook (*e.g.*les citoyens commentant les fuites d'eau à proximité)
- Des sources médiatiques telles que les flux RSS de différents journaux répertoriés dans le tableau ci-après (*e.g.* un article du journal Le Monde mentionnant un incendie)
- Des informations météorologiques de l'Open Weather Map (*e.g.* les conditions climatiques lors d'un événement spécifique)
- Des événements organisés de Open Agenda (*e.g.* des concerts, des expositions ou des événements sportifs)
- Des informations spécifiques issues de DBpedia (*e.g.*, type de quartiers).

Toutes ces sources de données sont consommées dans un puissant mécanisme multithreading utilisant des API open-source. Les concepts, sous-concepts et propriétés utilisés pour extraire les données sont représentés dans une ontologie qui formalise

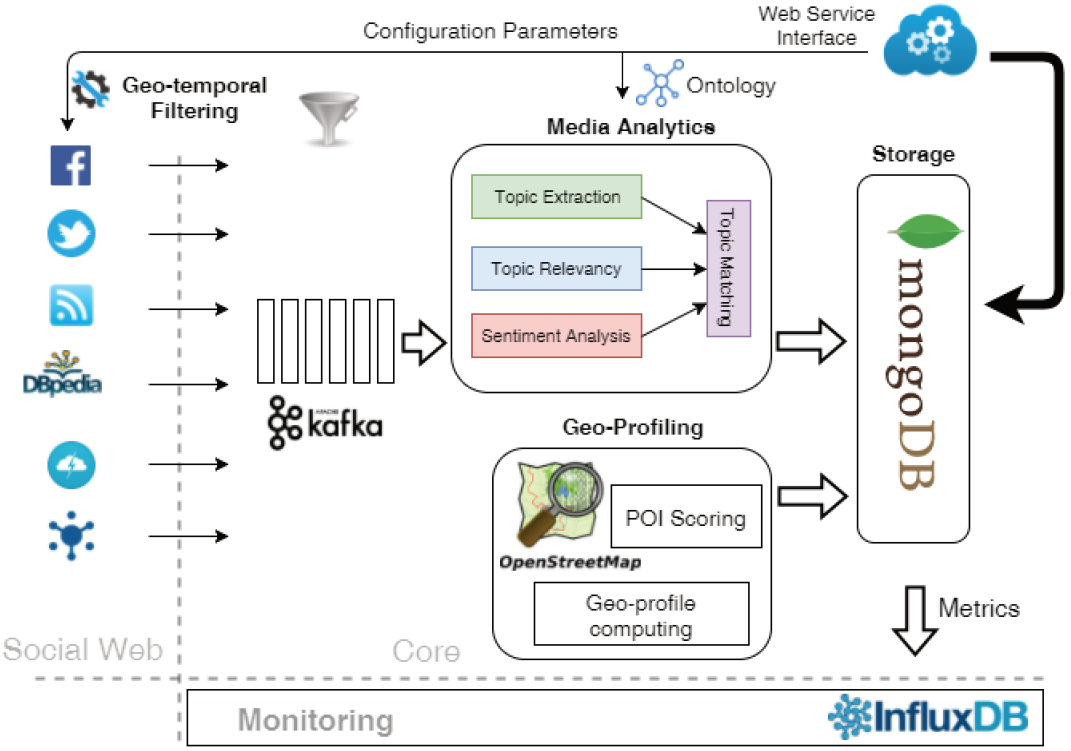


FIGURE 5.1: Architecture de Scouter

les différentes relations de manière à permettre une interprétation sans ambiguité par le système. Plus de détails sur l'ontologie sont fournis dans la suite du chapitre.

L'unité d'analyse des médias digère les flux extraits de Kafka et exploite l'infrastructure distribuée d'Apache Spark pour analyser les flux en temps réel. Ces flux sont enregistrés comme des événements annotés avec la localisation, les dates de début/fin et une description. Afin de filtrer les événements les plus pertinents sans aucun doublon dans la base de données, une approche d'extraction du sujet et une analyse de sentiment sont combinées.

L'extraction via la brique analytique analyse le texte des flux pour découvrir les occurrences de termes. Ensuite, le module de notation tire parti des poids définis par l'utilisateur, c'est-à-dire une valeur réelle dans la plage $[0, 1]$, associée aux concepts d'ontologie pour fournir une notation globale pour chaque texte. L'analyse des sentiments classifie les flux en catégories positives ou négatives en utilisant l'algorithme d'entropie maximum Adam Berger and Pietra, 1996. Il construit un modèle utilisant la régression logistique multinomiale pour déterminer la bonne catégorie pour un texte donné.

Simultanément, l'unité de géolocalisation fournit des caractéristiques géographiques pour la zone analysée. Il détermine le type de terrain entourant l'emplacement de l'anomalie. Sur la base des points d'intérêt et des terrains pour une zone donnée, un profil est généré qui décrit la catégorie de la région ciblée en utilisant une classification configurable tels que résidentiel, touristique, industriel

ou agricole.

Après l'étape de notation, les événements sont enregistrés dans une base de données de documents évolutive et distribuée (*MongoDB*). Par conséquent, nous obtenons des contextes spatio-temporels et notés en temps réel qui peuvent aider l'opérateur à expliquer une anomalie détectée dans le réseau d'eau souterrain. Scouter fournit également un outil de surveillance des mesures pour suivre les performances du système notamment les temps d'exécution des requêtes, les temps de traitement des événements, le nombre d'événements et les durées d'apprentissage de l'extraction des sujets. Ces métriques sont stockées dans une base de données de séries chronologiques avec un accès en lecture/écriture performant (*InfluxDB*). Enfin, le composant de services Web est utilisé pour configurer le système. Il fournit une interface intuitive intégrée avec des outils graphiques pour fournir des paramètres de configuration d'une manière conviviale et lisible.

## 5.3 Combinaison de TAL et Web Sémantique

Dans cette section, nous détaillons la méthodologie de collecte de données à partir des différents types de sources disponibles sur le web. Ce processus repose sur quatre composantes principales ayant vocation à:

- Extraire les événements les plus pertinents en fonction d'un graphe complexe de concepts et de propriétés.
- Identifier les résumés appropriés de chaque événement avec le maximum d'expressivité.
- Éviter les doublons stockés dans la base de données en comparant les résumés et en évaluant la similitude entre eux en utilisant l'analyse des sentiments.

### 5.3.1 Ontologie

Généralement, les outils de scrapping s'appuient sur un fichier de configuration qui répertorie les propriétés des concepts ou des événements qu'ils tentent d'extraire [S Sirisuriya, 2015]. Un traitement sémantique efficace des données ne peut être réalisé sans une modélisation unifiée des flux de données hétérogènes. Le but de cette étape est d'assurer l'unification syntaxique en utilisant un format unique, en l'occurrence *RDF*, et d'utiliser une ontologie au format du Web Sémantique pour spécifier un vocabulaire commun. Un aspect de notre système consiste à coder un ensemble d'ontologies spécifiques au domaine d'application.

Dans le cas de Scouter, les capacités d'extractions des différentes sources web dépendent fortement d'une ontologie prédéfinie qui répertorie les principaux concepts recherchés par l'utilisateur. En tirant parti de la flexibilité d'une ontologie et de

la possibilité de l'étendre en fonction d'autres travaux de recherche, nous avons découvert qu'elle améliorait grandement l'expérience de l'utilisateur et permettait de meilleures interactions avec d'autres modèles sémantiques intéressants.

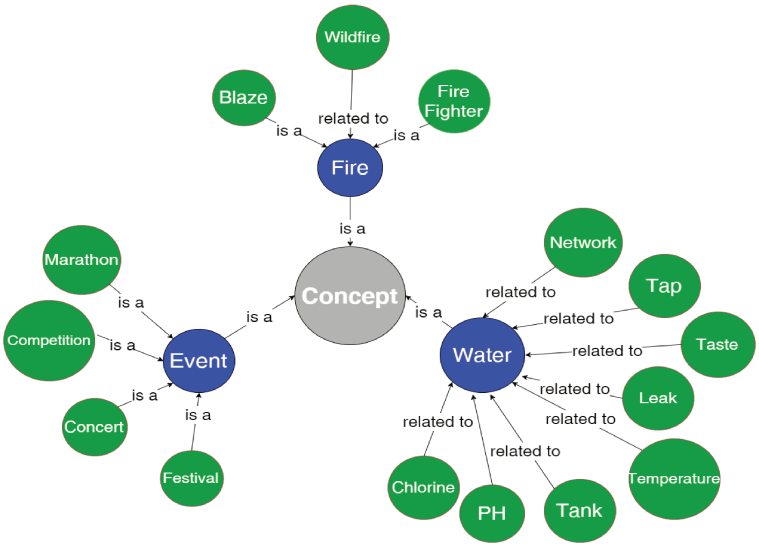


FIGURE 5.2: Extrait de l'Ontologie Scouter

L'ontologie définie permet d'organiser les relations entre les différents concepts et sous-concepts en deux dimensions:

**Hiérarchie Verticale** : Un concept donné (par exemple "Evenement") peut avoir plusieurs sous-concepts (par exemple "Evenement sportif", "Evenement culturel") ou des alias et des fautes d'orthographe.

**Dépendance Horizontale** : Un concept peut avoir plusieurs propriétés qui décrivent un état spécifique sur une période donnée. Par exemple, l'eau peut être potable, ou avoir une couleur spécifique.

En combinant les concepts et les propriétés à travers des prédicats, nous pouvons construire un graphe complexe tel que celui de la figure 5.2 utilisé pour le cas d'utilisation de la fuite d'eau. Nous pouvons facilement faire valoir que ce type de structure contient plus d'expressivité qu'une liste classique de mots-clés exposés dans un fichier de configuration.

### 5.3.2 Extraction de Thèmes

Après avoir récupéré les événements appropriés à partir de diverses sources en fonction des concepts et des sous-concepts de l'ontologie, l'étape suivante consiste à extraire des sujets significatifs des événements qui suivent le workflow décrit dans la Figure 5.3.

Le prétraitement principal concerne ici le nettoyage du texte d'entrée, l'identification des candidats potentiels, et enfin le traitement par TLN des phrases. Les fichiers

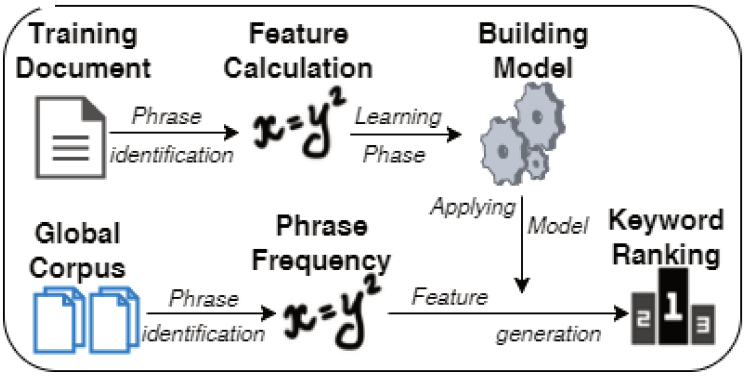


FIGURE 5.3: Workflow Extraction de Thèmes

d'entrée sont filtrés pour régulariser le texte et déterminer les limites des phrases initiales, puis la division en jetons (*token* à côté de plusieurs modifications sont effectuées (les apostrophes sont supprimées, les mots coupés sont scindés en deux, etc.).

Ensuite, nous considérons toutes les sous-séquences afin de déterminer celles qui sont des phrases candidates appropriées. Pour augmenter la précision, nous utilisons une liste de mots-clés français contenant plus de 500 mots dans différentes classes syntaxiques (conjonctions, articles, particules, etc.). Nous listons tous les mots et les analysons en utilisant la méthode Lovins itérée [Lovins, 1968] pour éliminer tout suffixe, et répéter le processus jusqu'à ce qu'il n'y ait plus de changement. Cette méthode itérative permet de traiter différentes variations d'une phrase afin d'éviter les doubles extractions et ne retenir que les éléments pertinents.

Le traitement principal implique deux statistiques calculées pour chaque phrase candidate:

- La fréquence de phrase dans le texte d'entrée par rapport à sa rareté dans l'usage général.
- La première occurrence qui est la distance dans le texte d'entrée de la première apparition de la phrase concernée.

.

Ces deux caractéristiques sont converties en données nominales pour le processus d'apprentissage automatique et une table de discrétisation pour chaque statistique est dérivée des données d'apprentissage. Enfin, nous générons un modèle qui donne les scores de chaque candidat et les classons en utilisant les techniques naïves bayésiennes [Domingos and Pazzani, 1997].

### 5.3.3 Pertinence des Thèmes

Plusieurs travaux de recherche abordent le problème de la récapitulation automatique [Ellouze, Jaoua, and Belguith, 2017]. Les outils proposés mélangent généralement plusieurs classes de fonctionnalités telles que la vraisemblance récapitulative, l'utilisation de signatures spécifiques ou l'analyse syntaxique [Lin, 2004]. Dans notre cas, nous avons choisi une approche directe basée sur la similarité de distribution qui compare les intrants et le contenu généré. En fait, nous considérons qu'un bon résumé doit être caractérisé par une faible divergence entre les distributions de probabilité des mots dans l'entrée et le résumé généré, et par une forte similitude avec l'entrée. Pour ce faire, nous avons utilisé deux mesures communes: la divergence de Kullback Leibler et la divergence de Jensen Shannon.

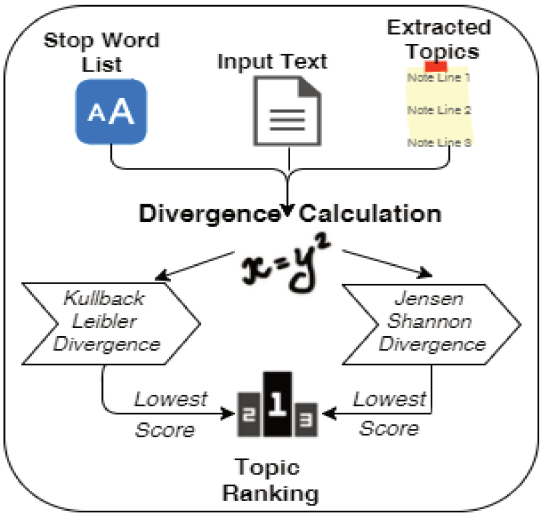


FIGURE 5.4: Workflow Evaluation de Pertinence des Thèmes

Tout d'abord, les mots à la fois en entrée et en résumé sont séparés et triés avant tout calcul. Ensuite, nous calculons les deux mesures:

- **Divergence Kullback Leibler (KL)** : Elle correspond au nombre moyen de bits utilisés en codant des échantillons appartenant à P et en utilisant une autre distribution Q. Cette mesure est donnée par l'équation suivante:

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

  Dans notre cas, les deux distributions de probabilités des mots sont estimées à partir de l'entrée et du résumé. Comme la divergence de KL n'est pas symétrique, le résumé d'entrée et les divergences d'entrée de résumé sont introduits comme métriques. En outre, nous effectuons un lissage simple en utilisant une fonction d'approximation qui capture les signaux importants tout en laissant de côté le bruit et d'autres structures à échelle fine.

- **Divergence Jensen Shannon (JS)** : Celle-ci s'appuie sur le fait que la distance entre deux distributions ne peut être très différente de la moyenne des distances de leur distribution moyenne. Elle est donnée par la formule suivante:

$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M) \tag{5.1}$$

$$\textit{where } M = \frac{1}{2}(P + Q) \tag{5.2}$$

  Contrairement à la divergence KL, la distance JS est symétrique et toujours définie. Nous calculons les versions lissées et non lissées de la divergence sous forme de scores récapitulatifs. La dernière étape consiste à utiliser la sortie de ces deux fonctions pour classer les sujets extraits et conserver uniquement ceux qui ont le meilleur score de synthèse (c'est-à-dire les divergences les plus faibles).

### 5.3.4 Analyse de sentiments

Au cours de la dernière décennie, l'analyse ssentimentale a connu un développement exponentiel en raison de l'utilisation croissante des réseaux sociaux et de la popularité des sites Web où les gens peuvent exprimer leur opinion sur différents produits/évènements et les évaluer. De nombreuses solutions ont été proposées et packagées dans plusieurs technologies [A. Collomb and Brunie, 2014]. Nous proposons dans cette section une approche simple basée sur certains outils fournis par la boîte à outils Stanford CoreNLP [Manning et al., 2014].

Avant d'appliquer le modèle, nous devons lancer plusieurs étapes de prétraitement qui amélioreront la précision du score de sortie. Ici, trois étapes sont impliquées:

- **Tokenisation** : Elle sépare le texte en une séquence de jetons et enregistre les décalages de caractères de chaque jeton dans le texte d'entrée. Ensuite, nous avons divisé une séquence de jetons en phrases significatives.
- **Reconnaissance d'entité** : Elle vérifie si les jetons sont cohérents et conformes à une norme prédéfinie avant d'essayer de déterminer les informations de genre probables aux noms basés sur un dictionnaire. Ensuite, l'algorithme de reconnaissance annote les jetons reconnus en tant que personnes, lieux, organisations, numéros, dates, heures ou durées.
- **Résolution syntaxique** : Le système utilise une analyse syntaxique complète, incluant à la fois la représentation des constituants et des dépendances, basée sur un analyseur probabiliste.

Puisque notre cas d'utilisation est d'analyser les médias et les réseaux sociaux sur le territoire français, nous avons utilisé un dictionnaire français intégré dans un wrapper pour analyser les mots.

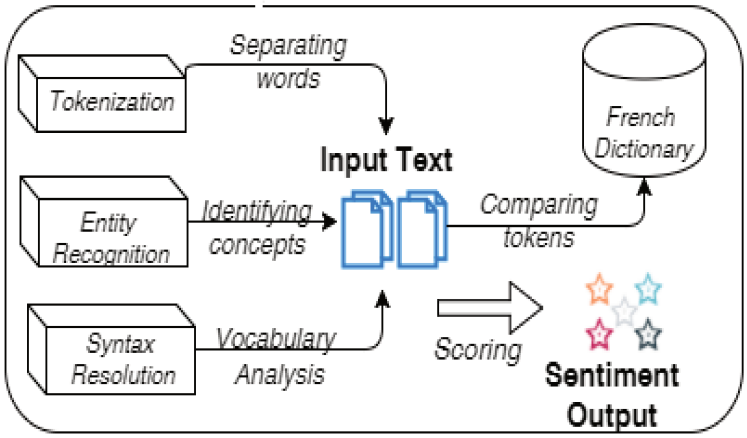


FIGURE 5.5: Worfklow d'Analyse Sentimentale

Après la phase de prétraitement, nous entrons dans l'étape de calcul principale où nous appliquons le modèle. Parmi plusieurs modèles, nous avons choisi les arbres composés basés sur l'apprentissage profond. Il repose sur les nœuds d'un arbre binarisé de chaque phrase, incluant notamment le nœud racine de chaque phrase, qui reçoivent un score de sentiment. Afin de capturer le sentiment d'un texte d'entrée, un modèle de réseau de tenseurs neuronaux récursifs (RNTN) est construit sur la base des caractéristiques des phrases d'entrée. Ces phrases sont représentées en utilisant des vecteurs de mots et un arbre d'analyse, puis nous calculons des vecteurs pour les nœuds supérieurs dans l'arbre en utilisant la même fonction de composition à base de tenseurs. Cette approche s'inspire des modèles profonds récursifs de la compositionnalité sémantique développée par Stanford [Socher et al., 2013].

### 5.3.5 Correspondance des Thèmes

Comme indiqué au début de cette section, l'objectif des différentes étapes de la partie analytique des médias est d'extraire les événements uniques les plus pertinents en les annotant avec un résumé significatif. Par conséquent, nous essayons d'éviter les événements en double qui se réfèrent au même événement ou à la même occurrence. Afin de compléter cette tâche avec une grande précision, nous mélangeons plusieurs approches reposant sur du TAL, de la construction d'ontologies et de l'analyse sentimentale. Le workflow global de notre module d'analyse des médias est détaillé dans la figure 5.6.

Le processus suit les étapes suivantes :

- Pour chaque événement récupéré à partir des différentes sources, la phase d'extraction du sujet proposera une liste de résumés potentiels basés sur une approche bayésienne.

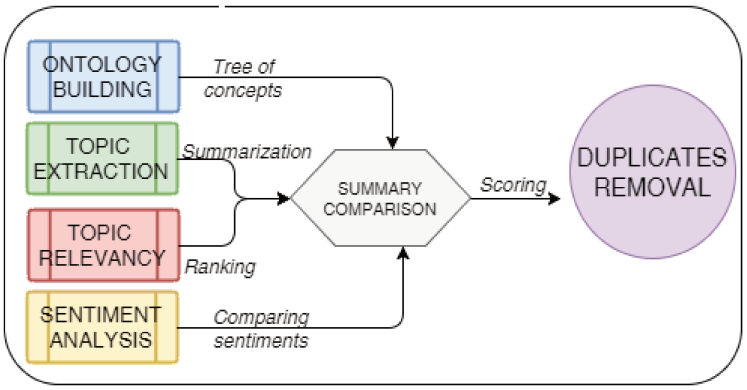


FIGURE 5.6: Workflow de Correspondance des Thèmes

- Ensuite, ces résumés seront classés en utilisant les divergences les plus faibles (c'est-à-dire la divergence KL et la divergence JS) afin d'évaluer leur précision. Parmi les mieux notés, nous vérifierons s'ils ont le même sentiment (c'est-à-dire positif, neutre ou négatif).
- Si l'un des sujets sélectionnés au cours de ce processus a le même sentiment, nous supposons alors qu'ils se réfère au même événement de la même manière. Par conséquent, nous concluons que ces événements sont des doublons et que nous ne conservons que le contenu d'un événement.
- De plus, nous annotons l'événement avec une référence de l'autre événement supprimé pour montrer à l'utilisateur final que cet événement spécifique est présent dans différentes sources.

Même si ce module fournit des fonctions puissantes pour filtrer des événements uniques pertinents, une dimension spatiale est nécessaire pour contextualiser complètement l'anomalie détectée. Cette partie sera expliquée dans la section suivante axées sur les données géographiques, dont les fonctionnalités ont été conjointement développées avec un autre doctorant (Jeremy Lhez).

## 5.4 Profilage Géographique

### 5.4.1 Méthodologie

Le but de Scouter est de fournir des informations pertinentes pour contextualiser et potentiellement expliquer les anomalies détectées. La figure 5.7 donne un aperçu de l'organisation du système de profilage. Le profilage utilise 5 types de terrains différents: résidentiel, agricole, naturel, industriel et touristique. Certaines zones sont plus facilement exploitables en utilisant des POI, d'autres seront représentées majoritairement par des polygones.

Les résultats finaux sont affichés dans le tableau 5.4. Nous avons utilisé la technique décrite en section 5.8, et nous avons fait la moyenne des deux méthodes si la densité

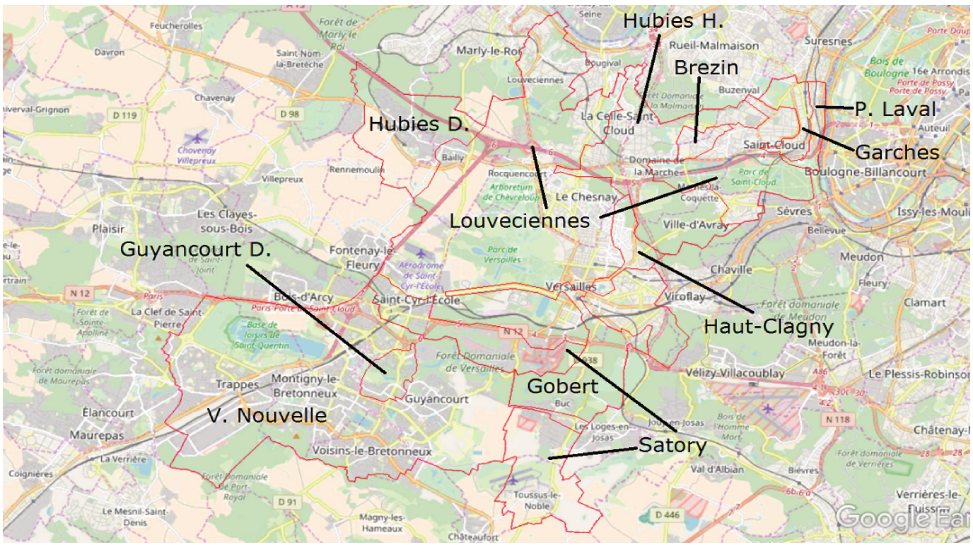


FIGURE 5.7: Secteurs de consommations de Versailles, superposés aux données d'OpenStreetMap

est jugée moyenne. Une fois que nos ajustements sont finalisés, nous avons présenté des résultats à un groupe d'experts du domaine pour recueillir leur avis. Ces évaluations sont majoritairement satisfaisantes, avec quelques remarques pour certains secteurs ne comportant pas de type de terrain majoritaire.

Le module d'analyse des sources médias aide à filtrer les événements pertinents et à supprimer les doublons. Toutefois, des informations géographiques sur la zone où l'anomalie est détectée pourraient affiner et améliorer la précision du contexte. Cette tâche est gérée par deux modules complémentaires: le module de géo-profilage et le module de raisonnement. Il peut être effectué avant le raisonnement, pour orienter la recherche d'événements, ou après, pour changer le classement des sources potentielles. Il est composé de deux méthodes, qui sont combinées et enrichies d'une troisième méthode basée sur la consommation pour de meilleurs résultats.

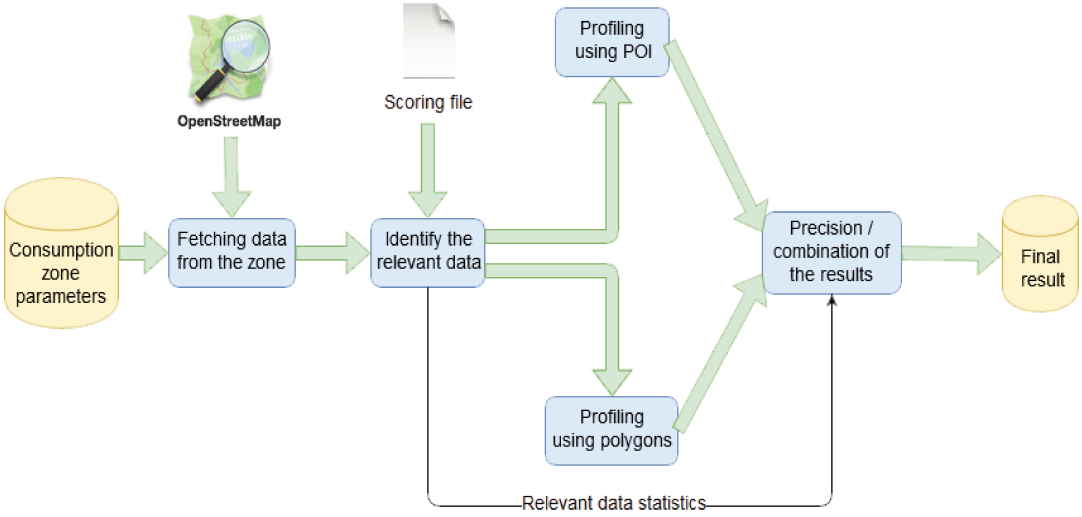


FIGURE 5.8: Workflow du Profilage Géographique

- **Méthode 1** : Elle extrait les points d'intérêt (POI) présents dans un secteur donné à partir de sources de données géographiques en ligne, pour déterminer la proportion des différents types de surfaces qui le composent. L'expert du domaine a sélectionné les types de surfaces correspondant à notre cas d'utilisation, nous donnant cinq paramètres de profilage: résidentiel, naturel, agricole, industriel et touristique. Ensuite, nous avons créé un fichier d'évaluation, en assignant des notes à chaque POI, afin de calculer un score pour chaque type de surface, et calculer sa proportion (c'est-à-dire une valeur réelle dans la plage [0,1]).
- **Méthode 2** : Elle est également basée sur des données géographiques, mais utilise des entités modélisées comme des polygones au lieu de POI. Les tests d'inclusion sont plus complets, car certains polygones peuvent être inclus complètement ou partiellement dans le secteur de la consommation. En outre, le calcul n'est pas effectué en utilisant le système de notation, mais les zones des polygones, qui sont moins arbitraires. Sinon, les deux méthodes produisent le même résultat: les proportions (également en valeur réelle dans la plage [0,1]) pour chaque type de surface.
- **Méthode 3** : Pour certains types de secteurs de consommation, nos deux méthodes peuvent différer légèrement. Pour décider quelle méthode devrait être utilisée dans chaque cas, nous avons ajouté une troisième méthode qui calcule ce que nous appelons *le ratio de consommation*. Pour chaque secteur, nous calculons le débit journalier et faisons une moyenne sur une longue période pour éviter les anomalies; nous divisons ensuite ce débit par la longueur du pipeline sur le secteur pour obtenir le ratio. Un ratio faible correspond à un secteur avec peu de consommateurs, comme les zones rurales, inversement un ratio élevé fait référence à un secteur dense. Le programme sélectionne le meilleur profilage en utilisant ces critères.

Dans le cas d'un résultat mixte, nous calculons la moyenne des méthodes, afin de refléter le fait qu'il peut y avoir à la fois des zones vides et des zones peuplées. Dans la figure 5.8, nous résumons nos stratégies de profilage en mettant en évidence leurs combinaison et les données d'entrée pour chaque méthode.

### 5.4.2 Outils & Ressources

Le module de géo-profilage repose essentiellement sur des données en accès libre issues d'OpenStreetMap: l'identification des sources d'anomalies ne peut être réalisée qu'avec des sources externes, de même que leur classification probabiliste. Le programme établit à partir des informations sur les secteurs de consommation les données à extraire, en construisant une *bounding box* adaptée.

Un fichier de configuration est également fourni au programme, afin de pouvoir établir quelles sont les grandes catégories de terrain à considérer, et quels tags définis par OSM appartiennent à ces catégories. A chaque tag est attribué une note, qui correspond à sa pertinence pour décrire la catégorie de terrain à laquelle il appartient. Le fichier, au format JSON, est donc organisé en une hiérarchie de catégories, incluant les tags en tant que feuille, mais aussi les différentes classes définies par OSM. Cela permet à l'utilisateur de créer ou modifier la configuration du profilage si nécessaire. Un exemple issu du fichier de scores pour le cas d'usage de Waves est fourni sur la figure 5.9, le fichier intégral est disponible en **Annexe C**.

```
"amenity": {
        "tourism": {
                "entertainment": {
                        "arts_centre": 0.8,
                        "casino": 1.0,
                        "community_centre": 0.8,
                        "fountain": 0.3,
                        "gambling": 1.0,
                        "planetarium": 0.8,
                        "theatre": 1.0}
        }
```

FIGURE 5.9: Extrait de l'arborescence du fichier de configuration du profilage de Waves

Les origines des problèmes détectés sont généralement des événements externes affectant les paramètres de qualité sur le réseau. Plusieurs types d'événements peuvent être responsables tels que les catastrophes naturelles, les événements sportifs ou les expositions culturelles. Certains ne sont pas prévisibles et seront remarqués en exploitant les sources appropriées, mais d'autres peuvent être détectés facilement :

- Les événements sportifs sont annoncés quelques semaines à l'avance,
- Un festival culturel peut être programmé plusieurs mois auparavant,
- Certains événements peuvent également être pris en compte, même s'ils ne causent pas directement d'anomalies: les données météorologiques aideront à décider de la pertinence d'autres événements (un festival en plein air sera plus pertinent en tant que cause d'anomalie dans le cas d'un temps chaud).
- Les données statiques peuvent également fournir des informations précises telles que Open Street Map. Il a été sélectionné en raison de sa complétude relative par rapport à d'autres données en ligne comme GeoNames.
- Les données en accès libre fournissent également une grande quantité de détails sur les emplacements entourant notre réseau d'eau. Certaines sont utiles pour améliorer le profilage ou la classification des sources (par exemple la densité de population ou les attractions touristiques), d'autres sont des événements originaux potentiels (par exemple des célébrations religieuses ou des événements typiques réguliers).

## 5.5 Evaluation

Dans cette section, nous discutons des métriques de performance utilisées dans notre système, nous évaluons la partie analytique des médias sur plusieurs dimensions telles que la qualité des événements collectés, puis nous nous concentrons sur le module de géolocalisation.

### 5.5.1 Analyses des Médias

Dans cette section, nous réalisons une expérimentation de collecte d'événements durant 9 heures continues de toutes les sources mentionnées à savoir les réseaux sociaux (Facebook et Twitter), les journaux (flux RSS), Open Agenda, DBPedia et Open Weather Map. La zone géographique d'intérêt cible est un groupe de villes dans la banlieue de Paris, en particulier Versailles. Les paramètres utilisés pour chacune des sources de données sont expliqués dans le tableau 5.1.

| Source | Fetch Frequency | Pages of Interest | Concepts Scores |
|---|---|---|---|
| Facebook | 12 hours | Mon Versailles<br>Versailles Officiel<br>Public Events | meter:1<br>damage:10<br>concert:10<br>spray:1<br>fire:10<br>water:10<br>blaze:1<br>wildfire:10<br>flow:5<br>tank:1<br>chlore:5<br>pressure:5 |
| Twitter | Streaming | @Versailles<br>@monversailles<br>@prefet78<br>#sdis78 | |
| Open Agenda | 24 hours | | |
| Open Weather Map | 4 hours | | |
| DBpedia | 24 hours | | |
| RSS News Papers | 12 hours | Le Parisien<br>78 Actu<br>versailles.fr<br>Sdis78<br>yvelines.gouv.fr | |

TABLE 5.1: Sources de Données de Scoring des Concepts

Twitter a été exécuté en utilisant une API de streaming sur tous les tweets situés dans la zone de délimitation, une attention particulière a été accordée aux événements extraits de certains flux (pages *@Versailles* et *@monversailles*) car il s'agit de comptes officiels de la ville. Les mots-clés utilisés pour interroger ces tweets sont un ensemble de 12 concepts (chacun ayant des sous-concepts dans l'ontologie) avec un poids affecté qui indique la pertinence du mot-clé. Pour cette évaluation, le système est évalué en fonction des mesures suivantes:

- **Nombre d'événements collectés et stockés** : Il représente le nombre de flux reçus provenant de toutes les sources de données pendant l'exécution. La figure 5.10 montre le nombre d'événements collectés et le nombre d'événements

stockés dans la base de données ayant un score supérieur à 0. Nous pouvons voir clairement que beaucoup d'événements collectés (environ 28 % pour cette expérience) ne sont pas pertinents, ils seront donc inutiles pour l'opérateur en tant qu'explication potentielle des anomalies. Néanmoins, une partie substantielle des évènements restants permet une première contextualisation des anomalies.

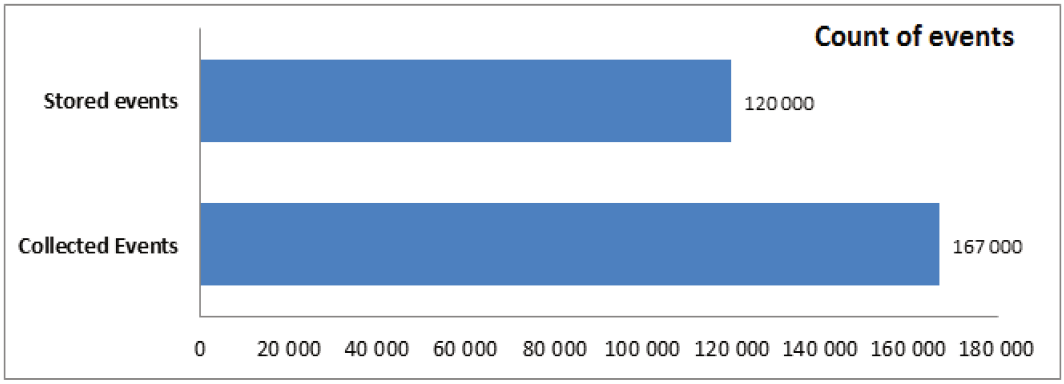


FIGURE 5.10: Evènements Collectés et Stockés durant 9 heures

- **Messages Kafka par seconde** : Il représente le nombre de messages (événements) envoyés au broker Kafka, il permet de connaitre le chargement dans la file d'attente de messagerie. Lorsque Scouter est en cours d'exécution, tous les processeurs commencent à ingérer des données, puis chacun d'eux s'teindra jusqu'au prochain lancement programmé après une certaine fréquence. Ceci explique le pic à l'heure de début de la figure 5.11.

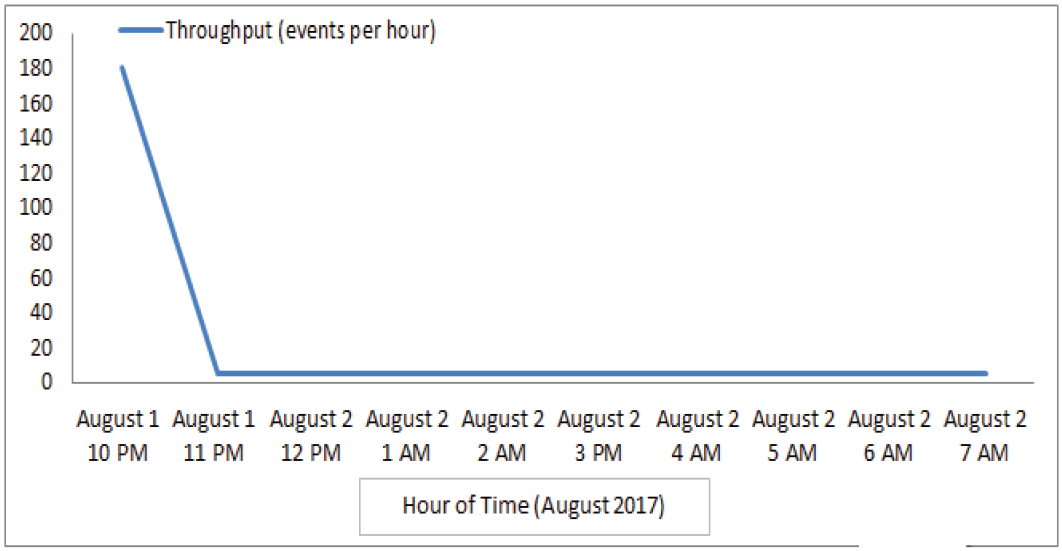


FIGURE 5.11: Débit Kafka

- **Temps d'Entraînement des Modèles d'Extraction de récapitulatifs** : Il représente le temps passé à construire le modèle de traitement de langage naturel pour extraire les sujets et traiter les événements entrants.

| Measure | Time in Millisecond |
|---|---|
| Average Processing Time | 7.43 |
| Topic Extraction Training Time | 474 |

TABLE 5.2: Temps de Traitement Scouter

Le tableau 5.2 indique le temps moyen nécessaire pour marquer tous les événements collectés. En effet, il est calculé en divisant la somme du temps de notation pour chacun des événements par le nombre d'événements collectés. Il montre également le temps de formation pour l'algorithme d'extraction de sujet pour construire le modèle qui est utilisé pour détecter les sujets. Nous pouvons voir que Scouter fonctionne très bien avec un nombre relativement important d'événements arrivant au système sans aucune défaillance ou retard.

### 5.5.2 Qualité des événements collectés

Pour évaluer la qualité des événements collectés, nous avons considéré le cas d'utilisation des fuites d'eau, où les événements ont été extraits des sources de données mentionnées sur 9 heures, en utilisant l'ontologie de la figure 5.2 et le tableau 5.1. L'expert du domaine a fourni l'horodatage et l'emplacement de toutes les anomalies rapportées en 2016 qui se sont élevées à 15 au total. À partir de la base de données, nous avons récupéré tous les événements stockés à proximité de l'horodatage et de l'emplacement de chaque anomalie et les avons présentés à cinq experts du domaine. Pour chaque événement, on leur a demandé s'ils croyaient que cet événement pouvait donner une explication appropriée et pertinente pour l'anomalie signalée. Une contrainte était imposée, la réponse devait être binaire, c'est-à-dire Oui ou Non, afin de simplifier l'interprétation des résultats.

TABLE 5.3: Evaluation des Experts Métier

| Evaluateur | Evénements | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | × | X | × | X | X | X | × | X | × | × | X | × | × | × | × |
| 2 | × | X | × | X | X | × | × | X | × | X | X | × | × | × | × |
| 3 | × | X | × | X | X | × | X | × | × | X | × | X | X | × | × |
| 4 | × | X | × | X | X | × | × | X | × | X | × | × | X | × | × |
| 5 | × | × | × | X | × | × | × | X | × | × | X | × | × | × | × |

Pour évaluer la fiabilité de l'annotation, nous avons utilisé la mesure Kappa de Fleiss [Fleiss, 1971]. C'est une mesure statistique qui vise à évaluer la fiabilité de l'accord entre un certain nombre d'annotateurs lors de l'attribution d'étiquettes à des sujets catégoriels. Cela peut être interprété comme l'expression de la mesure dans laquelle le niveau d'accord observé entre les évaluateurs dépasse ce qui serait attendu si tous les évaluateurs faisaient leurs évaluations de manière complètement aléatoire. Il suit l'équation ci-dessous avec les résultats calculés pour notre scénario de 5 évaluateurs:

$$kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} = \frac{0.84 - 0.5256888889}{1 - 0.5256888889} = 0.6626686657$$

$$where\ \bar{P}_e = \sum_{j=1}^{k} P_j^2 = \sum_{j=1}^{k} (\frac{1}{Nn} \sum_{i=1}^{N} n_{ij}))^2$$

$$And\ P_i = \frac{1}{n(n-1)} \sum_{j=1}^{k} n_{ij}(n_{ij} - 1)$$

$P$ est la moyenne de $Pi$, la mesure dans laquelle les annotateurs sont d'accord pour l'événement. $Pe$ est la somme de $Pj$ au carré, où $Pj$ est la proportion de toutes les assignations qui étaient à la *jième* catégorie. Dans ces équations, $N$ est le nombre d'événements, $n$ est le nombre d'annotateurs et $k$ est le nombre de catégories que nous avons (Oui et Non dans notre cas d'utilisation).

Sur la base de la table d'interprétation des valeurs kappa [Landis and Koch, 1977], nous pouvons dire que les annotateurs ont un accord substantiel sur les événements annotés comme étant une explication pertinente pour une anomalie de fuite d'eau. Par conséquent, nous pouvons dire que le système Scouter a été efficace dans la sélection des événements les plus pertinents et a fourni une aide substantielle pour contextualiser les anomalies liées aux fuites d'eau.

### 5.5.3 Profilage Géographique

Le module de profilage peut être exécuté "hors ligne" dans notre système. Cela signifie qu'il n'a pas besoin d'être intégré dans le traitement du flux. En fait, les données requises en entrée sont essentiellement statiques: les données géographiques sont rarement mises à jour et la configuration du réseau ne change pas beaucoup au fil du temps. Nous avons effectué une série de tests sur plusieurs ensembles de données livrés par l'expert du domaine au niveau national et international.

Les performances du module de raisonnement dépendent principalement de la taille des données extraites d'Open Street Map: les secteurs de consommation plus importants auront plus de données prenant plus de temps à la fin. Le profilage avec des polygones est le plus long car il nécessite l'extraction des POI et des polygones à traiter. En moyenne, le calcul du ratio de consommation est le plus rapide, puisqu'il n'a pas besoin d'être extrait d'Open Street Map.

Le tableau 5.4 détaille la performance de nos différentes méthodes pour le cas d'utilisation de la région de Versailles. Il s'agit d'une zone de 350.000 habitants dans la banlieue de Paris qui est composée de 11 secteurs de consommation. Pour chacun de ces secteurs, nous indiquons le nombre de capteurs de flux présents sur chaque secteur, la taille des données à extraire de Open Street Map (POI et polygones), puis nous indiquons le temps de calcul pour chaque méthode. Comme on peut le voir,

| Zone | Capteurs | Données (Mo) | Méthodes de Profilage | | |
|---|---|---|---|---|---|
| | | | Méthode 3 (ms) | Méthode 1 (ms) | Méthode 2 (ms) |
| P. Laval | 2 | 5.4 | 270 | 25 | 605 |
| V. Nouvelle | 16 | 53.8 | 620 | 282 | 630 |
| Hubies D. | 1 | 5.8 | 190 | 13 | 30 |
| Brezin | 1 | 3.1 | 150 | 8 | 72 |
| Guyancourt | 2 | 4.2 | 161 | 13 | 50 |
| Louveciennes | 19 | 123.2 | 850 | 1118 | 1290 |
| Hubies H. | 13 | 37.15 | 740 | 163 | 180 |
| Haut-Clagny | 4 | 8.6 | 260 | 21 | 32 |
| Garches | 3 | 7.0 | 220 | 205 | 46 |
| Gobert | 3 | 15.4 | 201 | 36 | 105 |
| Satory | 5 | 32.5 | 292 | 103 | 215 |

TABLE 5.4: Performances des 3 Méthodes de Profilage

les régions avec le plus de données et le plus de capteurs ont le temps de traitement le plus long: cela correspond aux zones les plus grandes.

## 5.6 Synthèse

### 5.6.1 Récapitulatif

Dans ce chapitre, nous avons présenté Scouter, un système qui a démontré son utilité dans le projet WAVES pour améliorer la contextualisation des anomalies identifiées. L'objectif principal était de proposer un système générique capable de s'adapter à n'importe quelle plate-forme Big Data quel que soit le moteur utilisé et sans perdre la capacité de traiter différents types de sources de données. Pour atteindre une telle cible, nous avons choisi une approche basée sur des connecteurs de données reposant fortement sur le système de file d'attente de messagerie et nous offrons plusieurs fonctionnalités TAL telles que l'extraction de sujets et l'analyse des sentiments.

Grâce à son packaging Docker sous format conteneur facile à prendre en main, Scouter est déjà utilisé dans d'autres prototypes chez Atos et nous souhaitons l'étendre avec de nouvelles fonctionnalités telles que l'enrichissement d'ontologies basé sur un dictionnaire de concepts et l'identification d'événements dupliqués provenant de différentes sources. Enfin, nous prévoyons d'améliorer l'implémentation en prenant en charge divers formats d'ontologies (par exemple ttl, N3, RDF / XML, etc.) et en ajoutant de nouvelles sources de données pour s'adapter à la plupart des cas d'utilisation (par exemple les informations sur le trafic, etc).

### 5.6.2 Enseignements Tirés

Au cours de ce travail, nous avons appris plusieurs leçons sont la nature diffère selon les divers aspects du projet. En effet, au lieu d'essayer d'adapter notre implémentation à des technologies hétérogènes dans le monde du Big Data, la meilleure approche consistait à créer un pont simple mais puissant qui rendrait l'intégration transparente. Le bon choix consistait à utiliser un système de file d'attente de messagerie tel qu'Apache Kafka, largement utilisé et connu pour sa robustesse. De plus, nous avons découvert que peu de sources de données avec un contenu de haute qualité surpassaient plusieurs sources de données avec un contenu de qualité moyenne à faible. Par conséquent, nous avons décidé de ne conserver que 4 catégories différentes et pour chacune, nous avons sélectionné la source la plus adaptée.

Puisque nous visons à concevoir un outil adaptable à plusieurs cas d'utilisation, opter pour une approche basée sur l'ontologie semble être une solution gagnante. Par conséquent, nous pourrions donner la possibilité à chaque expert de domaine d'utiliser sa propre ontologie avec les concepts spécifiques, les propriétés et les mots-clés qui répondent à ses besoins. Cependant, l'élément clé pour une implémentation réussie est de trouver les bons modèles et les bons scores. Même si beaucoup de bibliothèques étaient disponibles pour les fonctions PNL, beaucoup d'entre elles manquaient de robustesse et de flexibilité. Au lieu de consacrer beaucoup de temps à trouver les technologies les plus adaptées, nous sommes très optimistes quant à la possibilité d'améliorer grandement les résultats en investissant plus de temps dans la modélisation et la notation des ontologies.

Enfin, nous avons découvert que le meilleur moyen de supprimer la complexité consistait à empaqueter le code dans une application Web conviviale. Par conséquent, au lieu de demander aux utilisateurs de connecter Scouter à leur framework, ils devront simplement entrer l'emplacement de l'analyse, les sources de données spécifiques et l'ontologie de domaine appropriée. En ce qui concerne la partie déploiement, l'utilisation de technologies de conteneurisation telles que Docker tend à supprimer la charge de déploiement et peut être lancée en moins d'une minute.

# Chapter 6

# Conclusion

## 6.1 Synthèse

Les travaux de cette thèse ont donné naissance à deux systèmes complémentaires qui, combinés ensemble, permettent le déploiement d'une véritable plateforme innovante multi-paradigmes , c'est-à-dire rapprocher les mondes du sémantique, du TAL et de l'apprentissage automatique.

D'un côté, RAMSSES est un puissant système de détection d'anomalies basé sur Apache Spark avec une approche mixte et originale de l'apprentissage automatique et de la sémantique. Il surmonte le fardeau d'un processus d'apprentissage automatique classique composé de plusieurs tâches humaines longues et sujettes à des erreurs. Le système propose une méthode automatique pour nettoyer les données, réduire la dimensionnalité et la complexité temporelle des algorithmes lourds en utilisant une méthode d'allocation de données originale.

Il implémente un ensemble complexe de règles pour lancer automatiquement l'algorithme le plus efficace qui peut résoudre le cas d'utilisation en parallélisant les tâches entre différents nœuds. Construit au-dessus d'un puissant framework Big Data, RAMSSES offre plus de flexibilité en permettant aux utilisateurs de gérer la mémoire, le disque et l'utilisation du réseau afin d'obtenir un traitement plus efficace.

En résumé, les principales contributions de la plateforme RAMSSES sont:

- Une méthode automatique pour réaliser un prétraitement de données tel qu'une réduction de dimension gérée par un générateur de requêtes continues tirant parti des technologies de web sémantique.
- Une approche automatique pour sélectionner les attributs dans une grande liste multidimensionnelle en analysant le profil de données.

- Un ensemble complexe de règles pour sélectionner automatiquement l'algorithme d'apprentissage automatique approprié en analysant soigneusement le profil des données ingérées, telles que la dépendance aux variables ou le type de distribution.
- Un apprentissage itératif des annotations de l'expert de terrain sur la véracité des anomalies pour améliorer la précision de la détection.
- Une évaluation approfondie des ensembles de données réels et synthétiques pour évaluer la performance globale de la plateforme

De l'autre côté, Scouter est un système original de contextualisation de singularité basé sur Apache Kafka qui propose automatiquement des explications pertinentes aux anomalies détectées en utilisant une approche de filtrage continu. En effet, Scouter propose un puissant outil d'analyse de documents du Web s'appuyant sur les technologies de création d'ontologies et de Web sémantique pour récupérer et classer automatiquement les événements pertinents extraits du Web.

Scouter expose également un ensemble puissant de fonctions de traitement du langage naturel telles que l'extraction de sujets, la pertinence des sujets, la correspondance de thèmes et l'analyse des sentiments, ainsi qu'une méthode originale de géolocalisation pour classer les zones. Enfin, Scouter réduit la complexité des calculs lours en utilisant des frameworks Big Data et élimine la limitation de la taille de l'ensemble de données en implémentant une méthode de filtrage continu. En résumé, les principales contributions de Scouter sont:

- Une méthode efficace pour extraire à la fois des données statiques et de streaming de diverses sources du web à partir d'un graphe hiérarchique de concepts provenant d'une ontologie.
- Une méthode originale de géo-profilage qui donne une représentation détaillée des zones où les événements se produisent. Cette méthode a été conjointement développée avec les travaux d'un autre thésard du laboratoire (*Jeremy Lhez)*
- Une nouvelle approche pour sélectionner des événements non dupliqués pertinents en utilisant de puissantes fonctions de traitement du langage naturel dans un workflow tridimensionnel: extraction de sujet, pertinence de sujet et analyse de sentiment.
- Une technique de filtrage dynamique de données basée sur une combinaison d'analyse des médias et d'annotations de géolocalisation.

## 6.2 Enseignements tirés

Dans cette section, nous présentons les leçons apprises selon plusieurs dimensions distinctes.

- **Architecture** : Puisque Waves vise à être une grande plate-forme de données pour le traitement de flux massifs, la conception d'une architecture modulaire et extensible était la base d'un système puissant. Le fait que nous traitons plusieurs types de données (flux, contenu Web) et que nous gérons différents threads dans le pipeline nous a poussé à trouver la technologie de messagerie la mieux adaptée pour une intégration transparente. Parce que nous travaillons avec différents partenaires académiques, nous devions proposer un cadre prêt à l'emploi qui serait la base commune entre tous les modules développés dans les laboratoires. Le bon choix était d'utiliser un système de files d'attente de messagerie largement reconnu pour sa robustesse. Rapidement, Apache Kafka a semblé être le candidat le plus apte, d'autant plus que de nombreuses plateformes de Big Data en dépendent fortement pour ses capacités de distribution. Passer de simples fiches de données CSV à des services Web réels n'était pas une tâche facile. En effet, la connexion entre les réseaux Atos et Suez était très compliquée à cause des protocoles de sécurité qui étaient différents entre les entreprises. Atteindre les postes informatiques pour résoudre le problème et standardiser la sécurité prenait trop de temps. Nous devions donc utiliser des machines et des réseaux qui n'étaient pas configurés par les administrateurs informatiques de l'entreprise. De plus, l'implémentation des services web dans l'environnement Suez était une technologie SOAP et un code Microsoft C#, alors que les développeurs d'Atos étaient plus spécialistes REST et Java. Enfin, afin de pouvoir gérer toutes les différentes tâches de notre pipeline, l'utilisation d'une architecture de micro-services basée sur des technologies de conteneurisation telles que Docker a été la clé d'un déploiement réussi.

- **Cas d'Usage** : Trouver le bon cas d'utilisation avec suffisamment de données pour supporter la phase de test et de prototypage prenait beaucoup de temps. En effet, les chefs de projet et les développeurs de Suez n'étaient pas des spécialistes du sémantique et de *RDF*, ce qui a ajouté plus de complexité lors des réunions pour expliquer les concepts. De plus, comme nous cherchons également à fournir un contexte aux anomalies détectées, l'analyse du Web à l'aide de puissants outils TAL n'était pas simple en raison des diverses ressources disponibles. Ici, nous avons découvert que peu de sources de données avec un contenu de haute qualité surpassaient plusieurs sources de données avec un contenu de qualité moyenne à faible.

- **Apprentissage automatique** : Nous avons d'abord pensé à développer des algorithmes de clustering, puis nous avons décidé d'utiliser des librairies déjà optimisées disponibles pour différentes parties. Au départ, ous avons commencé par benchmarker FlinqML, mais rapidement nous sommes passés à Spark en raison des ses performances reconnues et du soutien de la communauté qui gère le projet.

- **Packaging** : Notre première approche consistait à fournir un module robuste et complet avec une implémentation déjà intégrée pour le système de file d'attente de messagerie Kafka. Cependant, après avoir examiné l'expérience des utilisateurs à partir de plusieurs autres ressources réutilisables, nous avons découvert que le meilleur moyen de supprimer la complexité consistait à empaqueter le code dans une application Web conviviale. De plus, pour améliorer la maintenabilité du code, nous avons opté pour un système basé sur des connecteurs flexibles qui peuvent facilement être activés ou désactivés en fonction des besoins du cas d'utilisation. Nous avons découvert que cette approche maximise la capacité des utilisateurs finaux à réutiliser notre outil pour divers objectifs tels que le prototypage, la recherche, etc.
- **Déploiement** : En ce qui concerne la partie déploiement, nous avons découvert que l'utilisation de technologies de conteneurisation telles que Docker supprime la charge de déploiement, peut être lancée en moins d'une minute et peut commencer à afficher les résultats sans se soucier de l'implémentation technique. En outre, les conteneurs Docker sont facilement configurables et connectables à d'autres conteneurs qui contiennent des processus différents, ce qui peut constituer un ensemble complexe de composants pour un système global très avancé.

# Appendix A

# Exemples de Requêtes WAVES

# Waves experimental queries

A benchmark testing CONSTRUCT, FILTER, UNION, and GROUP BY query capabilities of Waves system.

## Query 1 (complex)

Returns two types of observation values (low i.e < 5 and moderate i.e > 12) for flow sensors with specific time.

```
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX qudt:<http://data.nasa.gov/qudt/owl/qudt#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cuahsi: <http://his.cuahsi.org/ontology/cuahsi#>

CONSTRUCT{
    ?event ssn:isProducedBy ?sensor;
       ssn:hasValue ?observation;
       ssn:startTime ?time.
    ?observation qudt:numericValue ?value.
    ?sensor ssn:observes ?flow.
}
WHERE {
    {
        ?event ssn:isProducedBy ?sensor;
           ssn:hasValue ?observation;
           ssn:startTime ?time.
        ?observation qudt:numericValue ?value.
        ?sensor ssn:observes ?flow.
        FILTER( ?value > "12"^^xsd:double )
    }
    UNION
    {
        ?event ssn:isProducedBy ?sensor;
           ssn:hasValue ?observation;
           ssn:startTime ?time.
        ?observation qudt:numericValue ?value.
        ?sensor ssn:observes ?flow.
        FILTER( ?value < "5"^^xsd:double )
    }
}
```

## Query 2 (intermediate)

Detect the flow that its observation value is larger than the average of past observations. Computes

the average of past day observation values for flow sensors and returns the observation that exceeds that specific value.



```
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX qudt:<http://data.nasa.gov/qudt/owl/qudt#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cuahsi: <http://his.cuahsi.org/ontology/cuahsi#>

CONSTRUCT{
    ?event ssn:isProducedBy ?sensor.
    ?event ssn:startTime ?time.
    ?observation qudt:numericValue ?value.
    ?sensor ssn:observes ?flow.
}
WHERE {
    ?event ssn:isProducedBy ?sensor;
           ssn:hasValue ?observation;
           ssn:startTime ?time.
    ?observation qudt:numericValue ?value.
    ?sensor ssn:observes ?flow.
}
GROUP BY ?flow
HAVING ( ?value > AVG(?value) )
```

# Query 3 (low complexity)

Returns the list of sensors having an observation value superior to 10

```
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX qudt:<http://data.nasa.gov/qudt/owl/qudt#>

CONSTRUCT{
    ?event ssn:isProducedBy ?sensor;
            ssn:hasValue ?observation;
            ?observation qudt:numericValue ?value.
}
WHERE {
    ?event ssn:isProducedBy ?sensor;
            ssn:hasValue ?observation;
    ?observation qudt:numericValue ?value.
    FILTER(?value > "10"^^xsd:double)
}
```

## Appendix B

# Fichier de Configuration WAVES sous format .trig

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix waves: <http://www.waves-rsp.org/configuration#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://localhost:9091/waves/versailles/> .

<http://localhost:9091/waves/first-tests> {
   <cleaner/3> a waves:Cleaner ;
      waves:consumesStream <rawStream/2> ;
      waves:id "3"^^xsd:int ;
      waves:installation <installation> ;
      waves:producesStream <rawStream/4> .

   <converter/7> a waves:Converter ;
      waves:consumesStream <rawStream/6> ;
      waves:id "7"^^xsd:int ;
      waves:installation <installation> ;
      waves:model """@prefix ssn:<http://purl.oclc.org/NET/ssnx/ssn#> . \r
@prefix qudt:<http://data.nasa.gov/qudt/owl/qudt#> . \r
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> . \r
@prefix waves:<http://example.org/waves#> .\r
$eventId \r
   a ssn:SensorOutput;\r
   ssn:isProducedBy $uri(?id);\r
   ssn:hasValue $bnode(?obsId);\r
   ssn:startTime $timestamp .\r
```

```
$bnode(?obsId)\r
   a ssn:ObservationValue ;\r
   qudt:numericValue $double(?value, ?unit, ?scale) ;\r
   qudt:unit $value(?unit) .\r
      """ ;
      waves:producesStream <stream/9> ;
      waves:query """prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> \r
prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> \r
prefix ssn:<http://purl.oclc.org/NET/ssnx/ssn#> \r
prefix qudt:<http://data.nasa.gov/qudt/owl/qudt#> \r
select * \r
where { \r
   ?sensorUri rdf:type ssn:Sensor ; \r
      qudt:unit ?unit ; \r
      rdfs:label ?sensorName . \r
}\r
      """ .

   <rawSource/1> a waves:RawSource ;
      waves:id "1"^^xsd:int ;
      waves:installation <installation> ;
      waves:producesStream <rawStream/2> .

   <sampler/5> a waves:Sampler ;
      waves:consumesStream <rawStream/4> ;
      waves:id "5"^^xsd:int ;
      waves:installation <installation> ;
```

```
   waves:producesStream <rawStream/6> .

<scouter/8> a waves:Scouter ;
   waves:id "8"^^xsd:int ;
   waves:installation <installation> .

<stream/9> a waves:Stream ;
   waves:id "9"^^xsd:int ;
   waves:installation <installation> .

<rawStream/2> a waves:RawStream ;
   waves:id "2"^^xsd:int ;
   waves:installation <installation> .

<rawStream/4> a waves:RawStream ;
   waves:id "4"^^xsd:int ;
   waves:installation <installation> .

<rawStream/6> a waves:RawStream ;
   waves:id "6"^^xsd:int ;
   waves:installation <installation> .

<installation> a waves:Installation ;
   rdfs:label "installation" ;
   waves:clock [ waves:acceleration "80.0"^^xsd:float ;
         waves:localTimeZone "False" ;
         waves:startDate "2013-12-31T23:00:00+00:00"^^xsd:dateTime ] ;
```

```
    waves:createdAt "2018-05-22 15:59:01" ;
    waves:description "This is our first demo project for test." ;
    waves:influxHost "localhost:8086" ;
    waves:kafkaHosts "localhost:9092" ;
    waves:license "Open Data Common Open Database License (CC BY 3.0)" ;
    waves:metrics [ waves:frequency "PT15S"^^xsd:duration ;
        waves:reporters "influxdb, jvm" ] ;
    waves:mongoHost "localhost:27017" ;
    waves:name "first-tests" ;
    waves:redisHost "localhost:6379" ;
    waves:version "1.0.0" ;
    waves:zookeeperHosts "localhost:2181" .
}
```

## Appendix C

# Fichier de Configuration du Profilage

```
{
    "amenity" : {
        "tourism" : {
            "entertainment" : {
                "arts_centre" : 0.8 ,
                "casino" : 1.0 ,
                "community_centre" : 0.8 ,
                "fountain" : 0.3 ,
                "gambling" : 1.0 ,
                "planetarium" : 0.8 ,
                "theatre" : 1.0
            },
            "other" : {
                "sauna" : 0.5 ,
                "shower" : 0.7 ,
                "toilets" : 0.2 ,
                "water_point" : 0.7
            }
    },
        "natural" : {
            "sustenance" : {
                "bbq" : 0.8 ,
                "drinking_water" : 0.3
            }
    },
        "industrial" : {
            "transportation" : {
                "bicycle_parking" : 0.8 ,
                "charging_station" : 0.8
            },
            "other" : {
                "coworking_space" : 1.0
            }
    },
        "housing" : {
            "education" : {
                "kindergarten" : 0.8 ,
                "library" : 0.8 ,
                "school" : 0.8
            },
            "healthcare" : {
                "doctors" : 0.8 ,
                "pharmacy" : 0.8
            },
            "other" : {
                "grave_yard" : 0.7 ,
                "post_box" : 0. 7,
                "townhall" : 0.7
```

```
            }
        }
},
    "building"    : {
        "tourism"   : {
            "accomodation"  : {
                "hotel"   :  0.8
            },
            "civil"    : {
                "cathedral"    :  1.0 ,
                "temple"  :  1.0
            },
            "other"   : {
                "ruins"   :  0.6
            }
    },
        "industrial"      : {
            "commercial"  : {
                "commercial"  :  0.8 ,
                "industrial"      :  1.0 ,
                "warehouse" :  0.8
            }
    },
        "housing"  : {
            "accomodation"  : {
                "apartments"  :  1.0 ,
                "house" :  1.0 ,
                "residential"      :  1.0
            }
        }
},
    "geological"    : {
        "tourism"   : {
            "other"   : {
                "paleontological_site"        :  0.8
            }
    },
        "natural"    : {
            "other"   : {
                "moraine"  :  1.0 ,
                "outcrop"  :  1.0
            }
        }
},
    "historic"     : {
        "tourism"   : {
            "other"   : {
                "archaelogical_site"        :  0.8 ,
```

```
                    "castle"    :   1.0 ,
                    "city_gate"     :   0.8 ,
                    "citywalls"     :   0.8 ,
                    "monastery"  :   0.8 ,
                    "monument":   1.0
                }
            }
    },
        "landuse"   : {
            "tourism"    : {
                "other"   : {
                    "retail"     :   0.5
                }
        },
            "natural"    : {
                "other"   : {
                    "forest"    :   1.0 ,
                    "grass"  :   0.8 ,
                    "meadow":   1.0 ,
                    "village_green"     :   1.0
                }
        },
            "agricultural"       :{
                "other"   : {
                    "farmland"   :   1.0 ,
                    "farmyard"   :   0.8 ,
                    "greenfield"     :   1.0 ,
                    "orchard"   :   1.0 ,
                    "vineyard"   :   1.0
                }
        },
            "industrial"      : {
                "other"   : {
                    "commercial"   :   1.0 ,
                    "industrial"      :   1.0
                }
        },
            "housing"   : {
                "other"   : {
                    "residential"      :   1.0
                }
            }
    },
        "leisure"    : {
            "tourism"    : {
                "other"   : {
                    "beach_resort"    :   1.0 ,
                    "garden"  :   0.5 ,
```

```
                    "summer_camp": 1.0,
                    "swimming_area": 1.0,
                    "water_park": 0.8
                }
    },
        "natural": {
            "other": {
                "firepit": 1.0,
                "garden": 1.0,
                "golf_course": 1.0,
                "nature_reserve": 1.0,
                "park": 1.0,
                "wildlife_hide": 1.0
            }
    },
        "industrial": {
            "other": {
                "hackerspace": 1.0
            }
        }
},
    "natural": {
        "natural": {
            "surface": {
                "scrub": 1.0,
                "heath": 1.0,
                "moore": 1.0,
                "grassland": 1.0,
                "fell": 1.0,
                "bare_rock": 1.0,
                "scree": 1.0,
                "shingle": 1.0,
                "sand": 0.7,
                "mud": 1.0
            },
            "water": {
                "water": 0.8,
                "wetland": 0.8,
                "glacier": 1.0,
                "bay": 0.8,
                "beach": 0.7,
                "coastline": 0.8,
                "spring": 0.8,
                "hot_spring": 0.6,
                "geyser": 0.6
            },
            "landform": {
                "peak": 1.0,
```

```
                "volcano"  :  1.0 ,
                "valley"  :  0.8 ,
                "river_terace"  :  1.0 ,
                "ridge"  :  1.0 ,
                "arete"  :  1.0 ,
                "cliff"  :  1.0 ,
                "saddle"  :  1.0 ,
                "rock"  :  1.0
            }
        }
},
    "office"  : {
        "industrial"  : {
            "other"  : {
                "accountant"  :  1.0 ,
                "advertising_agency"  :  1.0 ,
                "architect"  :  1.0 ,
                "company" :  1.0 ,
                "employment_agency"  :  1.0 ,
                "insurance"  :  1.0 ,
                "it"  :  1.0 ,
                "lawyer"  :  1.0 ,
                "ngo" :  1.0 ,
                "notary"  :  1.0 ,
                "private_investigator"  :  1.0 ,
                "research"  :  0.8 ,
                "tax"  :  1.0 ,
                "tax_advisor"  :  1.0 ,
                "t elecommunication"  :  1.0 ,
                "water_utilisty"  :  1.0
            }
        }
},
    "shop" : {
        "housing"  : {
            "food"  : {
                "bakery"  :  0.8 ,
                "convenience"  :  0.6
            }
        }
},
    "sport"  : {
        "natural"  : {
            "other"  : {
                "beachvolleyball"  :  1.0 ,
                "bmx" :  1.0 ,
                "equestrian"  :  1.0 ,
                "free_flying"  :  1.0 ,
```

```
                    "golf"    :   1.0 ,
                    "model_aedrome" :   1.0 ,
                    "motocross"   :   1.0 ,
                    "orienteering"      :   1.0
                }
            }
        }
        "tourism"    :  {
            "tourism"    : {
                "other"    : {
                    "apartment"    :   1.0 ,
                    "attraction"       :   1.0 ,
                    "campsite"    :   1.0 ,
                    "chalet"    :   0.8 ,
                    "gallery"     :   0.7 ,
                    "guest_houses"    :   0.8 ,
                    "hostel"    :   0.8 ,
                    "hotel"    :   1.0 ,
                    "information"      :   1.0 ,
                    "museum":   0.7 ,
                    "picnic_site"       :   0.6 ,
                    "theme_park"   :   1.0 ,
                    "view_point"    :   1.0 ,
                    "zoo"   :   1.0
                }
            }
        }
}
```

# Bibliography

A. Collomb C. Costea, D. Joyeux O. Hasan and L. Brunie (2014). *A Study and Comparison of Sentiment Analysis Methods for Reputation Evaluation*. Tech. rep.

Adam Berger, Stephen Della Pietra and Vincent Della Pietra (1996). "A Maximum Entropy Approach to Natural Language Processing". In: *Computational Linguistics-MIT*, pp. 22–1.

Aggarwal, Charu C. and Chandan K. Reddy (2013). *Data Clustering: Algorithms and Applications*. 1st. Chapman & Hall/CRC. ISBN: 1466558210, 9781466558212.

Ankerst, Mihael et al. (1999). "OPTICS: Ordering Points to Identify the Clustering Structure". In: *SIGMOD Rec.* 28.2, pp. 49–60. ISSN: 0163-5808. DOI: 10.1145/304181.304187. URL: http://doi.acm.org/10.1145/304181.304187.

Bengio, Yoshua (2009). "Learning Deep Architectures for AI". In: *Foundations and Trends® in Machine Learning* 2.1, pp. 1–127. ISSN: 1935-8237. DOI: 10.1561/2200000006. URL: http://dx.doi.org/10.1561/2200000006.

Bolton, Richard J. and David (2002). "Statistical fraud detection: A review". In: *Statistical Science* 17.

Boutsidis, Christos, Michael W. Mahoney, and Petros Drineas (2008). "Unsupervised Feature Selection for Principal Components Analysis". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: ACM, pp. 61–69. ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401903. URL: http://doi.acm.org/10.1145/1401890.1401903.

Carbone, Paris et al. (2015). "Apache Flink™: Stream and Batch Processing in a Single Engine". In: 38.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (2009). "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3, 15:1–15:58. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: http://doi.acm.org/10.1145/1541880.1541882.

Chen, Guoqiang Jerry et al. (2016). "Realtime Data Processing at Facebook". In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. San Francisco, California, USA: ACM, pp. 1087–1098. ISBN: 978-1-4503-3531-7. DOI: 10.1145/2882903.2904441. URL: http://doi.acm.org/10.1145/2882903.2904441.

Craig Chambers Ashish Raniwala, Frances Perry Stephen Adams Robert R. Henry Robert Bradshaw Nathan Weizenbaum (2013). "FlumeJava: Easy, Efficient Data-Parallel Pipelines". In: *Google Inc.*

Deng, Li and Dong Yu (2014). "Deep Learning: Methods and Applications". In: *Foundations and Trends® in Signal Processing* 7.3–4, pp. 197–387. ISSN: 1932-8346. DOI: 10.1561/2000000039. URL: http://dx.doi.org/10.1561/2000000039.

Domingos, P. and M. Pazzani (1997). "On the optimality of the simple bayesian classifier under zero-one loss". In: *Machine Learning*.

D.T. Pham, S.Dimov and C.D. Nguyen (2005). "Selection of K in K-means clustering". In: *Proceedings of Mechanical Engineering Conference*. Vol. 219. IMECHE.

Ellouze, Samira, Maher Jaoua, and Hadrich Belguith (2017). "Machine Learning Approach to Evaluate MultiLingual Summaries". In: *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*.

Erfani, Sarah M. et al. (2016). "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning." In: *Pattern Recognition* 58, pp. 121–134.

Fischer, Lorenz, Thomas Scharrenbach, and Abraham Bernstein (2013). "Scalable Linked Data Stream Processing via Network-Aware Workload Scheduling". In: *Proceedings of the 9th International Workshop on Scalable Semantic Web Knowledge Base Systems*, pp. 81–96.

Fleiss, J.L. et al. (1971). "Measuring nominal scale agreement among many raters". In: *Psychological Bulletin* 76.5, pp. 378–382.

García, Norberto Fernández et al. (2014). "RDSZ: An Approach for Lossless RDF Stream Compression". In: *The Semantic Web: Trends and Challenges - 11th International Conference*, pp. 52–67.

Goldstein, Markus (2015). *Unsupervised Anomaly Detection Benchmark*. DOI: 10.7910/DVN/OPQMVF. URL: http://dx.doi.org/10.7910/DVN/OPQMVF.

Guarino, Nicola (1998). "Formal Ontology and Information Systems". In: *FOIS'98)* 20.1.

Hirzel, Martin et al. (2014). "A Catalog of Stream Processing Optimizations". In: *ACM Comput. Surv.* 46.4, 46:1–46:34. ISSN: 0360-0300. DOI: 10.1145/2528412. URL: http://doi.acm.org/10.1145/2528412.

Hodge, Victoria J. and Jim Austin (2004). "A Survey of Outlier Detection Methodologies." In: *Artif. Intell. Rev.* 22.2, pp. 85–126.

ITISE Conference, Selected Contributions from the (2016). *Time Series Analysis and Forecasting*. Springer.

Jeffrey Dean, Sanjay Ghemawat (2004). "MapReduce: Simplified Data Processing on Large Clusters". In: *Google*.

Kolchin, Maxim et al. (2016). "YABench: A Comprehensive Framework for RDF Stream Processor Correctness and Performance Assessment". In: *Web Engineering - 16th International Conference, ICWE, Lugano, Switzerland, June 6-9, 2016.* Pp. 280–298.

Landis, J. Richard and Gary G. Koch (1977). "The Measurement of Observer Agreement for Categorical Data". In: *Biometrics* 33.1, pp. 159–174. ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/2529310.

Laptev, Nikolay, Saeed Amizadeh, and Ian Flint (2015). "Generic and Scalable Framework for Automated Time-series Anomaly Detection". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 1939–1947.

Lazarevic, Aleksandar and Vipin Kumar (2006). "Feature bagging for outlier detection." In: *KDD*. ACM, pp. 157–166.

Lazarevic, Aleksandar et al. (2003). "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection." In: *SDM*. SIAM, pp. 25–36.

Lin, Chin-Yew (2004). "ROUGE: A Package for Automatic Evaluation of summaries". In: *Proc. ACL workshop on Text Summarization Branches Out*.

Liu, Huawen et al. (2015). "Penalized partial least square discriminant analysis with l1 for multi-label data." In: *Pattern Recognition* 48.5, pp. 1724–1733.

Lovins, J.B. (1968). "Development of a stemming algorithm." In: *Mechanical Translation and Computational Linguistics*.

Manning, Christopher D. et al. (2014). "The Stanford CoreNLP Natural Language Processing Toolkit." In: *ACL (System Demonstrations)*. The Association for Computer Linguistics.

Marra, Francesco et al. (2017). "A Deep Learning Approach for Iris Sensor Model Identification". In:

Matt Zwolenski Lee Weatherill, et al (2014). "The digital universe: Rich data and the increasing value of the internet of things". In: *Australian Journal of Telecommunications and the Digital Economy,*

Michael Ovsiannikov Silvius Rus, Damian Reeves and Paul Sutter (2011). "The Quantcast File System". In: *Quantcast*.

Middleton, Anthony M. (2011). "High-Performance Computing Cluster". In: *LexisNexis*.

Mohamed, Abdel rahman, George Dahl, and Geoffrey E. Hinton (2009). "Deep Belief Networks for phone recognition". In:

Nathan Marz, James Warren (2015). *Big Data Principles and best practices of scalable realtime data systems*. 1st. Manning. ISBN: 9781617290343.

Phuoc, Danh Le et al. (2013). "Elastic and Scalable Processing of Linked Stream Data in the Cloud". In: *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, pp. 280–297. DOI: 10.1007/978-3-642-41335-3_18. URL: http://dx.doi.org/10.1007/978-3-642-41335-3_18.

P.Trebuna, J.Halcinova and M.Fil'o (2014). "The importance of normalization and standardization in the process of clustering". In: *Proceedings of IEEE 12th International Symposium on Applied Machine Intelligence and Informatics*. SAMI.

*Redis, Dec 2015*.

S Sirisuriya, S.C.M. de (2015). "A Comparative Study on Web Scraping". In: *International Research Conference*.

Sander, Jörg et al. (1998). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications". In: *Data Min. Knowl. Discov.* 2.2, pp. 169–194. ISSN: 1384-5810. DOI: 10.1023/A:1009745219419. URL: http://dx.doi.org/10.1023/A:1009745219419.

Shvachko, Konstantin et al. (2010). "The Hadoop Distributed File System". In: *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. MSST '10. Washington, DC, USA: IEEE Computer Society, pp. 1–10. ISBN: 978-1-4244-7152-2. DOI: 10.1109/MSST.2010.5496972. URL: http://dx.doi.org/10.1109/MSST.2010.5496972.

Shyu, Mei-Ling et al. (2008). "Principal Component-based Anomaly Detection Scheme." In: *Foundations and Novel Approaches in Data Mining*. Vol. 9. Studies in Computational Intelligence. Springer, pp. 311–329. URL: http://dblp.uni-trier.de/db/series/sci/sci9.html#ShyuCSC06.

Singh, B., N. Kushwaha, and O. P. Vyas (2013). "Exploiting the anomaly detection for high dimensional data using descriptive approach of data mining". In: *2013 4th International Conference on Computer and Communication Technology (ICCCT)*, pp. 121–128.

Socher, Richard et al. (2013). "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Suzuki, Einoshin et al. (2003). "Detecting Interesting Exceptions from Medical Test Data with Visual Summarization." In: *ICDM*. IEEE Computer Society, pp. 315–322.

Toshniwal, Ankit et al. (2014a). "Storm@Twitter". In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. SIGMOD '14. Snowbird, Utah, USA: ACM, pp. 147–156. ISBN: 978-1-4503-2376-5. DOI: 10.1145/2588555.2595641. URL: http://doi.acm.org/10.1145/2588555.2595641.

– (2014b). "Storm@Twitter". In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. SIGMOD '14. Snowbird, Utah, USA: ACM, pp. 147–156.

Tyler Akidau Robert Bradshaw, Craig Chambers Slava Chernyak Rafael J. Fernandez-Moctezuma Reuven Lax Sam McVeety Daniel Mills Frances Perry Eric Schmidt Sam Whittle (2013a). "The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing". In: *Google Inc.*

Tyler Akidau Alex Balikov, Kaya Bekiroglu Slava Chernyak Josh Haberman Reuven Lax Sam McVeety Daniel Mills Paul Nordstrom Sam Whittle (2013b). "MillWheel: Fault-Tolerant Stream Processing at Internet Scale". In: *Google Inc.*

Vargas, Rocio, Amir Mosavi, and Ramon Ruiz (2017). "DEEP LEARNING: A REVIEW". In: 5.

Vavilapalli, Vinod Kumar et al. (2013). "Apache Hadoop YARN: Yet Another Resource Negotiator". In: *Proceedings of the 4th Annual Symposium on Cloud Computing*. SOCC '13. Santa Clara, California: ACM, 5:1–5:16. ISBN: 978-1-4503-2428-1. DOI: 10.1145/2523616.2523633. URL: http://doi.acm.org/10.1145/2523616.2523633.

Wang, Guozhang et al. (2015). "Building a Replicated Logging System with Apache Kafka." In: *Proc. VLDB Endow.* 8, pp. 1654–1665.

Xue, Zhenxia, Youlin Shang, and Aifen Feng (2010). "Semi-supervised outlier detection based on fuzzy rough C-means clustering." In: *Mathematics and Computers in Simulation* 80.9, pp. 1911–1921.

YAHOO (2015). *S5 - A Labeled Anomaly Detection Dataset*. URL: https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70.

Yazici, Berna and Senay Asma (2007). "A comparison of various tests of normality". In: 77, pp. 175–183.

Zaharia, Matei et al. (2010). "Spark: Cluster Computing with Working Sets". In: *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10*. URL: https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets.

Zaharia, Matei et al. (2012). "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing". In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. NSDI'12. San Jose, CA: USENIX Association, pp. 2–2. URL: http://dl.acm.org/citation.cfm?id=2228298.2228301.

Zhang, Jun, Henry Shu-Hung Chung, and Wai-Lun Lo (2007). "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms." In: *IEEE Trans. Evolutionary Computation* 11.3, pp. 326–335.

Zou, Qiong et al. (2010). "From a Stream of Relational Queries to Distributed Stream Processing". In: *Proc. VLDB Endow.* 3.1-2, pp. 1394–1405. ISSN: 2150-8097. DOI: 10.14778/1920841.1921012. URL: http://dx.doi.org/10.14778/1920841.1921012.

Çinar, Salim and Nurettin Acir (2017). "A novel system for automatic removal of ocular artefacts in EEG by using outlier detection methods and independent component analysis." In: *Expert Syst. Appl.* 68, pp. 36–44.