UQAC
Université du Québec
à Chicoutimi

ARCHITECTURE, LOGIQUE FLOUE, CLASSIFICATION ET CLUSTERING POUR L'EXPLORATION DE DONNÉES RÉELLES ISSUES DE MULTIPLES MAISONS INTELLIGENTES

PAR CÉDRIC DEMONGIVERT

THÈSE PRÉSENTÉE À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI COMME EXIGENCE PARTIELLE EN VUE DE L'OBTENTION DU GRADE DE PHILOSOPHIÆ DOCTOR (PH.D.) EN INFORMATIQUE

QUÉBEC, CANADA

© CÉDRIC DEMONGIVERT, 2022

RÉSUMÉ

Selon l'Organisation des Nations Unies (2019), il est estimé que le nombre de personnes âgées de 65 ans et plus dans le monde devrait doubler d'ici 2050 pour atteindre un total de 1,5 milliard d'individus. Cette évolution impacte le risque de déclarer une maladie neurodégénérative qui, selon le rapport de l'Alzheimer's Disease International (2015), augmenterait exponentiellement avec l'âge. Or, ces maladies provoquent une perte progressive d'autonomie ce qui induit des besoins en ressources matérielles et humaines. En parallèle de ces enjeux, des progrès allient technologie et santé, comme le dépistage automatique du cancer du sein, présenté dans le travail de Karabatak (2015). Avec l'amélioration continue des environnements intelligents et du matériel, notamment suite à l'apparition de technologies comme l'Arduino (2005), la Rasbperry Pi (2012) ou la Latte Panda (2016), on est en droit d'imaginer les applications possibles de la maison intelligente aux problématiques posées par l'accroissement des maladies dégénératives. C'est ce que nous proposons dans cette thèse, où nous faisons le point sur les environnements intelligents et la reconnaissance d'activités. Point duquel nous dérivons deux propositions : d'une part, une architecture basée sur la gestion d'un flux d'événements descriptifs par des compositions d'agents autonomes permettant le support de plus de 30 environnements hétérogènes. D'autre part, une approche basée sur la logique floue permettant de conduire un processus de reconnaissance d'activités malgré la grande diversité de nos jeux de données. Ces deux solutions participant à l'élaboration d'un outil permettant aux cliniciens de suivre à distance, l'évolution du comportement de patients atteints de maladies dégénératives.

According to the United Nations (2019), the world population aged 65 years and more will double up to 1.5 billion individuals before 2050. This trend will impact the growth of neurodegenerative disorders that are subject to an exponential risk of appearance with aging, as reported by Alzheimer's Disease International (2015). As this kind of disease induces a decrease in the autonomy of the elderly, this evolution will heavily increase the need for human and material resources around the world. In parallel, various research works combine technology and healthcare, like for the automatic breast cancer detection described in the article of Karabatak (2015). This tendency, in conjunction with hardware and intelligent environments improvement, notably with the Arduino (2005), the Rasbperry Pi (2012), and the Latte Panda (2016), affords us to imagine how smart-homes could solve the implications of the aforementioned growth of degenerative diseases. To investigate this question, this thesis derivates two proposals from a careful study of intelligent environments and activity recognition methods. The first is an architecture that supports more than 30 heterogeneous environments and that works by assembling autonomous agents for processing a flux of descriptive events. Finally, the second is a model built upon fuzzy logic that enables us to recognize activities despite the inherent diversity of our datasets. These two solutions answer some aspects of the process of making a tool that allows clinicians to monitor people with degenerative diseases from their homes.

TABLE DES MATIÈRES

RÉSUMÉ ii

LISTE DES TABLEAUX viii

LISTE DES FIGURES x

LISTE DES ABRÉVIATIONS xv

DÉDICACE xviii

REMERCIEMENTS xix

CHAPITRE I – INTRODUCTION 1

1.1 CONTEXTE DE RECHERCHE 1

1.2 INFORMATION ET SANTÉ 2

1.3 ENVIRONNEMENTS INTELLIGENTS 4

1.4 LE PROJET SAPA 5

1.5 PROBLÉMATIQUES DE RECHERCHE 7

1.5.1 GESTION DE MULTIPLES HABITATS INTELLIGENTS 8

1.5.2 RECONNAISSANCE D'ACTIVITÉS 10

1.6 CONTRIBUTIONS 13

1.7 MÉTHODOLOGIE 14

1.7.1 PHASE I : ÉTUDE DE LA LITTÉRATURE SCIENTIFIQUE. 14

1.7.2 PHASE II : DÉVELOPPEMENT D'UNE ARCHITECTURE. 14

1.7.3 PHASE III : TEST ET COLLECTE D'INFORMATION. 14

1.7.4 PHASE IV : PROPOSITION D'UN MODÈLE BASÉ SUR LA LOGIQUE FLOUE. 14

1.7.5 PHASE V : APPLICATION DE NOTRE MODÈLE. 15

1.8 ORGANISATION DU DOCUMENT 15

CHAPITRE II – LA MAISON INTELLIGENTE 16

2.1 INTRODUCTION 16

2.2 REPRÉSENTER L'INFORMATION 16
2.2.1 BLOB 17
2.2.2 XML 19
2.2.3 YAML 20
2.2.4 JSON 22
2.2.5 BILAN 23
2.3 SYSTÈMES ÉVÉNEMENTIELS 25
2.3.1 MQTT 25
2.3.2 RABBITMQ 28
2.3.3 KAFKA 29
2.3.4 REDIS 30
2.3.5 BILAN 31
2.4 API 31
2.4.1 SOAP 32
2.4.2 REST 34
2.4.3 GRAPHQL 36
2.4.4 BILAN 38
2.5 ARCHITECTURES 38
2.5.1 MAVHOME (2004) 39
2.5.2 GATORTECH (2008) 42
2.5.3 CASAS (2013) 45
2.5.4 OPENHAB 2.X (2017) 48
2.5.5 LIGHT NODE COMMUNICATION FRAMEWORK (2017) 50
2.5.6 WITS (2018) 53
2.5.7 SEARCH (2020) 56
2.6 UN MODÈLE EN COUCHES 57
2.6.1 LA COUCHE MATÉRIELLE 60

2.6.2 LA COUCHE SERVICE 61
2.6.3 LA COUCHE APPLICATIVE 63
2.7 CONCLUSION 63
CHAPITRE III – RECONNAISSANCE D'ACTIVITÉS 65
3.1 INTRODUCTION 65
3.2 ACTIVITÉ ET RECONNAISSANCE D'ACTIVITÉS 66
3.2.1 ACTIVITÉ 67
3.2.2 ACTIVITÉS DE LA VIE QUOTIDIENNE (AVQ) 67
3.2.3 ACTIVITÉS INSTRUMENTALES DE LA VIE QUOTIDIENNE (AIVQ) 69
3.2.4 RECONNAISSANCE D'ACTIVITÉS 70
3.3 MISE EN FORME DES DONNÉES 73
3.3.1 SÉRIE TEMPORELLE 73
3.3.2 SEGMENTATION 74
3.3.3 CARACTÉRISTIQUES DISCRIMINANTES 77
3.4 APPRENTISSAGE SUPERVISÉ 79
3.4.1 CLASSIFIEUR BAYÉSIEN NAÏF 80
3.4.2 MODÈLE DE MARKOV CACHÉ 84
3.4.3 MACHINES À VECTEURS DE SUPPORT 88
3.4.4 ARBRES DE DÉCISION 93
3.4.5 RÉSEAUX DE NEURONES 96
3.5 APPRENTISSAGE NON SUPERVISÉ 100
3.5.1 FORAGE DE SÉQUENCES 100
3.5.2 AGRÉGATION 103
3.5.3 MODÉLISATION DE SUJETS 105
3.6 ANALYSER LA QUALITÉ 107
3.7 CONCLUSION 110
CHAPITRE IV – ARCHITECTURE POUR LE SUPPORT DE MULTIPLE HABITATS INTELLIGENTS 112

4.1 LES HABITATS DU PROJET SAPA 113
4.2 ARCHITECTURE ORIENTÉE DONNÉES 114
4.2.1 HABITATS INTELLIGENTS AVEC DES AGENTS 118
4.2.2 ÉVÉNEMENTS ET AGENTS 122
4.3 IMPLÉMENTATION DE L'ARCHITECTURE 125
4.3.1 IMPLÉMENTATION DES AGENTS 127
4.4 VALIDATION DE L'ARCHITECTURE 129
4.4.1 DÉVELOPPEMENT ORIENTÉ COMPORTEMENTS 130
4.4.2 COUVERTURE DU CODE AVEC JACOCO 132
4.4.3 ÉVALUATION EMPIRIQUE 133
4.5 CONCLUSION 138
CHAPITRE V – CARACTÉRISTIQUES FLOUES POUR LA GÉNÉRALISATION ET LA QUALIFICATION DES INSTANCES D'ACTIVITÉ 151
5.1 ENSEMBLES DE DONNÉES 153
5.1.1 COLLECTE DE DONNÉES 155
5.1.2 TRAITEMENT DES DONNÉES 157
5.2 LOGIQUE FLOUE 160
5.2.1 APPLICATION DANS LA RECONNAISSANCE D'ACTIVITÉS 162
5.2.2 ENSEMBLES FLOUS SPÉCIFIQUES À NOTRE CONTEXTE 164
5.2.3 FONCTIONS D'APPARTENANCE 167
5.2.4 SCORE DE QUALITÉ 172
5.2.5 RÉSUMÉ DES ENSEMBLES DE DONNÉES GÉNÉRÉS 173
5.3 APPLICATION À L'ENSEMBLE CASAS 173
5.4 CONCLUSION 180
CHAPITRE VI – TRANSFÉRABILITÉ DE NOTRE MÉTHODE AU PROJET SAPA 182
6.1 EXPLORATION NON SUPERVISÉE DES DONNÉES 183

6.1.1 MESURES D'ÉVALUATION POUR L'APPRENTISSAGE NON SUPERVISÉ . . . 183
6.1.2 ALGORITHMES POUR L'EXPLORATION . . . 190
6.1.3 EXPLORATION DES DONNÉES BRUTES DE SAPA . . . 192
6.1.4 EXPLORATION DES ENSEMBLES AVEC CARACTÉRISTIQUES FLOUES . . . 200
6.1.5 CLUSTERING PAR QUALITÉ . . . 203
6.1.6 DISCUSSION . . . 206
6.2 CLASSIFICATION . . . 209
6.2.1 FILTRAGE EN TERMES DE QUALITÉ . . . 212
6.3 CONCLUSION . . . 213
CHAPITRE VII – CONCLUSION GÉNÉRALE . . . 224
7.1 RÉALISATION DES OBJECTIFS . . . 226
7.2 ÉVALUATION DES TRAVAUX . . . 229
7.3 TRAVAUX FUTURS . . . 234
7.4 APPORTS PERSONNELS . . . 236
BIBLIOGRAPHIE . . . 238

LISTE DES TABLEAUX

TABLEAU 2.1 : TABLEAU COMPARATIF DES ARCHITECTURES DE NOTRE REVUE DE LITTÉRATURE. 58

TABLEAU 4.1 : SCÉNARIOS POSSIBLES LORS DE LA RÉCEPTION D'UN ÉVÉNEMENT INDIQUANT L'INSERTION D'UNE NOUVELLE VALEUR BOOLÉENNE DANS UN FLUX DE VALEURS À TRANSFORMER EN FLUX D'ÉTIQUETTES. 133

TABLEAU 4.2 : STATISTIQUES CONCERNANT CHAQUE INSTALLATION TRAQUÉE ENTRE FÉVRIER ET JUIN 2019.. 135

TABLEAU 5.1 : EXEMPLES D'ENTRÉES D'UN FICHIER COMPRENANT LA LISTE DES ÉVÉNEMENTS ÉMIS PAR LES CAPTEURS D'UNE INSTALLATION. 157

TABLEAU 5.2 : EXEMPLES D'ENTRÉES D'UN FICHIER COMPRENANT LA LISTE DES ÉTIQUETTES ÉMISES PAR LES ALGORITHMES D'UNE INSTALLATION. 158

TABLEAU 5.3 : PROPRIÉTÉS EXTRAITES POUR CHAQUE FENÊTRE GLISSANTE SUR LES OBSERVATIONS. 159

TABLEAU 5.4 : ENSEMBLES FLOUS PAR CATÉGORIES. 164

TABLEAU 5.5 : CONFIGURATION DES ENSEMBLES FLOUS REPRÉSENTANT LES DURÉES. 168

TABLEAU 5.6 : CONFIGURATION DES ENSEMBLES FLOUS REPRÉSENTANT L'ACTIVITÉ DES CAPTEURS. 169

TABLEAU 5.7 : CONFIGURATION DES ENSEMBLES FLOUS REPRÉSENTANT LES VARIÉTÉS DE CAPTEURS. 170

TABLEAU 5.8 : JEUX DE DONNÉS GÉNÉRÉS. 175

TABLEAU 5.9 : RÉSULTATS OBTENUS SUR ARUBA VIA NOTRE MÉTHODE.. . 176

TABLEAU 5.10 : EXEMPLES D'ACTIVITÉS *CUISINER* AVEC LEUR SCORE DE QUALITÉ RESPECTIFS. 178

TABLEAU 6.1 : MOYENNES DES MESURES D'ÉVALUATION PAR ALGORITHME. 193

TABLEAU 6.2 : DIFFÉRENCES ENTRE LES MOYENNES DES MESURES D'ÉVALUATION PAR ALGORITHME OBTENUES AVEC LES PROPRIÉTÉS FLOUES, ET LES MOYENNES OBTENUES SANS LES PROPRIÉTÉS FLOUES. 200

TABLEAU 6.3 : DIFFÉRENCES ENTRE LES MOYENNES DES MESURES D'ÉVALUATION APRÈS FILTRAGE DANS LE CAS OÙ UNE SEULE CLASSE PEU FIABLE EST UTILISÉE ET LA MOYENNE DES MESURES OBTENUES AVEC LES PROPRIÉTÉS FLOUES. 205

TABLEAU 6.4 : DIFFÉRENCES ENTRE LES MOYENNES DES MESURES D'ÉVALUATION APRÈS FILTRAGE DANS LE CAS OÙ PLUSIEURS CLASSES PEU FIABLES SONT UTILISÉES ET LA MOYENNE DES MESURES OBTENUES AVEC LES PROPRIÉTÉS FLOUES. . 207

TABLEAU 6.5 : MEILLEURES MOYENNES DE LA JUSTESSE ET DU SCORE F1 APRÈS UNE VALIDATION CROISÉE À 10 PLIS D'UNE APPLICATION DES ALGORITHMES C4.5 (J48), FORÊT ALÉATOIRE, K PLUS PROCHES VOISINS, ET BAYES NAÏF SUR TOUS LES JEUX DE DONNÉES DISPONIBLES VIA WEKA 3.8.5. LES VALEURS SOULIGNÉES SONT LES SCORES MAXIMUMS DE CHAQUE CATÉGORIE. 210

TABLEAU 6.6 : DIFFÉRENCES DES RÉSULTATS OBTENUS APRÈS FILTRAGE. 212

LISTE DES FIGURES

FIGURE 1.1 – VUE D'ENSEMBLE DE LA PROBLÉMATIQUE DANS LAQUELLE S'INSÈRE CETTE THÈSE. EN HAUT DE L'IMAGE, ON PEUT VOIR LES DIFFÉRENTES ACTIONS À RÉALISER POUR LA RÉSOUDRE ET, EN BAS DE L'IMAGE, LES OBJETS ET ACTEURS DU PROJET. LA PARTIE REPRÉSENTÉE EN ROUGE EST LA LIMITE DE CE QUE CETTE THÈSE DOIT RÉSOUDRE D'UN POINT DE VUE ARCHITECTURAL. 10

FIGURE 2.1 – UN EXEMPLE DE CONTENU XML. 20

FIGURE 2.2 – UN EXEMPLE DE CONTENU YAML. 22

FIGURE 2.3 – UN EXEMPLE DE CONTENU JSON. 24

FIGURE 2.4 – SCHÉMA REPRÉSENTANT LES DIFFÉRENTES COUCHES DE L'ARCHITECTURE MAVHOME. 41

FIGURE 2.5 – SCHÉMA REPRÉSENTANT UN AGENT INTELLIGENT SELON STUART RUSSELL.. 42

FIGURE 2.6 – REPRÉSENTATION DES COUCHES DE L'ARCHITECTURE GATORTECH. 44

FIGURE 2.7 – REPRÉSENTATION DE L'ARCHITECTURE CASAS.. 47

FIGURE 2.8 – REPRÉSENTATION DE L'ARCHITECTURE OPENHAB. 51

FIGURE 2.9 – REPRÉSENTATION DE L'ARCHITECTURE WITS. 55

FIGURE 3.1 – SCHÉMA REPRÉSENTANT UNE SÉLECTION DE NŒUDS D'UN ARBRE DE DÉCISION. 94

FIGURE 4.1 – PLAN REPRÉSENTANT TROIS INSTALLATIONS DU PROJET SAPA, LES ZONES BLEUES REPRÉSENTENT LA ZONE D'ACTIVATION DES CAPTEURS INFRAROUGES. 142

FIGURE 4.2 – SCHÉMA DE L'INTERGICIEL. 143

FIGURE 4.3 – TIRER OU POUSSER DE L'INFORMATION. 143

FIGURE 4.4 – SCHÉMA DÉTAILLÉ DE L'INTERGICIEL. 144

FIGURE 4.5 – SCHÉMA D'UN ENVIRONNEMENT ORIENTÉ AGENT. 145

FIGURE 4.6 – SCHÉMA REPRÉSENTANT UN AGENT. 146

FIGURE 4.7 – SCHÉMA REPRÉSENTANT DES GRAPPES D'AGENTS EN RÉSEAU. 146

FIGURE 4.8 – SCHÉMA REPRÉSENTANT LE BUS D'ÉVÉNEMENTS DE LA PLATEFORME IMPLÉMENTÉE. 147

FIGURE 4.9 – EXEMPLE DE REQUÊTE À L'API PERMETTANT D'EXTRAIRE LES DIFFÉRENTES ÉTIQUETTES INDIQUANT L'UTILISATION D'UN RÉFRIGÉRATEUR. 147

FIGURE 4.10 – EXEMPLE DE REQUÊTE À L'API PERMETTANT D'EXTRAIRE LA DURÉE TOTALE D'UTILISATION DES SALLES DE BAIN D'UNE MAISON. 147

FIGURE 4.11 – SCHÉMA REPRÉSENTANT UN AGENT TRADUISANT UN FLUX DE VALEURS BOOLÉENNES EN FLUX D'ÉTIQUETTES. 148

FIGURE 4.12 – SCHÉMA REPRÉSENTANT UN AGENT TRADUISANT UN FLUX DE VALEURS RÉELLES EN FLUX DE BOOLÉENS. 148

FIGURE 4.13 – SCHÉMA REPRÉSENTANT UN AGENT TRADUISANT DES FLUX DE VALEURS BOOLÉENNES EN FLUX DE BOOLÉENS DÉCRIVANT LES PRÉSENCES DANS UNE PIÈCE. 148

FIGURE 4.14 – NOMBRE D'ÉVÉNEMENTS PUBLIÉS PAR LES ÉQUIPEMENTS DE CHAQUE ENVIRONNEMENT PAR JOUR ENTRE LE 5 FÉVRIER 2019 UTC (INCLUS) ET LE PREMIER JUIN 2019 UTC (EXCLUS) POUR CHAQUE ENVIRONNEMENT SUIVIT. 149

FIGURE 4.15 – NOMBRE D'ÉVÉNEMENTS GÉNÉRÉS PAR AGENTS DE CHAQUE ENVIRONNEMENT PAR JOUR ENTRE LE 5 FÉVRIER 2019 UTC (INCLUS) ET LE PREMIER JUIN 2019 UTC (EXCLUS) POUR CHAQUE ENVIRONNEMENT SUIVIT. 149

FIGURE 4.16 – NOMBRE D'ÉVÉNEMENTS IMPORTÉS EN FONCTION DE LA DURÉE NÉCESSAIRE POUR QU'ILS SOIENT ENREGISTRÉS EN INCLUANT LE PROCESSUS DE GÉNÉRATION DES INFORMATIONS DE PLUS HAUT NIVEAU. 150

FIGURE 5.1 – FONCTIONS D'APPARTENANCE DÉCRIVANT LA RELATION ENTRE DIFFÉRENTS MOMENTS DE LA JOURNÉE REPRÉSENTÉS EN ABSCISSE ET DIFFÉRENTS DEGRÉS D'APPARTENANCE AUX ENSEMBLES FLOUS PRÉCÉDEMMENT DÉFINIS REPRÉSENTÉS EN ORDONNÉE.. 168

FIGURE 5.2 – FONCTIONS D'APPARTENANCE À CHAQUE ENSEMBLE FLOU REPRÉSENTANT UNE DURÉE INDIQUANT LE DEGRÉ D'APPARTENANCE REPRÉSENTÉ EN ORDONNÉE EN FONCTION DE LA DURÉE MESURÉE, REPRÉSENTÉE EN ABSCISSE.. 169

FIGURE 5.3 – FONCTIONS D'APPARTENANCE DE CHAQUE ENSEMBLE FLOU REPRÉSENTANT L'ACTIVITÉ DES CAPTEURS AVEC LE DEGRÉ D'APPARTENANCE EN ORDONNÉE EN FONCTION DU NOMBRE D'ÉVÉNEMENTS REÇUS TOUTES LES 5 SECONDES REPRÉSENTÉ EN ABSCISSE. 170

FIGURE 5.4 – FONCTION D'APPARTENANCE DE CHAQUE ENSEMBLE FLOU REPRÉSENTANT UNE VARIÉTÉ DE CAPTEUR AVEC LE DEGRÉ D'APPARTENANCE EN ORDONNÉE EN FONCTION DU NOMBRE DE TYPES DE CAPTEURS DIFFÉRENTS AYANT ÉMIS, EN ABSCISSE.. 171

FIGURE 5.5 – VUE D'ENSEMBLE DU PROCESSUS DE GÉNÉRATION DES DONNÉES. 174

FIGURE 6.1 – VISUALISATION VIA UMAP DES RÉSULTATS OBTENUS VIA L'ALGORITHME DES K-MOYENNES SUR LE JEU DE DONNÉES DE L'ENVIRONNEMENT NUMÉRO 1.. 195

FIGURE 6.2 – VISUALISATION VIA UMAP DES RÉSULTATS OBTENUS VIA L'ALGORITHME DES K-MOYENNES SUR LE JEU DE DONNÉES DE L'ENVIRONNEMENT NUMÉRO 3.. 196

FIGURE 6.3 – VISUALISATION PAIR À PAIR DES INSTANCES DE L'ENVIRONNEMENT NUMÉRO 1. CHAQUE COLONNE ET CHAQUE LIGNE CORRESPONDENT À UNE PROPRIÉTÉ PARTICULIÈRE DE CHAQUE INSTANCE. CHAQUE CELLULE DE LA GRILLE EST UNE REPRÉSENTATION EN DEUX DIMENSIONS DE LA DISTRIBUTION DES INSTANCES EN FONCTION DES PROPRIÉTÉS DE SA COLONNE ET DE SA LIGNE.. 197

FIGURE 6.4 – VISUALISATION PAIR À PAIR DES INSTANCES DE L'ENVIRONNEMENT 1. CHAQUE COLONNE ET CHAQUE LIGNE CORRESPONDENT À UNE PROPRIÉTÉ PARTICULIÈRE DE CHAQUE INSTANCE. CHAQUE CELLULE DE LA GRILLE EST UNE REPRÉSENTATION EN DEUX DIMENSIONS DE LA DISTRIBUTION DES INSTANCES EN FONCTION DES PROPRIÉTÉS DE SA COLONNE ET DE SA LIGNE. LES INSTANCES DE L'ACTIVITÉ *AUTRE* ONT ÉTÉ ÉLIMINÉES DE CETTE REPRÉSENTATION. . . . 216

FIGURE 6.5 – VISUALISATION PAIR À PAIR DES INSTANCES DE L'ENVIRONNEMENT 17. CHAQUE COLONNE ET CHAQUE LIGNE CORRESPONDENT À UNE PROPRIÉTÉ PARTICULIÈRE DE CHAQUE INSTANCE. CHAQUE CELLULE DE LA GRILLE EST UNE REPRÉSENTATION EN DEUX DIMENSIONS DE LA DISTRIBUTION DES INSTANCES EN FONCTION DES PROPRIÉTÉS DE SA COLONNE ET DE SA LIGNE. LES INSTANCES DE L'ACTIVITÉ *AUTRE* ONT ÉTÉ ÉLIMINÉES DE CETTE REPRÉSENTATION. . . . 217

FIGURE 6.6 – VISUALISATION PAIR À PAIR DES INSTANCES DE L'ENVIRONNEMENT 23. CHAQUE COLONNE ET CHAQUE LIGNE CORRESPONDENT À UNE PROPRIÉTÉ PARTICULIÈRE DE CHAQUE INSTANCE. CHAQUE CELLULE DE LA GRILLE EST UNE REPRÉSENTATION EN DEUX DIMENSIONS DE LA DISTRIBUTION DES INSTANCES EN FONCTION DES PROPRIÉTÉS DE SA COLONNE ET DE SA LIGNE. LES INSTANCES DE L'ACTIVITÉ *AUTRE* ONT ÉTÉ ÉLIMINÉES DE CETTE REPRÉSENTATION. . . . 218

FIGURE 6.7 – VISUALISATION VIA UMAP DES RÉSULTATS OBTENUS VIA L'ALGORITHME DES K-MOYENNES SUR LE JEU DE DONNÉES DE L'ENVIRONNEMENT 1 COMPRENANT LES DEGRÉS D'APPARTENANCE À NOS ENSEMBLES FLOUS. 219

FIGURE 6.8 – VISUALISATION VIA UMAP DES RÉSULTATS OBTENUS VIA L'ALGORITHME DES K-MOYENNES SUR LE JEU DE DONNÉES DE L'ENVIRONNEMENT 3 COMPRENANT LES DEGRÉS D'APPARTENANCE À NOS ENSEMBLES FLOUS. 220

FIGURE 6.9 – VISUALISATION PAIR À PAIR DES INSTANCES DE L'ENVIRONNEMENT 17. CHAQUE COLONNE ET CHAQUE LIGNE CORRESPONDENT À UNE PROPRIÉTÉ PARTICULIÈRE DE CHAQUE INSTANCE. CHAQUE CELLULE DE LA GRILLE EST UNE REPRÉSENTATION EN DEUX DIMENSIONS DE LA DISTRIBUTION DES INSTANCES EN FONCTION DES PROPRIÉTÉS DE SA COLONNE ET DE SA LIGNE. LES INSTANCES DE L'ACTIVITÉ *AUTRE* ONT ÉTÉ ÉLIMINÉES DE CETTE REPRÉSENTATION. . . . 221

FIGURE 6.10 – SCORES MOYENS PAR NOMBRE DE CENTROÏDE OPTIMISÉ VIA L'ALGORITHME DES K-MOYENNES. 222

FIGURE 6.11 – ÉVOLUTION DE LA JUSTESSE ET DU F-SCORE MOYEN EN FONCTION DE LA TAILLE DU FENÊTRAGE ADOPTÉ POUR CHAQUE JEUX DE DONNÉES FLOUTÉ. 223

LISTE DES ABRÉVIATIONS

ACID	*Atomicity, Consistency, Isolation, Durability*
ADL	*Activity of Daily Living*
AIVQ	Activités Instrumentales de la Vie Quotidienne
AMQP	*Advanced Message Queuing Protocol*
API	*Application Programming Interface*
AVQ	Activités de la Vie Quotidienne
BLE	*Bluetooth Low Energy*
BLOB	*Binary Large OBject*
CART	*Classification And Regression Tree*
CASAS	*Center of Advanced Studies in Adaptative System*
CIUSSS	Centre Intégré Universitaire de Santé et de Services Sociaux
CNN	*Convolutional Neural Network*
CoAP	*Constrained Application Protocol*
CRUD	*Create Read Update Delete*
CSSS	Centre de Santé et de Services Sociaux
EBML	*Extensible Binary Markup Language*
FFT	*Fast Fourier Transform*
GIF	*Graphics Interchange Format*
GraphQL	*Graph Query Language*
HEC	Hautes Études Commerciales
HMM	*Hidden Markov Model*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IADL	*Instrumental Activity of Daily Living*
IDC	*International Data Corporation*
ID3	*Iterative Dichotomiser 3*
IETF	*Internet Engineering Task Force*
IHM	Interface Homme-Machine
IdO	Internet des Objets
IP	*Internet Protocol*
JPEG	*Joint Photographic Experts Group*

JSON *JavaScript Object Notation*
LDM Longueur de Description Minimale
LIARA Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activité
LSTM *Long Short Term Memory*
MavHome *Managing and Adaptive Versatile Home*
MIME *Multipurpose Internet Mail Extensions*
MIT *Massachusetts Institute of Technology*
MQTT *Message Queuing Telemetry Transport*
MLP *MultiLayered Perceptron*
OASIS *Organization for the Advancement of Structured Information Standards*
ONU Organisation des Nations Unies
OpenHAB *Open Home Automation Bus*
OSGi *Open Services Gateway initiative*
PNG *Portable Network Graphic*
QoS *Quality of Service*
RabbitMQ *Rabbit Message Queue*
REDIS *REmote DIctionary Server*
REST *REpresentational State Transfer*
RFC *Request For Comment*
RFID *Radio Frequency IDentification*
RNN *Recurrent Neural Network*
SAPA Soutien à l'Autonomie des Personnes Âgées
SAX *Simple API for XML*
SEArch *Smart Environment ARCHitecture*
SGML *Standard Generalized Markup Language*
SOAP *Simple Object Access Protocol*
SQL *Structured Query Langage*
STOMP *Simple Text Oriented Messaging Protocol*
SVM *Support Vector Machine*
TCP *Transmission Control Protocol*
UDP *User Datagram Protocol*
UMAP *Uniform Manifold Approximation and Projection for Dimension Reduction*
UQAC Université du Québec à Chicoutimi

URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
WITS	*Web-based Internet of Things Smart home*
W3C	*World Wide Web Consortium*
XML	*Extensible Markup Language*
XSD	*XML Schema Definition*
XSLT	*eXtensible Stylesheet Language Transformations*
YAML	*YAML Ain't Markup Language ou Yet Another Markup Language*

DÉDICACE

Je dédicace cette thèse à mon grand-père, qui m'a poussé juste ce qu'il faut pour que je dévale par moi-même la pente de la vie,

Je dédicace ma thèse à ma famille, qui a financé mon long et fructueux voyage en m'offrant l'environnement merveilleux dans lequel j'ai grandi,

Je dédicace ma thèse à mes amis, qui je dois bien l'avouer, ont toujours répondu présent pour égailler le chemin donnant jusqu'ici,

Et je dédicace finalement ma thèse à mes ennuis, sources d'infinies péripéties.

REMERCIEMENTS

Cette thèse, fruit de quatre années de travail, n'aurait pu voir le jour sans le soutien moral et les conseils de nombreuses personnes à qui je souhaiterais dire merci du fond du cœur.

S'il n'est pas possible de remercier tout le monde ici, je souhaiterais tout de même mentionner mes directeurs de thèse, Kévin Bouchard, Sébastien Gaboury et Bruno Bouchard, qui ont eu la patience de m'apporter le suivi nécessaire, même dans les moments les plus difficiles.

Je souhaiterais aussi remercier mes nombreux collègues, Benjamin Vigneau, Camille Bertuglia, Charles Cousyn, Edmond Lachance, Florentin Thullier, Geoffrey Glangine, Julien Maître, Kévin Chapron, Killian Lachaux, Patrick Lapointe, Rémi Clairet et Yannick Francillette qui ont peuplé cette aventure de nombreux visages familiers.

Je souhaiterais évidemment remercier des amis de longue date, Aymerick Rouxelin, Antonin Lebrard, Alexandre K\ourio, Mathieu Fichez, Nicolas Blazin, Pierre Blazin et Romain Ernandez qui ont tous à leur manière contribué à ma personnalité actuelle au travers de ma traversée scolaire, et à qui je dois un peu de mon arrivée en thèse.

Il me semble important de mentionner aussi ma famille, et plus particulièrement Éric Demongivert, Jean-Pierre Anthoine, Marc Demongivert, Micheline Anthoine, Nathalie Anthoine, Nicole Demongivert, Sonia Demongivert et Romain Demongivert qui ont su m'apporter en tout temps leur soutient et leur confiance, et sans qui, rien de tout cela n'eut été possible.

Je souhaiterais remercier plus spécialement Jacques Chambre, qui a su m'apprendre au bon moment à avoir confiance en moi et à me débrouiller tout seul, et finalement Sergio Dosantos, sans qui l'idée de me lancer dans un doctorat ne me serait jamais venu.

Finalement, je souhaiterais remercier les nombreuses personnes que je n'ai pas mentionnées ici, mais que je n'ai pas oubliées pour autant, et qui ont marqué de leur empreinte ces presque 30 dernières années de vie.

CHAPITRE I
INTRODUCTION

1.1 CONTEXTE DE RECHERCHE

Le vieillissement global de la population est une tendance démographique largement documentée par l'Organisation des Nations Unies (ONU) notamment grâce à des études régulières menées à l'échelle mondiale par son Département de l'Économie et des Affaires Sociales. Ces rapports, comme ceux de 2013, 2015 et 2017, permettent de se faire une idée précise de la tendence et des méthodes de mesure sur la dernière décennie. Selon l'Organisation des Nations Unies (2019), il y aurait aujourd'hui 703 millions de personnes âgées de 65 ans et plus dans le monde et il est estimé que ce nombre devrait doubler pour atteindre un total de 1,5 milliards d'individus d'ici 2050.

Une telle progression n'est pas sans conséquence, notamment pour la santé publique où elle est intimement liée au développement des maladies neurodégénératives, c'est-à-dire à des situations médicales caractérisées par une dégradation progressive du système nerveux. En effet, selon le rapport de l'Alzheimer's Disease International (2015), le risque de déclarer une maladie neurodégénérative augmenterait exponentiellement avec l'âge et doublerait tous les 6,3 années. Selon ce même rapport, 46,8 millions de personnes souffraient d'une maladie neurodégénérative en 2015 et les projections estiment que ce nombre doublera tous les 20 ans pour atteindre un total de 131,5 millions de personnes d'ici 2050 avec, en moyenne, 9,9 millions de nouveaux cas déclarés chaque année.

Au-delà d'une corrélation entre un phénomène démographique et un phénomène de santé publique, c'est bien d'espérance et de qualité de vie des populations dont il est question. Les maladies dégénératives impactent les malades, mais elles entraînent aussi une pression

conséquente sur leurs proches et sur les aides-soignants chargés de veiller sur eux. L'altération du comportement et la perte progressive d'autonomie des individus atteints, requièrent alors la mise en place d'une surveillance et d'un soutien constant. Une situation qui implique des besoins en ressources diverses, qu'elles soient matérielles ou humaines.

Tous ces points font, des maladies neurodégénératives, un vecteur de pression économique grandissant sur les différents états du monde. Toujours selon l'Alzheimer's Disease International (2015), les maladies neurodégénératives auront coûté à tous les pays du globe un total de 604 milliards de dollars en 2010 et ce nombre s'est élevé à 818 milliards de dollars en 2015. En reprenant la publication de l'Organisation des Nations Unies (2019), il est possible de constater que cette pression financière impacte directement les ménages qui doivent assumer seuls, ou, via un système redistributif, l'accès aux ressources nécessaires à la garantie de leur qualité de vie.

Les conséquences du vieillissement ne sont cependant pas limitées à celles sur la santé, et, comme décrit dans le livre de Schwarz *et al.* (2014), l'évolution de la démographie entraîne aussi une inversion de la pyramide des âges dans certains pays induisant une réduction du nombre d'actifs par habitant. Une pression financière grandissante corrélée à une réduction des moyens de production définissent donc la base des défis à relever pour financer et garantir la qualité de vie de la population mondiale.

1.2 INFORMATION ET SANTÉ

L'exploitation grandissante d'internet ainsi que les progrès technologiques des vingt dernières années, ont ensemble, ouvert des opportunités permettant d'extraire toujours plus d'informations du monde qui nous entoure. D'ailleurs selon un rapport de l'*International Data Corporation* (IDC) rédigé par Gantz & Reinsel (2011), le monde aurait produit en 2011

une quantité de données supérieure à 1,8 zettabits, soit plus de 1,8 mille milliards de gigabits d'information. Et, bien que toutes les données acquises soient loin d'être pertinentes, des méthodes montrent tout le potentiel de ces ressources et plus particulièrement des informations qui en sont dérivées. Par exemple, comme décrit dans l'article de Mylonas *et al.* (2013), les données générées par les téléphones portables peuvent servir de preuves pour résoudre des affaires criminelles, en repérant notamment la position d'un individu à l'aide d'éléments aussi abstraits que les informations d'accès de l'appareil à internet. De plus, dans l'article de Wang (2016), les données extraites de capteurs portés à même le corps permettent de déduire les activités réalisées par un individu et donc d'évaluer son activité quotidienne en tirant parti des informations physiques associées à certains éléments du corps, telle que leur vitesse en fonction du temps. Finalement, dans l'article de Chen *et al.* (2017), des informations médicales provenant d'hôpitaux permettent de prévoir les problèmes de santé qu'un patient est susceptible de déclarer dans le futur grâce à des méthodes basées sur l'apprentissage machine.

Ces exemples, parmi tant d'autres, démontrent l'intérêt que peut avoir la connaissance dérivée de données variées pour les entreprises, les particuliers et la société publique. Il est donc parfaitement légitime d'envisager les applications possibles de ce même procédé dans le milieu de la santé et d'imaginer qu'ils puissent apporter une partie des solutions permettant de relever les défis à venir. Ainsi, les progrès des méthodes permettant de faire de la reconnaissance d'images, détaillés dans l'étude de Russakovsky *et al.* (2014), ont ouvert la voie à des solutions utiles pour réaliser des diagnostics automatiques de divers problèmes de santé, comme pour le dépistage automatique du cancer du sein, tel que présenté dans l'étude de Karabatak (2015). L'évolution des méthodes basées sur l'apprentissage machine a aussi permis la production de systèmes d'assistance robotisés pour des tâches complexes comme les opérations de chirurgie, telles que présentées dans la revue de littérature de Kassahun *et al.* (2015). Finalement, les données extraites des smartphones ou produites par ceux-ci sont aussi

utiles pour aider à monitorer des maladies comme le diabète ou pour maintenir une hygiène de vie plus saine comme dans le cadre des applications présentées en revue de littérature de l'étude de Helbostad *et al.* (2017).

L'étude de Verma & Sood (2018) explore, quant à elle, comment les données d'une maison associées à des informations extraites de registres médicaux permettent de monitorer un score de santé associé à un patient pour faciliter la prise de décision à distance de professionnels de la santé. Les possibilités offertes par les méthodes modernes d'analyse de l'information semblent donc à même de répondre à diverses problématiques posées par le vieillissement mondial, le défi étant alors de trouver des méthodes pertinentes permettant d'extraire et de stocker cette donnée.

1.3 ENVIRONNEMENTS INTELLIGENTS

En parallèle du développement des méthodes permettant d'exploiter l'information, des technologies ont aussi émergé pour mieux interagir avec le réel. C'est notamment le cas des *environnements intelligents* . Selon Cook & Das (2005) il est possible de définir un environnement intelligent comme un environnement, ou un milieu, ayant la capacité d'extraire de la connaissance et de l'exploiter afin d'améliorer l'expérience de vie de ses occupants.

Bien que l'idée de fondre les technologies de l'information dans le quotidien soit assez ancienne, comme peut le démontrer l'article de Weiser (1999), les moyens de le faire n'ont, eux, émergé que très récemment. Avec, en premier lieu, une amélioration continue des technologies embarquées, dont l'histoire de l'apparition et la démocratisation des *téléphones intelligents* est un parfait marqueur. Ces appareils, comprenant de nombreux capteurs, sont capables, seuls, ou en coopérant entre-eux, de monitorer des informations complexes comme l'état du trafic routier tel que décrit dans l'article de Mohan *et al.* (2008). Ils permettent aussi d'offrir

des solutions de conseil ou de support pour les équipes cliniques grâces à des applications comme celles décrites dans l'étude de Mosa *et al.* (2012) permettant, entre autres, le diagnostic de maladies, la collecte d'informations détaillées sur des médicaments, le calcul des scores médicaux ainsi que la fluidification et le suivi de la communication entre les équipes.

L'apparition des micro-ordinateurs, véritables ordinateurs miniaturisés très peu coûteux et à la portabilité inégalée comme l'Arduino (2005), la Rasbperry Pi (2012) ou la Latte Panda (2016) ont aussi ouvert de nouvelles voies de développement. Ces solutions ont permis la mise au point de nouveaux projets complexes par des équipes plus réduites comme la production de centres de contrôle de capteurs pour les maisons, tel que décrit dans l'article de Vujović & Maksimović (2015) ou encore la production de systèmes permettant de monitorer des environnements à distance comme décrit dans la publication de Ferdoush & Li (2014).

Finalement, le développement et la production de nouveaux capteurs et effecteurs plus simples à déployer grâce à de nouveaux systèmes de communication ont achevé de rendre les systèmes ambiants accessible à tous. On retrouve dans cette catégorie d'outils, les réseaux Zigbee, notamment utilisés dans des installations comme celles de Gill *et al.* (2009).

Tous ces progrès ont largement contribué au développement d'habitats ayant la capacité de se réguler et de s'adapter en fonction du comportement de leurs occupants : les maisons intelligentes. Ces installations sont un moyen d'extraire de la connaissance du quotidien et de l'exploiter pour augmenter la sécurité et le confort des individus en les assistant dans leurs activités quotidiennes.

1.4 LE PROJET SAPA

Dans ce contexte de démocratisation des maisons intelligentes, Soutien à l'Autonomie des Personnes Âgées (SAPA), est un projet pluridisciplinaire issu d'une demande associée

au programme de soutien à domicile du Centre Intégré Universitaire de Santé et de Services Sociaux (CIUSSS) de Centre-Sud-de-l'Île-de-Montréal et mené par divers acteurs au Québec incluant à ce jour l'Université de Sherbrooke, l'Université de Montréal, l'Université McGill, Université du Québec à Chicoutimi (UQAC), le Centre de Santé et de Services Sociaux (CSSS) de Cavendish et l'école des Hautes Études Commerciales (HEC) Montréal. Les objectifs de ce projet sont de répondre à des problématiques de santé liées au vieillissement de la population, notamment en développant un assistant cognitif ambiant personnalisable basé sur les environnements intelligents, en le déployant en conditions réelles et en évaluant son efficacité.

Dans les faits, un tel assistant allégerait la charge qui pèse sur les professionnels de la santé en prolongeant l'autonomie des personnes atteintes de problèmes cognitifs handicapants. Par exemple, l'environnement pourrait sécuriser des espaces tels que la cuisine en éteignant automatiquement des plaques de cuisson laissées actives. Une telle installation pourrait aussi aider à la planification, en organisant la semaine de son occupant et en lui rappelant quand il doit prendre ses médicaments durant la journée. Finalement, on pourrait imaginer d'automatiser certaines tâches, comme commander automatiquement les courses en fonction des préférences observées de la personne suivie.

Les informations complémentaires extraites par un tel outil permettraient aux aidants de mieux comprendre les personnes qu'ils suivent, en leur offrant un accès à des rapports sur l'état de santé des occupants de chaque maison comme la qualité de leur sommeil ou leur capacité à réaliser certaines tâches, ceci en offrant plus d'informations que celles que la personne ou ses proches peuvent communiquer. Finalement, une telle solution permettrait aux proches de suivre l'état de santé de leurs aînés à distance et d'ainsi mieux organiser le soutien qu'ils peuvent leur apporter.

Du fait de son périmètre, le projet SAPA est donc complexe et va nous amener à concevoir et déployer une solution inédite pour unifier la gestion de plusieurs environnements hétérogènes. En effet, SAPA doit gérer des environnements variant en terme d'équipement et d'agencement afin d'inclure le maximum de participants possible. Cette situation nous obligera aussi à considérer une solution de transfert de connaissances, élément indispensable pour partager l'expérience accumulée entre chaque environnement afin de bonifier les résultats.

Il sera aussi difficile d'obtenir un étiquetage fiable de nos observations de terrain. Il est en effet impossible dans le contexte de SAPA d'avoir la vérité terrain sur les activités qui se déroulent dans les maisons, cela nous obligerait à soit demander aux personnes elles-mêmes de tenir un journal de leurs activités, soit filmé l'intérieur de chaque maison et observer l'entièreté de la vie de la personne sur une période de temps. Ce point nous amènera donc à nous pencher sur un aspect précis de l'apprentissage machine : l'apprentissage non supervisé. Ce dernier nous permettant potentiellement de répondre à l'absence d'étiquetage.

Cette thèse s'inscrit donc dans ce plus vaste ensemble et vise, plus particulièrement, à mettre en place une solution intermédiaire entre les lieux de collection de la donnée, et les lieux en ayant une utilité, tout en apportant une réponse aux problématiques de reconnaissance d'activités que nous venons d'exposer. Le travail réalisé dans cette recherche a obtenu les autorisations éthiques nécessaires à la réalisation de ce projet à l'Université de Montréal (CER VN 17-18-10).

1.5 PROBLÉMATIQUES DE RECHERCHE

Comme nous l'avons brièvement introduit précédemment, cette thèse devra répondre à la problématique visant à monitorer différents habitats intelligents hétérogènes à distance. Cette problématique amenant deux questionnements supplémentaires relatifs, d'une part,

à la manière de gérer ces habitats, et d'autre part, à l'absence d'étiquetage fiable pour la reconnaissance d'activité.

1.5.1 GESTION DE MULTIPLES HABITATS INTELLIGENTS

Le nombre de pièces et l'équipement des maisons intelligentes peuvent varier et cette variation est une complexité à gérer. Par exemple, si le nombre de pièces change d'une maison à l'autre, il faut alors formaliser cette information pour que des applications tierces puissent moduler le nombre d'onglets et d'informations à présenter. Or, si toutes les maisons se ressemblent, ce problème ne se pose pas puisque l'interface n'a pas à s'adapter d'une quelconque manière.

Une variation du nombre de pièces va aussi complexifier la nature des requêtes que l'on peut recevoir. Ainsi, si plusieurs salles de bains existent, il faudra distinguer les questions « y a-t-il quelqu'un dans une salle de bain ? » et « y a-t-il quelqu'un dans la salle de bain du premier étage ? », en comparaison de la seule question possible s'il n'y a qu'une unique salle de bain dans chaque maison : « y a-t-il quelqu'un dans la salle de bain ? ».

Aussi, une variation des capteurs entre deux maisons induit aussi une différence dans les niveaux d'informations que l'on déduira. Par exemple, une maison dont la cuisine est très bien équipée nous permettra de comprendre plus précisément ce qu'il se passe en détectant l'ouverture d'un frigo, de plusieurs placards et l'utilisation d'une plaque chauffante, alors qu'une maison moins équipée ne pourra permettre qu'une détection plus grossière basée sur un vague mouvement et l'ouverture d'un placard par exemple.

Les solutions proposées dans la littérature sont spécialisées dans l'exploitation d'une seule maison, ou de maisons ayant toutes les mêmes caractéristiques techniques. C'est notamment le cas des travaux de Cook *et al.* (2013a), Kreuzer (2017), Yao *et al.* (2018) et de

Augusto *et al.* (2020), que nous étudierons plus en détails dans le Chapitre 2. Pour ce qui est de notre solution, cette dernière devra gérer plusieurs environnements en même temps.

Afin de résoudre ce problème complexe, il est possible de le subdiviser en plusieurs tâches plus simples à réaliser. Cette liste qui, en fait, correspond à un processus de traitement de l'information encore à développer est la suivante : capturer, transporter, transformer et distribuer de l'information.

Capturer l'information, ou répondre à la question « comment extraire et stocker de l'information décrivant le réel depuis plusieurs environnements intelligents ? », c'est-à-dire mettre au point une solution logicielle capable de récupérer des données telles que la température, l'état de la consommation électrique ou encore des informations relatives au mouvement dans les pièces de maisons dont la structure et l'équipement sont hétérogènes.

Transporter l'information, ou répondre à la question « comment diffuser de l'information à travers plusieurs machines pour que des applications tierces puissent y accéder ? », c'est-à-dire développer une architecture logicielle qui puisse diffuser de l'information entre plusieurs machines différentes pour que les environnements intelligents et les applications puissent publier ou extraire de l'information auprès de machines qui pourront alors la relayer au sein d'une architecture bien plus complexe.

Transformer l'information, ou répondre à la question « comment exploiter de la donnée comme la température, la luminosité ou, le mouvement pour en extraire et stocker de l'information de plus haut niveau comme les activités réalisées dans une journée ? », c'est-à-dire permettre aux applications tierces de pouvoir accéder à de l'information difficile à produire en leur épargnant le coût de développement nécessaire pour y arriver. C'est aussi une étape importante pour que les environnements intelligents aient la capacité de diffuser le minimum d'informations nécessaires en fonction de leurs préférences tout en garantissant que le réseau

fera de son mieux pour en tirer parti.

Finalement **Distribuer** l'information, ou répondre à la question « comment exposer l'information reçue aux applications qui aideront in fine les professionnels de la santé dans leurs tâches quotidiennes ? », c'est-à-dire trouver une manière de présenter l'information ainsi produite et collectée à des consommateurs potentiels pour qu'ils puissent, via des requêtes, accéder à du contenu en temps réel ou explorer la base de connaissances déjà accumulées, le tout à un coût minimal en matière de développement.

La figure 1.1 résume alors schématiquement le problème à résoudre.

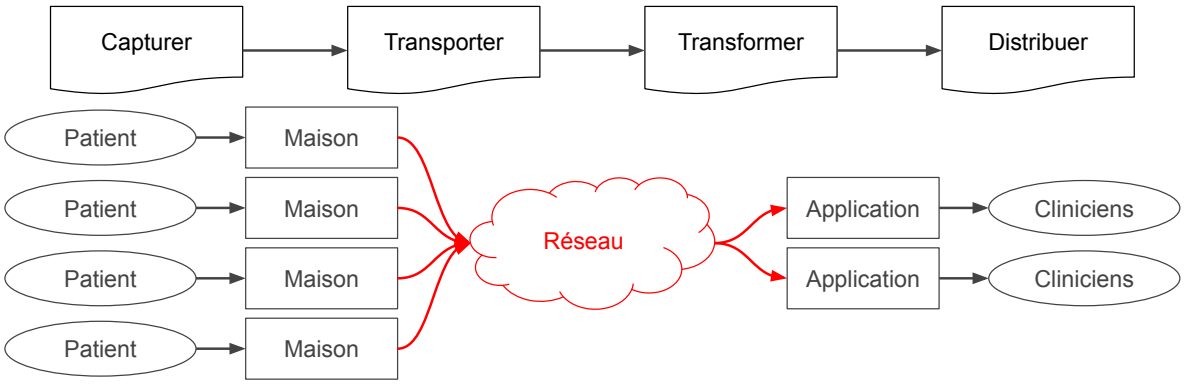

Figure 1.1 : Vue d'ensemble de la problématique dans laquelle s'insère cette thèse. En haut de l'image, on peut voir les différentes actions à réaliser pour la résoudre et, en bas de l'image, les objets et acteurs du projet. La partie représentée en rouge est la limite de ce que cette thèse doit résoudre d'un point de vue architectural.

1.5.2 RECONNAISSANCE D'ACTIVITÉS

La masse d'informations extraites d'environnements hétérogènes ne peut pas être entièrement annotée à la main, ce procédé étant bien trop demandant pour les résidents qui auraient à tenir un journal. Cependant, la plupart des modèles mathématiques permettant de reconnaître

des activités nécessitent une phase d'optimisation préalable afin de fonctionner. Et cette phase d'optimisation requiert le plus souvent d'avoir accès à un ensemble d'exemples de couples d'entrées / sorties. Aussi, la plupart des systèmes de reconnaissance ne se généralise pas d'un environnement à l'autre. En conséquence, il est assez courant de devoir optimiser de nouveau chaque algorithme pour chaque nouvel environnement suivi. Cette thèse vise donc aussi à explorer des solutions possibles permettant d'exploiter des ensembles d'informations non annotés ou annotés en partie par des règles d'inférence logique.

En parallèle de ce premier point, il existe aussi une problématique liée à l'hétérogénéité des environnements. En effet, des maisons intelligentes équipées de différents capteurs, aux agencements hétérogènes et aux profils d'occupants variés amènent à la production de données drastiquement différentes. On peut ainsi retrouver des changements structurels, c'est-à-dire qu'il peut exister, entre deux environnements, des capteurs additionnels ou des capteurs manquants. Mais il est aussi possible d'être confronté à des changements plus contextuels, comme l'absence ou la présence d'habitudes différentes d'un occupant à l'autre. Or, comme nous l'avons déjà dit, la plupart des systèmes de reconnaissance se généralisent plutôt mal, et ceci va rapidement poser problème dans le cadre de notre travail. Ainsi, dans cette thèse, il nous faudra donc aussi aborder des éléments relatifs au *transfert de connaissances*, ce point étant nécessaire pour nous permettre d'exploiter d'un seul tenant tous les jeux de données produits, plutôt que chacun en totale indépendance de tous les autres.

Finalement, la plupart des algorithmes de reconnaissance d'activités proposés dans la littérature se contentent d'annoter, c'est-à-dire d'associer des ensembles d'instants à des étiquettes indiquant que certaines activités y sont réalisées ou non, parfois de manière parfaitement compréhensible et parfois en utilisant des systèmes de classification plutôt opaques. C'est notamment le cas des algorithmes référencés dans les travaux de Ramasamy Ramamurthy & Roy (2018), Hussain *et al.* (2019) et Gu *et al.* (2021a).

Cependant, même des activités très similaires peuvent être réalisées de nombreuses manières différentes, et un individu peut ne pas réaliser toutes ses activités quotidiennes avec la même implication. Par exemple, faire réchauffer des plats déjà préparés dans un micro-onde et préparer des pâtisseries, de la pâte aux garnitures, sont deux scénarios dont l'on peut dire raisonnablement qu'ils représentent fidèlement l'activité de *faire la cuisine* . Cependant, le premier semble être une manière plus passive d'effectuer cette activité et le second semble impliquer que la personne investit une attention et une énergie bien plus conséquentes à la tâche. On peut alors parler de qualité de la réalisation d'une activité et il est possible d'imaginer bien d'autres qualificatifs, comme la complexité ou le fait qu'une activité est plutôt habituelle ou irrégulière.

Ces propriétés, qui ne sont pas réellement extraites, peuvent avoir une importance capitale pour évaluer l'évolution du comportement d'un individu. Ainsi, dans le cadre de l'exemple de nos deux activités *faire la cuisine , si une activité nécéssitant moins d'implication est de moins bonne qualité,* on pourrait imaginer qu'une baisse continue de la qualité puisse être corrélée à une aggravation de la dépendance de la personne et donc de sa maladie.

Aussi, dans des environnements hétérogènes, une activité que l'on reconnaît grâce à l'activation de nombreux capteurs, comme des capteurs électriques, de contact et de mouvement, peut être largement plus précise qu'une activité reconnue via un ensemble de capteurs bien plus réduit. Ainsi, il peut aussi être intéressant d'informer le clinicien du niveau d'équipement d'une installation pour qu'il puisse en tenir compte.

Ainsi, finalement, cette thèse proposera aussi des moyens permettant de mieux qualifier les annotations produites par des algorithmes de reconnaissance d'activités.

1.6 CONTRIBUTIONS

Cette thèse recoupe trois contributions théoriques et appliquées : un prototype de plateforme logicielle permettant à des applications tierces d'exploiter les données produites par plusieurs environnements intelligents hétérogènes ; un modèle d'estimation de la qualité basé sur la logique floue permettant d'extraire de manière unifiée de l'information malgré l'hétérogénéité de nos jeux ; et, enfin, une exploration non supervisée de jeux de données non annotés et hétérogènes.

La première de nos contributions, la plateforme, a fait l'objet d'une publication par Demongivert *et al.* (2021a). Cette contribution est plus amplement détaillée dans le Chapitre 4 de cette thèse. C'est cet outil qui nous a permis de stocker et d'explorer en temps réel différents jeux de données produits par des maisons habitées par des personnes atteintes de maladies dégénératives. La capacité de cette plateforme à gérer des milieux hétérogènes nous a permis d'extraire les nombreux jeux de données exploités par notre dernier travail.

La seconde contribution, le modèle d'estimation de la qualité basé sur la logique floue, a aussi fait l'objet d'une publication par Demongivert *et al.* (2020). Cette contribution est plus amplement détaillée dans le Chapitre 5. C'est ce modèle, basé sur les données produites par *Center of Advanced Studies in Adaptative System* (CASAS), notamment présenté dans l'article de Cook (2012), qui a pour objectif de transférer de la connaissance d'un jeu de données extérieur vers les nôtres.

Finalement, notre dernière contribution, une exploration de nos jeux de données à l'aide de notre modèle, a fait partiellement l'objet d'une publication par Demongivert *et al.* (2021b). Cette contribution est le sujet abordé au sein du Chapitre 6 et constitue une étude de l'application de notre modèle basé sur la logique floue à nos propres jeux de données.

1.7 MÉTHODOLOGIE

La méthodologie que nous avons appliquée peut être résumée en cinq grands points.

1.7.1 PHASE I : ÉTUDE DE LA LITTÉRATURE SCIENTIFIQUE.

Dans cette phase initiale, nous avons étudié la littérature afin de construire une base de connaissances nous permettant, par la suite, de conduire cette recherche dans les meilleures conditions. Ce travail préliminaire est l'objet des Chapitres 2 et 3 de cette thèse.

1.7.2 PHASE II : DÉVELOPPEMENT D'UNE ARCHITECTURE.

Dans cette seconde phase, nous avons proposé et mis en place une architecture permettant de répondre aux aspects de la problématique de cette thèse visant à capturer de l'information d'environnements intelligents hétérogènes, à transporter cette information extraite au sein d'un réseau de machines et, finalement, à distribuer cette information transportée à des applications tierces. C'est le sujet du Chapitre 4 de cette thèse.

1.7.3 PHASE III : TEST ET COLLECTE D'INFORMATION.

Dans cette troisième phase, nous avons testé notre architecture dans le cadre du projet SAPA. Nous l'avons amélioré en conséquence et nous avons profité de ce temps pour collecter nos différents jeux de données.

1.7.4 PHASE IV : PROPOSITION D'UN MODÈLE BASÉ SUR LA LOGIQUE FLOUE.

Dans cette quatrième phase, réalisée en parallèle de la phase précédente, nous avons utilisé un jeu de données de CASAS afin de produire un modèle de reconnaissance d'acti-

vités basé sur la logique floue. L'objectif étant que ce modèle puisse être transposé de cet environnement vers les nôtres. C'est le sujet du Chapitre 5 de cette thèse.

1.7.5 PHASE V : APPLICATION DE NOTRE MODÈLE.

Finalement, dans cette dernière phase de notre recherche, nous avons utilisé le modèle que nous avons proposé en phase IV pour explorer nos jeux de données produits en phase III. Cette dernière étape nous a permis d'évaluer la capacité de notre modèle à transférer de la connaissance. C'est le sujet du Chapitre 6 de cette thèse.

1.8 ORGANISATION DU DOCUMENT

Cette thèse est donc divisé en 7 Chapitres. Le premier chapitre, qui s'achève ici, est une introduction au sujet de la thèse et à son contexte. Le second chapitre traite, quant à lui, de la maison intelligente dans la littérature et constituera un travail préparatoire pour la première partie de la thèse. Le troisième chapitre traite de l'exploitation de l'information extraite d'environnements intelligents et constituera un travail préparatoire pour la seconde partie de la thèse. Ces trois premiers chapitres formeront notre revue de littérature.

La seconde partie de ce document, constituée des Chapitres 4, 5, 6 et 7 se focalisera principalement sur nos contributions. Ainsi, le Chapitre 4 traite de notre proposition d'architecture. Les Chapitres 5 et 6 présentent notre travail de reconnaissance d'activités, et le Chapitre 7 sert de conclusion à l'entièreté de ce document.

CHAPITRE II
LA MAISON INTELLIGENTE

2.1 INTRODUCTION

Nous avons défini en Section 1.3 un environnement intelligent comme un milieu ayant la capacité d'extraire de la connaissance et de l'exploiter afin d'améliorer les conditions de vie de ses occupants. Nous nous sommes aussi penchés sur le cas plus spécifique de la maison intelligente et nous avons présenté, en Section 1.5, les possibilités que pouvait offrir une telle technologie pour répondre aux problématiques du vieillissement mondial.

Dans ce chapitre, nous allons nous intéresser aux architectures des maisons intelligentes en commençant par étudier trois problématiques clefs puis en analysant différentes architectures proposées dans la littérature.

2.2 REPRÉSENTER L'INFORMATION

La communication d'information est un problème central dans le cadre de la maison intelligente. En effet, il faut pouvoir faire communiquer différents composants entre eux comme des capteurs, des effecteurs ou encore des composants logiciels. Mais il peut aussi être nécessaire de communiquer avec l'extérieur, en particulier si l'on souhaite observer la maison à distance. Ce dernier point amène rapidement à des questions relatives à la sécurité de l'information. Finalement, une maison intelligente peut avoir à stocker de l'information de toute sorte pour réaliser des calculs.

Pour manipuler et communiquer de l'information, comme dans le cadre des études de Romero *et al.* (2010), de Soliman *et al.* (2013), de Perumal *et al.* (2013) et de Bao

et al. (2016), les architectures d'environnements intelligents peuvent utiliser une ou plusieurs représentations. Dans le cadre des études présentées, on peut ainsi trouver des références à *YAML Ain't Markup Language ou Yet Another Markup Language* (YAML), *JavaScript Object Notation* (JSON), *Extensible Markup Language* (XML) ou à des formats plus bruts.

Dans le cadre de cette thèse, nous allons manipuler des données et les communiquer entre différents composants. Il semble donc indispensable de présenter les formats utilisés dans la littérature, et ce, pour pouvoir faire un choix, mais aussi, pour connaître à l'avance les formats supportés par les maisons intelligentes et les applications des cliniques.

Dans cette section, nous allons donc nous intéresser à certains formats permettant de représenter l'information que l'on peut retrouver au sein de différentes architectures de maison intelligente.

2.2.1 BLOB

Le gros objet binaire, ou le *Binary Large OBject* (BLOB) en anglais, est tout simplement une séquence continue de bits en mémoire dont la taille est fixe et définie au préalable. On peut retrouver des références à ce type de stockage dans le contexte des bases de données dans les livres de Beynon-Davies (1996) et de Paulraj (2005) ainsi que dans divers articles comme ceux de Shapiro & Miller (1999), de Nicolae *et al.* (2009) et de Conti *et al.* (2010).

Pour stocker de l'information dans un BLOB, il faut l'encoder en binaire en suivant un standard prédéfini, ce même standard servant, par la suite, à décoder l'information du-dit BLOB. Les représentations binaires sont souvent utilisées pour stocker des images, des vidéos, des musiques ou des informations propres à différents logiciels spécialisés.

Ainsi, les formats d'image bitmap, *Joint Photographic Experts Group* (JPEG), *Graphics*

Interchange Format (GIF) ou *Portable Network Graphic* (PNG), que l'on peut retrouver dans le livre de Murray & VanRyper (1996) sont des exemples de standards permettant de stocker des images au format binaire. Bien qu'il existe de nombreux standards pour stocker des informations multimédias sous la forme de BLOB, il existe aussi des standards permettant de stocker des informations plus génériques comme le format *Extensible Binary Markup Language* (EBML) décrit dans la *Request For Comment* (RFC) de Lhomme *et al.* (2020). Ce format est notamment la base du standard de stockage vidéo Matroska [1].

Utiliser des formats binaires, plutôt que des formats textes pour représenter de l'information permet de gagner de l'espace et du temps au détriment de la facilité d'accès et d'édition par des opérateurs humains. Pour ce qui est de la difficulté d'accès, il est, en effet, plus évident de modifier un nombre en base 10 comme 15 649 en utilisant des caractères et un éditeur de texte plutôt que de modifier sa représentation binaire (0b11110100100001) ou hexadécimale (0x3D21) en utilisant un éditeur spécialisé et en tenant compte des spécificités de représentation de la machine.

Pour ce qui est du gain d'espace, utiliser un format textuel est bien plus lourd : là où le nombre 15 649 tient sur un total de 14 bits (ou 2 octets), la représentation textuelle de ce nombre, en utilisant le standard Unicode et l'encodage le plus compact, c'est-à-dire UTF-8, tiendra pour sa part sur un total de 48 bits (ou 6 octets). La problématique de la taille semble négligeable, au premier abord, quand on l'applique à l'encodage d'une unique information, mais lorsque l'on souhaite transférer une quantité importante de données, comme la température mesurée dans un salon sur les 4 derniers mois, tripler la taille du message a un impact non négligeable sur la durée des requêtes d'un logiciel et peut résulter en des temps d'attentes indésirables pour l'utilisateur final.

1. Pour plus d'information se référer à https://www.matroska.org/index.html

2.2.2 XML

XML est un format texte proposé par le *World Wide Web Consortium* (W3C) [2] permettant de stocker des informations arbitraires. La première spécification acceptée de XML, faisant suite à plusieurs brouillons, a été publiée par Sperberg-McQueen *et al.* (1998). Diverses modifications mineures de cette première spécification ont amené à la dernière spécification à ce jour qui a été rédigée par Maler *et al.* (2008). XML est une dérivation du *Standard Generalized Markup Language* (SGML) définie dans la norme ISO (1986).

Les fichiers au format XML sont des fichiers textes, lisibles et éditables par des êtres humains, où l'information est stockée dans des éléments délimités par des balises. Les éléments peuvent ainsi prendre la forme d'une balise ouvrante, d'un contenu et d'une balise fermante ou la forme d'une balise dite auto-fermante. Les fichiers au format XML chiffrent donc l'information ainsi que la structure logique de celle-ci.

Les documents XML peuvent être considérés comme bien formés ou valides. Un document bien formé est un document texte qui respecte tout simplement la norme XML et un document valide est un document bien formé qui suit une structure précise prédéfinie par ses utilisateurs. Ce second standard peut être défini via des outils tiers ou directement via certaines fonctionnalités de la norme XML. *XML Schema Definition* (XSD), un format de fichier défini par Mendelsohn *et al.* (2012), est un bon exemple de solution permettant de définir le standard que doit suivre un document XML bien formé pour être considéré comme valide.

En bref, l'écosystème XML permet de stocker de l'information arbitraire ainsi que sa structure associée dans des fichiers textes. Il comprend aussi des outils pour permettre à des logiciels de comprendre et de manipuler la norme que suivent certains fichiers. Le format XML étant un format textuel, il profite des avantages et des inconvénients décrits à la fin de la

2. Voir https://www.w3.org/.

```
<event>
    <origin>127.0.0.1</origin>
    <timestamp>1610749936000</timestamp>
    <rooms>
        <room>bathroom</room>
        <room>kitchen</room>
        <room>garden</room>
    </rooms>
</event>
```

Figure 2.1 : Un exemple de contenu XML.

Section 2.2.1. La Figure 2.1 est un exemple de document XML bien formé.

2.2.3 YAML

YAML est un format texte permettant de stocker des informations arbitraires. La première spécification de YAML a été rédigée par Ben-Kiki *et al.* (2004) et la dernière spécification à ce jour est la spécification rédigée par Ben-Kiki *et al.* (2009).

YAML a pour objectif, tout comme XML, de proposer un format de stockage adapté à la lecture et à la modification par des humains, quitte à en réduire la simplicité de lecture et d'écriture par des machines. À la différence de XML, YAML se détache complètement de la norme SGML et préfère s'inspirer de différentes technologies incluant des langages comme C [3], Java [4], Perl [5], Python [6], Ruby [7], des standards réseaux tels que la RFC0822 (Courriel) [8],

3. Voir https://www.iso.org/standard/74528.html.
4. Voir https://www.java.com/.
5. Voir https://www.perl.org/.
6. Voir https://www.python.org/.
7. Voir https://www.ruby-lang.org/.
8. Voir https://datatracker.ietf.org/doc/html/rfc822.

la RFC1866 (*HyperText Markup Language* (HTML)) [9], la RFC2045 (*Multipurpose Internet Mail Extensions* (MIME)) [10], la RFC2396 (*Uniform Resource Identifier* (URI)) [11], et des technologies comme XML [12], *Simple API for XML* (SAX) [13], *Simple Object Access Protocol* (SOAP) [14], et JSON [15].

Concrètement YAML ne stocke pas l'information via un système de balisage comme XML mais via une arborescence de nœuds représentant différents types de données. Chaque type de nœud possède alors des règles d'écriture qui lui sont propres et optimisées pour que leur manipulation soit plus simple pour des opérateurs humains. Ainsi, par exemple, il est possible de définir dans un fichier YAML des nœuds décrivant des listes en utilisant directement des tirets et des sauts de ligne. Il est aussi possible d'écrire des nœuds décrivant des relations clefs/valeurs en séparant chaque entrée par un saut de ligne et chaque paire clef/valeur via le caractère « : ». Il est encore possible d'écrire des noeuds représentant directement des valeurs comme la valeur vide (null), des textes, des chaînes de caractères, des dates, des nombres ou des références à des morceaux du document.

Cette approche rend les fichiers YAML moins verbeux que des fichiers XML car l'écriture y est optimisée pour prendre peu de caractères et tous les noeuds ne requièrent pas, comme en XML, d'être ouverts puis fermés explicitement. Pour les mêmes raisons, les fichiers YAML sont plus lisibles et plus simples à rédiger que des fichiers XML mais en contrepartie, les systèmes permettant de décoder YAML sont beaucoup plus complexes, car là où tout est

9. Voir https://datatracker.ietf.org/doc/html/rfc1866.
10. Voir https://datatracker.ietf.org/doc/html/rfc2045.
11. Voir https://datatracker.ietf.org/doc/html/rfc2396.
12. Voir https://www.w3.org/XML/.
13. Voir http://www.saxproject.org/.
14. Voir https://www.w3.org/TR/soap/.
15. Voir https://www.ecma-international.org/publications-and-standards/standards/ecma-404/.

```yaml
event:
    origin: 127.0.0.1
    timestamp: 1610749936000
    rooms:
        - bathroom
        - kitchen
        - garden
```

Figure 2.2 : Un exemple de contenu YAML.

détection de trois types de balises en XML (ouvrante, fermante et auto-fermante), YAML possède une grammaire spécifique par type de données ou de structure de données encodée. Étant un format texte, YAML profite aussi des inconvénients décrits à la fin de la Section 2.2.1.

Comme dans le cadre de XML, YAML possède aussi son écosystème de technologies comprenant un système de schéma semblable à XSD et un système de requêtes similaire à XQuery ou *eXtensible Stylesheet Language Transformations* (XSLT). La Figure 2.2 est un exemple de document YAML.

2.2.4 JSON

JSON est un format texte décrit dans le standard de l'European association for standardizing information and communication systems (1999) et dérivé de la grammaire du langage JavaScript. C'est un format permettant de stocker des informations arbitraires qui a la particularité d'être standardisées par deux organismes en parallèle : par l'*Internet Engineering Task Force* (IETF) d'une part dans le document de Bray (2017) et par l'European association for standardizing information and communication systems (2017) d'autre part.

Le format JSON peut être vu comme un intermédiaire entre l'approche suivie par le standard XML et l'approche du standard YAML dans le sens où la capacité d'édition

par un être humain est importante mais pas au point de briser la rétrocompatibilité avec la grammaire JavaScript. JSON n'est pas aussi verbeux que XML pour des raisons similaires à celles décrites pour YAML en Section 2.2.3. Cependant, à la différence de YAML, le format JSON peut rencontrer des limites dans la facilité d'édition par l'absence de certaines fonctionnalités comme la capacité à référencer des éléments d'un même document ou d'une grammaire spéciale pour écrire de longs textes. Étant un format texte, JSON souffre aussi des inconvénients décrits à la fin de la Section 2.2.1.

JSON permet de représenter des structures de données composées de trois grands types de valeurs : des primitives qui peuvent être des chaînes de caractères, des nombres (qu'ils soient flottants ou non), des valeurs booléennes ou la valeur vide (null), des tableaux qui sont juste une séquence ordonnée de valeurs et des objets qui sont une succession de paires clef/valeurs.

Bien que JSON soit, à la base, pensé pour être compatible avec la technologie JavaScript, il existe aujourd'hui de nombreux portages permettant d'utiliser ce format dans de nombreuses autres technologies incluant notamment Python, Java et C++.

Comme dans le cadre de XML et YAML, JSON possède aussi son écosystème de technologies comprenant un système de schéma semblable à XSD et un système de requêtes similaire à XQuery ou eXtensible Stylesheet Language Transformations (XSLT). La Figure 2.3 est un exemple de document JSON.

2.2.5 BILAN

Nous avons présenté dans les sections précédentes quatres solutions pour représenter l'information : le BLOB, XML, YAML et JSON. Ces quatres solutions sont, ou peuvent, êtres

```
{
    "event": {
        "origin": "127.0.0.1",
        "timestamp": 1610749936000,
        "rooms": [
            "bathroom",
            "kitchen",
            "garden",
        ]

    }
}
```

Figure 2.3 : Un exemple de contenu JSON.

utilisées sous la forme d'un langage de balisage. Par exemple, dans le cas d'un BLOB , il est possible d'utiliser le standard Matroska, dans le cas de JSON ou YAML, les structures peuvent être réduite à des balises représentant des *objets ou des séquences*, finalement XML est par nature un langage de balisage.

Pour cette raison le format utilisé importe peu car avec un modèle commun de balisage, il est possible d'utiliser tous les formats en même temps, à la demande. Si l'objectif consiste principalement à réduire la taille du code transmis, le binaire reste le plus compacte, comme nous l'avons précisé en Section 2.2.1. Dans notre infrastructure, nous avons opté temporairement pour JSON ce dernier étant supporté de base par notre cadre de travail. Ainsi, même si le format est plus lourd que le binaire, le support offert par notre base de code nous permet de nous focaliser sur des points plus importants. JSON peut avoir d'autres avantages néanmoins : il est moins verbeux que XML, il permet par défaut de représenter les types des valeurs utilisées, et finalement, étant un format texte lisible par un opérateur humain, c'est une solution plus pratique pour analyser les éventuels problèmes de communication entre logiciels.

2.3 SYSTÈMES ÉVÉNEMENTIELS

Les maisons intelligentes observent leur milieu dans un cadre temps réel pouvant, par la suite, amener à un raisonnement et à des décisions. Le réseau y est aussi une problématique tout comme la coopération entre composants. Tous ces points font des systèmes événementiels une solution usitée dans le cadre des architectures de maison intelligente comme dans le cadre des études de Di Giorgio & Pimpinella (2012), de Li *et al.* (2013) et de Wan *et al.* (2015).

Parfois, il est aussi possible de voir des modules réalisant du traitement d'événements complexes, tel que décrit dans l'article de Hallé (2017), afin de pouvoir adopter un fonctionnement au mieux temps réel. C'est notamment le cas des études de Jarraya *et al.* (2016), de Mongiello *et al.* (2017) et de Shahad *et al.* (2018).

Dans le cadre de cette thèse, un système événementiel est une potentielle solution à la problématique du transport de l'information. Aussi, ce type de système permet d'échanger de l'information en temps réel, ou du moins, de manière réactive. Finalement, c'est un outil qui possède une compatibilité inhérente avec différents environnements intelligents de la littérature, notamment de par son intégration dans ceux-ci.

Dans cette section, nous allons donc nous intéresser à différents composants permettant de gérer des systèmes événementiels au sein d'une architecture de maison intelligente.

2.3.1 MQTT

Message Queuing Telemetry Transport (MQTT) est un protocole de communication de type publication / inscription permettant à différentes machines d'échanger des messages. Le protocole MQTT est standardisé par l'*Organization for the Advancement of Structured Information Standards* (OASIS) et sa dernière version à ce jour a été rédigée par le OASIS

Message Queuing Telemetry Transport (MQTT) Technical Committee (2019).

MQTT se veut être un protocole de communication simple à implémenter et peu consommateur de ressources afin de pouvoir être embarqué sur des technologies transportables ne pouvant offrir la capacité de calcul et la bande-passante d'une tour. MQTT est un protocole qui traite de la situation où plusieurs clients, c'est-à-dire des logiciels devant s'échanger des messages, souhaitent communiquer entre eux via un serveur, qui est un logiciel dont le principal rôle est de gérer les clients, réceptionner leurs messages et les retransmettre. Un serveur MQTT fonctionne sur un modèle publication / inscription, c'est-à-dire que les différents clients peuvent s'identifier comme étant intéressés à recevoir des informations concernant différents sujets (topics en anglais) en s'inscrivant auprès du serveur, puis, des clients peuvent publier des informations relatives à différents sujets, des informations qui seront ensuite retransmises par le serveur à tous les clients s'étant inscrits aux sujets du message.

Afin d'optimiser les ressources réseau disponible, le protocole MQTT propose trois niveaux de services ou *Quality of Service* (QoS) en anglais. Les niveaux de services représentent un compromis entre la ressource réseau utilisée et la garantie que l'on reçoive les messages publiés. Dans les faits, une haute qualité de service nous garantira de recevoir tous les messages dans les meilleures conditions, mais obligera le serveur à communiquer plus de messages avec ces clients, induisant une charge réseau plus lourde. Au contraire, une qualité de service basse peut entraîner la disparition de certains événements mais la quantité de messages échangés entre les serveurs et les clients est maintenue au plus bas. Les trois niveaux de service proposés par le protocole MQTT sont :

— **Au plus une fois** est le niveau de service le plus économique, il garantit de recevoir les messages au maximum une fois, mais au risque de ne pas les recevoirs tous. Ce type de qualité de service est très utile pour gérer des capteurs qui transmettent à très

haute fréquence, car dans cette situation, la perte d'une donnée n'est pas si grave, dans la mesure où une nouvelle ne devrait pas tarder à arriver. Aussi, quand on transmet beaucoup de messages, diviser la charge par deux, trois ou quatre en ignorant des communications sécurisantes peut avoir un très fort impact.

— **Au moins une fois** est un niveau de service qui garantit de recevoir tous les messages, mais au risque d'en recevoir certains plus d'une fois. Cette qualité de service est une solution intermédiaire utile dans les cas où l'on doit tout recevoir, mais où recevoir des doublons ne pose pas de problème. En bref, ici on garde les communications permettant de s'assurer que tout le monde reçoit bien une fois chaque message, mais on élimine celles qui nous garantissent qu'un message n'a pas été transmis plusieurs fois à la même personne.

— **Exactement une fois** est le niveau de service le plus fiable qui garantit de recevoir tous les messages, et chacun en un unique exemplaire. Ce type de qualité de service, coûteux en ressources, est plus adapté à des situations où recevoir des doublons posent un problème. La documentation MQTT donne l'exemple des échanges bancaires, où la répétition d'une transaction peut entraîner des paiements en double.

Finalement, MQTT est un standard de communication agnostique pour ce qui est du format des messages échangés, c'est-à-dire qu'en dehors des informations nécessaires pour que les échanges se fassent, le protocole n'impose strictement aucune règle sur le contenu des messages en eux-mêmes et ceux-ci peuvent être chiffrés avec n'importe quel standard.

MQTT n'étant qu'un standard, il faut en choisir une implémentation pour l'utiliser. Différents logiciels permettent de déployer un serveur MQTT (*Broker* en anglais) pour réaliser

des échanges tel que Mosquitto [16], ActiveMQ [17] ou encore HiveMQ [18].

2.3.2 RABBITMQ

Rabbit Message Queue (RabbitMQ) est une solution à code source libre permettant de déployer des serveurs d'échange de messages entre applications. La documentation rédigée par l'Équipe de RabbitMQ (2020) divise la solution en deux morceaux : un serveur permettant de jouer l'intermédiaire entre les clients et des librairies à intégrer aux applications souhaitant communiquer avec le serveur.

Afin de s'adapter à différents contextes d'utilisation, le serveur RabbitMQ intègre un système de modules permettant d'ajouter ou de retirer des fonctionnalités. Il est notamment possible de choisir le protocole d'échange à utiliser et RabbitMQ offre à ce jour des solutions permettant de supporter les protocoles MQTT, *Simple Text Oriented Messaging Protocol* (STOMP) ou encore le protocole originel de l'outil : *Advanced Message Queuing Protocol* (AMQP).

En dehors de l'implémentation du serveur, de la solution cliente et du système de module, RabbitMQ propose aussi différents outils complémentaires comprenant :

— Des outils d'administration permettant de *modifier ou monitorer* l'état des serveurs. Ces outils permettent notamment d'extraire des données relatives aux ressources consommées où à l'état des communications.

— Des solutions permettant de *distribuer les calculs* afin de supporter une quantité d'échanges et de clients bien plus grande qu'en utilisant une seule machine.

16. Pour plus d'information se référer à https://mosquitto.org/
17. Pour plus d'information se référer à https://activemq.apache.org/
18. Pour plus d'information se référer à https://www.hivemq.com/

— Des services permettant d'intégrer des problématiques *de sécurité* afin de limiter la communication de certains clients en fonction de leur statut.

— Des outils utiles pour *extraire de l'information* permettant de corriger les logiciels tiers utilisant le serveur d'échange de messages.

RabbitMQ est donc un logiciel dont l'objectif premier est d'offrir toute une gamme d'outils permettant de répondre à des contraintes propres aux architectures industrielles.

2.3.3 KAFKA

Apache Kafka est une solution à code source libre permettant de déployer des serveurs d'échange de messages entre applications. Comme pour RabbitMQ, la documentation rédigée par l'Équipe de Kafka (2020a) divise la solution en deux morceaux : une solution serveur, permettant de jouer l'intermédiaire entre les clients et des librairies à intégrer aux applications souhaitant communiquer avec le serveur.

Contrairement à RabbitMQ cependant, Apache Kafka ne mise pas sur la modularité mais sur une forme de spécialisation de ses composants. Ainsi, Apache Kafka n'offre pas le choix du protocole d'échange et utilise un protocole spécialisé défini par l'Équipe de Kafka (2020b) et optimisé pour ses choix d'implémentation. Ensuite, Kafka ne propose pas un système de module tel que celui de RabbitMQ mais plutôt une solution complète clés en main comprenant des outils d'administration, de sécurité et de maintenance.

Kafka fonctionne en suivant un schéma de publication / inscription avec des sujets et des niveaux de services similaires à ce que propose le protocole MQTT décrit dans la Section 2.3.1. La solution d'Apache met aussi en avant sa capacité à retenir les événements publiés au sein de ces sujets pour une durée prédéterminée afin de les retransmettre au besoin. Finalement,

Kafka est une solution prévue pour être distribuée sur des groupes de machines afin de pouvoir gérer une charge variable d'échanges.

Kafka, comme RabbitMQ, est donc une solution permettant de publier et recevoir des événements dont l'objectif premier est d'offrir tout une gamme d'outils permettant de répondre à des contraintes propres aux architectures industrielles.

2.3.4 REDIS

REmote DIctionary Server (REDIS) est une base de données clef/valeur, sans *Structured Query Langage* (SQL) et à code source libre, fonctionnant en mémoire vive. D'après la documentation rédigée par Équipe de REDIS (2020), REDIS est un outil permettant d'écrire et de lire de l'information plus rapidement qu'avec un système traditionnel de base de données utilisant la mémoire morte comme support de stockage. Cependant, comme la mémoire vive est particulièrement coûteuse, et donc limitée, REDIS ne peut stocker qu'une quantité limitée d'information. Finalement, REDIS est un outil permettant de manipuler nativement des types de données qu'une base de données standard n'offre pas. Ceci incluant des listes, des ensembles, des dictionnaires ou des flux.

D'après Nelson (2016), REDIS est un outil qui peut répondre à différentes problématiques au sein d'un système. Il peut, par exemple, être utilisé comme outil de mise en cache pour optimiser certains traitements longs en profitant notamment de certaines fonctionnalités qui lui sont propres comme le stockage de données pour une durée limitée et renouvelable, ce qui permet d'accélérer les calculs les plus couramment effectués en délaissant les autres.

REDIS peut aussi être utilisé comme outil d'échange par publication / inscription. Le protocole de communication reste très simple comparé à ceux présentés jusqu'à maintenant, mais peut profiter du fait de venir de pair avec les fonctionnalités de stockage de l'outil pour

échanger des informations complexes de manière réactive.

REDIS est donc aussi une solution permettant de publier et recevoir des événements. Bien qu'elle soit moins spécialisée que les outils présentés jusqu'ici, la simplicité d'installation et d'utilisation, ainsi que les fonctionnalités de base de données en mémoire vive, peuvent en faire un outil utile dans certains contextes précis.

2.3.5 BILAN

Nous avons présenté dans les sections précédentes quatres outils permettant d'intégrer un système évènementiel au sein d'une architecture : MQTT, RabbitMQ, Kafka et REDIS. À notre échelle, MQTT et REDIS sont les solutions les plus faciles à installer. Ces deux outils ne nécéssitent pas de passer par une étape de configuration importante, et ils ne viennent pas avec un large choix d'extension possibles.

Pour ce qui relève de notre recherche, nous n'avons pas été à l'origine du choix de technologie. Nous avons donc dû nous adapter à la sélection faite par les équipes chargées de cette partie du projet, c'est à dire MQTT.

2.4 API

Au delà du format utilisé pour échanger ou stocker l'information, développer et mettre à disposition une *Application Programming Interface* (API), dans le cadre des maisons intelligentes, peut permettre à des tiers de développer des applications pouvant profiter des données extraites par l'environnement, comme de ses capacités à altérer le réel. Par exemple, une API peut permettre à des applications tierces de notifier un habitant d'une intrusion ou lui permettre d'ouvrir tous les volets de la maison par la pression d'un seul bouton.

L'inverse est aussi vrai, une maison intelligente peut utiliser des API tierces pour extraire plus d'information ou pour réaliser des actions complexes. Par exemple, une maison intelligente peut utiliser l'API proposée par un service météo pour obtenir des informations concernant les prévisions météo à venir. Cela, sans avoir à faire elle-même des prévisions complexes basées sur les données des stations météorologiques installées spécifiquement pour la maison. De la même manière, une maison intelligente peut utiliser l'API d'un service de magasinage en ligne pour faire livrer de la nourriture ou du matériel.

Finalement, dans le cadre de cette thèse, la mise en place d'une API est une étape incontournable. Que cela soit pour interfacer des environnements intelligents avec notre système ou que cela soit pour communiquer avec les applications des cliniques.

Dans cette section, nous allons donc présenter certains standards utilisés pour produire ou manipuler des API dans le cadre des maisons intelligentes.

2.4.1 SOAP

Simple Object Access Protocol (SOAP) est un protocole d'échange de message basé sur XML et standardisé par le W3C, la dernière version à ce jour a été rédigée par Mendelsohn *et al.* (2007). SOAP est indépendant de tout langage et peut-être utilisé par le biais de divers protocoles de communication en autant que ceux-ci permettent d'échanger des données au format XML.

Le protocole SOAP permet d'échanger des messages entre différents nœuds qui sont des composants logiciels capables d'émettre, recevoir et faire transiter des messages SOAP. Les communications se font toujours entre un unique émetteur, zéro ou plusieurs nœuds intermédiaires, et un unique destinataire. Chaque nœud SOAP peut remplir un ou plusieurs rôles, certains sont définis dans la norme de base et d'autres peuvent être définis par l'utilisateur.

Les messages SOAP sont des fichiers XML décrivant une *enveloppe* contenant un en-tête et un corps de message. L'en-tête de l'enveloppe est optionnel et peut contenir différentes informations à destination des nœuds qui recevront le message. Les informations d'en-tête permettent soit d'aider à acheminer l'enveloppe en tant que telle, soit de moduler la lecture du corps de celle-ci. SOAP ne définit que quelques attributs génériques pouvant être utilisés dans les différents éléments d'en-tête. Ces attributs représentent : les types de nœuds qui ont le droit de traiter l'en-tête, si l'en-tête doit être supprimé ou relayé une fois qu'il a été traité, si l'en-tête peut être ignoré et quel protocole a été utilisé pour chiffrer et doit être utilisé pour déchiffrer le contenu de l'en-tête. Pour le reste, l'utilisateur est libre d'utiliser la structure XML qui lui convient.

À la réception de l'enveloppe, le destinataire final peut alors en lire le corps constitué d'une structure XML arbitraire ou d'une faute et agir en conséquence. Une faute est un corps standardisé permettant de décrire une erreur.

En dehors du modèle de communication en tant que tel, le protocole SOAP définit aussi différents outils annexes permettant de :

— Décrire le *cycle de vie* des messages en fonction de leur type en explicitant comment ceux-ci doivent être échangés par différents nœuds.

— Décrire les standards utilisés pour coder les en-têtes d'enveloppe et faire vérifier ceux-ci.

— Décrire les standards utilisés pour coder les corps d'enveloppe, vérifier et décoder automatiquement ceux-ci.

Le standard est suivi d'une seconde partie proposant un modèle par défaut permettant de décrire des en-têtes et des corps de messages génériques via des structures XML. La dernière version de cette seconde partie à ce jour a été rédigée par Gudgin *et al.* (2007).

2.4.2 REST

REpresentational State Transfer (REST) est un style architectural client/serveur développé dans la thèse de Fielding (2000) définissant un style d'API pour la communication entre composants. REST est plus un ensemble de propriétés désolidarisées de toute notion relative à leur implémentation plutôt qu'un standard établi pour certains protocoles ou environnements.

Ainsi, pour qu'un système de communication soit conforme à REST, il doit respecter certaines propriétés. La première d'entre elles est celle de fonctionner sans état, c'est-à-dire qu'une requête transmise par un client à un serveur REST doit être le seul élément nécessaire pour pouvoir exécuter ladite requête. D'après l'auteur, ce type d'approche permet d'assurer la scalabilité d'un système REST : les requêtes étant la seule source d'information nécessaire pour démarrer un calcul, elles peuvent ainsi être transmises à n'importe quel serveur et il est donc possible d'utiliser plusieurs machines en fonction de la charge de travail à réaliser. Utiliser une approche sans état facilite aussi l'analyse d'un flux d'échange, l'observateur n'ayant que les requêtes à regarder pour comprendre l'ensemble du processus. Finalement, une approche sans état augmente aussi la fiabilité d'un système : si une requête échoue, il est possible de la retraiter ou de la passer à un tiers directement sans avoir à remettre à zéro ou transférer un état gardé en mémoire.

La mise en cache est aussi une propriété importante du modèle REST et elle est assurée par la propriété des requêtes à s'autosuffire : si un serveur, ou un client, doit transmettre deux fois la même requête, le résultat de celle-ci peut aisément être gardé en mémoire et être resservi tel quel. De même, si plusieurs clients exécutent la même requête, un même serveur peut resservir le même résultat plusieurs fois et donc mutualiser les bénéfices de l'opération de mise en cache.

Pour qu'une API soit conforme au modèle REST, elle doit aussi utiliser une seule et

unique interface de communication : les règles doivent être les mêmes pour tout le monde. Cela est notamment assuré par la manière dont les requêtes et les réponses REST fonctionnent.

En dehors de ces propriétés, le fonctionnement de REST réside dans le traitement de ce qu'il définit comme des *ressources* . D'après l'auteur, toute information pouvant être nommée peut devenir une *ressource* REST. De même, tout concept qui peut recevoir un identifiant hypertexte (comme une *Uniform Resource Locator* (URL)) rentre dans la définition de ce qu'est une ressource.

Pour lire ou modifier une ressource, une architecture REST fonctionne via des *représentations* . Une représentation est un standard permettant de chiffrer de l'information sous forme de bits, comme les exemples de standards présentés en Section 2.2. Une représentation peut contenir de la méta-information, c'est-à-dire de l'information à propos de l'information et, parfois, de la méta-méta-information qui, selon l'auteur, consiste principalement en des sommes de contrôle permettant de vérifier l'intégrité d'un message. Finalement, la représentation peut contenir ce qu'on appelle de l'information de contrôle, c'est-à-dire de l'information décrivant l'action à réaliser. Le triplet, constitué d'un identifiant de ressource, d'une représentation et d'une information de contrôle forme ce qu'on peut appeler une requête REST entre composants.

REST est intéressant de par ses propriétés et notamment son implémentation dans le cadre de *HyperText Transfer Protocol* (HTTP). Les requêtes deviennent en effet assez simples, car elles sont alors constituées de trois éléments : un identifiant, une action et des données. L'identifiant prend la forme d'une URL permettant de décrire une ressource à manipuler, par exemle « http ://127.0.0.1/maison/10/capteurs » identifie la liste des capteurs de la maison numéro 10. L'action HTTP décrit la nature de l'action à réaliser sur la ressource, par exemple, GET permet de lister les capteurs, POST de créer un nouveau capteur, UPDATE de modifier

la liste des capteurs au complet et DELETE de détruire l'ensemble des capteurs de la maison. Finalement, la requête peut contenir des données dans un format identifié via un type MIME, tel que défini dans la RFC de Freed *et al.* (2013). Nous avons présentés des exemples de format en Section 2.2. Les données pourront alors être utilisées par le serveur, dans notre exemple pour créer le nouveau capteur dans le cadre d'une requête de type POST, ou pour filtrer la collection à retourner dans le cadre d'une requête de type GET.

Les réponses sont alors constituées d'un code de réponse HTTP, comme le code 200 pour indiquer que la requête a été correctement gérée, le code 404 pour dire que l'identifiant n'existe pas, le code 206 pour signifier que la réponse contient une représentation partielle de l'information demandée ou le code 400 pour signaler que la requête transmise est mal-formée. La réponse peut alors venir avec de l'information dans un format identifié via un type MIME apportant plus de précision relative à l'erreur ou au succès rencontré.

REST est donc à la fois un style architectural, et une classe d'API permettant à des composants d'échanger entre eux au sujet de différentes ressources.

2.4.3 GRAPHQL

Graph Query Language (GraphQL) est un standard développé par Facebook et l'de développement de GraphQL (2018), décrivant un langage de requête ainsi que la manière de le traiter. Les requêtes GraphQL ont la particularité d'identifier le type d'information que l'on souhaite obtenir tout en définissant aussi la représentation que l'on souhaite que les données prennent. Selon la spécification, GraphQL est développé en premier lieu pour faciliter l'extraction de données dans l'objectif de produire des interfaces homme-machine.

GraphQL est donc utile dans la situation où un client souhaite manipuler ou extraire de l'information d'un serveur distant. Pour ce faire, le serveur doit définir et mettre à disposition

des clients l'ensemble des types de données manipulables sous la forme d'un fichier appelé schéma. Un schéma peut contenir différentes structures ainsi que leurs relations les unes vis-à-vis des autres. Par exemple, un schéma peut contenir une structure représentant des capteurs. Cette structure peut être constituée d'un nom sous la forme d'une chaîne de caractères, d'un identifiant sous la forme d'un nombre entier, d'une date de création sous la forme d'un nombre entier et d'une maison qui est une relation vers une autre structure définie dans le même schéma.

Les clients, en utilisant le schéma offert par le serveur, peuvent alors envoyer des documents GraphQL pour obtenir de l'information ou réaliser des tâches diverses. Un document GraphQL peut contenir différents messages qui peuvent être des requêtes, des mutations ou des inscriptions. Les requêtes GraphQL permettent tout simplement d'extraire de l'information d'un serveur en explicitant ce que l'on souhaite obtenir et la forme que la réponse doit avoir. Les mutations sont des requêtes visant à modifier l'état du serveur GraphQL. Finalement, les inscriptions sont des requêtes types publication / inscription où le client se dit intéressé de recevoir un certain type d'information en continu en fonction des événements internes ayant lieu au sein du serveur GraphQL.

GraphQL est donc un langage de dialogue pouvant être utilisé pour permettre à des composants logiciels de s'échanger de l'information. C'est une solution qui a de l'intérêt dans le cadre du développement d'interfaces graphiques car elle permet aux clients de faire des requêtes très complexes en une seule fois, de recevoir de la part du serveur distant une information préformatée et d'en faire ainsi le rendu directement sans passer par des étapes de transformation intermédiaires de l'information obtenue. GraphQL est cependant un système difficile à configurer et nécessitant des ressources que certaines plateformes ne peuvent offrir.

2.4.4 BILAN

Nous avons présenté dans les sections précédentes trois modèles d'API : SOAP, REST et GraphQL. REST est la solution la plus pratique dans notre contexte : elle est supportée par défaut par notre cadre de travail, tout comme JSON, et elle demande moins de configuration qu'une solution SOAP ou GraphQL. De plus, si un modèle plus avancé comme GraphQL est nécéssaire à un moment donné, ce dernier peut être construit sur les bases d'une solution REST.

2.5 ARCHITECTURES

Dans les trois sections précédentes, nous nous sommes intéressés à certaines problématiques clefs des architectures de maisons intelligentes. Dans cette section, nous allons décrire de plus près les architectures elles-mêmes.

Le développement d'architectures pour la maison intelligente a principalement commencé au tournant des années 2000. Bien que l'idée soit en réalité bien plus ancienne, ce sont bien des progrès matériels et logiciels qui ont facilité le développement de nouvelles architectures de maisons intelligentes en parallèle d'autres projets visant à développer des solutions pour d'autres types d'environnements. L'introduction du rapport de Youngblood *et al.* (2004) sur l'architecture de maison intelligente *Managing and Adaptive Versatile Home* (MavHome) développée à l'Université du Texas à Arlington est assez éclairante en ce sens. En effet, on constate que, durant ce projet, pas moins de quatre autres projets de recherches sont menées en parallèle sur le même thème des environnements intelligents par d'autres universités. On compte, notamment, un projet de maison intelligente par la Georgia Institute of Technology décrit dans un article de Kientz *et al.* (2008), des projets du *Massachusetts Institute of Technology* (MIT) et de l'Université de Stanford sur les espaces de travail intelligents et une solution

de l'Université du Colorado à Boulder par Mozer (1999) sur les environnements intelligents en général.

D'autres projets concernant la maison intelligente ont été menés après MavHome, avec notamment l'architecture GatorTech de Helal *et al.* (2008), produite à l'Université de Floride, et visant à construire une solution modulaire pouvant s'adapter aux changements technologiques. L'architecture CASAS « a smart home in a box » de Cook *et al.* (2013a), visant à produire une architecture pouvant être déployée facilement pour extraire de l'information concernant le comportement d'individus dans diverses maisons. Le protocole réseau Light Node Communication Framework de Plantevin *et al.* (2017), conçue à l'Université du Québec à Chicoutimi (UQAC), et visant à optimiser et à améliorer la sécurité des communications au sein des maisons intelligentes. Finalement l'architecture *Web-based Internet of Things Smart home* (WITS) de Yao *et al.* (2018), imaginée à l'Université de New South Wales à Sydney, profitant des nouvelles avancées techniques avec un point particulier sur l'expérience utilisateur et la simplicité de développement d'applications pour le quotidien. En parallèle de ces projets, des solutions à code source libre ont aussi émergé, permettant de déployer des installations fonctionnant sur des plateformes maintenues par une communauté active, avec notamment la solution *Open Home Automation Bus* (OpenHAB) évaluée dans l'article de Smirek *et al.* (2014). Toutes ces solutions cherchant à résoudre des problématiques différentes, il est important d'étudier leur fonctionnement pour construire un modèle plus générique.

2.5.1 MAVHOME (2004)

MavHome est une architecture de maison intelligente proposée en 2004 par une équipe de l'Université du Texas à Arlington. L'objectif de MavHome est de permettre à des maisons de percevoir, raisonner, et agir pour maximiser le confort et la sécurité de ses occupants tout en minimisant sa propre consommation en ressources. Bien que l'approche proposée

ait été principalement testée dans le cadre de la maison intelligente, elle est, dans les faits, généralisable à tout type d'environnement intelligent.

MavHome est une architecture ancienne et peu complexe qui pose des bases utilisées dans toutes les études que nous allons explorer. Cela en fait une proposition dont l'étude est un bon point de départ pour introduire des éléments généraux et nous focaliser, ensuite, sur les spécificités de chaque environnement.

MavHome est une architecture en couche comprenant six niveaux. La couche la plus basse, **la couche physique**, est définie comme l'ensemble des capteurs, effecteurs, alimentations et autres appareils nécessaires au bon fonctionnement de l'environnement. Au-dessus de cette couche physique vient **la couche d'interface physique** avec l'ensemble des éléments permettant aux différents appareils de communiquer entre eux. Cette couche comprend des éléments matériels et logiques comme du câblage, des routeurs, des ordinateurs ou des logiciels pilotes. La troisième couche, la **couche d'interface logique**, quant à elle, se charge d'exposer tout ce matériel aux couches supérieures sous la forme de micro-services offrant une méthode uniforme pour communiquer avec l'ensemble du matériel, que cela soit les capteurs, les effecteurs ou autres éléments du réseau.

La couche suivante, la **couche intergiciel** est, quant à elle, composée uniquement de logiciels qui remplissent deux rôles : recenser les services proposés par la couche d'interface logique et exposer un service unique permettant de lire et d'agréger plus facilement les données exposées par la multitude d'interfaces de la couche inférieure. Vient alors l'avant-dernière couche, la **couche de service**, qui propose des outils permettant de transformer, miner et stocker de l'information extraite des intergiciels. Cette couche est alors utilisée par la dernière couche de l'architecture MavHome, la **couche applicative**, qui s'occupe de la prise de décision et de tout le travail d'apprentissage nécessaire pour que le système puisse analyser

son environnement. La Figure 2.4 présente un schéma de l'ensemble de cette architecture.

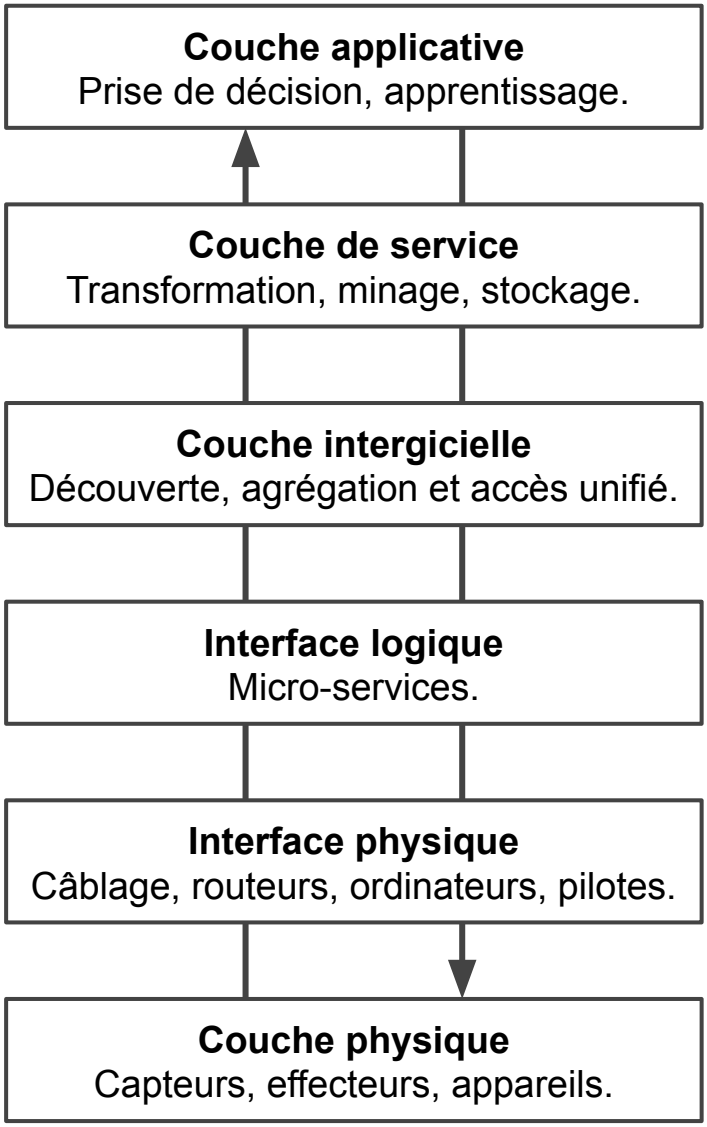

Figure 2.4 : Schéma représentant les différentes couches de l'architecture mavhome.

D'un point de vue plus général, MavHome modélise la maison intelligente comme un agent intelligent qui, selon Stuart Russell (2009), est un système capable de percevoir son environnement à l'aide de capteurs, de raisonner via une logique qui lui est propre et d'agir sur la réalité qui l'entoure via des effecteurs. Le raisonnement interne d'un agent intelligent

est une fonction qui met en relation des perceptions et des actions à entreprendre, on l'appelle la fonction agent. Cette définition d'un agent intelligent est résumée par dans la Figure 2.5.

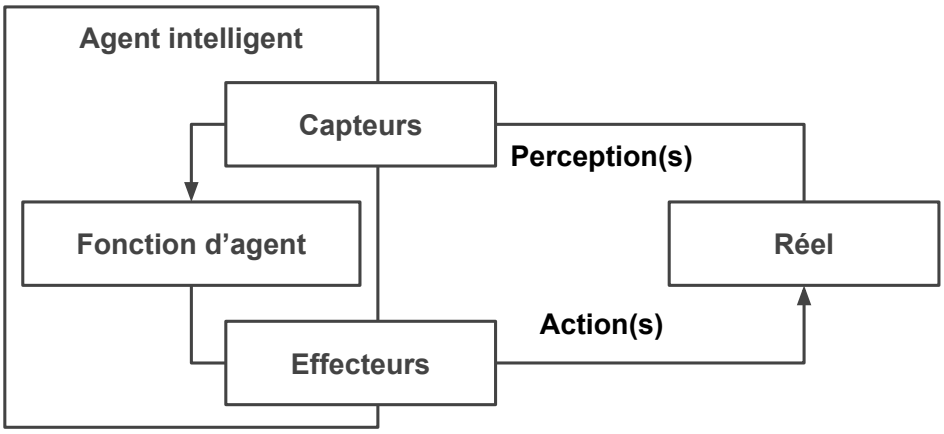

Figure 2.5 : Schéma représentant un agent intelligent selon Stuart Russell.

Cette façon de concevoir l'environnement se retrouve alors dans l'implémentation en couches : tout le processus de perception devient une série de signaux ascendants, l'information est d'abord perçue par des capteurs, puis communiquée par le réseau aux ordinateurs chargés de réaliser le calcul, cette information passe ensuite par chaque couche jusqu'à atteindre la couche applicative qui peut finalement prendre une décision. La décision prise par la couche applicative peut alors être traduite en action en transmettant une série de signaux descendants de la couche applicative jusqu'aux actuateurs.

2.5.2 GATORTECH (2008)

GatorTech est une architecture de maison intelligente conçue à l'Université de Floride en 2005. Cette solution a pour objectif de permettre d'intégrer facilement de nouveaux composants pour étendre les capacités de la maison tout en offrant un cadre de travail ergonomique pour développer des applications à destination de ses occupants. L'accent mis sur la modularité et l'accessibilité des moyens de configuration fait de cette architecture un premier pas vers un

système installable par des personnes sans bagage technique.

L'architecture GatorTech fonctionne aussi en couches successives. Tout comme dans le cas de MavHome, la première couche de l'architecture est **la couche physique** comprenant l'ensemble des équipements matériels permettant à l'environnement de percevoir et agir. Cette couche physique est alors suivie d'une **couche appelée *plateforme capteur*** , composée de logiciels qui ont pour objectif d'exposer tous les capteurs et effecteurs de la maison sous la forme de services uniformisés. Cette solution permet d'ajouter ou d'extraire facilement des technologies de la maison via une technique proche du motif de conception adaptateur tel que décrit dans le livre de Gamma *et al.* (1994). Chaque adaptateur communique directement avec **une couche dite de service** qui est en réalité un cadre de travail *Open Services Gateway initiative* (OSGi) [19] assurant l'aspect modulaire du projet. L'outil OSGi va se charger en temps réel de lister, enregistrer, démarrer puis, le moment venu, détruire ou arrêter diverses portions de codes utilisés par les couches supérieures. En plus des modules représentant des capteurs et effecteurs, d'autres services, comme la reconnaissance vocale, la synthèse vocale et notamment le système de diffusion d'éléments multimédias comme des musiques ou des vidéos sont aussi enregistrés sous forme de module dans cette couche qui va assurer le couplage le plus lâche possible entre chaque morceau de logiciel, permettant ainsi de les changer à loisir sans devoir reprogrammer toute l'architecture.

Arrivent finalement les couches réalisant le travail d'intelligence. La première d'entre elles, **la couche de connaissance**, est principalement constituée d'un moteur logique permettant de raisonner sur la structure de l'environnement et la nature des services proposés dans la couche de service. C'est une couche qui peut, par exemple, adapter la sortie de certains capteurs automatiquement en convertissant l'unité de mesure utilisée par ceux-ci en fonction du besoin final. **La couche de gestion du contexte** utilise alors toutes les couches inférieures

19. Pour plus d'information se référer à https://www.osgi.org/

pour comprendre dans quel état général se trouve la maison, si ces états sont sécuritaires pour l'occupant et si les actions à réaliser dans un futur proche ne compromettent pas le bon fonctionnement de l'environnement. C'est cette partie de l'architecture qui va assurer la sécurité de la maison et qui va se charger de toutes les prises de décisions automatiques permettant de maximiser le confort de ses occupants en fonction des moyens disponibles. La dernière couche, **la couche applicative**, expose finalement une interface homme-machine aux occupants pour leur permettre de configurer et d'interagir avec la maison en proposant un outil graphique, facile d'utilisation, permettant à des personnes sans bagage technique de programmer des tâches simples. Toute cette architecture est représentée dans la Figure 2.6

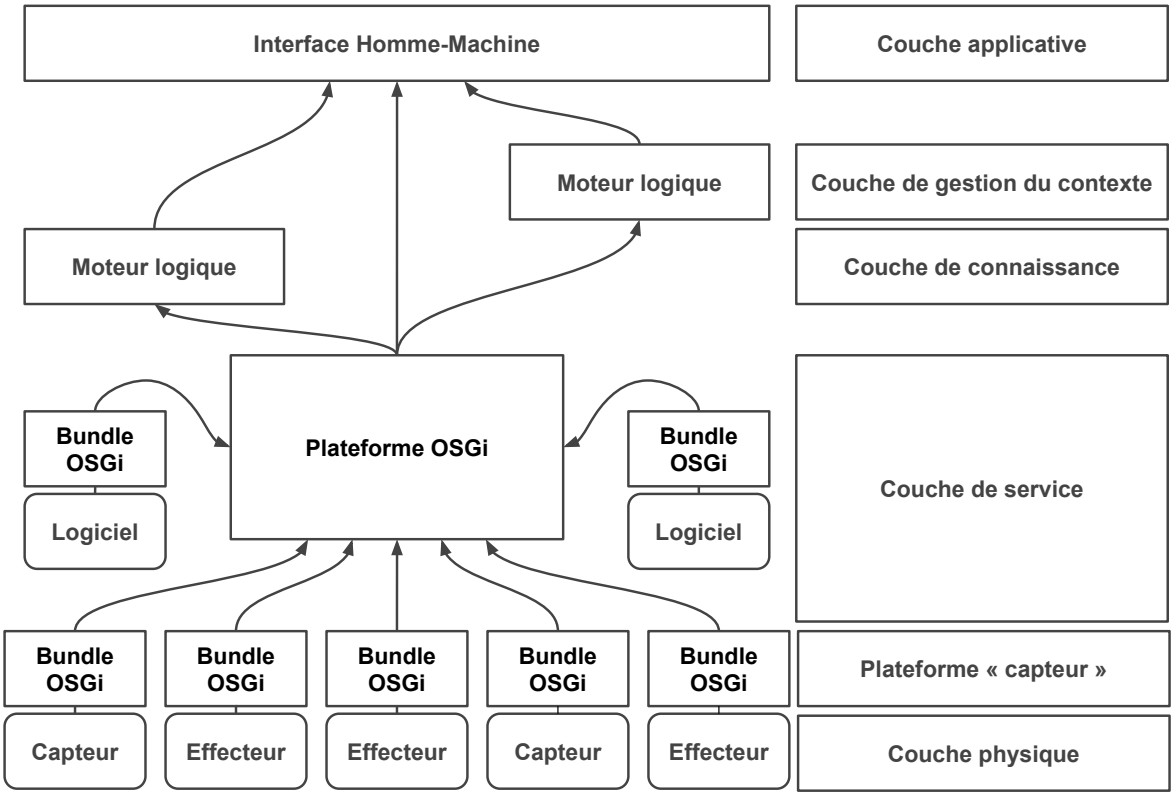

Figure 2.6 : Représentation des couches de l'architecture GatorTech.

En dehors de sa structure, cette architecture approche la manipulation des données de

manière intéressante en tentant au maximum d'abstraire toute l'information extraite par la maison. Ainsi, dans le cadre de l'architecture GatorTech, on cherchera à utiliser une approche plutôt qualitative que quantitative. Par exemple, plutôt que de considérer la température exacte d'une pièce dans une unité de mesure quelconque le système préfèrera évaluer à quel degré la pièce est *chaude* , *froide* ou *tempérée* . Cette approche permet de simplifier la définition des règles par les utilisateurs qui remplaceront une formule de la forme « si la température de la cuisine est supérieure à 26 degrés alors activer l'air climatisé » en « si la température de la cuisine est chaude, alors activer l'air climatisé ».

Le système GatorTech adopte la même approche avec les effecteurs en décrivant ceux-ci par l'intention qu'il y aurait dans leur activation plutôt que par leur nom effectif. Par exemple, l'air climatisé sera plutôt considéré comme un système *permettant de baisser la température* . Ce type de formalisme permet de modéliser les liens possibles entre un état donné de la maison et un état désiré par le biais de la réalisation de certaines actions. Ainsi, il devient possible de résoudre des scénarios du type : « il n'y a plus de nourriture », « il n'y a pas de système permettant de mettre de la nourriture dans le frigo » alors « appeler de l'aide externe » ou encore « la température de la cuisine est chaude », « il y a des systèmes permettant de réduire la température de la cuisine » alors « activer les systèmes permettant de réduire la température de la cuisine ». Remarquez que, dans ce second exemple, l'architecture pourra alors activer l'air climatisé comme la fermeture des stores si ceux-ci sont aussi décrits comme des systèmes permettant de réduire la température.

2.5.3 CASAS (2013)

CASAS est une architecture de maison intelligente mise au point par une équipe du laboratoire éponyme de la Washington State University en 2013. L'objectif premier de l'architecture CASAS est de proposer une solution légère, facile à déployer et prête à l'emploi. Dans

les faits, cela se traduit par un kit matériel qui tient dans une simple boîte, une architecture en couche simplifiée au maximum et l'absence de configuration ou de calibrage de l'infrastructure durant son installation.

L'architecture CASAS, comme l'architecture MavHome et GatorTech est une architecture en couches. La première couche est toujours **la couche physique** qui représente le matériel nécessaire pour que la maison puisse percevoir et agir avec notamment l'ensemble des capteurs, effecteurs et micro-ordinateurs. La seconde couche, **la couche intergiciel** est principalement constituée d'un système de communication événementielle sous la forme d'une plateforme de publication / inscription permettant à un service d'envoyer des messages à un nombre arbitraire d'applications clientes et à des applications clientes d'écouter un nombre arbitraire de services en utilisant un système de canaux. Certaines fonctionnalités supplémentaires, telles qu'un système permettant de garder en mémoire le dernier état de la maison, sont aussi intégrées à cette seconde couche. De manière similaire au choix de l'architecture GatorTech, tous les composants physiques communiquent avec la couche intergiciel via des micros-services qui fonctionnent comme des ponts tel que décrit dans le motif de conception du même nom du livre de Gamma *et al.* (1994), cette technique permettant à CASAS d'accepter n'importe quel nouveau composant s'il est livré avec un service logiciel compatible. Finalement, la dernière couche, **la couche applicative**, regroupe l'ensemble des logiciels à destination de l'environnement. À la livraison de l'architecture, cette couche contient une solution pour reconnaître un ensemble pré-établi d'activités en utilisant des séquences d'événements ainsi qu'une solution pour découvrir de nouvelles activités qui ne seraient pas encore reconnues.

Comme dans les cas précédents, la perception est un processus ascendant, de la couche physique jusqu'à la couche logicielle, et l'action est un processus descendant partant d'une décision prise dans la couche logicielle qui redescend jusqu'à la couche physique. La Figure

2.7 schématise complètement cette architecture.

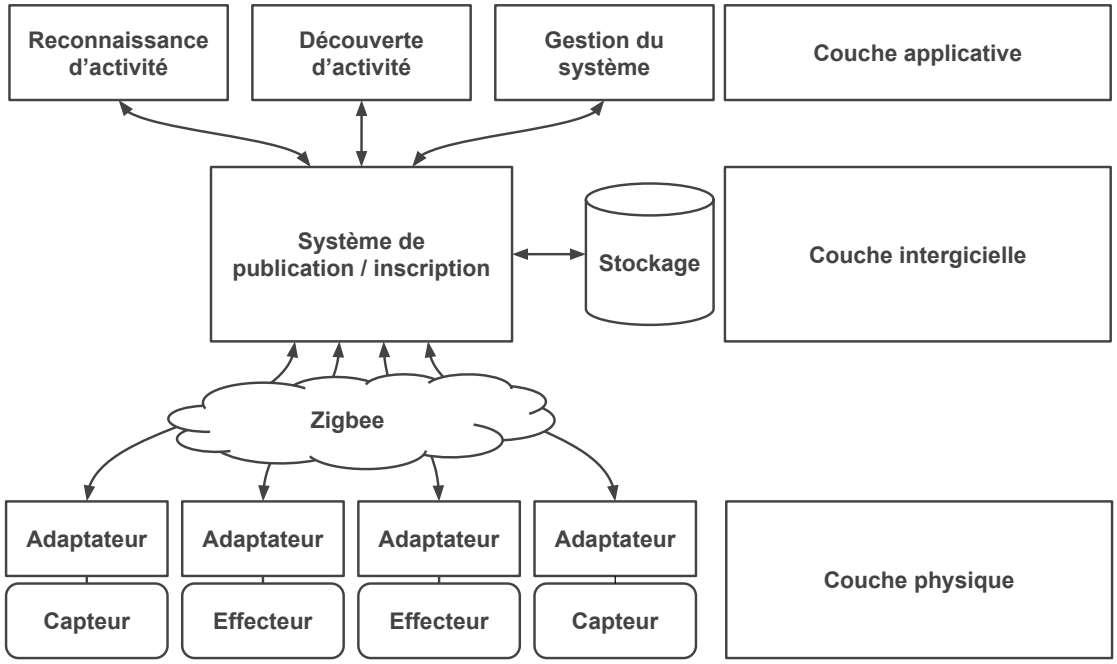

Figure 2.7 : Représentation de l'architecture CASAS.

La communication au sein de l'architecture CASAS est assurée par un maillage ZigBee. ZigBee, tel que décrit dans le livre de Gislason (2008), est un protocole de communication conçu principalement pour travailler avec des groupes de capteurs et d'effecteurs. C'est une solution qui répond à des contraintes spécifiques à ce domaine, comme le besoin de limiter au maximum la consommation énergétique afin de ne pas devoir remplacer les batteries des capteurs trop fréquemment, le besoin d'avoir un coût de déploiement abordable car celui-ci est alors multiplié par le nombre de capteurs ayant besoin de communiquer au sein de la maison ou encore le besoin d'éviter au maximum la perte d'information ou l'impossibilité de communiquer au sein du réseau pour ne pas rendre la maison aveugle. ZigBee permet notamment, par son modèle en maillage de capteurs qui communiquent de proche en proche, d'assurer la communication même si certaines pièces tombent en panne ou si certains élé-

ments environnementaux empêchent la communication en adaptant sa manière de transmettre l'information en temps réel.

Pour ce qui est de sa simplicité, l'architecture a été déployée dans un total de 32 maisons pour l'étude de 2013 avec une durée d'installation moyenne de deux heures et une durée de désinstallation moyenne d'une demi-heure. Cette rapidité est le résultat d'un processus de standardisation menant à la production d'une architecture qui tient dans une boîte avec un ensemble de capteurs fixes préconfigurés et étiquetés avec leur lieu d'installation. La plus grosse difficulté, citée dans leur article de 2013, consiste principalement à trouver le bon positionnement de capteurs de mouvement permettant d'éviter des détections par plusieurs capteurs aux mêmes moments ou des fausses détections, comme une détection dans la cuisine alors que la personne passe seulement devant la porte de celle-ci.

2.5.4 OPENHAB 2.X (2017)

OpenHAB est un projet de solution logicielle open source initialisé en 2010 dont la version stable 2.0, annoncée dans le billet de blog de Kreuzer (2017), est sortie en 2017. D'après la documentation en ligne d'OpenHAB 2.5, la solution se présente comme une architecture en deux couches comprenant une couche qui est complètement liée à la représentation des différents composants et une seconde couche qui représente toutes les applications utilisant la maison intelligente. Cette manière de présenter l'architecture est ensuite rapidement supplantée par la définition des différents éléments logiques qui la composent, ils parlent alors de **couche physique** et de **couche virtuelle**.

La couche physique est un ensemble d'éléments appelés *composants* (*things* en anglais) qui représentent chacun un morceau insécable de la maison intelligente. Ces morceaux peuvent être matériels ou logiciels, les *composants* peuvent ainsi représenter des capteurs, des lumières,

une cuisinière ou une horloge tout comme une connexion à un service météo ou une connexion à un service de livraison. Chaque *composant* peut alors être configuré individuellement, comme en fournissant l'identifiant et le mot de passe du compte permettant d'accéder à la météo en ligne dans l'exemple du service météo.

Tout *composant* de l'infrastructure peut alors exposer en lecture ou en écriture une ou plusieurs fonctionnalités offertes par la maison sous la forme de *canaux* (*channels* en anglais). Un capteur complexe peut ainsi exposer un canal pour lire la température d'une pièce et un canal pour accéder à sa luminosité, un service météo peut exposer un canal pour accéder à la météo des prochaines 24 heures et une lampe connectée peut offrir un canal pour définir la couleur de la lumière émise et un canal pour définir si celle-ci doit être allumée ou éteinte.

Chaque *composant* est aussi une machine à états suivant sa propre logique permettant notamment de traquer l'ajout, la destruction, l'initialisation, le dysfonctionnement ou la modification d'élément de la maison en temps réel dans le logiciel. Finalement, comme dans le cadre de GatorTech et de CASAS, tous les composants sont livrés avec des *connecteurs* (bindings en anglais) qui sont des petites pièces logicielles qui fonctionnent comme des adaptateurs au sens du livre de Gamma *et al.* (1994) et qui assurent l'extensibilité de la plateforme.

La couche virtuelle est, quant à elle, un ensemble d'éléments appelés *objets* (items en anglais) qui représentent des fonctionnalités utilisées par des logiciels tiers. Chaque *objet* a un état spécifique et représente un type de valeur particulier comme un booléen, un nombre, ou une chaîne de caractères et ne peut être manipulé que par le biais d'événements communiqués à la plateforme. La luminosité du salon et la température de la cuisine sont deux exemples *d'objets* possibles.

Chaque *objet* de la couche virtuelle est associé à un *canal* de la couche physique via

un *lien* (link en anglais) qui permet de garantir une indépendance entre la représentation virtuelle de la maison et son implémentation réelle. Ce type de solution permet, par exemple, de ne pas provoquer d'erreur quand on change le capteur de température d'une cuisine au profit d'un nouveau plus efficace. Comme la représentation virtuelle n'arrête pas de capter la température de la cuisine, les services qui dépendent de cette fonctionalité n'ont pas à se soucier d'une coupure ou de devoir gérer la reconnection avec le nouveau matériel installé.

Finalement, des *règles* (*rules* en anglais) permettent d'automatiser des comportements simples en posant des conditions sur l'état de la maison provoquant une série d'actions préétablies si elles sont évaluées positives à un instant particulier. OpenHAB permet ce fonctionnement en temps réel par le biais de l'usage d'un système événementiel, toute action dans l'application est alors la conséquence de la publication d'un événement complexe par du code tiers provoquant alors une réaction de la part d'autres morceaux de codes qui surveillent l'apparition d'événements types donnés. Comme pour le cas de GatorTech, OpenHAB utilise le framework OSGi afin d'assurer sa modularité et pour ainsi permettre à ses utilisateurs d'ajouter de nouveaux paquetages de fonctionnalités à la volée. La Figure 2.8 schématise grossièrement ce type d'architecture.

2.5.5 LIGHT NODE COMMUNICATION FRAMEWORK (2017)

Le *Light Node Communication Framework* de Plantevin *et al.* (2017) est un protocole réseau conçu au Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activité (LIARA) à l'Université du Québec à Chicoutimi en 2017. Ce protocole, contrairement aux architectures présentées jusqu'à maintenant, ne traite que d'une partie spécifique de l'infrastructure de la maison intelligente. Son objectif premier est de proposer une solution légère, implémentable sur une vaste variété de plateformes, permettant de mettre en place un système

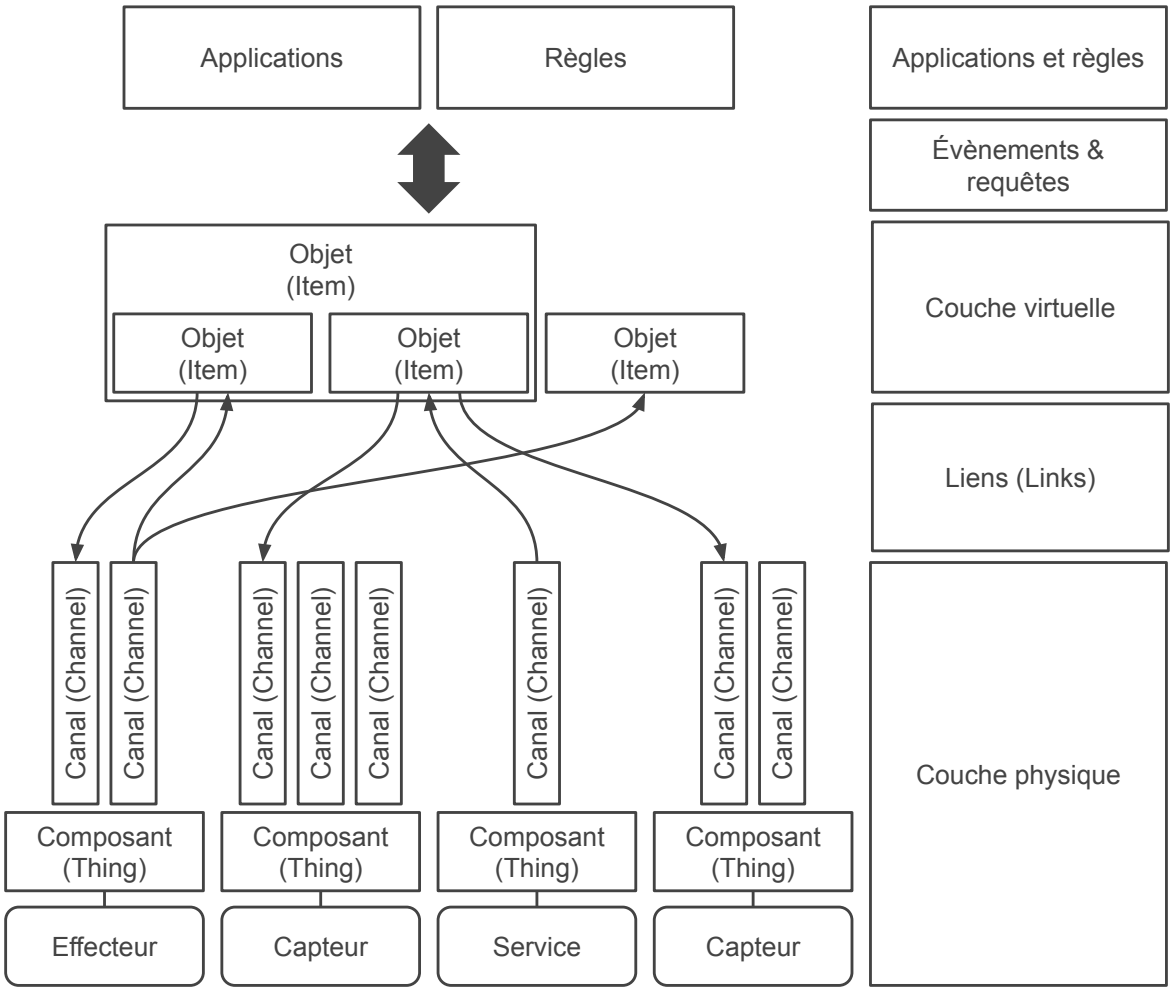

Figure 2.8 : Représentation de l'architecture OpenHAB.

de communication fiable, adaptable et complètement décentralisé au sein de la maison.

CASAS, GatorTech, MavHome et OpenHAB sont des solutions utilisant toutes un système événementiel pour fonctionner. Ce système peut prendre la forme d'un bus dans le cas d'OpenHAB, c'est-à-dire d'un système qui va communiquer les événements qui y entrent à tous les morceaux de logiciels qui sont branchés dessus. Il peut encore prendre la forme d'un système centralisé de publication / inscription comme dans le cas de CASAS, GatorTech et MavHome où des morceaux de l'infrastructure vont se dire intéressés par des

sujets particuliers comme *les mouvements dans le salon* et ne recevoir que les événements publiés par des systèmes tiers dans les sujets qui l'intéresse, ce fonctionnement peut être aisément comparé à celui d'un salon de clavardage. Ces systèmes ont le défaut principal d'être centralisés, c'est-à-dire qu'une pièce logicielle unique est nécessaire au bon fonctionnement de toutes les communications du système. Si cette pièce rencontre un problème, il n'y a alors plus aucune communication dans toute la maison, et la maison au complet peut être considérée à l'arrêt.

Pour pallier à ce problème, le *Light Node Communication Framework* propose de laisser les capteurs, effecteurs et micro-ordinateurs faire tout le travail de communication en utilisant un protocole réseau spécifique. La grande variété des dispositifs pouvant communiquer, logiciels comme matériels, requiert d'avoir le minimum de préconditions possibles afin de pouvoir implémenter le protocole sur un maximum de pièces. Pour cela, le protocole proposé ne requiert d'avoir accès qu'à une implémentation fonctionnelle de l'*Internet Protocol* (IP).

Le *Light Node Communication Framework* offre deux solutions. La première permet de configurer les éléments de la maison à distance via *Constrained Application Protocol* (CoAP), un protocole ayant des similarités avec HTTP. CoAP divise toutes les commandes de configuration en deux éléments : un élément représentant une action comme lire, créer, mettre à jour et un élément représentant une ressource comme la version d'un capteur particulier ou le nom d'un capteur particulier. La seconde solution vise, quant à elle, à communiquer de l'information dans toute la maison, chiffrée ou non, en passant par un mélange entre un système très léger de publication / inscription fonctionnant en réalité comme un système permettant de filtrer des messages distribués à toutes les pièces matérielles connectées à un même réseau (broadcast en anglais). Cette seconde solution offre aussi une méthode pour découvrir les éléments matériels présents dans le système via des requêtes simples.

Cette solution bien que se focalisant seulement sur une partie plus spécifique de la maison intelligente permet de présenter une autre approche de la standardisation basée sur un standard réseau plutôt qu'un système de pont sous la forme de services tels que présenté dans le livre de Gamma *et al.* (1994). Ce protocole permet aussi d'explorer un modèle événementiel décentralisé où l'intelligence est répartie sur tous les éléments du système plutôt que sur des gros blocs monolithiques pouvant provoquer des pannes totales de l'architecture s'ils venaient à faillir.

2.5.6 WITS (2018)

WITS est une architecture conçue et proposée par l'Université de New South Wales de Sydney en 2018. C'est une solution qui vise à exploiter les technologies web qui ont émergé durant la dernière décennie ainsi que des solutions liées au domaine de l'Internet des Objets (IdO) pour répondre aux problèmes d'interopérabilité entre composants d'une même maison ainsi que pour réaliser de la reconnaissance d'activités en temps réel. L'architecture proposée offre aussi une interface homme-machine visant des objectifs similaires à l'interface proposée par la solution GatorTech, c'est-à-dire de permettre à des tiers sans bagage technique et des médecins sans connaissances informatiques de configurer leur maison.

WITS est toujours une architecture en couches, elle comporte quatre composantes. La première couche est toujours **la couche physique** comprenant l'ensemble du matériel permettant de faire de l'environnement un environnement intelligent. Cette première couche est toujours composée des différents capteurs, effecteurs et micro-ordinateurs. Vient ensuite **la couche de gestion de la détection** (*sensing management* en anglais) dont l'objectif principal est d'uniformiser la manière de communiquer avec les différentes pièces matérielles de la maison via une API qui fonctionne comme les solutions de ponts de CASAS ou GatorTech. Contrairement aux solutions précédentes, WITS fait le choix de transformer toutes les com-

munications de la maison en communications synchrones. Dans les faits, tout échange dans l'infrastructure proposée est donc censé être bloquant : les étapes à résoudre pour parvenir à un résultat final se réalisent donc systématiquement les unes après les autres, et jamais au même moment. La plateforme utilisée pour gérer la détection est un service suivant le standard REST tel que défini dans la thèse de Fielding (2000). Finalement, cette couche de l'infrastructure se charge aussi de précalculer un ensemble de statistiques sur les blocs d'information reçus par la maison. Parmi ceux-ci, on retrouve des minimums, des maximums, des moyennes, des variances, ainsi qu'un ensemble d'autres informations statistiques standards.

La couche intermédiaire de WITS est appelée la **couche de gestion du contexte** (*context management* en anglais) dont l'objectif premier est de transformer toutes les informations brutes de la couche précédente en information de plus haut niveau décrivant ce qui se passe dans la maison. Dans WITS, il existe trois types d'information dérivés de ce que les capteurs perçoivent : la position des personnes dans la maison, les interactions en cours avec des objets et finalement un ensemble d'activités atomiques représentant notamment les différents mouvements et gestes réalisés par les individus dans les pièces. Cette première reconnaissance est réalisée via des algorithmes d'apprentissage machine et tous les résultats sont alors stockés dans une base de données centrale dans la maison pour effectuer des calculs ultérieurs.

L'avant-dernière couche de l'infrastructure WITS est alors la **couche de gestion des règles** (*rule management* en anglais). C'est cette couche qui va réaliser la reconnaissance d'activités complexes comme *faire la cuisine* ou *préparer un café* en appliquant un ensemble de règles de la forme déclencheur / action à la base de données centrale de la maison. Ainsi, quand tout nouvel élément est découvert, cette couche vérifie chaque règle qu'elle contient pour ensuite produire l'information finale en réaction. Cette solution vise premièrement à combler un déficit d'information permettant l'optimisation d'algorithmes d'apprentissage plus complexes.

La dernière couche de WITS est une couche Interface Homme-Machine (IHM) sous la forme d'une application web permettant à des tiers de monitorer la maison et de configurer la base de règles de l'architecture via un outil graphique. Cette interface est disponible sur plusieurs types de terminaux comprenant ordinateurs, téléphones et tablettes. La Figure 2.9 est une représentation schématisée du fonctionnement de l'architecture WITS.

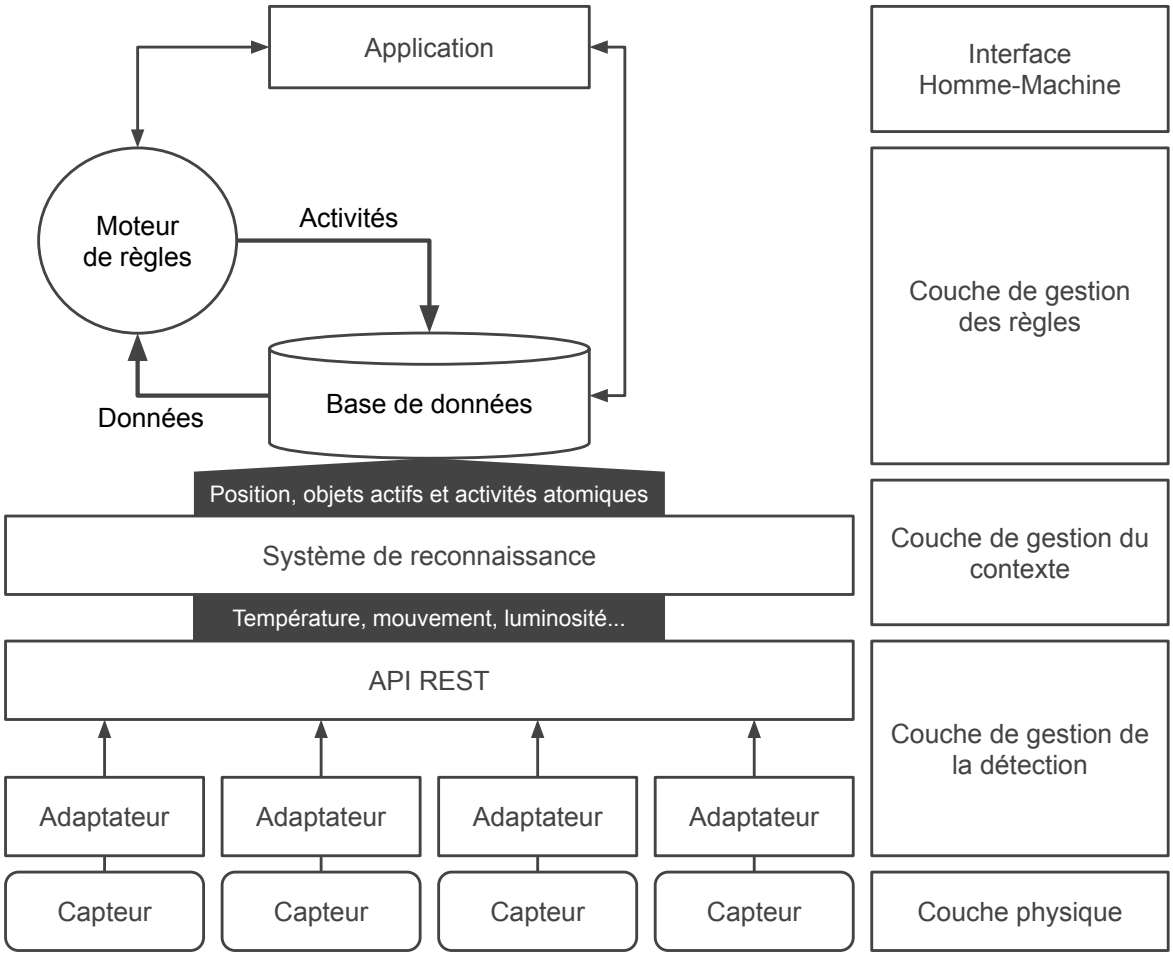

Figure 2.9 : Représentation de l'architecture WITS.

Contrairement à d'autres solutions présentées précédemment, WITS est une architecture conçue pour extraire de l'information de maisons dans un but médical et délaisse les outils

permettant à la maison de prendre des décisions et d'agir sur son environnement.

2.5.7 SEARCH (2020)

Smart Environment ARCHitecture (SEArch) est une architecture de maison intelligente proposée par l'université Middlesex of London en 2020 dans le papier de Augusto *et al.* (2020). C'est une architecture qui s'est construite au fur et à mesure en répondant à différents projets réalisés en collaboration avec plusieurs acteurs extérieurs. L'architecture SEArch se veut construite par l'expérimentation sur le terrain tout en proposant une approche exclusivement focalisée sur les problèmes posés par les utilisateurs finaux.

Cette architecture est capable de gérer à la fois la collecte des données via des capteurs et la mise à jour de l'environnement via différents actuateurs. Ces deux aspects sont gérés par des composants travaillant de concert, tous implémentés à la volée pour répondre aux différents besoins des projets dans lesquels s'inscrit l'architecture. Tous les éléments matériels sont gérés par une boîte domotique, et communiquent via un réseau sans-fil Z-Wave. Tous les éléments utilisés ne supportent pas nativement le protocole Z-Wave, et les auteurs intègrent les exceptions en employant un patron de conception de pont, tel que décrit dans le livre de Gamma *et al.* (1994).

Pour ce qui est de l'aspect reconnaissance d'activités, l'architecture SEArch implémente un moteur de règles qui opère sur le flux d'événements collectés depuis l'environnement monitoré. Ce moteur est agrémenté par différents services additionnels présents pour soit répondre à des situations plus difficiles à gérer, soit offrir aux utilisateurs finaux des fonctionnalités additionnelles utiles pour comprendre le fonctionnement de l'environnement en général. La combinaison de tous ces éléments logiciels intermédiaires forme alors le moteur de raisonnement de l'architecture. Plus précisément, c'est la pièce logicielle qui prend en

entrée les observations réalisées par les capteurs, et qui, en définitive, décide des réactions à entreprendre en accord avec ces dernières.

SEArch propose aussi un moteur d'apprentissage qui travaille à configurer les règles internes de son moteur de raisonnement en tentant de déduire ces dernières d'une observation continue de l'environnement. Finalement, l'architecture empaquette différents outils supplémentaires permettant de répondre à des problématiques qui ne s'inscrivent pas dans les deux éléments que nous venons de décrire. On peut ainsi citer l'exemple du configurateur automatique ayant pour objectif de permettre à des utilisateurs sans connaissances techniques préalables de pré-configurer l'architecture en répondant à un ensemble de questions.

En conclusion, SEArch est une architecture capable de s'adapter à différentes situations en observant son comportement interne. Cette forme de réflexivité est implémentée par un système auxiliaire d'apprentissage travaillant sur un comportement codé par un ensemble de règles strictes.

2.6 UN MODÈLE EN COUCHES

Le Tableau 2.1 permet de comparer l'ensemble des architectures que nous avons présenté, notamment, les points que nous estimons les plus importants dans notre contexte. Les critères que nous avons choisis sont : la présence d'une API permettant de manipuler des données en temps réel (API TR), la distribution des calculs (Dist.), le transport des données à des tierces parties (Trs. Dt.), la gestion d'une hétérogénéité en termes d'équipement et de disposition (Htr.), la gestion de plusieurs environnements en même temps (Pls. Ev.), l'utilisation d'un système de publication / inscription (Pub./Ins.), la reconnaissance d'activité en temps réel (Ra. TR), la capacité à percevoir et / ou agir (L/E) et la gestion du cycle de vie et de l'hétérogénéité des capteurs (G. Ca.).

On peut remarquer qu'aucune architecture ne propose de gérer plusieurs environnements en même temps. Aussi, l'hétérogénéité est géré au cas par cas : chaque architecture étant extensible, il est possible d'adapter ces dernières à différents environnement, mais cela nécessite de systématiquement coder des éléments supplémentaires, ou de toucher à minima à la configuration interne de chaque solution.

Tableau 2.1 : Tableau comparatif des architectures de notre revue de littérature.

Architecture	**Type**	**G. Ca.**	**UI**	**API**	**API TR**	**Ra. TR**	**Pls. Ev.**	**Htr.**	**Trs. Dt.**	**Dist.**
MavHome	L/E	Oui	Non	CORBA	Non	Oui	Non	Oui	Non	Non
OpenHAb	L/E	Oui	Oui	OSGi	Pub./Ins.	Oui	Non	Oui	Non	Non
GatorTech	L/E	Oui	Oui	OSGi	Non	Non	Non	Oui	Non	Non
CASAS	L/E	Oui	Oui	Non	Pub./Ins.	Oui	Non	Oui	Non	Non
Light Node	L/E	Oui	Non	Non	Non	Non	Non	Oui	Oui	Oui
WITS	L/E	Oui	Oui	REST	Non	Oui	Non	Oui	Non	Non
SEArch	L/E	Oui	Oui	Non	Non	Non	Non	Oui	Non	Non

API TR → API Temps Réel, Dist. → Distribution, Trs. Dt. → Transport des données, Htr. → Hétérogénéité, Pls. Ev. → Plusieurs Environnements, Pub./Ins. → Publication/Inscription, Ra. TR → Reconnaissance temps réel, L/E → Lecture (Perception) / Écriture (Action), G. Ca. → Gestion des capteurs.

Malgré les nombreuses différences relevées au sein du Tableau 2.1, la majorité des solutions que nous avons présentées en Section 2.5 et, notamment, les architectures 2.5.1 (MavHome), 2.5.2 (GatorTech), 2.5.3 (CASAS), 2.5.4 (OpenHAB) et 2.5.6 (WITS) sont des architectures dites *en couches*.

Selon Weilkiens *et al.* (2015), les architectures en couches sont des solutions respectant le concept de *séparation des responsabilités* tout en assurant un couplage minimal entre ses différents composants.

Pour y parvenir, ce type d'architecture va chercher à isoler tous les différents éléments d'une solution répondant à une sous problématique spécifique et faisant l'objet d'un arbitrage architectural en modules appelés *couche* . Les couches vont alors être empilées de telle manière que chacune d'elles ne puisse communiquer qu'avec ses voisines immédiates, permettant ainsi de réduire le nombre de dépendances entre chaque élément de l'infrastructure au strict

minimum. La communication entre deux couches adjacentes est alors assurée via une interface qui est un standard à respecter pour pouvoir transmettre des signaux. De cette manière, chaque couche de l'infrastructure peut ainsi être facilement corrigée, étendue, et même totalement remplacée tant que chaque standard de communication est respecté.

Une approche en couche est donc particulièrement adaptée dans le cas de la maison intelligente car elle répond très bien aux problématiques de perception et d'action des différents environnements intelligents. Pour ce qui est de la problématique de perception, les couches permettent d'abstraire progressivement l'information avant de la servir. Il s'agit bien ici de transformer des données de très bas niveau comme un ensemble de températures, mouvements, luminosités et consommations électriques en un ensemble d'informations de bien plus haut niveau comme les activités réalisées dans la maison. Cela en utilisant autant d'étapes que nécessaire afin de permettre, à terme, à des applications tierces de prendre des décisions en se basant sur des concepts aussi génériques que possible. Enfin, pour ce qui est de la problématique d'action, les couches successives permettent de transformer des ordres très abstraits comme *prépare la cuisine* en actions bien plus concrètes comme un jeu d'ordres à transmettre à un ensemble fini d'effecteurs respectant tous des standards de communication différents.

Finalement, le couplage lâche assuré par les modèles en couche permet aux maisons d'évoluer, car la configuration matérielle de celles-ci peut changer en tout temps. Soit en supprimant, ajoutant ou en remplaçant des capteurs ou effecteurs, en changeant de protocole de communication, en mettant à jour les algorithmes permettant de reconnaître ce qui se passe au sein de la maison ou en ajoutant des services à destination des occupants. Le modèle en couche est une solution permettant donc de s'assurer que la maison peut évoluer en même temps que la grande variété de technologies qui la compose tout en s'adaptant à la fluctuation des besoins auxquels elle répond dans le temps.

Il est d'ailleurs possible d'extraire des couches très générales que l'on peut retrouver dans toutes les architectures déjà présentées.

2.6.1 LA COUCHE MATÉRIELLE

La couche physique est systématiquement présentée comme la première couche de toutes les architectures étudiées. Rien de réellement étonnant à cela, car il ne peut y avoir de maison intelligente sans effecteurs, capteurs, routeurs, micro-ordinateurs et objets connectés en tout genre. Sans matériel au sein de la maison, il n'y a ni information recueillie, ni décision prise et encore moins d'actions entreprises en conséquence. Aussi, une fois tout ce matériel installé, rien ne garantit que l'infrastructure ne risque pas de changer de nouveau que ce soit en ajoutant du nouveau matériel, retirant du matériel obsolète ou encore en remplaçant du matériel existant.

Le problème n'est par ailleurs pas uniquement matériel, car une fois tous les appareils installés encore faut-il qu'ils communiquent ensemble. Et il existe de nombreuses solutions technologies pour cela : entre le Bluetooth, la variante *Bluetooth Low Energy* (BLE), les Maillages Zigbee, le protocole Z-Wave, le Wifi, les connexions câblées ou encore des protocoles spécialisés comme celui présenté en section 2.5.5, les variantes architecturales sont légion et certaines peuvent même utiliser plusieurs protocoles en même temps.

Finalement, une fois les choix matériels et réseau posés, il reste les nombreux problèmes d'interopérabilité à tous les niveaux, notamment mis en évidence dans l'article de Noura *et al.* (2019). Chaque appareil ayant ses propres prérequis en termes de protocole réseau, de format de données, de standard d'API, de sémantique, de système d'exploitation et de structure résultant des choix stratégiques des différents constructeurs, il n'existe alors pas de méthode évidente permettant de gérer tout nouveau matériel installé. Un choix architectural doit donc

être réalisé pour permettre aux logiciels dépendants du hardware de pouvoir communiquer de manière uniforme avec tout ce qui existe dans la maison peu importe leurs spécificités afin de ne pas devoir recoder l'ensemble des logiciels à chaque changement de configuration.

2.6.2 LA COUCHE SERVICE

Avec une couche physique fonctionnelle, il est possible de recevoir autant d'information que nécessaire. Ceci-dit, recevoir l'information et l'utiliser sont deux choses bien différentes, car il va bien falloir transformer les données reçues, stocker tous les résultats, et les servir à des applications tierces qui y trouveront une utilité.

On arrive donc à la, ou les, couches de services. Ces couches intermédiaires ont pour objectifs d'offrir tout le support nécessaire pour que les applications tierces, situées dans les couches supérieures de l'architecture, n'aient à s'occuper que des problématiques de très haut niveau.

Et ici, comme pour la couche physique, il existe de nombreuses solutions. Pour le stockage par exemple, la nature des bases de données peut largement varier avec des bases de données *Atomicity, Consistency, Isolation, Durability* (ACID) telles que MySQL, MariaDB, PostgreSQL ou SQLite, les bases de données NoSQL comme MongoDB ou CouchDB et des solutions plus spécialisées comme les bases de données pour séries temporelles comme InfluxDB. Et toutes ces solutions, venant évidemment avec leur panel d'avantages et d'inconvénients, ne sont pas l'unique problématique liée au stockage. Là encore, il est possible de n'avoir qu'une base unique dans l'architecture comme une multitude de bases dans l'objectif de distribuer la charge de calcul et de stockage sur plusieurs machines. Et pour chaque option, vient son propre protocole de communication.

Une fois la question du stockage résolue vient la question de la transformation des

données bas niveau en information de haut niveau. Cette transformation va être effectuée par différents algorithmes, eux aussi, sujets à des évolutions soit parce que des méthodes plus fiables ou plus rapides ont été conçues, soit pour de simples questions de maintenance. La nature même de ces calculs peut être le sujet d'un choix d'implémentation. Par exemple, il est possible de réaliser certains calculs à la demande tout comme il est possible de les optimiser via des techniques de mémoïzation, procédé présenté dans l'article de Michie (1968) visant à stocker les résultats en mémoires plutôt que de les recalculer à chaque demande. Avec plus de hauteur, l'agencement même des différents algorithmes de calcul peut aussi faire l'objet de considérations propres au génie logiciel permettant de ne pas devoir tout reconfigurer à l'ajout, au retrait ou à la modification d'un élément logiciel.

Finalement, vient la question du service. Comment présenter l'information stockée aux applications tierces pour leur faciliter la tâche ? Une fois de plus, le choix des standards peut devenir un réel casse-tête entre des solutions plus récentes comme REST ou GraphQL et des solutions plus anciennes comme SOAP telles que nous les avons présentées en Section 2.4. Et on ne parle pas de la multitude de standards permettant de représenter l'information décrite notamment en Section 2.2 comme XML, JSON, YAML ou les BLOB. À cela vient s'ajouter la nature des flux d'information : faut-il être systématiquement réactif, c'est-à-dire ne faire que répondre à des requêtes, ou être proactif et contacter les applications tierces pour les notifier d'événements importants en temps réel ? Et avec les flux temps réel, se présente tout un nouveau choix de solutions décrites notamment en Section 2.3.

Ainsi, les composants répondant aux problématiques consistant à manipuler la donnée pour des applications tierces sont des candidats parfaits pour produire une couche intermédiaire à l'ensemble des architectures de maisons intelligentes.

2.6.3 LA COUCHE APPLICATIVE

La couche applicative est systématiquement présentée comme la dernière couche de toutes les architectures étudiées. C'est un amoncellement de composants comprenant toutes les solutions qui constituent le de cerveau à la maison intelligente ou qui répondent à des besoins spécifiques de tierces parties comme, par exemple, le besoin de monitorer la santé d'un malade exprimé par un groupe de cliniciens.

Dans la plupart des architectures présentées, les outils de prise de décision se trouvent dans cette couche, souvent en conjonction avec des outils d'apprentissage profond permettant de détecter des concepts de très haut niveau. On y trouve aussi toutes les éventuelles interfaces graphiques permettant à des êtres humains de paramétrer la totalité du système que cela soit via des téléviseurs, des ordinateurs, des tablettes ou des téléphones.

La stabilité ainsi que la puissance de calcul de cette couche est d'abord assurée par l'interface qu'elle partage avec la couche de service qui est dans la plupart des cas une API suivant un standard spécifique.

2.7 CONCLUSION

Nous avons présenté dans cette section plusieurs problématiques clefs des architectures de maisons intelligentes incluant la représentation d'information en Section 2.2, les systèmes événementiels en Section 2.3 ainsi que les APIs en Section 2.4. Nous avons aussi étudié différentes architectures existantes telles que MavHome en Section 2.5.1, GatorTech en Section 2.5.2, CASAS en Section 2.5.3, OpenHAB en Section 2.5.4, le Light Node Communication Framework en Section 2.5.5 et WITS en Section 2.5.6.

Bien que toutes ces architectures aient été conçues avec des objectifs différents, nous

avons montré en Section 2.6 que toutes suivaient un modèle précis, le modèle en couche. Nous avons ensuite présenté les avantages de ce type d'architecture dans le domaine particulier des maisons intelligentes ainsi qu'une sélection de trois couches types que l'on peut aisément retrouver dans toutes les solutions étudiées en Section 2.5 : la couche physique, la couche de service et la couche applicative.

Pour chacune de ces couches types, nous avons présenté les grands enjeux et des choix de solutions architecturales. Cette analyse s'est finalement conclue par la couche applicative pour laquelle nous avons brièvement présenté des solutions qu'elle pouvait contenir. Il nous reste alors à tenter de comprendre quels algorithmes types peuvent être implémentés par-dessus une base d'architecture de maison intelligente ainsi que leurs objectifs, avantages et inconvénients.

CHAPITRE III
RECONNAISSANCE D'ACTIVITÉS

3.1 INTRODUCTION

Nous avons présenté, dans le second chapitre de cette thèse, des composants et des architectures de maisons intelligentes. Malgré l'hétérogénéité apparente des architectures présentées, nous avons vu qu'il était tout de même possible d'extraire un schéma générique de la stratégie adoptée par les différentes propositions existantes dans la littérature.

Nous savons donc comment installer des environnements intelligents pour produire de l'information. De cette manière, nous pouvons recevoir et stocker des données concernant les mouvements, la température, la luminosité, la consommation électrique, les prévisions météorologiques, la consommation d'eau et les interactions avec différents meubles. Nous pouvons aussi extraire et transformer cette information stockée en fonction des besoins.

Alors, que faire de cette ressource ? Que pouvons-nous installer dans la couche application du modèle présenté en Section 2.6 ? Dans les faits, quantité d'applications domotiques peuvent être installées au sein d'un système élémentaire de maison intelligente comme la gestion automatique des différentes lumières telles que décrit dans l'étude de Tang *et al.* (2017), l'optimisation de la consommation électrique telle que présentée dans l'article de Albuquerque *et al.* (2018), le monitoring et la gestion centralisée de la température de la maison comme montrés dans les travaux de Pallotta *et al.* (2008) ou encore la proposition de services multimédias tels qu'ils sont décrits dans l'article de Zhiwen Yu *et al.* (2006) et même un système de commande vocale comme dans le travail de Soda *et al.* (2012).

Mais il est possible d'aller plus loin dans l'assistance en ajoutant des systèmes permettant

de percevoir les enchaînements d'actions et les objectifs des différents acteurs présents dans l'environnement grâce au flux d'événements bas niveau produit par les installations présentées en Chapitre 2. Avec ce type d'information, il est alors possible de produire des tableaux de bord pour permettre aux cliniciens d'évaluer la progression de maladies dégénératives en suivant la perte progressive d'autonomie d'un individu. Plus encore, il est aussi possible de faire de la maison une forme d'orthèse cognitive permettant de pallier aux insuffisances détectées liées à une mémoire défaillante ou à des capacités motrices détériorées. En termes de projection, on pourrait même imaginer faire de cette maison un support visant à limiter la progression de la maladie.

Cependant, pour pouvoir envisager tous ces cas d'utilisation, il faut dans un premier temps mettre en place des systèmes informatiques capables de transformer l'information brute recueillie par l'environnement intelligent en information de plus haut niveau décrivant le comportement, les objectifs et les actions des habitants. Il faut aussi développer des modèles capables d'effectuer des projections de l'état de santé d'un individu dans l'avenir afin de définir des stratégies de soutien. On parle alors de mettre en place, dans cette dernière couche architecturale, un système de reconnaissance d'activités.

3.2 ACTIVITÉ ET RECONNAISSANCE D'ACTIVITÉS

Avant d'explorer la littérature, il faut tout d'abord établir ce qu'est la reconnaissance d'activités. Comme il n'existe pas à ce jour de consensus scientifique à ce sujet, nous tâcherons d'incorporer plusieurs points de vue différents quand cela sera nécessaire avant de poser la définition finale utilisée dans ce document.

3.2.1 ACTIVITÉ

Nous allons donc parler de *reconnaissance d'activités*, une notion qui peut être séparée en deux éléments comprenant l'action de *reconnaître*, d'une part, et l'objet de cette action, une *activité*, d'autre part. Avant de discuter de la nature de l'action en elle-même, il semble parfaitement légitime de définir la nature de l'objet que l'on souhaite étudier et donc poser la question, qu'est-ce une *activité* ?

Liu *et al.* (2016) et Chen & Nugent (2019) décrivent une activité comme une séquence d'*actions* exécutées dans le temps où les *actions* sont des éléments atomiques que l'on peut définir en fonction de la précision désirée. Ainsi, par exemple, l'activité *faire chauffer de l'eau* peut être vue comme la séquence d'actions *prendre un récipient* , *remplir le récipient d'eau* , *faire chauffer le récipient* , *attendre l'état d'ébullition* . La réalisation d'une séquence d'*actions* et donc d'une *activité* peut alors être *séquentielle* dans le cas où les actions se succèdent directement les unes aux autres, *concurrente* quand les actions sont réalisées au même moment ou finalement *intercalées* quand les actions de deux activités différentes s'entremêlent. Chaque activité peut aussi être catégorisée par le fait qu'elle est réalisée par un ou plusieurs acteurs.

Cook (2015) introduit dans sa définition une distinction entre les *instances* d'une activité et l'*activité* en elle-même. Ainsi, une *instance* est une séquence d'*actions* comme défini précédemment et une *activité* est une classe, c'est-à-dire un ensemble d'*instances* partageant des traits communs.

3.2.2 ACTIVITÉS DE LA VIE QUOTIDIENNE (AVQ)

Nous avons défini ce qu'était une activité. Il semble donc raisonnable de se demander quelles sont les activités importantes à étudier pour suivre l'état de santé d'une personne. À

cette fin, Équipe du Benjamin Rose Hospital (1959) a proposé, très tôt, une classification des activités quotidiennes permettant d'estimer la perte d'autonomie de personnes âgées : les Activités de la Vie Quotidienne (AVQ) ou *Activity of Daily Living* (ADL) en anglais. Les AVQ sont un ensemble réduit d'activités de la vie courante indispensable à la survie comprenant la capacité à *se laver*, à *s'habiller*, à *utiliser des toilettes*, à *s'asseoir* ou *se coucher* puis à *se relever seul*, *la continence* et, finalement, la capacité à *se nourrir*.

Au-delà de cette liste d'activités définie par Équipe du Benjamin Rose Hospital (1959), c'est surtout celle de Katz *et al.* (1963) qui est utilisée aujourd'hui. Selon Restrepo (1999), cette liste d'activités peut cependant faire l'objet d'adaptations d'un milieu hospitalier à l'autre. Ces adaptations respectent cependant toujours l'idée à la base de la liste originale. Les AVQ sont utilisées dans de nombreuses recherches médicales comme dans le cadre des travaux de Lehrner *et al.* (1999), Garrod *et al.* (2000), Doruk *et al.* (2004), Covinsky *et al.* (2006) ou encore de Wennie Huang *et al.* (2010).

Les AVQ ont fait l'objet de différents travaux de reconnaissance d'activités sur les vingt dernières années. Parmi ces travaux, on peut retrouver la proposition de Stikic *et al.* (2008) visant à reconnaître les AVQ en appliquant trois approches de classification (classifieur bayésien naïf, modèle de Markov caché et un algorithme de *boosting*) à des jeux de données extraits de puces *Radio Frequency IDentification* (RFID) et d'accéléromètres. On peut aussi citer la recherche de Bae (2014) visant à reconnaître des AVQ via une ontologie en se basant sur les données extraites de capteurs de mouvement, de capteurs de contact, de capteurs d'électricité et de capteurs de gaz. Il existe aussi l'étude de Roy *et al.* (2016) visant à reconnaître les AVQ via des téléphones intelligents dans un contexte multi-résident en utilisant des modèles de Markov cachés couplés. Finalement, on peut citer les travaux de Ercolano *et al.* (2017) proposant, cette fois, de reconnaître des AVQ via des méthodes d'apprentissage profond en se

basant sur des jeux de données capturées via des systèmes Kinect [20] et comprenant des flux vidéo RGB-D et un squelette simplifié des personnes réalisant chaque activité.

3.2.3 ACTIVITÉS INSTRUMENTALES DE LA VIE QUOTIDIENNE (AIVQ)

Les Activités Instrumentales de la Vie Quotidienne (AIVQ) ou *Instrumental Activity of Daily Living* (IADL) en anglais, sont un standard complémentaire des AVQ défini dans un article de Lawton & Brody (1969). Les AIVQ sont un ensemble spécifique d'activités complexes nécessitant une certaine planification et l'usage d'outils tiers. Les AIVQ ne sont pas, contrairement aux AVQ, indispensables à la survie d'une personne, cependant, les AIVQ décrivent des activités impliquant des interactions avec l'environnement d'une complexité juste supérieure à celle des AVQ. De manière générale, une dégradation des AIVQ tend à précéder une dégradation des AVQ comme expliqué dans l'article de Millán-Calenti *et al.* (2010).

Dans le travail original, les AIVQ englobent des activités comme la capacité à communiquer via le téléphone, à sortir faire des courses, à cuisiner, à maintenir sa maison en bon état, à entretenir ses vêtements, à se déplacer via des moyens complexes, à assurer une prise autonome de traitements médicaux et à assurer le bon état de ses finances. Les AIVQ sont évaluées via une échelle de mesure permettant une appréciation plus fine de l'autonomie réelle d'une personne.

Suivre les AIVQ permet de repérer une dégradation progressive de l'autonomie et des capacités cognitives d'un individu. Cependant, contrairement aux AVQ, les AIVQ sont bien plus complexes à définir et à tester. Certaines ne sont, par ailleurs, pas réellement observables au sein de la maison comme la capacité d'une personne à employer des moyens de transport

20. Pour plus d'information se référer à https://azure.microsoft.com/en-us/services/kinect-dk/

variés. Tout comme les AVQ, les AIVQ ont l'intérêt d'être un standard usité dans le milieu hospitalier et dans la recherche comme dans les études de Judge *et al.* (1996), de Spector & Fleishman (1998), de Baumgartner *et al.* (2004) et de Jefferson *et al.* (2006).

3.2.4 RECONNAISSANCE D'ACTIVITÉS

Nous avons, jusqu'à maintenant, traité de la notion d'activité en la définissant et en présentant des normes applicables dans notre contexte. Reste à définir ce qu'est la reconnaissance d'activités en tant que tel. Selon Kim *et al.* (2010), la reconnaissance d'activités est l'ensemble des techniques permettant d'extraire de l'observation d'un environnement, l'ensemble des activités humaines qui y sont réalisées. Les auteurs précisent, de plus, que le problème de reconnaître des activités en se basant sur des observations n'est pas trivial, notamment, de par le comportement des acteurs observés. En effet, ceux-ci ne réalisent pas un ensemble d'activités en séquence les unes après les autres, mais plutôt de manière concurrente, intercalée et ambigüe.

Tout d'abord, de manière concurrente, car une personne peut parfois réaliser plusieurs activités au même moment (ex., préparer à manger en regardant la télévision). On dit aussi de manière intercalée, car un acteur peut arrêter l'exécution d'une activité précise à un moment donné pour aller réaliser une activité tierce avant de revenir à son activité initiale (ex., une personne interrompt son repassage pour aller réceptionner un colis). Et finalement ambigüe, car certaines actions peuvent être interprétées de plusieurs manières différentes, avec, par exemple, le cas de l'action consistant à faire bouillir de l'eau, qui peut tout autant indiquer une volonté de faire cuire des pâtes, de préparer un thé ou de préparer une bouillotte. La problématique pouvant devenir plus complexe encore quand l'on tente de reconnaître des activités dans un milieu ou évolue plusieurs acteurs différents.

L'approche de Chen *et al.* (2012) étoffe un peu plus celle de Kim *et al.* (2010) en définissant la reconnaissance d'activités comme un processus complexe qui peut être grossièrement résumé en 4 grandes tâches comprenant la mise en place d'un système permettant d'observer le comportement d'un ou plusieurs acteurs, la création d'un système de gestion des données extraites via l'observation, la proposition d'un modèle permettant de décrire les activités, et, finalement, la mise en place d'outils permettant d'extraire les modèles proposés depuis les données stockées.

Chen & Nugent (2019) proposent, quant à eux, une classification de la reconnaissance d'activités basée sur deux divisions au niveau de la méthode d'observation et de la méthode de reconnaissance. Pour ce qui est de l'observation, cette classification sépare les solutions *basées sur la vision*, des méthodes *basées sur les capteurs* et, pour ce qui est des méthodes de reconnaissance, cette classification sépare les méthodes *basées sur les données* des méthodes *basées sur les connaissances*.

Ainsi, les méthodes d'observation *basées sur la vision* regroupent les solutions travaillant sur des flux vidéo via des outils d'analyse d'images comme dans le cadre de l'étude de Weiyao Lin *et al.* (2008) où les auteurs travaillent sur des flux de vidéosurveillance, ou comme dans l'article de Mo *et al.* (2016) où les auteurs utilisent les résultats d'un système Kinect de Microsoft.

Les méthodes d'observation *basées sur les capteurs* regroupent, quant à elles, des solutions qui reconnaissent des activités via des indices indirects capturés par un réseau d'appareils comme la température, le mouvement, la luminosité ou l'état des meubles. Dans cette seconde catégorie, on peut notamment citer l'article de Ronao & Cho (2016) utilisant les capteurs d'un téléphone intelligent pour reconnaître des activités, ou encore l'étude de Chen *et al.* (2010) où les auteurs utilisent les données émises par les capteurs d'une maison

intelligente pour extraire des activités.

Le groupe des méthodes *basées sur les données* englobe les solutions d'apprentissage supervisé ou d'apprentissage non supervisé comme dans le cadre de l'article de Liu *et al.* (2017) où les auteurs utilisent une adaptation du réseau bayésien pour reconnaître des activités telles que marcher, courir, prendre un café ou parler au téléphone via des données vidéo, ou comme dans l'étude de Novac *et al.* (2020) où les auteurs utilisent des réseaux de neurones pour reconnaître des activités comme marcher, s'asseoir ou monter/descendre des escaliers en utilisant des données accélérométriques.

Finalement, les méthodes *basées sur les connaissances* regroupent les solutions utilisant des modèles figés créées de toute pièce par des experts. L'étude de Civitarese *et al.* (2018) où les auteurs proposent de reconnaître des activités via une ontologie de la maison intelligente et un système dérivé des réseaux de Markov logique en est un bon exemple. Leur proposition est ensuite appliquée sur un jeu de données de CASAS comprenant différents capteurs de mouvement, capteurs de contact, capteurs de température, capteurs d'humidité et capteurs de luminosité. Il est aussi possible de citer l'article de Riboni & Murtas (2019) où les auteurs présentent une approche similaire mais, cette fois-ci, en produisant leur ontologie via de la reconnaissance d'images sur un jeu de données extrait d'internet.

Cette thèse vise donc principalement à effectuer de la reconnaissance d'activités dans les termes présentés jusqu'ici. De plus, si l'on se réfère à la classification proposée par Chen *et al.* (2012), l'approche proposée sera limitée aux méthodes *basées sur des données* extraites de capteurs tels que des capteurs de mouvement, des capteurs de contact, des capteurs de température, des capteurs de luminosité, des capteurs d'humidité et des capteurs électriques. Cette approche permet, notamment, de ne pas troubler la vie privée et le confort des participants en leur imposant un système beaucoup trop intrusif dû à la présence d'un ensemble de caméras

et de microphones. Dans la suite de ce document, les approches basées sur la reconnaissance d'images, de vidéos ou de sons seront ignorées.

3.3 MISE EN FORME DES DONNÉES

Nous avons défini ce qu'était la reconnaissance d'activités, et, avant de traiter des méthodes, il nous faut aborder le sujet des données.

3.3.1 SÉRIE TEMPORELLE

Que cela soit des actions, des températures, de la consommation électrique ou du mouvement, les données permettant de reconnaître des activités sont des observations faites par un système à un moment précis. Elles sont d'ailleurs principalement modélisées sous forme de séries temporelles qui sont, selon Chan (2010) et Shumway & Stoffer (2017), la représentation d'une séquence de valeurs indexées sur le temps.

L'utilisation des séries temporelles est très généralisée dans le cadre de la reconnaissance d'activités. Celles-ci sont notamment utilisées dans l'ensemble du livre de Cook (2015) et sont décrites comme étant la principale méthode de représentation des informations dans le contexte de la reconnaissance d'activités via des données capteurs dans le livre de Chen & Nugent (2019). De plus, c'est la modélisation que l'on retrouve principalement dans les articles publiés depuis les vingt dernières années que cela soit dans l'analyse des méthodes basées sur l'apprentissage profond de Wang *et al.* (2019), la reconnaissance de mouvement via des données accélérométriques de Shoaib *et al.* (2016) ou encore, la revue de littérature des méthodes basées sur les flux vidéo de Ke *et al.* (2013).

3.3.2 SEGMENTATION

Les algorithmes de reconnaissance d'activités s'appuient sur des séries temporelles représentant l'observation d'un milieu au cours du temps. Ces séries peuvent varier en longueur et si certaines représentent quelques secondes d'observations, d'autres peuvent durer plusieurs années. Les séries peuvent aussi varier en densité allant de quelques entrées par jour (p. ex., le nombre de sorties à l'extérieur du domicile en une journée) à plusieurs centaines d'informations par seconde (p. ex., données accélérométriques échantillonnées à un taux de 100 lectures par seconde).

Pour fonctionner, la plupart des algorithmes de reconnaissance d'activités n'utilisent pas la totalité d'un jeu de données extrait par un environnement intelligent mais plutôt différents morceaux prédécoupés de celui-ci. On parle alors de segmentation des données.

Selon Banos *et al.* (2014), les opérations de segmentation peuvent être catégorisées en trois différentes approches : les segmentations basées sur les activités, les segmentations basées sur les événements et les segmentations sous forme de fenêtres glissantes. Krishnan & Cook (2014) proposent, quant à eux, une catégorisation similaire, si ce n'est qu'ils ne font aucune distinction entre les segmentations basées sur les événements et les segmentations sous forme de fenêtres glissantes.

Dans la suite de cette section, nous adopterons la vision en deux blocs présentée dans l'article de Krishnan & Cook (2014). Nous exposerons donc dans un premier temps les méthodes visant à segmenter des données de manière « thématique », puis les méthodes visant à segmenter les données en utilisant une « taille » fixe.

SEGMENTATION PAR ACTIVITÉ

La segmentation par activité, telle que définie par Krishnan & Cook (2014) et Banos *et al.* (2014), est une découpe irrégulière d'une séquence d'événements en morceaux représentant chacun la réalisation d'une seule et unique activité. Le résultat de ce type de découpe peut alors être transféré à un algorithme de reconnaissance d'activités pour qu'il en évalue plus spécifiquement la nature.

Cook (2015) décrit plus en détails chaque approche possible et l'article de Banos *et al.* (2014) présente différentes implémentations. Ainsi, si l'on souhaite reconnaître un ensemble d'activités A, il est possible de découper une séquence d'événements :

— En utilisant les événements précis marquant le début et la fin spécifique de chaque activité à classer. Dans une telle situation, on doit alors apprendre à reconnaître $2 \times Card(A)$ états différents.

— En utilisant les transitions entre chaque activité, c'est-à-dire les couples fins/débuts plutôt que les débuts et les fins spécifiquement. Dans une telle situation, il faut alors être capable de distinguer $Card(A^2)$ situations possibles, c'est-à-dire une situation par couple d'activités existantes.

— En utilisant la totalité de l'activité réalisée, les divisions sont alors pratiquées quand l'algorithme n'est plus capable de choisir entre deux activités différentes. Dans ce cas précis, il faut alors produire un classeur à $Card(A)$ classes.

Bien que chaque approche soit assez spécifique, toutes relèvent, en réalité, de la détection de points de casse, c'est-à-dire d'instants spécifiques d'une séquence d'événements constituant l'observation d'un système où ledit système change d'état.

SEGMENTATION PAR FENÊTRAGE

La segmentation par fenêtrage telle que décrite dans l'article de Krishnan & Cook (2014) consiste en la décomposition d'une séquence d'événements en sous-séquences pouvant faire l'objet de superpositions appelées fenêtres.

Une des approches les plus naturelles consiste à créer des fenêtres de durée fixe comprenant, par exemple, tous les événements obtenus par intervalles de deux minutes. Ce type d'approche est très utilisé quand la séquence représente l'observation de capteurs qui émettent des événements à fréquence fixe comme pour les gyroscopes ou les accéléromètres car dans de telles situations, les sous-séquences obtenues sont régulières et comprennent le même nombre d'événements. C'est l'approche que l'on peut retrouver dans le travail de Li *et al.* (2016) où les auteurs utilisent une segmentation à durée fixe équivalente à la moitié de la durée de l'activité la plus courte parmi les activités à reconnaître. C'est aussi l'approche utilisée par Li *et al.* (2019) où les données sont segmentées par blocs de 6 secondes. Le résultat de cette segmentation sert ensuite de base à une opération d'agrégation visant à fusionner les segments représentatifs de la réalisation d'une même activité.

Une autre approche consiste à se baser sur des fenêtres comprenant un nombre d'événements fixe. Cette fois-ci, le problème est évidemment inversé, c'est-à-dire que toutes les sous-séquences obtenues sont garanties régulières mais ne représenteront pas la même durée d'observation. C'est l'approche employée dans l'article de Skocir *et al.* (2016) où des fenêtres comprenant les cinq derniers événements extraits de capteurs de mouvement et de capteurs de contact sont utilisées pour reconnaître des entrées et des sorties de pièces de la maison.

Finalement, une troisième méthode plus exotique, quant à elle, consiste à produire des fenêtres de tailles totalement variables en décidant du nombre d'événements à inclure sur

le moment. Cette approche est celle exploitée par la méthode proposée par Laguna *et al.* (2011) qui fait varier la taille des fenêtres en fonction de l'apparition de certains événements significatifs. C'est aussi la méthode employée dans les travaux de Noor *et al.* (2017) où la taille de la fenêtre varie en fonction de la probabilité qu'elle contienne une certaine activité.

Quoi qu'il en soit, le dimensionnement des fenêtres peut influencer les résultats obtenus par les algorithmes de classification en aval, que cela soit par le choix du nombre d'événements à inclure ou la durée à considérer. L'article de Banos *et al.* (2014) propose une observation empirique de ce phénomène et tente d'extraire des règles quant à son impact sur l'analyse des séries.

3.3.3 CARACTÉRISTIQUES DISCRIMINANTES

Les algorithmes de reconnaissance d'activités peuvent se baser sur différentes informations d'entrée pour en déduire l'activité de sortie. On appelle ces informations *caractéristiques*. Bien que les données extraites par les capteurs soient des caractéristiques, il peut être intéressant d'en calculer d'autres, comme décrit dans l'étude de Hassan *et al.* (2018) et comme précisé dans l'introduction de l'article de Sousa *et al.* (2017). De plus, dans le cadre d'une segmentation où la taille des segments varie d'un segment à l'autre, calculer certaines informations plus synthétiques permet de transférer une quantité de paramètres fixe aux algorithmes de reconnaissance, ce qui peut être un prérequis.

Dans le cadre de la présence d'un signal quelconque au sein d'un segment comme une température, une luminosité, une accélération, une consommation électrique, une vitesse ou même du mouvement, il peut être intéressant de remplacer ou de compléter ledit signal en utilisant des informations purement statistiques le concernant. Parmi les caractéristiques communément utilisées, on peut retrouver la valeur maximale, la valeur minimale, la moyenne

du signal, la somme, la médiane, les quartiles, les centiles, l'écart-type, l'écart moyen absolu, la variance, le nombre d'intersections avec la médiane, le coefficient d'asymétrie, le coefficient d'aplatissement, l'énergie, l'amplitude ainsi que le coefficient de corrélation Siirtola & Röning (2013), Hassan *et al.* (2018), Chen *et al.* (2018).

Dans le cas d'un signal numérique, il est possible de travailler avec la représentation fréquentielle du signal plutôt que sur le signal en lui-même. Cette représentation peut être obtenue en appliquant la transformée de Fourier rapide, ou la *Fast Fourier Transform* (FFT) en anglais, qui assigne à tout signal $S = (s_0, s_1, ..., s_{n-1}), s_i \in \mathbb{R}, i = 0, 1, ..., n-1.$, une série de fréquences $F = (f_0, f_1, ..., f_{n-1}), f_j \in \mathbb{C}, j = 0, 1, ..., n-1$. Parmi les caractéristiques communément utilisées, on peut retrouver le centre de masse, l'énergie et l'entropie Xia *et al.* (2018), Ni *et al.* (2019), Wang & Liu (2020).

Finalement, comme précisé dans l'article de Rosati *et al.* (2018), la transformée en ondelettes discrètes peut aussi servir de source pour produire des caractéristiques discriminantes basées sur la représentation temporelle-fréquentielle de l'information. Cette représentation se distingue de la transformée de Fourier en produisant des coefficients encodant à la fois les informations temporelles et spectrales du signal traité.

SACS DE MOTS, SACS DE CAPTEURS

Les sacs de mots sont des structures initialement utilisées dans la reconnaissance de texte. L'article de Salton *et al.* (1975) où les auteurs utilisent des vecteurs de nombres réels dont chaque dimension apprécie la proéminence d'un mot particulier au sein d'un document via un poids entre 0 et 1 est une des premières apparitions de cette idée dans le cadre de la classification de document. Ce type d'approche permet notamment de calculer la similarité entre différents documents. La même méthode peut être utilisée en comptant des mots plutôt

qu'en usant de poids réels, comme précisé dans la définition du livre de Manning *et al.* (2008).

En dehors du cadre de la reconnaissance de texte, les sacs de mots peuvent aussi trouver des applications dans le cadre de la reconnaissance d'activités comme dans l'approche proposée par Bettadapura *et al.* (2013) utilisant des sacs de mots pour représenter des séquences d'événements et leur chronologie. L'article cite d'ailleurs plusieurs autres travaux employant de telles structures pour réaliser la reconnaissance d'activités.

Les travaux de Crandall & Cook (2010) et de Malazi & Davari (2018), utilisent, quant à eux, des sacs de mots pour représenter les différents capteurs actifs dans une période de temps donnée ainsi que le nombre d'événements qu'ils ont émis. Une telle représentation est appelée par les auteurs un sac de capteurs et peut être une approche utile pour transformer un maillage de capteurs de mouvement en carte de densité résumant les différents déplacements d'un individu sur une période de temps donnée.

3.4 APPRENTISSAGE SUPERVISÉ

L'apprentissage supervisé (Mitchell, 2004) est une technique visant à approcher une certaine fonction mathématique inconnue mettant en relation un ensemble d'entrées et un ensemble de sorties par l'utilisation d'une collection d'exemples de couples entrées / sorties.

On peut dire d'une estimation d'une fonction qu'elle généralise bien cette fonction si elle prédit correctement les sorties de couples entrées/sorties n'appartenant pas à l'ensemble d'apprentissage utilisé pour produire l'estimation Cao *et al.* (2007). Aussi, pour optimiser un modèle, on utilise une fonction de coût qu'il faut minimiser. La fonction de coût peut varier d'un problème à l'autre mais toutes ont pour rôle d'estimer, directement ou indirectement, la quantité d'erreur entre la prédiction d'une estimation et la sortie réelle attendue.

On dit d'un algorithme d'apprentissage supervisé qu'il est un problème de classification s'il doit évaluer approximativement une fonction qui prend des valeurs dans un ensemble discret de sorties appelé des classes Santafe *et al.* (2015).

Un problème de classification peut être un problème de classification binaire si la fonction dont on cherche une approximation ne peut prendre que deux valeurs de sortie distinctes Japkowicz (2001). De plus, on dira d'un algorithme d'apprentissage supervisé qu'il est un problème de régression quand il s'agit de formuler approximativement une fonction prenant des valeurs sur un ensemble continu de sorties. L'article de Qiu *et al.* (2014), est un bon exemple de problème de régression car les auteurs y présentent des modèles pour prédire le comportement de séries temporelles variées décrivant des mécanismes biologiques, la demande électrique, le prix de biens immobiliers ou les résultats de modèles mathématiques tiers.

Dans le cadre de la reconnaissance d'activités, il est plus courant de devoir s'attaquer à des problèmes de classification, c'est-à-dire des problèmes où l'on cherche à faire correspondre des observations avec un nombre fini de classes appelées activités. Dans la suite de cette section, nous présenterons plusieurs exemples d'algorithmes d'apprentissage supervisé usuels dans le cadre de la reconnaissance d'activités.

3.4.1 CLASSIFIEUR BAYÉSIEN NAÏF

Le classifieur bayésien naïf est un algorithme d'apprentissage machine supervisé basé sur une modélisation probabiliste du problème de classification utilisant le théorème de Bayes (1763). C'est un modèle que l'on peut retrouver dans la littérature comme dans les travaux de Lewis (1998), de Rish (2001) et de Martinez-Arroyo & Sucar (2006). Le théorème de Bayes est un résultat de base en théorie des probabilités et constitue la pierre d'assise de l'inférence

bayésienne qui s'est montrée très efficace en intelligence artificielle. Le théorème s'énonce comme suit :

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}, \quad P(B) \neq 0, \tag{3.1}$$

Où $P(A \mid B)$ désigne la probabilité conditionnelle de l'événement A sachant que l'événement B s'est réalisé.

Dans le cadre de la reconnaissance d'activités, l'idée derrière le classifieur bayésien naïf est d'utiliser le théorème de Bayes (3.1) pour estimer la probabilité qu'une classe particulière d'activité soit attribuée à un instant donné sachant qu'un ensemble d'observations s'est réalisé. C'est l'approche que l'on retrouve dans les articles de T. van Kasteren & Krose (2007), de Zappi *et al.* (2007) et de Sarkar *et al.* (2010).

Formellement, considérons un ensemble de n caractéristiques discriminantes $X = \{X_1, X_2, ..., X_n\}$, où $X_1, X_2, ..., X_n$ sont des variables aléatoires représentant, par exemple, la température d'une cuisine, la présence d'une personne dans un salon ou la consommation électrique de la maison. Alors, si on observe, à un moment précis, l'événement $x = (x_1, x_2, ..., x_n)$, où les x_i sont les valeurs prises par les variables aléatoires X_i à ce moment par les capteurs (chaque variable aléatoire étant associée à un capteur) au sein d'un environnement intelligent, il est possible d'adopter une approche probabiliste pour évaluer quelle activité a la plus grande probabilité d'avoir lieu parmi un ensemble de k activités $A = \{a_1, a_2, ..., a_k\}$ en définissant une variable aléatoire Y prenant des valeurs sur A et en évaluant les probabilités suivantes :

$$P(Y = a_j \mid x) = P(Y = a_j \mid X_1 = x_1 \cap X_2 = x_2 \cap ... \cap X_n = x_n), \quad j = 1, 2, ..., k. \tag{3.2}$$

Comme précisé par Lewis (1998), l'algorithme de classification bayésien naïf statue que chaque caractéristique utilisée est conditionnellement indépendante de toutes les autres pour chaque activité. Cette assertion est bien évidemment erronée à un degré variable pour la plupart des jeux de données, mais elle permet cependant d'évaluer l'expression (3.2) de la manière suivante :

$$P(Y = a_j \mid x) = \frac{P(Y = a_j) \cdot \prod_{i=1}^{n} P(X_i = x_i \mid Y = a_j)}{P(x)}, \quad j = 1, 2, ..., k. \tag{3.3}$$

Cette dernière formule peut alors être estimée grossièrement en dénombrant les différents couples d'entrées/sorties d'un ensemble d'apprentissage en fonction de leurs caractéristiques. Mieux encore, comme le fait remarquer Domingos & Pazzani (1997), seul le numérateur est important à calculer dans le cas où l'on souhaite obtenir l'activité la plus probable, car le dénominateur est une constante.

Comme présenté par Domingos & Pazzani (1996), le classifieur bayésien naïf est un algorithme simple à mettre en oeuvre donnant d'assez bons résultats en pratique malgré les assertions qu'il fait sur la nature des données traitées (l'hypothèse d'indépendance entre les observations).

Dans le cadre de la reconnaissance d'activités, le classifieur bayésien naïf peut être retrouvé dans différentes études. Dans l'article de Singh *et al.* (2017), il est utilisé comme base de comparaison pour estimer les performances d'un algorithme tier dans le cadre de la reconnaissance d'AVQ au sein de trois maisons intelligentes équipées de capteurs de mouvement, de capteurs infrarouges et de capteurs de niveau d'eau. La classification bayésienne naïve donne d'assez bons résultats en comparaison des autres méthodes testées avec une justesse variant de 46% à 95% en fonction des jeux de donnés testés et de la nature de l'opération de

reconnaissance.

Shoaib *et al.* (2016) utilisent le classifieur bayésien naïf parmi d'autres solutions pour reconnaître des activités comme fumer, monter des escaliers, s'asseoir, rester debout ou faire du vélo via l'accéléromètre et le gyroscope de différents téléphones intelligents. Contrairement à l'étude de Singh *et al.* (2017), l'étude de Shoaib *et al.* (2016) se concentre principalement sur les conséquences que peuvent avoir le choix des capteurs ou du type de segmentation sur les résultats d'une reconnaissance d'activités.

Dans l'article de Sarkar *et al.* (2010), le classifieur bayésien naïf est utilisé pour détecter des activités comme manger, dormir ou se doucher. Les auteurs proposent dans leur étude une version modifiée de l'algorithme permettant de gérer les problèmes liés aux probabilités conditionnelles nulles qui résultent de l'absence d'un certain couple caractéristique discriminante/activité durant l'entraînement de l'algorithme. Étant donné que l'estimation du modèle final est le résultat d'un produit comme montré par l'équation (3.3), l'apparition d'un quelconque terme nul entraîne systématiquement une estimation finale nulle et empêche donc de classer efficacement certaines activités. Pour résoudre ce problème, les auteurs proposent de compléter l'estimation de chaque probabilité conditionnelle en y ajoutant la probabilité qu'une certaine caractéristique apparaisse tout court dans l'ensemble d'entraînement, que cela soit en faisant une interpolation linéaire entre les deux termes ou en utilisant une distribution de Dirichlet. Les auteurs présentent une comparaison avec un classifieur bayésien sans lissage dans le cadre de trois jeux de données différents, la version lissée du modèle est d'une bien meilleure justesse dans les trois cas testés avec un gain pouvant aller jusqu'au triple de la justesse originale.

Dans l'article de Zappi *et al.* (2007), la méthode bayésienne naïve est utilisée pour agréger les résultats de plusieurs algorithmes de reconnaissance d'activités travaillant sur des

données extraites d'accéléromètres portés à même le corps afin d'améliorer la qualité de la reconnaissance finale. Les auteurs comparent dans leur article la méthode bayésienne naïve à une fusion au vote majoritaire dans le cadre de la reconnaissance d'activités de l'industrie automobile telles que *ouvrir/fermer le capot* , *ouvrir les portières* , *vérifier l'état du volant* ou *vérifier la bonne fermeture du coffre* . L'approche bayésienne surpasse le vote majoritaire dans le cas où le nombre d'algorithmes à fusionner est faible. Plus le nombre d'algorithmes à fusionner est important, plus la différence de performance entre les deux méthodes de fusion est faible. Les auteurs expliquent cette différence principalement par le fait que la fusion bayésienne naïve pondère les résultats de chaque algorithme et met naturellement en exergue les algorithmes les plus performants là où le vote majoritaire accorde la même importance à toutes les voix.

Finalement, dans l'article de Nef *et al.* (2015) le classifieur bayésien naïf est utilisé avec d'autres algorithmes pour reconnaître des AVQ dans le contexte d'une maison intelligente. Les résultats du classifieur bayésien naïf restent cependant mitigés en comparaison des autres algorithmes testés, avec un F-score inférieur à 30% pour la majorité des activités reconnues dans l'article.

3.4.2 MODÈLE DE MARKOV CACHÉ

Le *Hidden Markov Model* (HMM) est un modèle probabiliste d'apprentissage machine supervisé que l'on peut retrouver dans divers travaux de la littérature comme dans les articles de Rabiner (1989), de Zia *et al.* (2011) et de Nguyen (2018). C'est un modèle qui décrit l'évolution d'un système dans le temps en utilisant deux processus stochastiques : un processus qui représente l'évolution de l'état interne du système modélisé et un processus représentant le comportement extérieur du même système.

Dans le cadre de la reconnaissance d'activités, le processus interne d'un HMM modélise l'enchaînement des activités réalisées dans l'environnement. C'est un processus qui n'est pas directement observable, et donc, l'inconnue de notre système. Le processus externe quant-à-lui décrit les seuls éléments observables de la réalisation des activités, c'est-à-dire l'activation des différents capteurs dans l'environnement.

Le processus stochastique caché, décrivant l'enchaînement des activités à reconnaître, est une chaîne de Markov (1906), c'est-à-dire que le modèle suppose, par hypothèse, que la réalisation d'une certaine activité ne dépend que de l'activité qui l'a précédé et pas de toute ou partie des activités déjà réalisées.

L'observation des activités du processus caché forme alors le second processus stochastique, c'est-à-dire l'enchaînement des activations de capteurs. L'objectif consiste à trouver l'enchaînement d'activités le plus probable ayant amené à une certaine observation.

Plus formellement, un HMM est défini par un ensemble d'éléments $\{S, K, \Pi, A, B\}$. Parmi ces éléments, S, est l'ensemble des états que peut prendre le processus stochastique caché, c'est-à-dire, dans notre contexte, des activités. K est l'ensemble des états que peut prendre le processus stochastique observable, c'est donc, l'ensemble des états que peuvent prendre les capteurs observant l'environnement. Π est une loi de probabilité définie sur S indiquant les chances que le système démarre par la réalisation d'une certaine activité. A est une matrice définissant pour tout couple d'activité $(s_1, s_2) \in S^2$ la probabilité que l'activité s_2 succède à l'activité s_1. Finalement, B, est une matrice définissant pour tout couple d'activité / observation $(s, k) \in S \times K$ la probabilité que l'activité s produise l'observation k.

De cette manière, dans l'étude de Zia *et al.* (2011), un modèle de HMM est proposé pour déduire quel appareil électrique est en cours d'utilisation en fonction de l'évolution de la consommation électrique du réseau d'un habitat intelligent.

Ronao & Cho (2014), quant à eux, vont mettre en relation plusieurs HMM pour reconnaître des activités telles que *marcher* , *monter des escaliers* ou *s'asseoir* en utilisant les données de gyroscopes et d'accéléromètres. Les auteurs utilisent ainsi trois modèles différents : un modèle générique et deux modèles spécialisés. Le modèle générique classe une première fois la séquence d'observations extraite des capteurs, et en fonction de ce premier résultat, la séquence d'observations est ensuite redirigée vers le modèle le plus apte à poser un diagnostic final. Lors des tests, la solution proposée par les auteurs identifie les activités réalisées avec une justesse variant de 85% à 95% en fonction de la classe d'activité considérée.

L'étude de Lee & Cho (2011) propose, elle, de reconnaître des activités comme *se déplacer* , *prendre le bus* ou *magasiner* en utilisant les capteurs d'un téléphone. Dans ce cas-ci, les auteurs utilisent deux HMM mais en séquence. Le premier modèle va transformer la séquence d'observations des capteurs en une séquence d'actions simples comme *se déplacer* , *marcher* , *courir* ou *monter des escaliers* , puis le second modèle va transformer cette séquence d'actions simples en activités plus complexes comme *magasiner* , *prendre le bus* ou *se promener* . En quelque sorte, dans cette étude, l'évolution de l'état interne du premier modèle sert de séquence d'observations pour le second modèle. Lors des tests, l'approche proposée par les auteurs reconnaît correctement les activités *se déplacer* et *prendre le bus* dans 100% des cas. Cependant, pour ce qui est de l'activité *magasiner* , l'approche proposé ne reconnaît que 50% des étiquettes. Dans l'ensemble, le HMM hiérarchique est plus précis que le HMM standard pour la détection des activités de haut niveau.

Les travaux de Nguyen *et al.* (2005) et de Asghari *et al.* (2020) reconnaissent des activités complexes telles que *manger* , *se laver* , *cuisiner* ou encore *prendre des médicaments* en utilisant une variante des HMM appelée HMM hiérarchique. Cette variante modélise l'évolution du comportement interne du système à résoudre sous la forme d'une arborescence plutôt que d'une simple séquence. Les états d'un HMM hiérarchique peuvent ainsi contenir

d'autres HMM hiérarchiques jusqu'à ne contenir, pour les feuilles de l'arborescence, qu'un HMM standard. Dans ce type de modélisation, quand un système entre dans un nouvel état, on descend alors dans l'arborescence et on résout le modèle que cet état contient, jusqu'à ce que le modèle fils entre dans un état terminal, indiquant qu'il faut alors remonter de nouveau l'arborescence. Dans un HMM hiérarchique, seuls les états des modèles feuilles de l'arborescence produisent le comportement extérieur du système. Ce type de modèle a l'avantage d'exprimer la notion de hiérarchie dans la composition de certaines activités, plus précisément, chaque étage de l'arbre représente un nouveau niveau d'abstraction, allant des actions simples au plus bas de l'arbre jusqu'aux activités plus générales au plus haut de l'arbre. Ce type de modèle est cependant bien plus complexe à résoudre qu'un HMM. L'article de Nguyen *et al.* (2005) compare différentes implémentations de HMM hiérarchique. Dans l'ensemble, les implémentations reconnaissent les types d'activités avec une justesse qui varie de 90% à 100% sur les jeux de données testés. L'article se focalise cependant plus sur l'évaluation à la fois de la capacité des modèles proposés à détecter le plus tôt possible les activités, et aussi leur capacité à évaluer correctement la durée de chaque activité. Le modèle proposé par Asghari *et al.* (2020) a été validé sur deux jeux de données et reconnaît les activités testées avec une justesse de 65% sur le premier jeu et une justesse de 60% sur le second. Les résultats sont obtenus dans le cadre d'une application en temps réel du modèle.

Finalement, les études de Brand *et al.* (1997) et de Natarajan & Nevatia (2007) utilisent des HMM couplés pour reconnaître des activités. Un HMM couplé est une variante de HMM où plusieurs modèles évoluent en parallèle et où l'état d'un modèle à un instant particulier dépend de son état précédent et de l'état précédent de tous les modèles qui lui sont couplés. Ce type de modélisation peut être pratique pour reconnaître des activités qui s'influencent les unes les autres, comme la gestuelle des mains par exemple. L'article de Natarajan & Nevatia (2007) étend cette approche en présentant les semi-HMM, qui sont des variations qui prennent

en compte la durée pendant laquelle le système reste dans un état précis.

Dans l'ensemble les HMM donnent d'assez bons résultats tout en offrant l'option de pouvoir travailler en temps réel. Aussi, les HMM proposent une modélisation concrète de la reconnaissance d'activités à la fois en termes de résultats attendus mais aussi en termes d'infrastructure déployée pour la reconnaissance, c'est-à-dire en représentant bien la problématique de reconnaissance comme la tentative d'un ensemble de capteurs d'observer indirectement la réalisation d'activités au sein d'un environnement. Finalement, les HMM peuvent être adaptés à différentes situations en utilisant des variations telles que les HMM hiérarchiques ou les semi-HMM. Malheureusement, comme présenté dans le travail de Rabiner (1989), les HMM ne sont pas simple à appréhender, et l'article de San-Segundo *et al.* (2016) montre bien la complexité qu'induit la mise en place d'un HMM notamment dans les étapes préparatoires nécessaires à la détermination des probabilités requises par l'opération de résolution de ce type de solution. Cette complexité initiale augmente, par ailleurs, d'autant plus quand l'on souhaite mettre en place une des nombreuses variations de HMM.

3.4.3 MACHINES À VECTEURS DE SUPPORT

Les *Support Vector Machine* (SVM) sont un ensemble de techniques d'apprentissage supervisé, présentées par Vapnik & Chervonenkis (1974) pouvant être utilisées à la fois pour résoudre des problèmes de régression et de classification. Comme nous nous intéressons à la reconnaissance d'activités dans cette thèse, nous nous limiterons à l'aspect classification.

Les SVM sont des modèles qui suivent le principe de *Minimisation du Risque Structurel* présenté dans les articles de Vapnik & Chervonenkis (1968) et de Vapnik (1982) dont l'objectif est de définir formellement la capacité de généralisation d'une configuration obtenue via l'apprentissage supervisé. Pour y parvenir, les auteurs proposent de séparer le taux d'erreur

d'un modèle en deux : l'erreur empirique observée en testant le modèle sur un jeu de données particulier, et la confiance que l'on peut avoir sur sa capacité à fonctionner sur des données nouvelles sous la forme d'une probabilité qui est fonction du nombre d'exemples utilisés lors du test empirique et de la capacité du modèle à discriminer correctement une quantité arbitraire de situations.

Les SVM répondent à cette contrainte de minimisation dans le cadre mathématique des problèmes de classification binaire linéairement séparables Chowdhary *et al.* (2020). Ce sont des problèmes où l'on cherche à discriminer deux groupes de points dans un espace réel dans lequel il est possible de définir clairement une frontière sous la forme d'une droite, d'un plan ou d'un hyperplan entre les instances de chaque groupe. Cette frontière sépare alors deux volumes contenant les points de chaque classe, la classification se faisant alors en déterminant à quel côté de la frontière chaque instance appartient. Pour répondre aux contraintes du principe de *Minimisation du Risque Structurel*, les SVM cherchent alors à trouver une frontière unique : la droite, le plan ou l'hyperplan maximisant la marge entre les vecteurs supports, c'est-à-dire, l'espace vide qui sépare la frontière des instances d'apprentissage des deux groupes à classer qui lui sont les plus proches. En conséquence, si l'ensemble des vecteurs de support change, la solution qui en résulte a toutes ses chances de changer elle aussi.

D'un point de vue plus formel, supposons que l'on est dans un espace de dimension n et que l'on cherche à classer un ensemble d'instances $X \in \mathbb{R}^n$, représentées par des points dans cet espace, en deux groupes de points ayant pour étiquette $Y = \{y_1, y_2\}$. Avec $w \cdot x$ le produit scalaire entre w et x, l'objectif d'un algorithme cherchant à résoudre un modèle de machine à vecteurs de support consiste alors à déterminer les paramètres $w \in \mathbb{R}^n$ et $b \in \mathbb{R}$ de l'hyperplan H défini par l'expression suivante :

$$H = \{x \in \mathbb{R}^n : w \cdot x - b = 0\}, \tag{3.4}$$

De telle manière que pour tout exemple d'instance $e \in X \times Y, e = (x, y)$:

$$\begin{cases} y = y_1 \Leftrightarrow w \cdot x - b \leq -1 \\ y = y_2 \Leftrightarrow w \cdot x - b \geq 1, \end{cases} \tag{3.5}$$

Et où la marge est maximale. C'est-à-dire, trouver l'unique hyperplan H défini par les valeurs $w \in \mathbb{R}^n, b \in \mathbb{R}$ telles que la valeur $\frac{2}{\|w\|}$ soit maximale, ce qui est équivalent à trouver ce même hyperplan tel que la valeur $\frac{\|w\|}{2}$ soit minimale ($\|w\|$ désigne la norme du vecteur w). Il est aussi possible, comme dans la description faite par Stoean & Stoean (2014), de minimiser $\frac{\|w\|^2}{2}$ pour augmenter la vitesse de convergence et minimiser la quantité de calcul.

Les vecteurs *support* sont donc les points les plus proches de l'hyperplan ainsi obtenu. Ce sont eux qui déterminent la solution, et si les vecteurs support changent il est très probable que la solution change elle aussi.

Dans l'étude de Yin *et al.* (2008), les auteurs utilisent les SVM dans le cadre de la détection d'activités *anormales* via des accéléromètres, des magnétomètres, des capteurs de luminosité, des capteurs de température et des micros. Le modèle sert principalement à distinguer les instances d'activités faciles à classer des activités qui ne le sont pas afin de passer les instances complexes à des algorithmes de détection spécialisés. Dans cette étude, chaque composante des points à diviser est le résultat de l'application d'un HMM entraîné à reconnaître un certain type d'activité *normale* . Après que l'ensemble des points soit calculé, la SVM est optimisée en utilisant un algorithme standard plus amplement décrit dans l'article de Schölkopf *et al.* (2001) afin de pouvoir trier les activités *normales* des activités qui ne le sont pas. L'article montre que ce type d'approche, quand elle est utilisée seule, ne détecte

l'ensemble des activités anormales qu'aux prix d'un nombre de faux positifs important. Les auteurs utilisent donc la SVM pour effectuer un premier filtrage grossier permettant d'entraîner d'autres modèles qui viendront améliorer la classification initiale.

Dans l'article de He & Jin (2009), les auteurs utilisent les SVM pour classifier différentes activités comme *courir* , *marcher* ou *sauter* via des accéléromètres. Pour y parvenir, les signaux émis par les capteurs sont décomposés en séquences de coefficients via une transformée en cosinus discrète. Le nombre de coefficients est ensuite réduit au maximum par le biais d'une analyse en composantes principales. Les vecteurs qui résultent de cette opération sont utilisés pour entraîner des SVM ou comme inconnue à classer. Comme une SVM ne peut effectuer qu'une classification binaire, les auteurs ont employé dans leur étude une stratégie de classification *un contre tous* . Ce type de stratégie permet d'utiliser plusieurs classeurs binaires pour gérer des problèmes de classification à plusieurs classes. Plus précisément, dans l'article, la classification s'effectue en construisant un classeur par paires de classes détectable et en adoptant la classe ayant cumulée le plus de confiance de la part de tous les classeurs. Sur le jeu de donné testé, avec la méthode proposée et l'utilisation des SVM, la méthode classifie les différentes instances avec une justesse maximale de 97.51%.

L'article de Anguita *et al.* (2012) traite d'une problématique similaire à celle de He & Jin (2009) mais cette fois en utilisant les capteurs d'un téléphone intelligent. Les auteurs utilisent l' *astuce du noyau* qui est une technique qui permet d'améliorer la qualité d'un classeur faible en se basant sur le théorème de Cover (1965) qui énonce que tout problème de classification non-linéairement séparable dans un espace donné a plus de chance de l'être dans un espace de dimension supérieure s'il y est projeté par une transformation non-linéaire. Dans le cadre de l'étude de Anguita *et al.* (2012), les auteurs proposent d'utiliser un noyau Laplacien plutôt qu'un noyau Gaussien afin d'optimiser la consommation de l'algorithme en ressources. L'étude motive ce choix par la facilité de calcul du noyau Laplacien sur du

matériel embarqué via des opérations de décalage de bits. D'autres modifications sont aussi apportées au modèle pour pouvoir réaliser une optimisation arithmétique à virgule fixe plutôt qu'en utilisant des nombres flottants. Pour une légère dégradation de la justesse moyenne de 89.3% à 89.0%, les auteurs effectuent leurs calculs sur des valeurs encodées sur 8 bits en lieu de 64 bits originellement. Ce changement permet de réduire grandement l'impact mémoire, et d'autant plus dans une étude où une stratégie *un contre tous* est utilisée. En effet, dans ce cas précis, la réduction de mémoire s'applique à un ensemble de classeurs réduisant ainsi une allocation supposée de 2,048 bits en une allocation de 256 bits.

Finalement, l'étude de Abidine *et al.* (2018) propose d'utiliser les SVM pour reconnaître des AVQ incluant notamment des activités telles que *manger* , *dormir* , *se laver* ou *cuisiner* . Pour ce faire, les auteurs utilisent les données extraites de capteurs de contacts, de capteurs de mouvement et de capteurs d'humidité en les transformant en vecteurs de nombres réels dont les composants représentent chacun différentes caractéristiques discriminatives de signaux telles que présentées en Section 3.3.3. L'article décrit l'utilisation d'une variante des SVM appelée SVM Pondérée dont l'objectif est de pallier à des problématiques qui naissent de l'entraînement d'un classifieur à distinguer deux classes avec un déséquilibre important du nombre d'instances d'entraînement qui les représente. Dans ce type de situation, la classe sous-représentée est sujette à une dégradation de la qualité de la classification, car il devient moins coûteux lors de l'optimisation du modèle de se tromper dans la classification de la classe sous-représentée que dans la classification de la classe surreprésentée. Les SVM Pondérées sont une adaptation du modèle classique qui accroissent l'importance de la classe sous-représentée dans le calcul de la fonction de coût utilisée pour optimiser le modèle. Une fois encore, la stratégie consistant à utiliser plusieurs classifieurs binaires pour chaque paire de classes est privilégiée, et les auteurs utilisent aussi l' *astuce du noyau* . L'article compare l'approche proposée avec 7 modèles sur 4 jeux de données différents. Dans l'ensemble, les auteurs obtiennent des résultats

équivalents aux autres approches et, une F-mesure toujours supérieure d'environ 10% à celle d'une SVM standard.

3.4.4 ARBRES DE DÉCISION

L'arbre de décision est un modèle d'apprentissage supervisé pouvant être utilisé à la fois pour résoudre des problèmes de régression et de classification. C'est le modèle utilisé dans les travaux de Du & Zhan (2002), Huang & Townshend (2003), Hssina *et al.* (2014) et Song & Ying (2015). Comme nous nous intéressons à la reconnaissance d'activités dans cette thèse, nous nous limiterons à l'aspect classification.

Un arbre de décision est un arbre où chaque nœud est associé à une collection d'instances d'apprentissage et où chaque relation est associée à un prédicat permettant d'extraire de la collection d'instances du nœud parent les instances appartenant à la collection du nœud enfant. La Figure 3.1 représente ce type de structure.

La classification d'une instance en utilisant un arbre de décision se réalise de la manière suivante : on commence par évaluer les prédicats menant du nœud actuel (en commençant par la racine de l'arbre) à tous les nœuds enfants directs en utilisant l'instance à classer. Quand l'instance à classer satisfait un prédicat, on réitère le processus décrit précédemment en se basant cette fois sur le nœud fils pointé par la relation associée au prédicat évalué. En procédant de cette manière, on finit toujours par atteindre une feuille de l'arbre. Il suffit alors d'attribuer à l'instance inconnue, la classe majoritaire dans la collection d'instances d'apprentissage associée à la feuille.

La difficulté dans cet exercice réside donc principalement dans la construction de l'arbre de décision en question. Plusieurs algorithmes permettant de résoudre ce problème

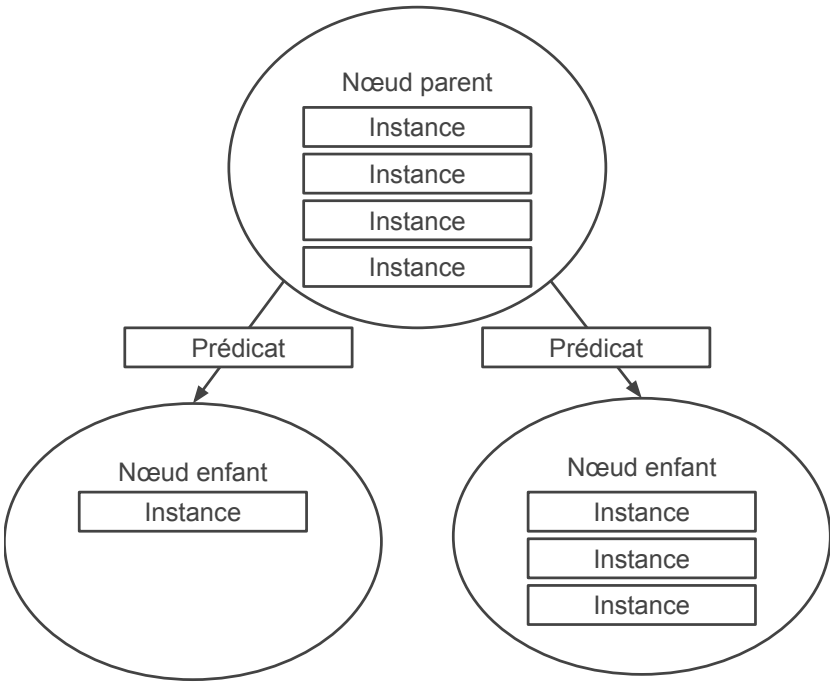

Figure 3.1 : Schéma représentant une sélection de nœuds d'un arbre de décision.

ont par ailleurs déjà été proposés dans la littérature dont notamment l'algorithme Iterative Dichotomiser 3 *(ID3)* de Quinlan (1986), son extension *C4.5* de Quinlan (1993) ainsi que l'algorithme Classification And Regression Tree *(CART)* de Breiman *et al.* (1984).

Ces trois solutions varient principalement dans la manière d'optimiser les nœuds et dans les types d'attributs supportés. Dans leur livre, Han *et al.* (2011), décrivent ID3, C4.5 et CART comme des algorithmes gloutons produisant les arbres de la racine jusqu'aux feuilles de manière itérative en appliquant une stratégie *diviser pour régner* . À chaque étape, les trois algorithmes vont chercher à partitionner un nœud par l'usage d'une sélection de prédicat. Chaque ensemble de la partition ainsi obtenue représentant alors un nouveau nœud enfant. L'algorithme va ensuite réitérer la procédure pour chaque nouveau nœud créé, ce procédé ne trouvant une fin que si :

— l'ensemble d'apprentissage associé au nœud à subdiviser ne contient que des instances

d'une même classe,

— il n'est plus possible de produire de nouveaux prédicats pour subdiviser l'ensemble d'apprentissage associé au nœud à subdiviser,

— tous les ensembles de prédicats pouvant être utilisés pour subdiviser l'ensemble d'apprentissage associé au nœud à subdiviser produisent toujours un nœud fils vide.

La sélection du meilleur ensemble de prédicats est réalisée en fonction d'une mesure qui varie d'un algorithme à l'autre. Cette mesure représente une estimation de la qualité de la découpe réalisée par un ensemble de prédicats.

Dans ce contexte, l'algorithme ID3 maximise l'expression appelée *gain d'information* basée sur la formule de l'entropie de Shannon (1948). Optimiser le résultat de cette expression garantit d'utiliser à chaque étape l'ensemble de prédicats produisant les nœuds les moins hétérogènes.

Le *ratio de gain* est la variante de l'expression du gain d'information utilisée par l'algorithme C4.5. Cette variante, selon Cherfi *et al.* (2018), permet notamment de minimiser un biais pouvant subvenir lors de la production d'un arbre de décision pour un problème de classification comprenant un grand nombre de classes.

Finalement, l' *index Gini*, tel que défini dans l'article de Raileanu & Stoffel (2004), est la mesure utilisée par l'algorithme CART dans le cas particulier des arbres de décision binaires. L'index Gini est aussi une forme d'estimation de l'impureté d'un ensemble d'apprentissage.

Aussi, pendant la construction d'un arbre, ou après celle-ci, certaines opérations complémentaires peuvent être réalisées pour améliorer certaines caractéristiques du modèle. C'est notamment le cas des opérations d'élagage, comme celles décrites dans l'article de Mehta *et al.* (1995), permettant d'éviter le sur-apprentissage en fusionnant certains nœuds dans l'objectif

de diminuer la profondeur totale de l'arbre.

Dans le cadre de la reconnaissance d'activités, comme dans les études de Bao & Intille (2004), de Maurer *et al.* (2006), de Prossegger & Bouchachia (2014), de Castro *et al.* (2017) et de Lopez Medina *et al.* (2019), les arbres de décision sont utilisés pour classifier des groupes de caractéristiques discriminantes telles que présentées en Section 3.3.3. Chaque étude procède de la même manière : les différents signaux observés via des capteurs comme des gyroscopes ou des accéléromètres sont segmentés puis réduits à un nombre de caractéristiques fixe. Ces groupes de caractéristiques décrivent grossièrement chaque courbe observée et forment alors chacun une instance à classifier par l'arborescence. Les instances d'apprentissage sont pré-étiquetées et permettent de construire l'arbre. Les instances à classer, non-étiquetées, descendent l'arborescence apprise en comparant leurs caractéristiques à celles de chaque branche. Une fois une feuille atteinte, l'instance est finalement étiquetée en utilisant la classe qui y est la plus représentée.

Il existe une relation entre les arbres de décision et les listes de règles SI-ALORS pouvant être générées notamment par l'algorithme PRISM de Cendrowska (1987). Il est possible de transformer un arbre en liste de conditions et une liste de conditions en arbre, une approche est présentée dans l'article de Quinlan (1987).

3.4.5 RÉSEAUX DE NEURONES

Si l'on se base sur les travaux de Blalock *et al.* (2020), de Goldberg (2017), ou, plus anciennement, le livre de Goodfellow *et al.* (2016), il est possible de définir les réseaux neuronaux comme des modèles permettant d'approximer des fonctions mathématiques quelconques via un ensemble d'exemples en sélectionnant une certaine fonction au sein d'une famille de fonctions candidates, toutes dérivables. Cette sélection se fait par un apprentissage qui consiste

principalement en l'optimisation d'un ensemble de constantes appelées des poids permettant de moduler le comportement interne du réseau. Cette définition, relève d'une approche plutôt mathématique des réseaux de neurones, d'autres travaux, comme ceux de Abiodun *et al.* (2018) et de Abiodun *et al.* (2019) privilégient, quant à eux, une définition mettant plus l'emphase sur la similarité entre cette approche, et le fonctionnement du cerveau humain.

Quoiqu'il en soit, les réseaux de neurones peuvent être utilisés pour répondre à tout une variété de taches incluant notamment, mais pas seulement, la classification, comme dans le travail de Yadav & Jadhav (2019), la régression, comme dans le travail de Cranmer *et al.* (2020), ou encore l'agrégation, comme dans l'article de Li *et al.* (2017). Cette versatilité fait de ce modèle une solution de choix pour répondre à toute une variété de problématiques. Par exemple, on retrouve parmis ces dernières le traitement du langage naturel, comme dans le travail de Otter *et al.* (2020), la vision par ordinateur, comme dans l'étude de Voulodimos *et al.* (2018), la reconnaissance de contenu audio, comme dans le travail de Zhang *et al.* (2018c), la prédiction, comme dans l'article de Torrisi *et al.* (2020), ou encore, l'automatisation robotique, comme dans l'étude de Chen *et al.* (2020). Le travail de Pouyanfar *et al.* (2018) propose, par ailleurs, une revue plus précise de ces différentes utilisations.

Bien que les réseaux de neurones soient construits autour d'une approche supervisée, rien n'interdit de les utiliser de manière non supervisée, comme dans le cadre des travaux de Gu *et al.* (2021b) et Zhang *et al.* (2018a) utilisant respectivement un auto-encodeur ou un réseau antagoniste génératif. Ces deux solutions non supervisées sont d'ailleurs deux exemples d'architectures possibles parmi les nombreuses autres approches que peuvent proposer les réseaux neuronaux. Le travail de Pouyanfar *et al.* (2018) présente de nombreux exemples d'architectures accompagnées des cas d'utilisations où elles sont privilégiées. Dans le cadre de cette thèse, nous nous occuperons principalement de traiter de l'utilisation supervisée des réseaux de neurones, et ce, plus particulièrement, dans le cadre de la reconnaissance

d'activités.

Dans le cadre de la reconnaissance d'activités, les réseaux de neurones peuvent être utilisés pour classifier des séquences d'évènements émis par des environnements intelligents. C'est le cas du travail de Liciotti *et al.* (2020) où différentes variations de réseaux *Long Short Term Memory* (LSTM) sont utilisées pour reconnaître des activités au sein de différents jeux de données produits par le laboratoire CASAS. Plus précisément, les auteurs travaillent à reconnaître des activités comme *se laver*, *dormir*, *manger*, *cuisiner*, *travailler*, *se relaxer* ou *prendre des médicaments* via les données émises par des capteurs de mouvement, de contact, ou de température. Certains des jeux de données utilisés peuvent inclure des capteurs supplémentaires, comme des capteurs de consommation électrique ou des capteurs spécialisés pour détecter l'usage d'eau chaude / d'eau froide, ou de certains appareils ménagers. Les jeux de données, présentés sous la forme d'une séquence d'événements annotés, sont pré-traités puis transmis à un réseau de neurones LSTM chargé de reconnaître chaque activité. Plus spécifiquement, un réseau LSTM est une architecture particulière de réseau neuronal présentée initialement dans l'article de Hochreiter & Schmidhuber (1997). C'est une évolution des *Recurrent Neural Network* (RNN), eux présentés initialement dans le travail de Rumelhart *et al.* (1986). Un RNN est un réseau spécialisé dans le traitement de données séquentielles ayant la capacité d'adapter son comportement au fur et à mesure qu'il avance dans le traitement de sa séquence d'entrée. L'entraînement d'un tel type de réseau pour la gestion des longues séquences d'éléments peut faire l'objet de différentes problématiques empêchant purement et simplement au réseau d'apprendre correctement. Parmi ces dernières, les problèmes d'explosion et de dissipation du gradient sont celles ayant motivé le développement des réseaux LSTM . Pour une approche plus approfondie de ce sujet, l'article de Sherstinsky (2020) propose un développement progressif de ces deux notions, développement que nous ne pouvons réaliser ici pour des raisons de concision. Pour en revenir à l'article de Liciotti

et al. (2020), les auteurs proposent de tester différentes configurations de LSTM, incluant notamment un réseau simple et un réseau double, permettant de traiter les séquences du début vers la fin et inversement en même temps ainsi que différentes combinaisons de ces deux approches en cascade ou en parallèle. L'efficacité des modèles utilisés varie en fonction des jeux de données, certains étant plus adaptés que d'autres aux différents contextes explorés. Dans l'ensemble, les auteurs parviennent à obtenir une justesse et une précision variant entre 85% et 95% sur 4 des 5 jeux de données testés, le dernier jeu, quant à lui, semble poser des problèmes importants faisant chuter ces scores autour de 75%.

On peut aussi trouver des utilisations des réseaux neuronaux dans le cadre de la reconnaissance d'activités en utilisant des capteurs à même le corps, comme dans le travail de Wan *et al.* (2020). Dans cette étude, les auteurs comparent l'efficacité de certaines architectures neuronales comme les LSTM, *MultiLayered Perceptron* (MLP) et *Convolutional Neural Network* (CNN) avec une application de SVM dans le cadre de la reconnaissance d'activités comme *marcher*, *s'asseoir*, *descendre des escaliers* ou *monter des escaliers* via les accéléromètres, les gyroscopes, les magnétomètres, les capteurs de température et les capteurs de rythme cardiaque présents sur plusieurs téléphones intelligents placés sur le corps du participant. Un MLP est ici un réseau de neurones tout à fait classique comprenant plusieurs couches interconnectées. Les CNN, quant à eux, sont des architectures plus complexes initialement proposées par LeCun *et al.* (1989) et fonctionnant par l'apprentissage d'opérations de convolution. Ce sont des réseaux particulièrement pratiques pour la reconnaissance d'images, car ils permettent d'apprendre des transformations ou des relations locales entre éléments. Dans le cadre de cette étude, ils peuvent être utiles en une dimension pour tenter d'extraire des relations entre plusieurs événements consécutifs. Une définition plus rigoureuse de ce type d'architecture est proposée dans l'article de Koushik (2016). En fonction des jeux de données utilisés, les CNN sont meilleurs que toutes les autres solutions. En effet, ces réseaux

dépassent la justesse et la précision des SVM, second au classement, d'entre 3 et 10 points, pour atteindre un résultat avoisinant les 93%.

3.5 APPRENTISSAGE NON SUPERVISÉ

L'apprentissage non supervisé (Srivastava *et al.*, 2015) est le regroupement des méthodes ne nécessitant aucun retour, que cela soit sous la forme d'exemples ou de récompenses, pour extraire de l'information d'un jeu de données. Dans le cadre de la reconnaissance d'activités, les méthodes d'apprentissage non supervisé sont utiles pour découvrir des nouvelles activités inconnues et en extraire leurs spécifications par l'observation d'un milieu. C'est, par ailleurs, l'approche que l'on peut retrouver dans les travaux de Wang *et al.* (2006), de Cook *et al.* (2013b) et de Gjoreski & Roggen (2017).

3.5.1 FORAGE DE SÉQUENCES

Le forage de séquences regroupe les méthodes d'apprentissage non supervisé dont l'objectif est d'extraire des séquences d'éléments récurrents au sein d'une séquence plus large d'éléments. C'est une technique qui peut être appliquée dans différents domaines, par exemple, la revue de littérature de Buczak & Guven (2016) présente un application de la méthode pour la détection d'anomalies au sein des systèmes informatiques par l'étude des journaux d'événements produits par les serveurs. Aussi, le travail de Kantety *et al.* (2002) exploite le forage séquences pour étudier des génomes en analysant les séquences d'ADN.

Comme présenté précédemment, il est commun, dans le cadre de la reconnaissance d'activités, de chercher à rapprocher tout ou partie de journaux d'événements émis par différents capteurs et/ou algorithmes avec des activités définies à l'avance. Dans le cadre de la reconnaissance d'activités, le forage de séquences est utilisé pour reconnaître des

enchaînements récurrents d'actions ou d'interactions au sein de journaux d'événements extraits de différents environnements intelligents. Les séquences ainsi obtenues sont alors regroupées puis annotées en se basant sur leurs caractéristiques, les couples exemple/classe peuvent ensuite être utilisés afin d'entraîner ou calibrer divers algorithmes de classifications ce qui permet, ainsi, de les adapter aux spécificités de chaque environnement. C'est notamment l'approche adoptée dans l'article de Bouchard *et al.* (2013).

Plus spécifiquement, les travaux de Chikhaoui *et al.* (2011) et de Wen *et al.* (2015) se basent sur l'approche fréquentielle du problème de forage de séquences et utilisent l'algorithme *Apriori* proposé par Agrawal & Srikant (1994) et basé sur la propriété du même nom. Chikhaoui *et al.* (2011) extraient ainsi des séquences d'événements fréquents et leur attribuent automatiquement une activité en fonction des différentes interactions avec des objets qui y apparaissent. Wen *et al.* (2015) proposent une version altérée de l'algorithme *Apriori* et utilisent les séquences minées pour classer des séquences inconnues. Le modèle proposé performe mieux dans les tests réalisés en comparaison de classeurs comme le classifieur bayésien naïf, l'arbre de décision ou le HMM.

D'autres variations de l'algorithme *Apriori* sont aussi présentées dans les livres de Dong & Pei (2007) et de Cios (2007). Ces différentes variations permettent de répondre à des contraintes différentes de temps, d'espace, ou de nature des données. Utiliser des variantes peut permettre de répondre à des contraintes spécifiques à la reconnaissance d'activités comme l'extraction d'activités entremêlées.

Les travaux de Rashidi *et al.* (2010) et de Rashidi & Cook (2013) s'inscrivent d'ailleurs dans cette logique en proposant une méthode de forage de séquences basée sur le principe de Longueur de Description Minimale (LDM) initialement présenté par Rissanen (1978) puis Grünwald & Grunwald (2007). Cette méthode a pour objectif de permettre de forer des

séquences d'événements répétées en tenant compte du fait que certaines séquences peuvent être entremêlées ou scindées par des séquences de bruit. Cette approche permet aussi de forer des séquences de petite taille de meilleure qualité comme décrit dans les articles de Vreeken *et al.* (2011) et de Galbrun *et al.* (2018). Dans le cadre des articles de Rashidi *et al.* (2010) et de Rashidi & Cook (2013), les auteurs forent des séquences, les regroupent via des algorithmes d'agrégation et utilisent les groupes ainsi obtenus pour entraîner des HMM. L'approche proposée a été testée sur des jeux de données afin de regrouper puis reconnaître automatiquement des activités comme manger, prendre des médicaments, sortir ou se laver. Dans l'étude de Rashidi & Cook (2013), en ce qui relève du forage de séquences, les auteurs parviennent à extraire, par maison intelligente testée, un total de 5.000 séquences récurrentes sur 30 jours de données. Ce total avoisine les 17.500 séquences quand les auteurs explorent 75 jours de données. Après un processus de simplification et de regroupement des séquences, cet ensemble initial peut être réduit à un total variant entre 5.000 et 7.000 séquences récurrentes réellement distinctes. Dans l'ensemble, l'exploration de journaux plus longs permet de découvrir plus de variations d'une même activité, d'où une élimination de plus en plus importante du nombre de séquences. Cette analyse peut aussi être appliquée aux résultats de l'étude de Rashidi *et al.* (2010). Dans cette première étude, une augmentation de la sensibilité de l'algorithme de minage implique une augmentation du nombre de séquences extraites. Et comme pour l'étude de Rashidi & Cook (2013), la simplification est plus importante quand l'on extrait beaucoup de séquences.

Finalement, l'article de Shou & Di (2018) utilise l'approche fréquentielle et l'algorithme *Prefix-Span* proposé par Han *et al.* (2001) pour extraire des séquences d'activités récurrentes réalisées par différents individus. Les auteurs étudient des séquences d'activités génériques comme *aller travailler* , *faire ses courses* ou *aller à l'école* , toutes reconnues via les services de géolocalisation de téléphones intelligents. Les séquences fréquentes obtenues sont

alors groupées et utilisées pour explorer les relations entre la démographie des participants et leurs activités récurrentes. De 19.130 séquences d'activités produites via la méthode proposée dans l'article, l'algorithme *Prefix-Span* extrait 5.002 séquences fréquentes différentes parmi lesquelles, environ 50% comprennent 3 éléments, 30% comprennent 4 éléments, 10% comprennent 5 éléments et 10% comprennent entre 6 et 8 éléments.

3.5.2 AGRÉGATION

L'agrégation est un problème d'apprentissage non supervisé pouvant être assimilé à de la classification non supervisée (Saxena *et al.*, 2017a). Les algorithmes d'agrégation visent à extraire des *agrégats* d'un ensemble d'instances d'apprentissage, c'est-à-dire, des groupements d'objets partageant des traits communs.

Malgré une certaine facilité intuitive à se représenter ce qu'est une classe d'objets, définir ce qu'est un *agrégat* plus formellement est un problème non trivial et le livre de Everitt *et al.* (2011) le démontre en utilisant plusieurs représentations de configurations types dans son introduction. Il existe cependant dans la littérature plusieurs tentatives de définitions avec notamment celle de Bonner (1964) empruntant au test de Turing (1950) définissant un agrégat comme un ensemble d'objets à partir duquel un être humain est capable d'extraire une relation commune. Cormack (1971) et Gordon (1999) définissent, quant à eux, un agrégat en utilisant la notion de similarité/dissimilarité, c'est-à-dire, en spécifiant qu'un agrégat d'objets est un ensemble d'éléments similaires entre eux et le plus dissimilaire possible des éléments appartenant à d'autres agrégats.

Les algorithmes d'agrégations sont généralement divisés en trois grands groupes de méthodes (Saxena *et al.*, 2017a) : les méthodes dites de *partitionnement*, les méthodes dites *hiérarchiques* et les méthodes tierces n'entrant dans aucunes des deux premières catégories.

Plus spécifiquement, les méthodes dites de *partitionnement* agrègent des objets appartenant à un ensemble en créant une partition de cet ensemble. Les méthodes dites *hiérarchiques* vont, quant à elles, produire une arborescence de classes soit en scindant un groupe d'objets jusqu'à n'avoir plus que des ensembles ne comprenant qu'un unique objet, soit en fusionnant des ensembles ne comprenant qu'un unique objet jusqu'à obtenir un ensemble les incluant tous. Finalement, la dernière classe de méthode regroupe des approches variées incluant notamment des approches basées sur la théorie des graphes, les réseaux de neurones profonds, l'utilisation de *grilles* ou encore l'utilisation d'arbres de décision.

Dans le cadre de la reconnaissance d'activités, l'agrégation peut servir à grouper des instances d'activités similaires pour former de futurs ensembles d'apprentissage. Dans l'article de Rashidi *et al.* (2010), les auteurs agrègent des séquences d'événements produites par un ensemble de capteurs de mouvement, de capteurs de température et de capteurs de contact en utilisant l'algorithme des *k-moyennes*. Cet algorithme du groupe des méthodes de *partitionnement* peut être retrouvé dans les articles de Forgy (1965) et de MacQueen (1967). Les auteurs utilisent une variation de la distance de Levenshtein (1966) pour estimer la similarité ou la dissimilarité entre deux séquences afin de produire différents agrégats. Les centroïdes de chaque agrégat sont alors utilisés pour entraîner des algorithmes de classification tiers. L'approche proposée est testée dans le cadre de la reconnaissance de différents types d'activités incluant notamment *téléphoner* , *préparer à manger* , *regarder la télévision* , *manger* ou *nettoyer* .

L'étude de Rashidi & Cook (2013), dans la continuité de l'étude de Rashidi *et al.* (2010), préfère l'approche *hiérarchique* à l'approche par *partitionnement*. Les auteurs utilisent un algorithme procédant par agglomération d'instances mais n'exécutent pas l'algorithme jusqu'au bout. Une condition d'arrêt basée sur la similarité des derniers agrégats générés leur permet de ne pas avoir à spécifier manuellement le nombre souhaité d'agrégats à extraire,

au contraire de l'approche utilisant l'algorithme des *k-moyennes*. Les centroïdes de chaque agrégat sont ensuite utilisés pour entraîner des classeurs. L'approche est appliquée à des environnements similaires à celui utilisé dans l'étude de 2010. Les activités reconnues changent cependant pour une liste d'AVQ comprenant les activités *se laver* , *manger* , *entrer / sortir de la maison* , *cuisiner* , *dormir* , *se relaxer* et *prendre des médicaments* .

Avec cette approche, les deux études reconnaissent entre 70% et 80% des classes d'activités étiquetées au sein des jeux de données testés. L'article de Rashidi & Cook (2013) précise la forme de certaines erreurs comme la fusion de deux classes d'activités en une seule ou l'absence d'une classe d'activité dont la réalisation est rare. L'étude de Rashidi *et al.* (2010) précise que si l'on étiquette chaque événement du jeu de données testé avec l'étiquette attribuée à l'instance d'activité à laquelle l'événement appartient, alors 87.5% des événements ont été correctement aggrégés, c'est-à-dire regroupés au sein de classes correspondant à leur étiquette.

3.5.3 MODÉLISATION DE SUJETS

La modélisation de sujets est une méthode empruntée au domaine du minage de texte qui permet de décrire la relation entre des sujets que peut couvrir un texte, comme *la cuisine* , *le bricolage* ou *la finance* et la probabilité que ceux-ci provoquent l'apparition de certains mots. Il semble, par exemple, plus probable qu'un texte traitant de *cuisine* contienne des mots comme *plat* , *légume* , *fruit* qu'un texte traitant de *bricolage* (Berry *et al.*, 2010).

La modélisation de sujets permet d'extraire non seulement le nombre de sujets, mais aussi les probabilités que l'existence de chaque sujet dans un document provoque l'apparition de certains mots. Ce type d'approche pourra, par exemple, produire un résultat indiquant qu'un sujet provoque l'apparition du mot *plat* dans 30% des cas, du mot *légume* dans 20% des cas,

du mot *ustensile* dans 25% des cas et du mot *fruit* dans 25% des cas. Ce type de résultat permettra donc de se faire une idée de la nature des sujets en question, ici potentiellement un sujet concernant *la cuisine* .

Dans le cadre de la reconnaissance d'activités, la découverte de sujet permet d'extraire des activités ou des routines en analysant des journaux d'événements. Dans ce type de contexte, les journaux deviennent des documents à miner, et les événements, des mots. L'intérêt de la méthode, en regard d'une méthode d'agrégation, repose principalement dans sa capacité à produire la description de chaque activité ou routine ainsi extraite. Cette définition peut alors être utilisée pour classer chaque sujet ou pour utiliser chaque sujet en se basant sur leur description.

Plus spécifiquement, dans leur étude, Huynh *et al.* (2008), utilisent la modélisation de sujets pour extraire les routines d'individus en se basant sur des activités déjà reconnues comme *préparer à manger* , *répondre au téléphone* , *se laver les mains* , *faire la vaisselle* ou *souper* . La méthode leur permet d'extraire les routines et les activités qui leur sont associées comme la routine consistant à *se déplacer* , *acheter de la nourriture à la cafétéria* , *s'asseoir à un bureau* et *manger* . L'approche proposée est testée pour reconnaître des routines comme *manger* ou *travailler* dans le cadre de l'analyse, soit, de séquences d'étiquettes générées à l'aide d'algorithmes de reconnaissances d'activités supervisés, soit, de séquences d'étiquettes générées via des approches non supervisées. Dans le cadre des séquences générées par des algorithmes supervisés, le modèle proposé reconnaît les routines avec une justesse moyenne d'environ 86%, dans le cadre non supervisé cette justesse est réduite à environ 77%.

Dans l'étude de Freedman *et al.* (2014), la modélisation de sujets permet de découvrir des activités en se basant sur des squelettes simplifiés obtenus via une observation d'individus par des capteurs. Le système extrait alors les activités sous la forme de sujets et leurs descriptions

sous la forme de squelettes. Les auteurs arrivent ainsi à extraire des activités comme *se déplacer* , *lancer un objet* ou *transporter des objets* . Les auteurs ont testés leur approche avec différents paramétrages. Dans le cas où le nombre de sujets à extraire est trop petit, l'algorithme tente de regrouper des activités différentes en une seule catégorie, quitte à ce qu'elles n'aient pas grand chose à voir entre elles en dehors du partage de certaines poses très communes. À l'inverse, dans le cas où le nombre de sujet à extraire est trop grand, l'algorithme tend à produire une catégorie pour chaque séquence de poses possible, ce qui peut amener à produire des sujets qui se ressemblent beaucoup. Avec un nombre de sujet raisonables, l'algorithme tend à produire des séquences de poses suffisamment évocatrices pour qu'un sujet humain reconnaisse aisément chaque type d'activité décrit.

Finalement, dans l'article de Boukhers *et al.* (2020), la modélisation de sujets est utilisée pour reconnaître des activités comme *s'asseoir*, *marcher* ou *monter des escaliers* via les données extraites de différents accéléromètres et gyroscopes. Pour y parvenir, les auteurs commencent par transformer les séries temporelles des capteurs en séries de *symboles* en discrétisant les segments de chaque série à convertir via une opération d'aggrégation. Les auteurs construisent ensuite des mots en concaténant les symboles appartenant à des canaux similaires, par exemple, en agrégant les symboles obtenus de toutes les accélérations en abscisse de chaque accéléromètre. Dans un dernier temps, les mots sont utilisés pour reconnaître les activités de façon non supervisée. La méthode présentée obtient un *F-score* de 91% lors des tests en se trompant majoritairement sur les activités *s'asseoir* et *rester debout* qui sont des activités plutôt statiques pour lesquelles les traces des capteurs sont similaires.

3.6 ANALYSER LA QUALITÉ

Il existe, dans la littérature des vingt dernières années, différentes études présentant le domaine de la reconnaissance d'activités au travers de prismes de lectures variés. On peut,

notamment, trouver l'étude de Aggarwal & Ryoo (2011) se focalisant notamment sur les découvertes faites dans la reconnaissance d'activités via des outils de reconnaissance d'images, l'étude de Ranasinghe *et al.* (2016) mettant en exergue les avancées dans la reconnaissance d'activités appliquée aux flux de données et l'étude de Abdallah *et al.* (2018) tentant de faire un rapide tour des différentes applications des méthodes de reconnaissance d'activités au sein d'autres domaines de recherche.

Après lecture de ces documents, on peut établir que la reconnaissance d'activités est un problème complexe à résoudre et qu'il est possible de l'aborder de bien des manières comme vu dans les sections précédentes, dans l'étude de Aggarwal & Ryoo (2011) et de Ranasinghe *et al.* (2016) ainsi que dans les livres de Cook (2015) et de Chen & Nugent (2019). Finalement, la reconnaissance d'activités peut être appliquée dans des situations variées comme dans le cadre de la mise en place de maisons intelligentes, dans la création d'outils pour les professionnels de la santé ou pour résoudre des problématiques de sécurité et de surveillance telles que présentées dans l'étude de Abdallah *et al.* (2018).

Toutes ces approches et ces études cherchent d'abord à établir si des activités ont lieu et à définir les moments précis où celles-ci commencent et se terminent. Pour ce faire, les approches proposées rencontrent des limites concrètes et complexes, comme gérer la variété possible de réalisation de la même activité dans le temps, les problématiques de généralisation des modèles appris à différents milieux et la très complexe tâche de reconnaître des activités dans des situations multi-résidentes. Il semble donc normal, à la vue de ces limites et en l'état de nos connaissances, que peu d'articles de la littérature fassent un focus sur des outils permettant de décrire plus précisément la nature de chaque instance d'activité découverte.

Être capable de décrire plus précisément chaque instance d'activité, en lui attachant des étiquettes représentant des informations comme la quantité d'actions effectuées au total, la

variété des interactions qui y ont lieu ou si l'activité est habituelle ou inhabituelle permettrait d'établir des analyses plus précises de l'état de santé ou du comportement d'un individu en particulier après une étude de l'évolution de ces différents paramètres. Dans le cadre des maladies dégénératives, par exemple, une personne en bonne santé aura plus de facilité à réaliser des tâches complexes impliquant plusieurs interactions avec son environnement ainsi qu'une planification minutieuse et une concentration sur une longue durée, alors qu'une personne malade, à l'inverse, préfèrera certainement limiter ses mouvements ou le nombre de sous-tâches à réaliser.

Ainsi, l'augmentation de la fréquence d'activités nécessitant moins de ressources cognitives peut être un bon indicateur pour les médecins de l'état de dégradation de la santé d'un malade.

Dans la littérature, l'étude de Fatima *et al.* (2013) est ce qui semble se rapprocher le plus de cette idée, en proposant différents attributs abstraits permettant de mieux décrire la nature des différentes instances d'activités présentes dans un ensemble d'apprentissage, dans l'objectif d'établir un lien entre la manière dont les activités sont exécutées et la nature des résultats obtenus en utilisant différentes méthodes de reconnaissance d'activités.

Adopter ce type d'approches pourrait aussi permettre de réaliser une forme de reconnaissance d'activités non supervisées via des modèles exprimant la qualité de la réalisation d'une certaine classe d'activité. Par exemple, en exprimant d'une certaine manière qu'une activité comme *faire la cuisine* est plus qualitative si l'acteur manipule une variété importante de matériel, si il est dynamique et s'il passe un moment raisonable dans la cuisine, il devient alors potentiellement possible d'éliminer les passages en coup de vent dans la cuisine qui seront au final des activités de bien trop basse qualité. De la même manière, il devient possible de reconnaître des activités en segmentant la journée de telle manière que les différents modèles

obtenus expriment une qualité qui soit maximale.

Utiliser ce type de modèle permettrait, de plus, de justifier la reconnaissance réalisée et, ainsi, d'apporter une information précieuse aux cliniciens qui pourront se faire une idée des indices ayant amené à la production d'une certaine étiquette. Ce serait une manière d'éviter l'effet *boîte noire* telle que décrit dans le cadre des réseaux de neurones par l'article de Zhang *et al.* (2018b). Aussi, ce type d'information pourrait alimenter les échanges entre les différents acteurs qu'ils soient cliniciens ou malades.

Le problème étant alors d'utiliser une représentation suffisamment générique pour qu'elle puisse être appliquée à un ensemble de lieux hétérogènes. Des outils, comme les tables de règles en logique floue des travaux de Pradhan *et al.* (2009) peuvent être une manière de construire des règles compréhensibles par des êtres humains. En les complétant avec des fonctions d'appartenances comme décrites dans l'article de Lam (2018) pour assurer une certaine abstraction entre les données réelles et l'espace de raisonnement, ce type de modèle pourrait être une base intéressante pour produire des méthodes de reconnaissance basées sur la qualité.

Finalement, des outils comme le forage de séquences telle que nous l'avons présenté en Section 3.5.1, l'agrégation tel que nous l'avons présenté en Section 3.5.2 et les arbres de décision tels que nous les avons présentés en Section 3.4.4 pourrait être des solutions intéréssantes pour découvrir des activités et les transformer en règles de logique floue.

3.7 CONCLUSION

Nous avons, en Section 3.2 de ce chapitre, proposé une définition de ce qu'était une activité et la reconnaissance d'activités en général. Nous avons par la suite, en Section 3.3, fait un rapide résumé des différentes manières de représenter l'information dans l'objectif de

l'utiliser pour reconnaître différentes activités.

En Section 3.4, nous avons présenté des exemples d'approches permettant de reconnaître des activités en usant d'outils basés sur l'apprentissage supervisé comme le classifieur bayésien naïf, les HMM, les SVM et les arbres de décision. Nous avons aussi présenté, en Section 3.5, des solutions non supervisées permettant de découvrir de nouvelles activités comme le forage de séquences, l'agrégation et la découverte de sujets.

La reconnaissance d'activités est un sujet complexe à définir et nécessitant de s'adapter aux différents environnements et problèmes à résoudre. Ces deux caractéristiques font des méthodes basées sur l'apprentissage d'excellents outils pour apporter des réponses en analysant directement la réalité grâce aux méthodes non supervisées ou en tentant de créer des modèles complexes en usant d'une base d'exemples via des solutions d'apprentissage supervisé. Finalement, l'apprentissage permet aussi d'adapter les solutions aux différentes situations rencontrées.

Finalement, en Section 3.6 nous avons discuté brièvement des limites que les solutions de reconnaissance d'activités rencontrent, et en quoi ces problèmes majeurs, qui doivent être résolus en priorité, peuvent expliquer le peu de travaux publiés sur certains sujets, comme celui de la création d'outils ou de modèles permettant de décrire la nature profonde des différentes activités observées. Pourtant, de tels outils pourraient être particulièrement utiles pour certaines applications dans le cadre de la santé.

CHAPITRE IV
ARCHITECTURE POUR LE SUPPORT DE MULTIPLE HABITATS INTELLIGENTS

Comme nous l'avons présenté dans le Chapitre 1, l'objectif du projet SAPA est de développer un assistant ambiant personnalisable afin de répondre à des problématiques de santé publique liées au vieillissement de la population mondiale. Pour y parvenir, ce projet propose de capitaliser sur les possibilités offertes par les environnements intelligents, notamment en ce qui concerne la collecte des données, sujet que nous avons traité plus particulièrement dans le Chapitre 2. Ces données peuvent finalement être analysées via des outils appartenant au domaine de la reconnaissance d'activités afin d'apporter de l'information utile aux cliniciens, sujet que nous avons traité dans le Chapitre 3.

Cette thèse s'insère dans la préparation du développement d'une partie du projet SAPA qui est le sujet de la Section 1.5 et de la Figure 1.1 de ce document. Notre travail est donc cantonné à un périmètre intermédiaire englobant des composants chargés de collecter, de transporter, de transformer et de distribuer les données extraites des environnements pour les rendre disponible à différentes applications cliniques. Cette solution doit alléger les responsabilités des projets situés en amont et en aval du flux de données pour que ceux-ci puissent se focaliser sur des problématiques qui leur sont propres.

Les maisons intelligentes de la Section 2.5 apportent toutes, à leur manière, une part de la solution. Cependant, aucune d'elle ne répond à toutes nos problématiques de manière exhaustive, notamment en se concentrant uniquement sur la gestion d'un seul et unique environnement. Notre travail de recherche consiste donc à élaborer une solution capable de gérer plusieurs environnements hétérogènes tout en répondant à l'ensemble de nos problématiques.

4.1 LES HABITATS DU PROJET SAPA

En date du 1er Juillet 2021, le projet SAPA listait un total de 37 environnements différents parmi lesquels 31 étaient encore en service. Les six environnements inactifs ayant fait l'objet de différentes désinstallations entre mars 2020 et mai 2021. Sur l'ensemble des environnements listés, trois étaient des installations en laboratoire, une servait de test au domicile d'un des chercheurs du projet, et les 33 restants étaient des maisons de participants équipés de capteurs. Entre le début du projet en 2018 et la rédaction de ce document, certains environnements additionnels, totalement virtuels, ont aussi servi pour réaliser différents tests.

Comme représenté au sein de la Figure 4.1, les appartements des participants sont plutôt hétérogènes. Sur les trois plans présentés, si la différence d'agencement des pièces et des meubles est la plus évidente à voir, il apparaît aussi une différence en terme de quantité de pièces, avec, notamment, une chambre additionnelle dans l'installation représentée en bas à droite de l'image. Au-delà de ces premières variations, il existe aussi des différences d'équipement. On retrouve ainsi jusqu'à 16 types de capteurs dont des capteurs infrarouges pour la détection des présences comme le FGMS-001 [21], le DMMS1 [22] ou le HSP02 [23], des capteurs de contact comme le HSM02 [24] ou le DMWD1 [25] ou encore des capteurs de consommation électrique comme le ZW096 [26] ou le DSC06 [27]. Ces capteurs peuvent exister au sein de neuf types de pièces, comprenant notamment des chambres, des cuisines, des salons ou des salles à manger. Enfin, 21 types d'objets sont suivis, comprenant parmis-eux des

21. Voir https://fccid.io/2AA9MFGMS002.
22. Voir https://fccid.io/Z52NAS-PD02Z.
23. Voir https://fccid.io/FU5HSP02.
24. Voir https://fccid.io/FU5HSM02.
25. Voir https://fccid.io/Z52NAS-DS01Z.
26. Voir https://fccid.io/XBAFT096.
27. Voir https://fccid.io/XBAZW078.

machines à café, des meubles-lavabos, des télévisions, des micro-ondes, des réfrigérateurs ou des cuisinières. Tous les éléments cités ici pouvant ne pas apparaître, ou apparaître en plusieurs exemplaires au sein de chaque installation.

Aussi, des variations au niveau logiciel sont à prendre en compte. Tout comme le matériel, la nature des informations à induire peut changer d'un environnement à l'autre car celles-ci dépendent grandement des objectifs des cliniciens et des chercheurs. Ainsi, garantir que l'environnement est suffisamment sécuritaire, monitorer le sommeil, s'assurer que l'individu s'alimente, établir la fréquence des sorties à l'extérieur et diagnostiquer si l'occupant est sujet ou non à des moments d'errance sont autant d'exemples de questions auxquelles l'outil doit pouvoir apporter une réponse.

Finalement, cette variété dans les objectifs à atteindre est la conséquence directe d'une dernière source d'hétérogénéité : celle des profils des participants. Des profils pouvant faire l'objet de différentes pathologies comme des troubles de la mémoire, des troubles cognitifs, des troubles psychotiques, ou encore une multitude d'autres conditions physiques ou mentales.

4.2 ARCHITECTURE ORIENTÉE DONNÉES

Dans le cadre du projet SAPA, il faut donc concevoir et mettre en place un composant intermédiaire entre les environnements intelligents et les cliniques. Cet intergiciel ayant pour rôle de collecter des données provenant des maisons sources, de les transformer en information de plus haut niveau puis de distribuer le tout aux applications finales permettant aux cliniciens de monitorer la santé des personnes suivies.

Bien que les architectures de la littérature présentées en Section 2.5 proposent un cycle de perception et de prise de décision, avec une abstraction de l'information des couches les

plus basses vers les couches les plus hautes et une traduction de commandes abstraites en actions concrètes des couches les plus hautes vers les couches les plus basses. Ce composant intermédiaire se démarque lui, déjà, par l'absence totale de circuit de prise de décisions descendant, l'objectif étant ici uniquement de collecter et distribuer de l'information.

De la même manière, où les architectures de la littérature se concentrent uniquement sur l'exploitation des données provenant d'une unique maison, notre composant, lui, se démarque une nouvelle fois en nécessitant d'être capable de capturer et transformer de la donnée venant de différents environnements. Environnements qui sont, par ailleurs, particulièrement hétérogènes, pour les raisons présentées dans la Section 4.1.

En dehors de ces déviations par rapport à la littérature, il existe aussi des problématiques communes, parmi lesquelles on retrouve la question de l'interopérabilité, largement traitée dans les articles déjà présentés dans le Chapitre 2. La question de l'adaptation aux changements matériels et logiciels au sein des environnements est aussi importante, celle-ci étant notamment traitée par des architectures comme OpenHAB. Finalement, la question de la distribution des données via des interfaces variées est aussi une problématique que nous avons en commun avec la littérature.

En partant de ces premiers constats, il est alors possible de schématiser grossièrement le composant à concevoir tel que présenté au sein de la Figure 4.2. Comme tout intergiciel, il existe deux interfaces à définir : une interface à destination des sources et une interface à destination des clients. Ces interfaces sont critiques car elles doivent imposer le moins de contraintes aux différents acteurs. La moindre contrainte sur les sources pouvant empêcher d'introduire un nouvel environnement dans le système, et les contraintes sur les applications clientes pouvant rendre le développement de certaines fonctionnalités plus coûteuses en temps et en calcul.

Pour ce qui est de l'interface entre les maisons et notre application, il est possible de collecter les données de deux manières : soit proactivement, en requêtant l'information auprès de chaque environnement, soit passivement, en laissant les maisons publier elles-mêmes l'information. C'est choisir entre *tirer* l'information des maisons, ou de laisser les maisons *pousser* l'information à nous. Ces deux types de flux d'information sont schématisés au sein de la Figure 4.3.

Au sein d'une architecture de maison intelligente, les capteurs poussent l'information au moment où ils la détectent. Il semble en quelque sorte naturel de répliquer ce comportement en laissant les maisons pousser l'information vers nous. En dehors de cette continuité comportementale, il est possible de trouver d'autres arguments au profit de l'approche où les maisons nous transmettent par elles-mêmes l'information.

Par exemple, l'approche inverse surchargerait le réseau de requêtes régulières visant à obtenir les mises à jour de chaque maison, l'intergiciel ne pouvant pas savoir de lui-même quand une nouvelle information est disponible. Dans cette situation, certaines requêtes peuvent même n'apporter aucune nouvelle information à l'intergiciel. Quand les maisons poussent l'information, si le réseau est surchargé, chaque environnement peut décider par lui-même, et en fonction de sa situation, d'accumuler une quantité suffisante d'information avant chaque nouvel envoi, garantissant au passage que toutes les requêtes apporteront une mise à jour.

Aussi, pour requêter des maisons, encore faut-il savoir qu'elles existent, il faudrait donc que les environnements s'inscrivent auprès de l'intergiciel pour que celui-ci connaisse leur existence, ce qui obligerait les maisons à pousser tout de même certaines informations. Cette situation requiert, par ailleurs, le développement d'un service supplémentaire permettant de maintenir un registre des environnements à requêter.

Finalement, laisser les maisons publier à leur bon vouloir c'est aussi les laisser maître des informations qu'elles fournissent, ce qui peut avoir une importance dans un contexte où l'on manipule des données personnelles.

Pour ce qui est de l'interopérabilité, l'approche où les maisons poussent l'information permet d'intégrer n'importe quelle architecture au prix du développement d'un composant capable d'effectuer la publication auprès de l'intergiciel. L'approche inverse permet aussi cela au prix de la mise en place d'une interface standardisée pour que l'intergiciel puisse requêter les mises à jour.

S'il semble ainsi plus bénéfique de procéder en laissant les maisons pousser leurs informations via une interface standardisée, il reste tout de même à répondre à la même question pour ce qui est de l'interface à destination des applications.

Cette fois-ci, s'il faut limiter au plus les contraintes, aucune situation ne se démarque vraiment. En effet, pour une interface, il peut être nécessaire de tirer certaines informations, et de laisser le serveur en pousser d'autres. Par exemple, pour produire les graphiques d'un tableau de bord, il est fréquent de requérir des informations sur le long terme, comme l'occupation des pièces sur les trois derniers mois. Ce type d'information ne peut exister sans une mémoire des événements, ce qui est facile quand on tire l'information d'une base de données, mais beaucoup moins quand on utilise un flux d'information poussé, ce dernier cas nécessitant de télécharger l'entièreté des événements déjà émis, de les filtrer et de les stocker.

À l'inverse, si l'on veut connaître l'état de certains objets en quasi temps réel, comme l'état d'un four, d'un évier ou d'autres électroménagers dans l'objectif de prévenir un dégât d'eau ou un incendie, il est largement préférable de laisser le serveur nous tenir informé des changements quand ceux-ci ont lieu. Tirer l'information dans cette situation nous ramène alors aux problématiques que nous avons traitées plus haut pour l'interface à destination des

maisons intelligentes.

Si l'interface à destination des applications doit pouvoir offrir les deux types de flux, il devient alors possible de scinder l'intergiciel en deux composants : un premier composant dont l'objectif est de collecter et transformer un flux d'événements décrivant ce qui se passe dans les environnements en quasi temps réel, flux que n'importe quelle application peut écouter si elle le souhaite. Puis un second composant, dont l'objectif est de collecter les événements émis par le premier pour compléter en continu une base de données servant de mémoire pour les applications tierces. Celles-ci pouvant récupérer les informations qui les intéressent via une interface de programmation applicative. La Figure 4.4 est une schématisation de cette approche.

Ainsi, dans la suite de ce chapitre, nous allons commencer par une description du composant traitant le flux d'information entrant, puis, nous finirons par une description du second composant servant de mémoire pour les applications finales.

4.2.1 HABITATS INTELLIGENTS AVEC DES AGENTS

Les environnements intelligents peuvent être définis comme des lieux améliorant l'expérience de vie de leurs occupants par le biais de contrôleurs orchestrés par des agents logiciels. Ainsi, comme présenté dans le Chapitre 2, des capteurs présents dans l'environnement vont commencer par collecter des données bas niveau relatives à la température, la luminosité, l'activité électrique ou au mouvement. Ces données vont ensuite naviguer au sein de l'environnement intelligent par le biais d'un flux d'événements. Des éléments logiciels, connectés à ce flux, vont alors pouvoir induire des informations plus abstraites comme la présence dans certaines pièces, des interactions avec du mobilier, ou encore, à un certain niveau, la réalisation de certaines activités. Ces informations abstraites vont ensuite permettre à d'autres éléments

logiciels de prendre des décisions, qui seront alors, finalement, communiquées aux effecteurs présents dans l'environnement afin que ces derniers modifient le réel en conséquence.

C'est l'approche de Stuart Russell (2009) dont nous avons déjà parlé en Section 2.5.1 et qui est schématisée au sein de la Figure 2.5. Ce que nous proposons, dans le cadre du premier composant de notre intergiciel, c'est d'abstraire un peu plus cette vision des maisons intelligentes en considérant tous les éléments précédemment cités, c'est-à-dire les capteurs, algorithmes et effecteurs, comme des agents autonomes capables de recevoir et de diffuser des messages, depuis et vers, tous les agents avoisinants.

La Figure 4.5 schématise cette approche. Dans cette situation, la description que nous avons faite en début de section du flux d'information est réduite à la communication entre différents agents présents dans un même milieu. Ces communications étant la résultante de leur propre logique interne, déclenchée par des observations qu'ils font, soit directement du monde réel, soit par le biais des communications qu'ils reçoivent. Chaque communication est ensuite transmise à tous les agents de l'environnement, ceux-ci pouvant à leur tour enclencher un nouveau processus logique et transmettre de nouvelles informations. Ce cycle amenant à la production d'un flux continu d'événements décrivant l'état interne de l'environnement en temps réel.

Avec un modèle basé sur des agents, on peut alors remplacer les trois blocs interfaces traitant le flux de données (en rouge) de la Figure 4.4 par un standard permettant de décrire chaque événement. Par ailleurs, il est aussi possible de remplacer l'opération de transformation du flux par une simple grappe d'agents logiciels communiquant en continu.

En uniformisant tous les composants sous la forme d'agents, il devient possible de lister ceux-ci aisément, et ainsi d'obtenir une vue globale de chaque installation connectée. Cette

vue globale peut ensuite s'affiner en mettant chaque agent en relation avec une série temporelle de messages émis ou reçus en fonction que ce dernier observe ou altère son environnement.

Cette fonctionnalité peut vite s'avérer très pratique, notamment pour les interfaces des applications, celles-ci pouvant ainsi s'adapter aux spécificités de chaque environnement en proposant des modules en fonction des agents disponibles. Quant aux techniciens, cette approche peut leur permettre d'avoir en tout temps une visibilité sur l'équipement et le statut de chaque environnement, ce qui est indispensable pour orchestrer les opérations de maintenance sur chaque installation, que celles-ci se résument au changement d'une batterie morte, ou au remplacement de capteurs défectueux.

La composition d'agents est aussi une solution permettant d'adapter l'architecture aux spécificités de chaque environnement. Par exemple, si un agent est disponible pour générer un étiquetage en fonction d'un flux de valeurs booléennes, il devient possible d'étiqueter facilement des interactions avec des équipements électriques si on le paire avec un agent qui génère un flux de valeurs booléennes en observant la consommation de certains équipements. De la même manière, il devient possible d'étiqueter la présence dans certaines pièces si on paire cet agent d'étiquetage avec d'autres agents générant un flux de booléens via des capteurs de mouvement. Ainsi, il devient possible de composer des chaînes de comportement en fonction de l'équipement disponible et des spécificités de chaque installation même si celles-ci varient.

Cette approche orientée agent simplifie aussi l'ajout et la suppression d'éléments logiciels pour les utilisateurs. Les agents étant autonomes, ils sont alors capables de se reconfigurer, de créer d'autres agents ou encore de se détruire eux-même en fonction des messages reçus. Si un agent est détruit, créé ou mis à jour, il peut alors communiquer cet état de fait par un événement, qui, une fois transmis à tous les agents observant le flux, peut à son tour provoquer

d'autres destructions, créations ou mis à jour en cascade. Cette forme de réaction en chaîne permettant à l'ensemble de la structure de se rétablir d'elle-même quand des mutations lui sont apportées.

Via cette fonctionnalité, un utilisateur n'a ainsi plus à connaître l'ensemble des dépendences d'un agent pour le supprimer, la cascade d'événements provoquant automatiquement la destruction des agents ne pouvant plus fonctionner, ou mieux encore, leur reconfiguration afin d'assurer un service minimum. Aussi, il devient possible d'automatiser une partie de la configuration en installant des agents permettant de générer automatiquement des morceaux complets d'infrastructure. Dans cette situation, quand des techniciens installent de nouveaux capteurs, certains agents peuvent alors créer automatiquement d'autres agents permettant de traiter cette nouvelle source d'information. Ainsi, l'apparition de capteurs de contacts permettant de détecter l'ouverture de placards dans une certaine pièce peut provoquer automatiquement l'apparition d'agents permettant d'identifier les interactions avec les meubles comme d'agents pour détecter la présence dans certaines pièces, et ainsi de suite jusqu'à la détection de certaines activités.

Utiliser un standard de communication permet, par ailleurs, de rendre le système totalement agnostique : de la même manière que des protocoles comme HTTP permettent de communiquer avec des serveurs développés dans différentes technologies, un protocole permettant de décrire des événements peut aussi servir à faire communiquer entre eux des agents codés dans différentes technologies. Ainsi, il devient possible d'exploiter les technologies les plus avantageuses en fonction de chaque problématique à résoudre, du moment que celle-ci permet de recevoir et d'envoyer des messages. Cette propriété vaut aussi pour l'équipement matériel, qui peut alors être intégré aisément via un patron adaptateur, tel que décrit dans le livre de Gamma *et al.* (1994).

Plus intéressant encore dans notre situation, cette méthode permet d'inclure des architectures complètes de maison intelligente, en installant dans chacune un composant capable de traduire leur comportement interne sous la forme d'un flux d'événements standardisés. Cette capacité d'intégrer n'importe quelle solution en décrivant son comportement interne par un flux ouvre par ailleurs aussi la possibilité de chaîner des intergiciels en redirigant le flux d'un intergiciel à l'autre via des pairs d'agents écouteurs / émetteurs, ce qui permet ainsi de créer des réseaux de grappes d'agents travaillant en coopération. Ces réseaux permettant alors de distribuer ou répliquer géographiquement des calculs.

Finalement, les agents logiciels peuvent aussi être isolés et testés, ce qui permet d'assurer que le projet fonctionne dans son ensemble. Ces tests se résumant alors à leur transmettre un flux de messages, et à observer les communications sortantes. Cette dernière possibilité est indispensable pour localiser l'origine d'erreurs quand beaucoup d'éléments coopèrent ensemble, dans notre cas des maisons, des intergiciels puis des applications.

Tous ces avantages font des architectures orientées agents une idée attrayante, bien que tout ceci vienne au prix de la production d'algorithmes plus complexes. Là où une boucle parcourant un tableau fini et trié de données peut suffire quand on fait de la reconnaissance d'activités hors-ligne, un algorithme bien plus complexe est nécessaire pour traiter de manière réactive des informations arrivant une à une dans un désordre plus ou moins important en fonction des aléas du réseau.

4.2.2 ÉVÉNEMENTS ET AGENTS

Une architecture basée sur les agents semble donc être une idée intéressante à explorer pour les raisons décrites dans la section précédente. Reste alors à définir un standard pour permettre à nos agents de communiquer entre eux. Ce standard est d'une importance capi-

tale dans la mesure où il est l'ensemble des conditions imposées pour inclure de nouvelles architectures et de nouveaux agents dans notre système.

Comme standard nous proposons quelque chose de simple : intégrer une action *Create Read Update Delete* (CRUD) avec un identifiant d'agent, un horodatage et une charge utile arbitraire suivant n'importe quelle représentation. Les actions CRUD servent alors à décrire le type d'événement, ces derniers pouvant représenter des additions, des suppressions ou des mutations d'information au sein de la série temporelle liée à l'agent émetteur. Le modèle CRUD est par ailleurs déjà utilisé dans d'autres technologies courantes, au sein des standards d'API on peut penser à REST que nous avons déjà présenté en section 2.4.2 mais aussi aux standards d'accès aux bases de données relationnelles comme SQL. Pour ce qui est du format événementiel, c'est aussi celui privilégié par des architectures récentes d'interfaces graphiques comme Redux (2020). Au besoin, il est d'ailleurs aussi possible d'ajouter des types permettant aux agents de se requêter entre eux.

La charge utile binaire peut contenir à peu près n'importe quoi, tout comme un datagramme dans la norme IP. Nous n'imposons d'ailleurs pas de standard de représentation pour les événements, cette décision revenant aux personnes développant les agents. Si besoin, ceux-ci peuvent accepter plusieurs standards communs comme ceux que nous avons présentés en Section 2.2. Il est aussi possible de développer des agents dont le travail consiste à traduire des événements d'un standard à un autre pour permettre à des groupes d'agents utilisant des standards différents de dialoguer entre eux. Pour ce qui est de notre implémentation, nous avons préféré faire simple en traitant toutes les communications en JSON, et cela malgré qu'une représentation binaire puisse être largement préférable, au prix, cependant, d'une difficulté supplémentaire à analyser les messages transmis. Comme les agents sont configurés pour connaître leurs coéquipiers via leur identifiant, ils sont aussi sensés connaître à l'avance le type de représentation et le type de données qu'ils recevront.

Par ailleurs, nous utilisons des identifiants plus complexes que de simple nombres, les agents étant organisés dans une arborescence ressemblant à un système de fichiers où les dossiers sont des espaces de la maison. Ainsi un capteur de mouvement peut appartenir à la cuisine d'une maison. Cette approche ayant l'avantage d'offrir aux techniciens une représentation de chaque maison, et aux agents la possibilité de coopérer avec d'autres agents appartenant à un même ensemble géographique. Cette dernière possibilité s'avérant plutôt pratique quand il s'agit de reconnaître les présences dans une pièce : il suffit alors de configurer l'agent pour travailler sur une pièce plutôt que de lui préciser chaque capteur de mouvement, de contact et capteur électrique devant lui servir de source.

Ainsi, comme décrit précédemment, et comme représenté au sein de la Figure 4.6, tous les agents deviennent des pièces logicielles autonomes pouvant écouter et émettre des événements dans un milieu. Cette description suivant ainsi la définition de Stuart Russell (2011).

Aussi, comme nous l'avons rapidement mentionné en fin de Section 4.2.1, il est possible de transformer des grappes d'agents en réseaux, capablent à la fois de répartir la charge de calcul géographiquement et de diffuser l'information collectée et générée vers différents lieux. Pour créer des réseaux, il suffit d'installer un agent d'écoute dans une grappe source et un agent d'émission dans une grappe de destination, le premier n'ayant qu'à communiquer au second tout ou partie des événements qu'il reçoit. De cette manière, il devient possible de créer un flux continu d'événements des maisons jusqu'aux cliniques tout en profitant de la robustesse d'un petit réseau internet. Les risques de surcharge d'un serveur centralisé peuvent être évités en réalisant une partie des calculs au sein des maisons, d'autres au sein de serveurs intermédiaires, et les calculs finaux au sein même des cliniques. Pour ce qui est des coupures d'information, il suffit alors de dupliquer certaines grappes afin de pallier au risque d'une

éventuelle défaillance matérielle. La Figure 4.7 schématise cette approche.

La construction d'un système de mémoire s'inscrit par ailleurs dans cette idée en installant au sein d'une grappe un agent spécialisé dont le rôle consiste principalement à transformer le flux d'événements entrant en écritures dans une base de données. Base dont l'accès peut être proposé par l'intermédiaire d'une API suivant un des standards que nous avons décrit en Section 2.4. Le flux continu peut alors être traité par lots, ou élément par élément en fonction de la charge. Les événements reçus permettant de construire une arborescence de séries temporelles, le tout pouvant être traité par les opérations standards d'une base de données.

4.3 IMPLÉMENTATION DE L'ARCHITECTURE

Dans le cadre du projet SAPA, nous avons implémenté l'architecture que nous avons décrite dans les sections précédentes sous la forme d'une plateforme monolithique codée en Java avec le cadre de travail Spring décrit dans les travaux de Gutierrez (2019) et de Arthur & Azadegan (2005). Cette plateforme gère à la fois une grappe d'agents et une solution de mémoire basée sur l'utilisation d'une base de données MySQL [28] ou MariaDB [29].

La partie de la plateforme responsable de générer l'information a été implémentée sous la forme d'un simple bus d'événements sur lequel des agents et d'autres écouteurs sont branchés. La Figure 4.8 schématise cette approche. Les événements reçus par la plateforme sont publiés dans le bus un à un, puis, les agents peuvent à leur tour émettre de nouveaux événements en réaction. Quand la plateforme reçoit un nouvel événement extérieur, elle démarre alors une nouvelle transaction SQL servant de réceptacle à l'ensemble des requêtes

28. Voir https://www.mysql.com/.

29. Voir https://mariadb.org/.

générées par des écouteurs situés en fin de bus. La transaction englobe ainsi les conséquences de l'événement reçu, mais aussi les conséquences de tous les événements émis en réaction par les agents enregistrés dans la plateforme. De cette manière, en cas de problème majeur lors de l'enregistrement des modifications, le système peut annuler toutes les conséquences de la réception d'un unique événement induisant ainsi soit l'apparition de toutes les modifications, soit l'apparition d'aucune d'entre-elles.

Les agents peuvent être instanciés par des applications clientes via une API REST en fournissant le type d'agent à produire et sa configuration. La description de la grappe d'agents est ensuite sauvegardée dans la base de données de la plateforme pour permettre leur réinstantiation en cas de redémarrage de l'application.

Afin de supporter les maisons du projet gérées via *OpenHab*, nous avons installé, dans chacune d'elles, une extension publiant vers notre plateforme un flux d'événements décrivant leur comportement interne. La plateforme génère ensuite automatiquement des compositions d'agents permettant de produire de l'information de plus haut niveau en se basant sur les données reçues. Dans sa version initiale, le projet comprenait des agents permettant de calculer ou recevoir des informations d'assez bas niveau, comme l'utilisation de fournitures électriques ou les interactions avec certains meubles, la création d'alertes basées sur l'état des batteries de chaque capteur, le passage de certains seuils de luminosité, ou encore, la présence dans certaines pièces.

La seconde partie de la plateforme propose quant à elle une API REST via HTTP permettant aux applications cliniques d'obtenir les informations stockées en base de données. Une autre API a aussi été mise en place afin d'exposer le flux d'événements interne de l'application en temps réel via une solution basée sur la technologie WebSocket. L'API REST a été pensée pour être la base du développement d'une future API GraphQL dont l'objectif vise

à offrir plus d'options aux applications clientes pour ce qui est de la représentation des données reçues. L'interface REST proposée traduit principalement des requêtes HTTP en requêtes SQL et propose des solutions pour gérer la pagination, le filtrage, les tris et les jointures en utilisant une nomenclature basée sur un graphe des objets requêtables et de leurs associations possibles. Un ensemble d'opérateurs basés sur des grammaires adaptées aux limitations imposées par la norme définissant les URL sont proposées aux applications finales pour personnaliser les requêtes. Une solution est aussi exposée pour requêter des agrégations ou des groupes d'agrégation de données à la demande. En utilisant des opérateurs génériques et flexibles, l'API développée permet de répondre à une grande variété de requêtes en implémentant un ensemble limité de fonctionnalités. Les Figures 4.9 et 4.10 illustrent quelques exemples de requêtes.

4.3.1 IMPLÉMENTATION DES AGENTS

Nous avons développé pour la première version de la plateforme trois types d'agents élémentaires. Toutes les informations de haut niveau produites étant le résultat de différentes combinaisons de ces agents.

Le premier type d'agent permet de transformer des flux de valeurs booléennes en flux d'étiquettes qui sont des n-uplets dont le rôle est de représenter un type d'événement dans le temps. Ces étiquettes sont composées d'un horodatage de départ, d'un horodatage de fin et d'une chaîne de caractères. Elles sont des structures intermédiaires importantes pour permettre aux utilisateurs finaux de représenter des emplois du temps, d'établir certaines statistiques ou d'effectuer des opérations de reconnaissance d'activités complémentaires. La source ainsi que la chaîne de caractères assignée à chaque portion de temps peuvent être toutes les deux

personnalisées lors de la configuration de l'agent, ce dernier acceptant un identifiant d'agent source et un type d'étiquette à produire. La Figure 4.11 schématise ce premier type d'agent. On peut y voir à gauche les entrées sous la forme d'un flux de valeurs booléennes comprenant des VRAI sous la forme de flèches ascendantes et des FAUX sous la forme de flèches descendantes. Finalement, à droite sont représentées les sorties sous la forme d'un flux d'étiquettes indiquant les espaces de temps délimités par chaque paire VRAI-FAUX reçue.

Un second type d'agent, décrit au sein de la Figure 4.12, a pour objectif de transformer des flux de valeurs numériques comme un flux de consommation électrique en flux de valeurs booléennes en utilisant un seuil. Quand ce seuil est dépassé, l'agent émet un VRAI booléen, et, à l'inverse, quand la variable numérique repasse sous le seuil, l'agent émet un FAUX booléen. Ce type d'agent est disponible en deux versions acceptant soit une valeur minimum à dépasser pour que l'agent émette un VRAI, soit une valeur minimum à tenir pour que l'agent émette un VRAI. Dans les deux cas, la source de valeurs numériques doit être spécifiée tout comme le seuil.

Finalement, un dernier type d'agent est utilisé pour identifier la présence d'individus dans des pièces. Ce type d'agent utilise plusieurs sources booléennes en entrée et émet un flux de booléens en sortie qui décrit les présences pour une pièce particulière. Ce dernier agent, présenté en Figure 4.13 peut être configuré en utilisant deux listes d'agents équivalentes à une liste blanche et une liste noire en sécurité informatique. Cette implémentation nous permet d'utiliser tous types de capteurs produisant un flux booléen pour détecter la présence d'individu dans des pièces, que ce soit via des capteurs de mouvement, des capteurs de contact, ou même des capteurs de consommation électrique quand leur flux de données numériques est transformé en flux de booléens. L'utilisation de plusieurs types de capteurs permettant d'avoir des informations plus fiables, les capteurs de mouvement pouvant occasionnellement louper,

de par leur comportement interne, certains passages quand ceux-ci sont fréquents.

Des agents plus sophistiqués peuvent aussi être implémentés dans le cadre de cette plateforme, notamment en adaptant certains travaux de la littérature comme celui de Valerio *et al.* (2017), décrivant des solutions pour distribuer des algorithmes sur des réseaux de machines.

4.4 VALIDATION DE L'ARCHITECTURE

L'agrégation de petites unités logiques pour produire de l'information de haut niveau à pour défaut majeur de rendre les opérations de maintenance plus difficiles. En effet, le comportement de chaque composante, ainsi que la structure de la combinaison elle-même, peuvent être des sources d'erreurs. Or, on peut considérer qu'il existe deux types de dysfonctionnement possible au sein de la plateforme : les dysfonctionnements provoquant une interruption, et les dysfonctionnements silencieux. Dans le cas facile où l'agent dysfonctionnel échoue et provoque une interruption, la trace de l'erreur peut permettre de trouver avec plus ou moins de difficulté la source du problème, qu'elle soit au sein de l'agent ayant émis l'erreur, d'un agent en amont ou d'une maison. Ce travail déjà laborieux peut pourtant devenir bien pire dans le cas où l'erreur est silencieuse et où le processus de génération d'informations continue en produisant des données plus ou moins aberrantes. Dans cette dernière situation, si l'erreur est identifiée en bout de chaîne, moyennant un premier délai néfaste où les informations erronées peuvent être utilisées au sein des cliniques, il faut alors identifier l'origine de l'erreur en analysant manuellement le chemin suivi par la donnée erronée sans aucune trace exacte de l'exécution. Quel que soit le scénario, un utilisateur final ne sera de toute manière pas capable d'analyser facilement la source du problème. D'ailleurs, toute mise à jour de ce type de système complexe peut amener à son tour son lot de problèmes, qu'ils soient immédiats ou

à rebours.

4.4.1 DÉVELOPPEMENT ORIENTÉ COMPORTEMENTS

Différentes méthodes ont été proposées dans la littérature pour résoudre ce type de problématique, comme dans le cadre de l'étude de de Sousa Santos *et al.* (2017). Pour notre plateforme, nous avons choisi de la couvrir de tests unitaires orientés comportement, tel que ceux décrits dans l'étude de Lucassen *et al.* (2017). Avec ces tests, il devient possible de garantir que chaque agent se comporte de manière valide quand il est pris indépendamment des autres. Isoler et tester de cette manière le comportement de chaque élément est par ailleurs bien plus simple que de tester le comportement de l'application ou de grappes d'agents au complet pour les raisons présentées précédemment. Aussi, lors de la modification ou de l'ajout d'un nouvel élément, il n'est alors nécessaire que de mettre à jour ou d'ajouter un seul et unique test de part le fait qu'il n'existe aucun couplage entre ceux-ci. Chaque test peut par ailleurs toujours être rejoué pour prouver que la plateforme est fonctionnelle. Finalement, dans le cadre des tests orientés comportement, chaque scénario encode le cahier des charges, ou le standard, décrivant le comportement attendu de chaque agent, une ressource utile pour permettre à des développeurs tiers de travailler sur le projet.

Évidemment, cela vient au prix d'un travail supplémentaire de rédaction des tests, mais aussi de conception, pour garantir que chaque nouvel élément développé n'ait pas de couplage trop important afin de pouvoir être isolé puis testé. Ce travail impacte notablement le plan de recherche et nous supposons que sa durée est inférieure au temps nécessaire pour complètement identifier un problème une fois la plateforme installée. De plus, ce type d'outil permet d'éviter d'autres erreurs majeures liées à l'utilisation d'informations corrompues.

Un test unitaire consiste principalement en un script *Groovy*, technologie décrite dans

les travaux de Kaewkasi & Gurd (2008). Chaque script est basé sur la librairie de test Spock offrant un cadre de développement et des outils d'exécution décrits notamment dans les travaux de Solis & Wang (2011) et sur le site Spock (2019). Chaque script est chargé de la validation d'un élément de la plateforme comme une classe Java et contient une série de fonctions suivant un standard de description permettant de comprendre facilement ce qu'elles sont censées démontrer comme par exemple « Cela produit un événement si l'événement reçu est différent de l'événement qui le précède ou de l'événement qui lui succède ». Chaque fonction d'un script donné peut alors être exécutée indépendamment des autres.

Dans le cadre du test d'un agent, chaque fonction va procéder en trois étapes : une étape d'*initialisation*, une étape d'*exécution* et une étape de *validation*. À l'initialisation, la fonction va instancier tous les objets nécessaires pour opérer le test. Parmi ces objets, on retrouve principalement l'agent et sa configuration, un bus événementiel prévu pour faciliter les tests unitaires, et, éventuellement, des usines ou des constructeurs adéquats pour faciliter la mise en place de certains scénarios complexes. Il se peut aussi que certains tests manipulent l'agent pour que celui-ci soit dans un état adéquat, en lui transmettant, par exemple, un historique de messages préalables. Après l'initialisation vient l'exécution où l'on va utiliser tous les éléments disponibles pour réaliser le scénario que la fonction est censée décrire, ce qui se résume généralement à l'envoi d'un ou plusieurs messages à l'agent testé. Vient alors la dernière étape, étape de validation, où la fonction va effectuer un ensemble d'assertions sur les messages émis par l'agent et sur son état interne.

L'idée clef ici est de profiter au maximum de l'effet domino. Si l'on assume qu'un agent, pris indépendamment, émet une sortie valide pour toutes les entrées valides possibles, alors il est possible de considérer que ces sorties seront les entrées valides d'un autre agent, et ainsi de suite, n'ayant plus qu'en bout de chaîne un journal d'événements attendus. Le résultat final n'étant alors dépendant que de la nature des événements transmis par les sources, il

devient alors possible d'éliminer la plateforme lors de la recherche de problématiques qui sont normalement couvertes par les tests.

4.4.2 COUVERTURE DU CODE AVEC JACOCO

Pour s'assurer au maximum que l'ensemble des agents fonctionnent correctement, les tests sont doublés d'une estimation de la couverture du code via l'outil JaCoCo plus amplement présenté dans le travail de Wielenga (2015) et de Zheng *et al.* (2016). Pour qu'une suite de tests soit valide, il faut donc que celle-ci exécute à la fois l'ensemble des instructions de son programme et l'ensemble de ses structures de contrôle. Bien que cette dernière solution ne garantisse toujours pas d'éliminer l'entièreté des problèmes présents, elle a l'avantage de sortir la totalité des problèmes dûs à un code invalide non exécuté, et notamment les problèmes liés à d'éventuels champs nuls. La couverture de code est aussi un bon moyen de lister les fonctionnalités non testées ainsi que les portions de code mort.

Avec les tests et la couverture de code, il devient donc possible de tester un agent tel que celui décrit au sein de la Figure 4.11 de la Section 4.2.1. Pour cela, il suffit de lister l'ensemble des couples possibles d'entrée et d'état interne de l'agent, puis, de vérifier pour chacun que le comportement de l'agent testé correspond au schéma théorique. Pour l'agent de la Figure 4.11 dont l'objectif est d'émettre des étiquettes en fonction d'un flux de valeurs booléennes, les scénarios ne dépendent que de trois éléments : la valeur booléenne modifiée, la valeur booléenne qui la précède et la valeur booléenne qui lui succède. Il est à noter qu'il se peut que la valeur qui succède ou qui précède la valeur reçue n'existe pas, c'est assez simple à imaginer notamment dans la situation où l'on reçoit la toute première valeur booléenne de la série à transformer. Le Tableau 4.1 résume ainsi l'ensemble des 18 scénarios possibles lors de la réception d'un événement indiquant l'insertion d'une nouvelle valeur booléenne dans la série source.

Tableau 4.1 : Scénarios possibles lors de la réception d'un événement indiquant l'insertion d'une nouvelle valeur booléenne dans un flux de valeurs à transformer en flux d'étiquettes.

Scénario			
N-1	**N**	**N+1**	**Comportement**
VRAI	VRAI	VRAI	Ignorer
VRAI	VRAI	FAUX	Ignorer
VRAI	VRAI	NULLE	Ignorer
FAUX	FAUX	VRAI	Ignorer
FAUX	FAUX	FAUX	Ignorer
FAUX	FAUX	NULLE	Ignorer
NULLE	FAUX	VRAI	Ignorer
NULLE	FAUX	FAUX	Ignorer
NULLE	FAUX	NULLE	Ignorer
FAUX	VRAI	NULLE	Émettre une étiquette non-finie.
NULLE	VRAI	NULLE	Émettre une étiquette non-finie.
FAUX	VRAI	FAUX	Émettre une étiquette complète.
NULLE	VRAI	FAUX	Émettre une étiquette complète.
VRAI	FAUX	VRAI	Scinder l'étiquette courante en deux.
VRAI	FAUX	FAUX	Terminer l'étiquette courante à la valeur reçue.
VRAI	FAUX	NULLE	Terminer l'étiquette courante à la valeur reçue.
FAUX	VRAI	VRAI	Faire démarrer l'étiquette suivante à la valeur reçue.
NULLE	VRAI	VRAI	Faire démarrer l'étiquette suivante à la valeur reçue.

Pour chacun des scénarios du Tableau 4.1, on instancie dans un premier temps un bus d'événements et l'agent à tester. Dans un second temps, on initialise la mémoire interne de l'agent avec les valeurs des colonnes *prédécesseur* et *successeur*. Il suffit alors d'émettre dans le bus l'événement correspondant à la valeur de la colonne *entrée* puis de vérifier que la sortie de l'agent est bien concordante avec le résultat indiqué dans la colonne *comportement*.

4.4.3 ÉVALUATION EMPIRIQUE

Entre le 5 février 2019 UTC (inclus) et le 1er juin 2019 UTC (exclus), la plateforme a reçu un total de 1.636.106 états provenant de capteurs réels installés dans un total de dix

installations hétérogènes en terme d'organisation et d'équipement. Comme présenté dans la Section 4.1, parmi ces premières dix installations, certaines étaient des maisons intelligentes, d'autres des laboratoires et finalement d'autres des environnements émulés. Les environnements émulés sont des logiciels qui ne font que publier un flux d'événements en continu décrivant virtuellement le comportement d'une maison, ce sont des éléments utiles pour réaliser certains tests, et notamment les tests de montée en charge. Toutes les installations réelles utilisaient quant à elles OpenHab avec un plugiciel personnalisé traduisant et transmettant le flux d'événements internes de l'architecture sous la forme d'un flux d'événements intelligibles par la plateforme.

Le Tableau 4.2 récapitule les statistiques en terme de séries temporelles et d'événements reçus et générés pour l'ensemble des environnements suivis. Ainsi, pour chaque environnement, la colonne *Séries* décrit le nombre de séries temporelles existant au sein de chaque installation. La sous colonne *Natives* présente le nombre de séries générées depuis des équipements installés dans chaque environnement, chaque équipement pouvant être responsable de la génération d'une ou plusieurs séries. La sous colonne *Virtuelles*, quant à elle, donne le nombre de séries générées par des agents, chaque agent étant responsable de la gestion d'une seule et unique série.

La colonne *événements* indique le nombre d'événements émis au sein du flux décrivant l'installation. Ces événements, comme les séries, sont séparés en un nombre d'événements dits *Natifs* émis par des équipements au sein de l'installation et des événements dits *Virtuels* émis par des agents.

La colonne *Pièces* présente le nombre de pièces traquées au sein de l'installation, ces pièces pouvant être par exemple des chambres, des salles de bain ou encore des cuisines. Il est possible qu'une installation ait plusieurs instances du même type de pièce. Par ailleurs,

ce nombre de pièces est aussi une bonne estimation du nombre de noeuds appartenant au premier étage de l'arbre décrivant chaque environnement tel que mentionné en fin de Section 4.2.2. Finalement, la colonne *Objets* décrit quant à elle le nombre d'objets traqués au sein de l'installation. Ces objets peuvent être, par exemple, des télévisions, des micro-ondes, des frigos ou des armoires. Comme pour les pièces, cette colonne donne une idée du nombre de noeuds appartenant au deuxième étage de l'arborescence décrivant chaque environnement.

Tableau 4.2 : Statistiques concernant chaque installation traquée entre février et juin 2019.

	Séries		**Événements**			
#	**Natives**	**Virtuelles**	**Natifs**	**Virtuels**	**Pièces**	**Objets**
1	66	66	725.674	414.783	6	12
2	66	57	163.534	143.611	8	10
3	60	70	82.581	86.520	5	15
4	61	75	194.229	170.255	6	14
5	48	65	36.506	63.582	5	11
6	55	67	97.460	75.803	6	12
7	71	81	165.074	119.242	9	14
8	62	63	170.618	117.362	5	11
9	35	40	277	139	6	6
10	43	46	153	110	6	12

Ainsi, la publication par les équipements des différents environnements de 1.636.106 événements a provoqué la génération de 1.191.407 événements supplémentaires décrivant des informations de plus haut niveau par des compositions d'agents tels que ceux décrits au sein de la Section 4.3.1 pour un total de 2.827.513 événements enregistrés en base de données. La Figure 4.14 et la Figure 4.15 décrivent plus précisément la charge de la plateforme en termes d'événements générés et reçus entre le 5 février 2019 UTC (inclus) et le 1er juin 2019 UTC (exclus) pour chaque environnement.

En observant ces deux figures, on peut remarquer que la première installation produit

plus d'événements par jour que les autres. D'ailleurs, elle dépasse même la production du 7ème environnement qui possède plus de capteurs. Cette situation est explicable par la nature multirésidente du premier environnement. Le nombre d'occupants étant plus important, les capteurs sont alors plus souvent activés durant la journée : ils détectent plus de mouvement, et souvent dans deux pièces en parallèle. Il en va de même pour les capteurs de contact et les capteurs électriques qui détectent plus d'interactions avec les équipements de la maison.

Cette première installation a par ailleurs été débranchée entre Février et Avril, ce qui explique le creux qui peut être observé au début de la courbe. C'est par ailleurs la seule installation multirésidente des dix environnements présentés et elle a pour objectif d'accumuler suffisamment de données pour développer des solutions permettant de discriminer les situations où un unique acteur évolue dans l'environnement des situations où plusieurs acteurs évoluent simultanément. Cette information est importante pour les cliniciens qui ont besoin d'isoler les périodes de temps où leur patient est seul afin d'évaluer sa capacité à réaliser certaines activités. Sans cet isolement, la précision des résultats serait par ailleurs bien moins bonne, voire nulle en fonction de la disponibilité ou non de solutions de reconnaissance d'activités en contexte multirésidents.

Pour les environnements plus classiques, on reçoit autour de 2.000 événements par jour. Les grappes d'agents de chaque environnement produisent alors autour de 1.500 événements sur le même laps de temps. Il est aussi possible de remarquer sur la Figure 4.15 que les environnements commencent à générer de l'information à différents moments. Par exemple, l'environnement 4 commence à générer des données en février et l'environnement 8 en mars. Ceci est la conséquence de l'installation en continu des différents environnements, ceux-ci commençant à publier et générer des données une fois qu'ils sont opérationnels.

Finalement, des variations de niveau d'activité peuvent être observées pour chaque

courbe. Dans les faits, ces pics et creux sont dus à des variations journalières du comportement des occupants : les environnements générant, par exemple, plus d'informations quand des acteurs sont présents dans la maison. De la même manière, quand un occupant dort, les capteurs de mouvement restent globalement inactifs. Finalement, certains creux peuvent aussi être la conséquence d'arrêts manuels du système, les différents participants pouvant décider à n'importe quel moment d'arrêter le système de gestion de la maison intelligente.

La Figure 4.16 donne une idée de la capacité de la plateforme à importer des événements depuis chaque environnement suivi. L'histogramme présenté est généré en utilisant un journal décrivant le fonctionnement interne de la plateforme sur une durée d'à peu près une demi-heure. Le fichier décrit le processus d'importation de 9.408 événements. À partir de cette description, nous avons calculé le temps requis pour que l'application traduise chaque événement en une insertion en base de données, ce qui inclut le temps nécessaire aux agents pour produire l'information de plus haut niveau. Il est important de noter que dans cette situation, la plateforme et la base de données coopèrent sur une même machine, ce qui réduit grandement le temps de communication entre ces deux logiciels.

Ainsi, parmi les 9.408 événements importés, un total de 5.169 d'entre eux ont été traités en moins de 20 millisecondes et 9.161 en moins d'une seconde. Les états restants ont pris entre 1 seconde et 2,5 secondes pour être totalement intégrés. Comme attendu, il est évident que l'architecture proposée ne performe pas autant qu'une importation par lots qui profite à la fois du fait que les données arrivent dans l'ordre et qu'aucune donnée ne manque. Il est cependant parfaitement possible, en cas de nécessité, d'intégrer cette dernière méthode en laissant les agents publier et traiter des lots d'événements plutôt que des événements tout court.

4.5 CONCLUSION

L'architecture que nous avons présentée est utilisée par des applications tierces pour générer différents tableaux de bord à destination d'acteurs humains variés. Via notre plateforme, ces applications extraient des informations comme l'organisation de chaque environnement, le taux d'occupation des pièces, les interactions avec les différents équipements ou encore, la nature des activités réalisées au jour le jour. L'aspect temps réel permet de plus à ces applications tierces de mettre à jour leurs différentes représentations dès que de nouvelles informations sont disponibles, ce qui oblige notre application à se plier à certains standards de performance pour assurer une bonne expérience de navigation à l'ensemble des utilisateurs finaux.

Concernant l'enregistrement des différents événements reçus, ceux-ci bénéficieraient grandement d'une modification des méthodes de communication entre les logiciels qui poussent les données et la plateforme qui les reçoit. Pour des raisons de prototypage, c'est via une API REST et le protocole HTTP que les données sont transmises, et, il est évident que les temps de communication seraient grandement réduits si l'on implémentait l'utilisation de protocoles plus rapides et potentiellement moins fiables comme le *Transmission Control Protocol* (TCP), ou mieux, une adaptation du *User Datagram Protocol* (UDP).

Aussi, la plateforme initiale ne parallélisait pas spécialement l'ensemble du processus de génération, cette parallélisation n'étant pas nécessaire. Ceci étant, comme nous l'avons décrit au début de ce chapitre, il serait intéressant de paralléliser le processus de génération en mettant en place une file d'événements par environnement connecté. De manière plus optimale, il serait aussi possible de construire un graphe de dépendance entre agents afin d'isoler le plus de grappes possible et ainsi de paralléliser à l'extrême le processus de génération, d'autres solutions peuvent aussi être implémentées telles que celles présentées dans l'étude de

Bainomugisha *et al.* (2013).

Si, dans les cas extrêmes, l'optimisation du réseau d'agents n'est pas suffisante, il est toujours possible de remplacer l'approche itérative de transmission et de traitement des événements par une approche par lots d'une ou deux secondes. Comme nous ne recevions en moyenne qu'un événement par seconde dans les environnements les plus actifs, cette approche ne nous était pas spécialement bénéfique. Ceci étant, une approche par lots devient indispensable pour traiter certains types de données précis, comme les informations liées au rythme cardiaque.

Pour conclure cette analyse, l'implémentation initiale du projet utilisait uniquement une base de données relationnelle générique (MySQL 8.0.16 ou MariaDB). Il est largement possible d'améliorer la vitesse d'importation des données en remplaçant cette structure par une base de données spécialisée, comme une base de données temps réel par exemple. Ce dernier point reste cependant à démontrer, le changement d'une base de données pour un type plus spécialisé pouvant avoir des impacts importants sur une architecture.

Cette recherche a donc mis en lumière différentes limitations. Parmi elles, la limitation liée au temps requis pour répondre aux requêtes des applications clientes reste la plus importante. En effet, la performance de notre plateforme est fortement liée à la base de données utilisée pour stocker et extraire de l'information. La vitesse de réponse moyenne tend à s'allonger au fur et à mesure que la base de données grandit en taille, ce qui peut s'expliquer facilement à la manière dont les bases de données retrouvent l'information via des arborescences. Dans notre cas précis, ces arborescences pourraient être remplacées par des solutions plus adaptées, notamment en exploitant le fait que les séries temporelles sont les structures de données les plus manipulées dans le système. En dehors des requêtes, cette problématique touche aussi la vitesse d'importation du flux d'événements, bien que comme

expliqué précédemment, une approche par lot puisse être largement bénéfique afin de réduire un peu plus ce temps de traitement précis. Finalement, un travail sur le protocole de communication utilisé pour transférer le flux d'un environnement à l'autre permettrait d'optimiser grandement tout le processus de collecte et de génération de l'information.

Pour ce qui est des avantages de notre approche, il est possible de commencer par traiter de sa versatilité, un de nos points forts majeur. En effet, lors de la mise en place de notre solution, nous avons été capables de suivre dix environnements intelligents totalement hétérogènes incluant différentes pièces et différents équipements. En date du 1er juillet 2021, cette architecture nous aura permis de suivre un total de 37 environnements dont 31 sont toujours actifs ce qui démontre la capacité de notre solution à s'adapter verticalement pour supporter toujours plus d'environnements différents.

Il est aussi possible de traiter de notre capacité à générer et collecter l'information de manière réactive. Tous les environnements présentés ont été installés en continu alors que la plateforme était déjà en ligne. Chaque nouvel environnement étant reconnu et affiché aux utilisateurs finaux aussitôt les premiers événements reçus. Avec une majorité des agents générés automatiquement en accord avec le flux d'événements reçu, les équipes chargées des installations ont pu tester chaque équipement sur place et les recalibrer en conséquence via des interfaces localisées directement en bout de chaîne. Aussi, avec une approche réactive, il devient aisé de modifier ou ajouter de nouveaux équipements en fonction des besoins des cliniciens, la plateforme se reconfigurant automatiquement en conséquence.

Finalement, un dernier avantage de notre solution est liée à son modèle événementiel basé sur le principe CRUD : il devient aussi possible d'ajouter, modifier ou supprimer des données manquantes ou erronées en publiant manuellement des événements dans le flux, ce qui permet à notre plateforme d'être complètement éditable en tout temps.

Nous avons donc présenté dans ce chapitre une architecture qui utilise des grappes d'agents qui coopèrent via la diffusion d'événements basés sur le principe CRUD dans l'objectif d'extraire et de générer de l'information depuis plusieurs environnements intelligents hétérogènes, et cela, de manière totalement réactive. Le flux d'information produit par les environnements sources pouvant alors être diffusé au sein d'un réseau de machines dans l'objectif de distribuer géographiquement le coût total en capacité de calcul. Dans certains noeuds de ce réseau, comme dans le cas spécifique des cliniques, il devient alors possible d'installer des agents spécialisés ayant pour objectif de stocker l'ensemble du flux d'événements dans une base de données spécialisée tout en offrant aux applications tierces un accès facile à toutes les informations collectées et générées. Nous avons aussi décrit, dans une seconde partie de ce chapitre, comment cette architecture a été implémentée et testée en Java en se basant sur le cadre de travail Spring. Finalement, nous avons présenté les résultats extraits de plusieurs mois d'exécution de l'architecture et nous avons discuté, un peu plus haut, des limites et avantages de notre solution. Il est possible de retrouver l'ensemble du code source de la première version de notre solution sur le dépôt du laboratoire : https://github.com/LIARALab. Pour conclure, l'entièreté de ce travail à fait l'objet d'une publication par Demongivert *et al.* (2021a).

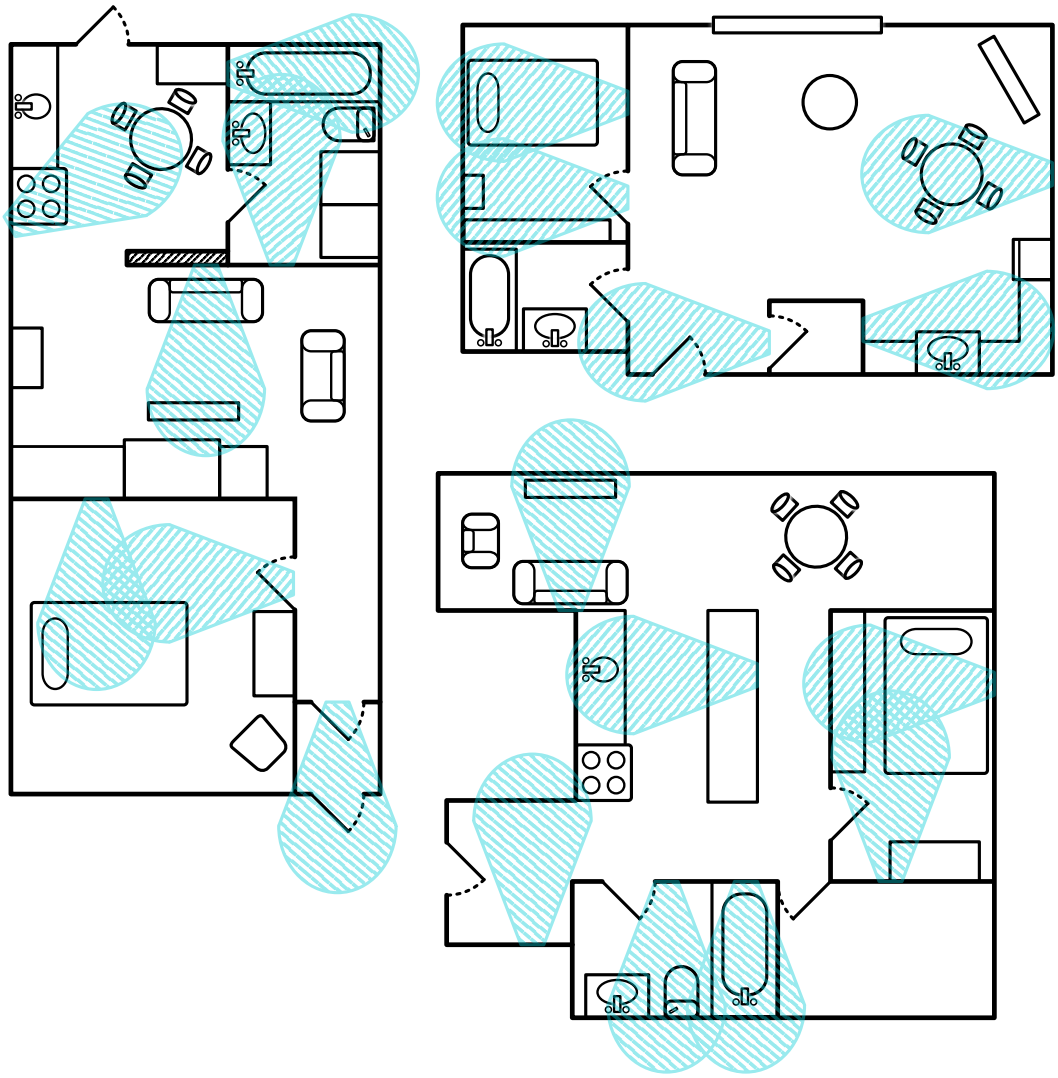

Figure 4.1 : Plan représentant trois installations du projet SAPA, les zones bleues représentent la zone d'activation des capteurs infrarouges.

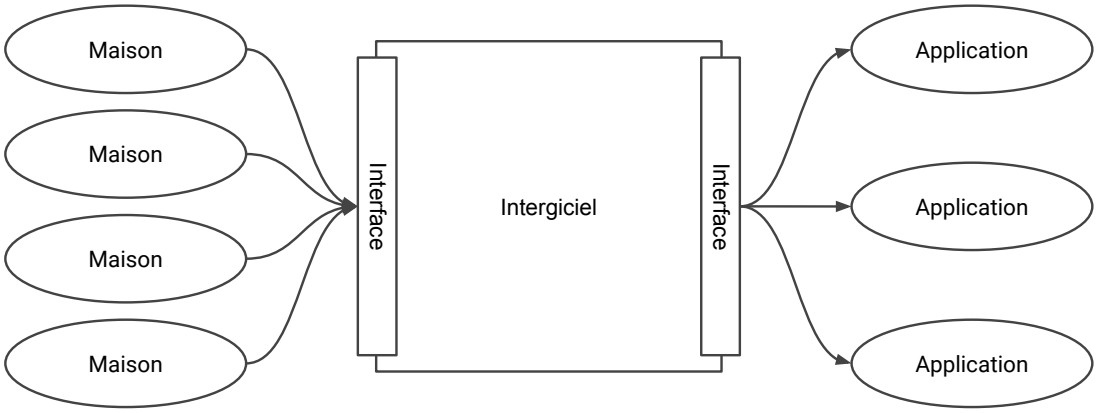

Figure 4.2 : Schéma de l'intergiciel.

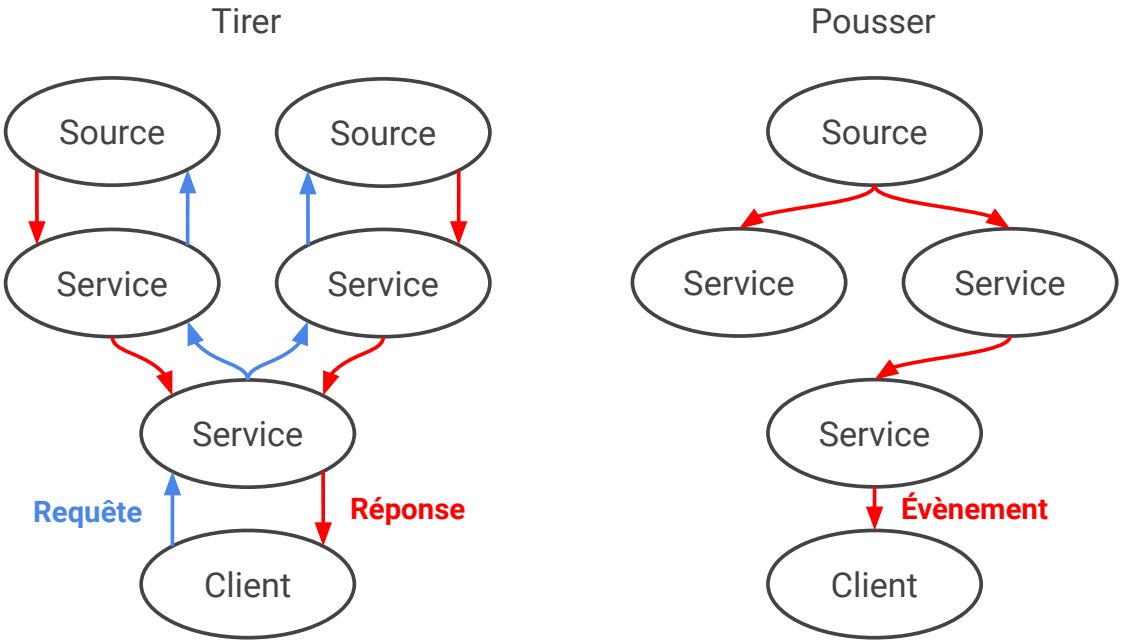

Figure 4.3 : Tirer ou pousser de l'information.

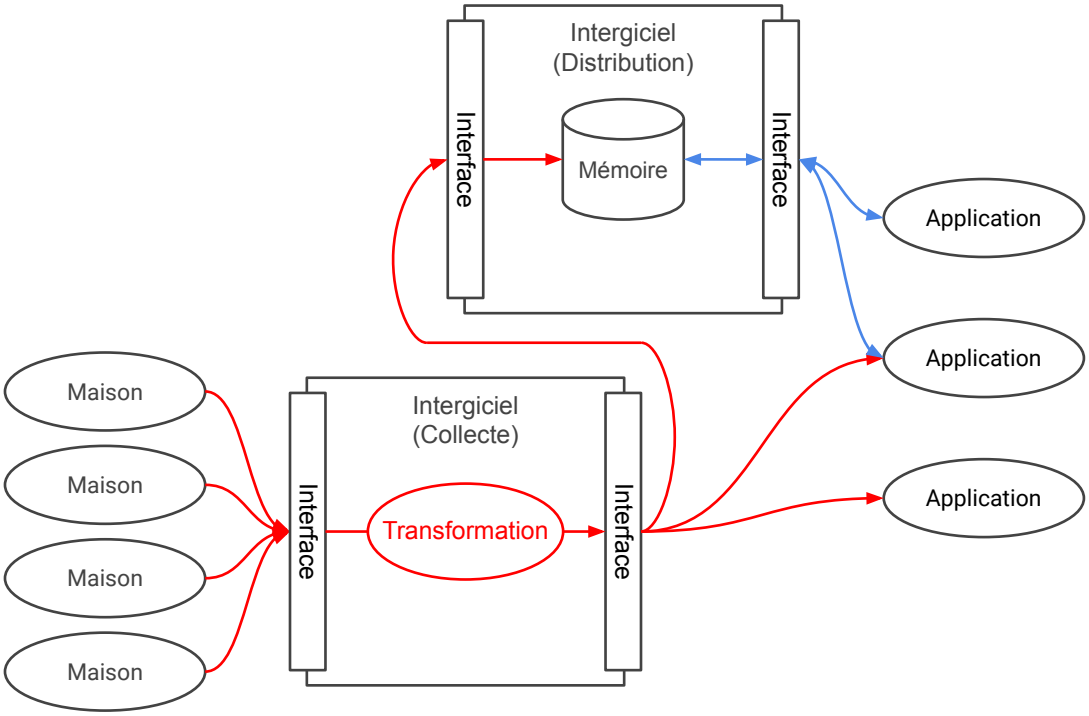

Figure 4.4 : Schéma détaillé de l'intergiciel.

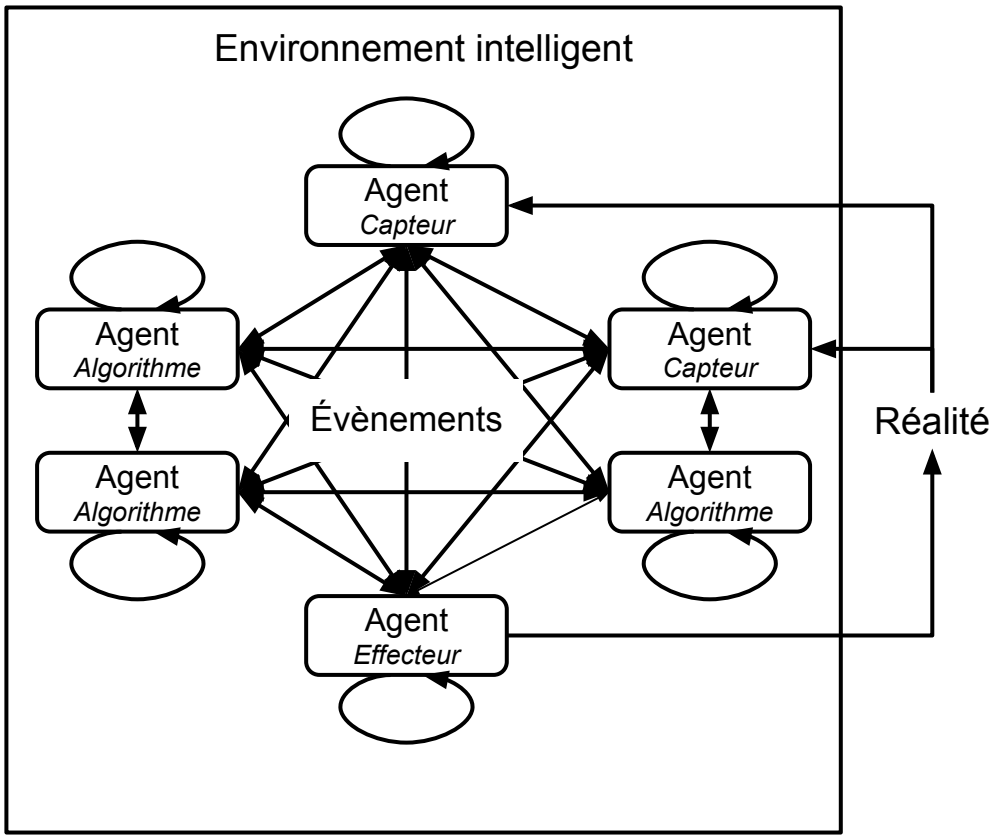

Figure 4.5 : Schéma d'un environnement orienté agent

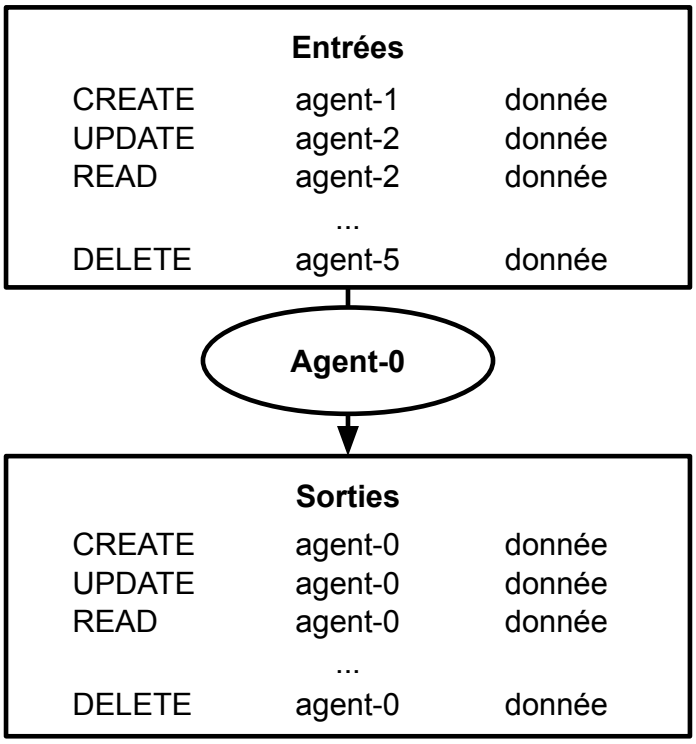

Figure 4.6 : Schéma représentant un agent

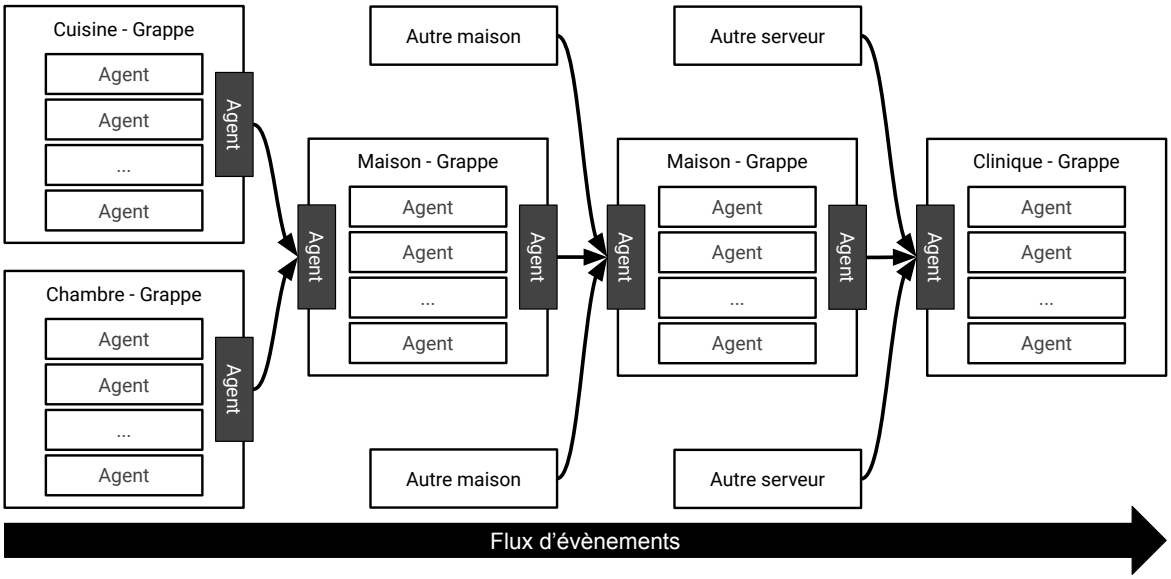

Figure 4.7 : Schéma représentant des grappes d'agents en réseau.

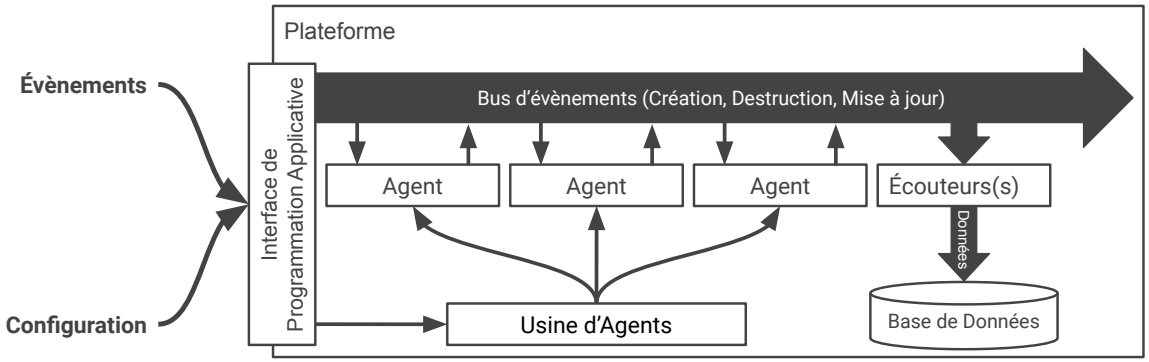

Figure 4.8 : Schéma représentant le bus d'événements de la plateforme implémentée.

```
GET http://domain/                                  ← Action, protocole et nom de domaine
    states<label>?                                  ← Donnée à manipuler, ici des étiquettes
        all&                                        ← Retourner toutes les étiquettes
        orderby=start:asc&                          ← Ordonnées par date de départ
        sensor.node.parents.identifier=0&           ← Des agents de la maison 0
        name=use:fridge&                            ← Retournant des étiquettes d'utilisation du frigo
        duration=gte:5m&                            ← D'une durée supérieure ou égale à 5 minutes
        date=format:(HH:mm:ss'['VV']')
            (10:00:00[America/Montreal]):(14:00:00[America/Montreal])
                                                        ↑
                                    Tous les jours entre 10h et 14h, créneau horaire de Montréal
```

Figure 4.9 : Exemple de requête à l'API permettant d'extraire les différentes étiquettes indiquant l'utilisation d'un réfrigérateur.

```
GET http://domain/                                  ← Action, protocole et nom de domaine
    states<label>/                                  ← Donnée à manipuler, ici les étiquettes
    aggregate?                                      ← Agréger les résultats plutôt que les retourner
        fields=duration(sum)&                       ← Sommer la durée
        sensor.node.parents.identifier=0&           ← Des agents de la maison 0
        name=presence:bathroom&                     ← Retournant des présences dans la salle de bain
        date=format:(yyyy-MM-dd'T'HH:mm:ss'['VV']')
            (2021-08-20T12:30:00[America/Montreal]):(2021-11-20T09:30:00[America/Montreal])
                                                        ↑
                    Entre le 20 Août 2021 à 12h30 et le 20 Novembre 2021 à 9h30, créneau horaire de Montréal
```

Figure 4.10 : Exemple de requête à l'API permettant d'extraire la durée totale d'utilisation des salles de bain d'une maison.

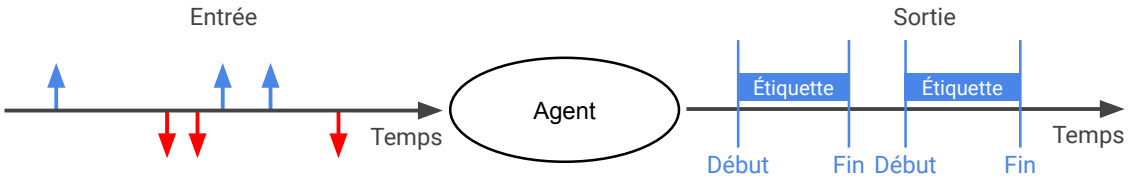

Figure 4.11 : Schéma représentant un agent traduisant un flux de valeurs booléennes en flux d'étiquettes.

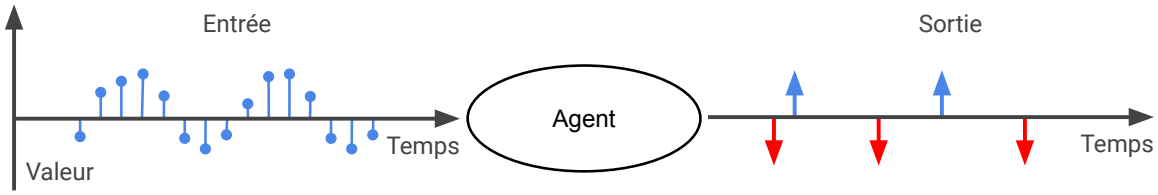

Figure 4.12 : Schéma représentant un agent traduisant un flux de valeurs réelles en flux de booléens.

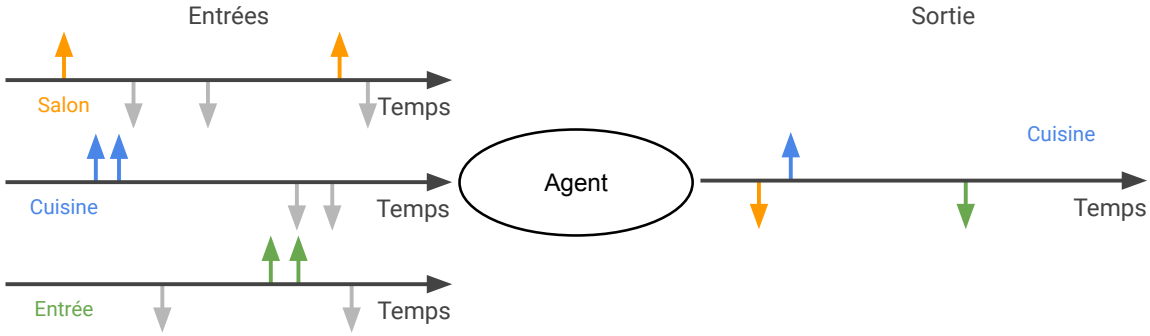

Figure 4.13 : Schéma représentant un agent traduisant des flux de valeurs booléennes en flux de booléens décrivant les présences dans une pièce.

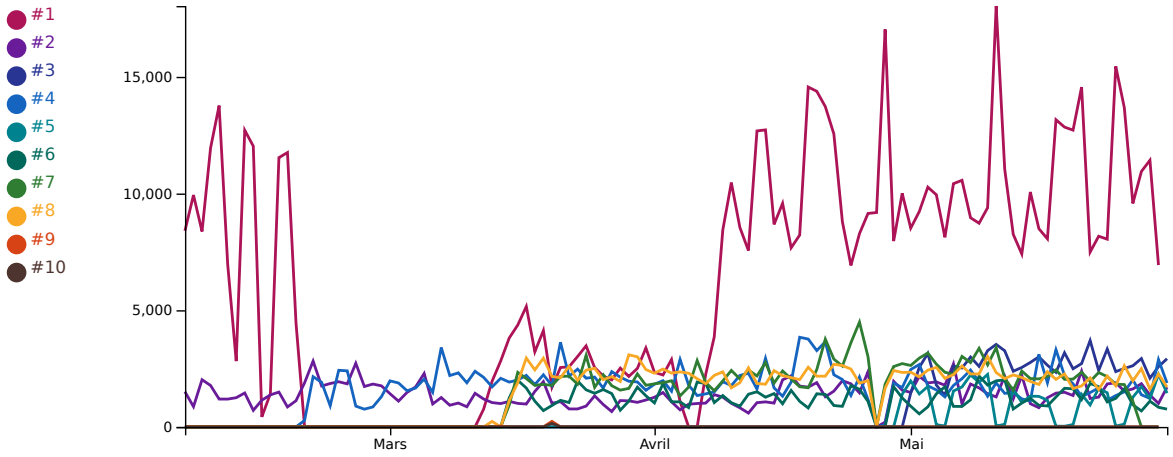

Figure 4.14 : Nombre d'événements publiés par les équipements de chaque environnement par jour entre le 5 février 2019 UTC (inclus) et le premier juin 2019 UTC (exclus) pour chaque environnement suivit.

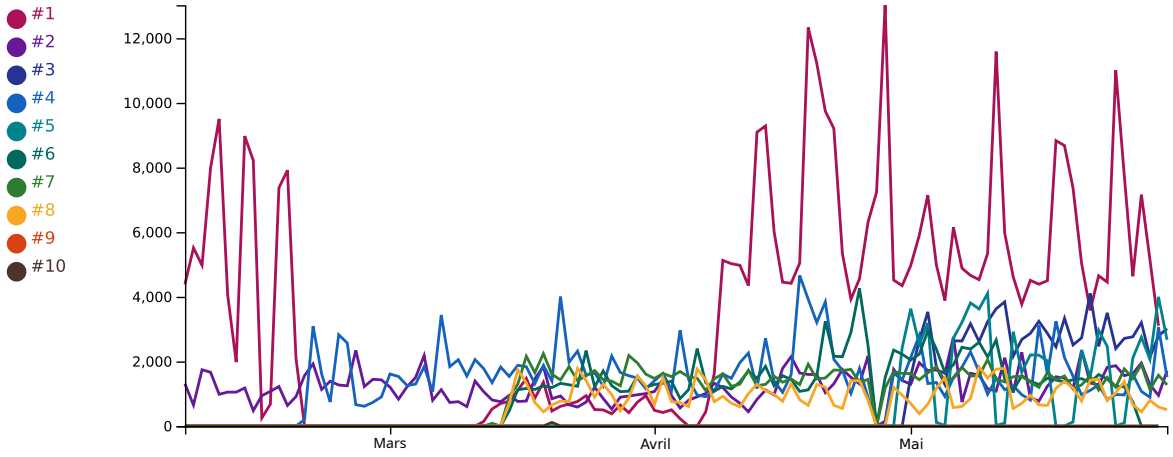

Figure 4.15 : Nombre d'événements générés par agents de chaque environnement par jour entre le 5 Février 2019 UTC (inclus) et le premier Juin 2019 UTC (exclus) pour chaque environnement suivit.

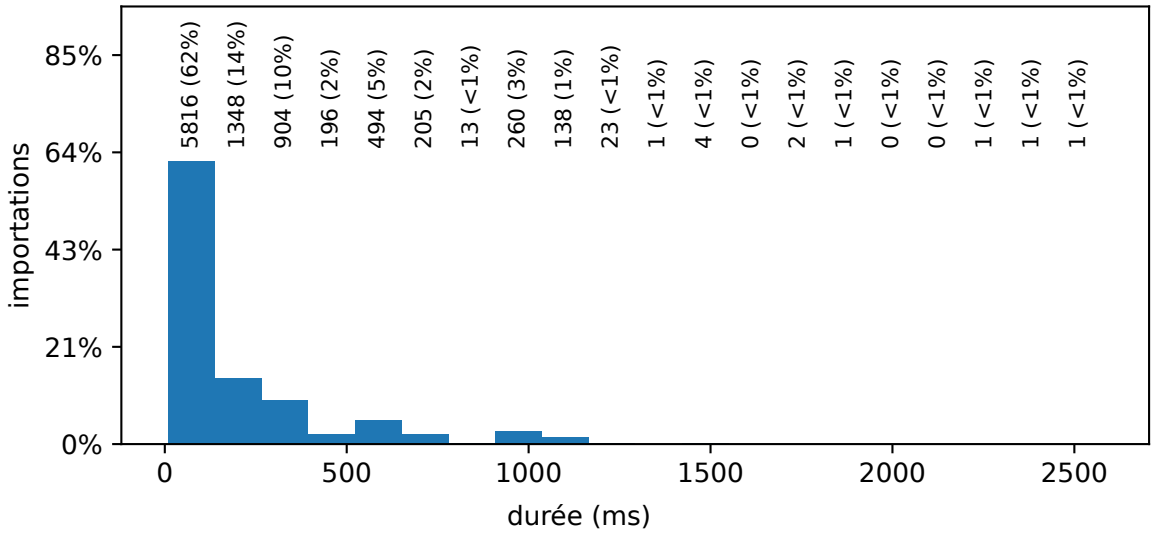

Figure 4.16 : Nombre d'événements importés en fonction de la durée nécessaire pour qu'ils soient enregistrés en incluant le processus de génération des informations de plus haut niveau.

CHAPITRE V
CARACTÉRISTIQUES FLOUES POUR LA GÉNÉRALISATION ET LA QUALIFICATION DES INSTANCES D'ACTIVITÉ

Comme nous l'avons vu dans le Chapitre 1 et le Chapitre 2 de ce document, la recherche sur les environnements intelligents a largement profité du développement et de la commercialisation de nouveaux matériaux, protocoles et logiciels. Tous ces éléments permettant de déployer toujours plus rapidement des infrastructures toujours plus personnalisables. Tout comme l'ensemble des architectures déjà publiées dans la littérature et présentées dans le Chapitre 2, notre propre solution profite, elle aussi, de ces avancées. Ce qui nous a permis de nous éloigner des pures considérations matérielles et de nous pencher plus attentivement sur les problématiques liées à l'adaptation à différents contextes hétérogènes. Et c'est pour répondre à cette problématique que nous avons déployé jusqu'à 37 environnements différents qui sont, aujourd'hui, la source d'autant de jeux de données à explorer.

Ces jeux de données, fruits de plusieurs années d'exploitation de notre solution, peuvent alors être explorés via des outils propres au domaine de l'intelligence artificielle. L'objectif de cette dernière étant de monitorer à distance l'autonomie et la santé de personnes atteintes de maladies dégénératives, sujet que nous avons déjà abordé au sein du Chapitre 1. On peut alors penser plus particulièrement aux solutions de reconnaissance d'activités basées sur l'apprentissage machine, solutions que nous avons présentées dans le Chapitre 3. Ces outils permettant d'extraire des informations abstraites nécessaires aux cliniciens pour monitorer l'état de santé des résidents sur de longues périodes de temps. Cependant, ce type de reconnaissance rencontre vite ses limites quand il s'agit de généraliser des modèles à plusieurs environnements que cela soit causé par la grande hétérogénéité des jeux de données ou que cela soit causé par des adaptations trop importantes aux spécificités de chaque contexte.

Ainsi, si l'on désire maintenir la qualité de la reconnaissance d'activités d'un environnement à l'autre, la plupart des solutions basées sur l'apprentissage machine doivent alors faire l'objet soit d'une réadaptation à chaque nouvel environnement, soit d'un entraînement basé sur tous les environnements disponibles. Or, ces deux options requièrent des jeux de données d'apprentissage complètement annotés afin de procéder à une optimisation. Dans notre situation, quand il faut gérer jusqu'à 37 environnements différents, la production de ce type de jeux de données à la main, et ce, sur de longues périodes de temps n'est pas chose aisée. Et si l'on extrapole un peu, plus le nombre d'installations en parallèle augmente plus cette opération d'annotation devient coûteuse en temps et en ressources. Par ailleurs, cette difficulté à annoter n'est pas propre à notre projet, mais peut aussi s'observer au sein de la littérature scientifique, où un nombre important de jeux de données ouverts viennent sans aucune annotation. Ceux-ci laissant ainsi à la portée de tous une importante banque de ressources seulement partiellement exploitée, voire laissée à l'abandon.

Finalement, de nombreux jeux de données étiquetées ne proposent pas non plus beaucoup d'informations au-delà de la position des instances de certaines classes d'activités. Ceci rendant plus difficile d'estimer la qualité de l'annotation elle-même, ou l'engagement des acteurs dans la réalisation de chaque activité reconnue. Néanmoins, ce type d'information peut rapidement s'avérer utile pour analyser précisément la performance des algorithmes de classification ainsi que les éventuels cas limites.

Dans ce chapitre, nous proposons une solution basée sur l'utilisation de la logique floue pour évaluer la qualité d'un étiquetage automatique de plusieurs jeux de données hétérogènes en transposant de la connaissance d'un environnement annoté par l'humain. Ce travail a, par ailleurs, fait l'objet d'une publication évaluée par un comité de pairs par Demongivert *et al.* (2020).

5.1 ENSEMBLES DE DONNÉES

Comme nous l'avons décrit dans le Chapitre 4, en date du 1er Juillet 2021, le projet SAPA listait un total de 37 environnements différents. Chacune de ces installations ayant produit, en continu, un jeu de données comprenant à la fois une description de la structure de chaque environnement et une représentation des observations réalisées. La structure étant une arborescence et la description des observations un long journal d'événements.

Chaque journal ainsi récupéré est en réalité le regroupement de différentes séries temporelles, toutes générées par des capteurs ou des algorithmes. Chaque série peut alors contenir des données variées, soit de très bas niveau comme la luminosité dans les pièces, soit plus abstraites, comme des interactions avec le mobilier. Les environnements observés étant hétérogènes à la fois dans leurs infrastructures, dans les profils médicaux des participants et dans leurs équipements, il en résulte alors des journaux tous différents les uns des autres.

Ces différences peuvent alors prendre plusieurs formes, avec notamment, celles de modifications structurelles. On parle ici, par exemple, de l'absence ou de l'addition de certaines séries temporelles d'un jeu à l'autre. Ces changements étant notamment dus à des variations d'équipement, de nombre de pièces ou de configuration d'agents. Il est aussi possible d'observer des différences dans la quantité d'observations capturées. Ces dernières étant la conséquence, principalement, de la longévité de chaque environnement et du nombre de capteurs qui y sont installés. Finalement, il y a les différences dans le contexte lui-même, ici affectées par les situations et comportements de chacun. Par exemple, certains individus font très peu la cuisine, d'autres ne dorment quasiment jamais dans leur chambre, d'autres sont plus ou moins victimes de longs moments d'errances dans les appartements, d'autres encore sont plus ou moins sujets à des visites de leurs familles ou de leurs aides-soignants entraînant, dans ces dernier cas, l'apparition temporaire de situations d'observation en contexte multirésidents.

Évidemment, en plus de cette hétérogénéité entre chaque jeu de données, tous ces environnements ne génèrent pas non plus des journaux d'observation parfaits. Et les sources d'erreurs peuvent être nombreuses. Il y a les erreurs d'installation avec, par exemple, les capteurs de mouvement mal placés, détectant alors des individus dans plusieurs zones géographiques différentes plutôt que dans l'unique zone qu'ils sont censés monitorer. Similairement, il peut aussi exister des interactions entre les participants et les équipements, comme la désinstallation de certains capteurs facilement accessibles comme les capteurs de mouvement ou les capteurs de consommation électrique. Aussi, la logique interne de chaque capteur, qu'elle soit physique ou logicielle, peut devenir une source évidente de problèmes avec, par exemple, l'incapacité des capteurs de mouvement à observer efficacement un milieu très animé, ou tout simplement la désactivation de capteurs ayant peu de batteries. Finalement, comme nous l'avons déjà précisé, les moments d'observation multirésidents, notamment lors de la visite d'un aide-soignant, peuvent amener à la création de journaux erronés peuplés de données générées par des algorithmes incapables de gérer ce type de situation.

Finalement, chaque jeu de données obtenu n'est pas annoté. En effet, un parc d'installations aussi varié implique l'allocation de ressources humaines importantes, que ce soit pour installer, désinstaller et réparer les environnements, communiquer avec les équipes cliniques, ou, finalement, pour maintenir la base logicielle existante. Dans ce contexte, il devient évidemment difficile d'allouer plus de moyens pour former et monopoliser des équipes d'aides-soignants supplémentaires afin de renseigner correctement des journaux d'observations au jour le jour. De plus, cela va aussi à l'opposé des problématiques déjà exposées dans le Chapitre 1, l'objectif de ce type de solution étant avant tout de réduire la charge de travail des cliniciens afin que ceux-ci puissent s'investir dans d'autres tâches plus importantes.

Pour conclure, nous nous heurtons donc rapidement à deux problématiques majeures : l'hétérogénéité des jeux de données, nécessitant de réadapter chaque algorithme à chaque

nouvelle configuration d'entrées, et l'absence d'annotation humaine, rendant l'apprentissage supervisé complexe, voire, pratiquement impossible.

5.1.1 COLLECTE DE DONNÉES

Dans le cadre de cette thèse, nous avons accumulé un total de 27 jeux de données, chacun d'entre eux recensant les observations effectuées au sein de différents appartements équipés de capteurs et installés à différents moments entre fin 2019 et début 2021. Au moment de notre analyse, il y avait 20 participants encore monitorés par notre système. Comme nous l'avons déjà présenté précédemment, l'équipement varie d'un environnement à l'autre. Pour être plus précis cependant, chaque maison intelligente contient une composition de capteurs infrarouges de détection de mouvement, de capteurs de contacts, de capteurs électriques et de capteurs d'humidité. Les capteurs infrarouges permettent d'inférer les déplacements et positions des individus dans les pièces des maisons. Pour ce qui est des capteurs de contacts, ceux-ci permettent principalement de monitorer les états des différentes portes et placards et donc de déduire les différentes interactions avec le mobilier. Les capteurs électriques analysent la consommation et permettent de détecter les interactions avec différents équipements tels que des micro-ondes, télévisions, fours et plaques chauffantes. Finalement, les capteurs d'humidité permettent notamment de détecter les potentiels dégâts d'eau. La plupart du temps, la luminosité et la température de chaque pièce sont aussi traquées par le réseau de capteurs infrarouges.

Dans chaque environnement suivi, les capteurs communiquent avec une Raspberry Pi 3 Modèle B+ [30] comprenant une installation personnalisée du logiciel OpenHab. OpenHab sert ici principalement de plateforme permettant de manipuler la configuration de la maison

30. Voir https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/.

au complet en gérant l'ajout, la suppression ou la modification des équipements. La reconnaissance d'activités est réalisée en temps réel par notre architecture en analysant chaque nouveau message reçu de la part de chaque environnement installé. Comme nous l'avons déjà présenté dans le Chapitre 4, la reconnaissance d'activités est assurée par un ensemble d'agents spécialisés émettant et recevant des messages depuis et vers un flux d'événement interne à notre plateforme. Parmi ces agents, certains émettent des étiquettes pouvant représenter des présences dans des pièces, des interactions avec différents équipements de la maison, ou la réalisation d'activités par les résidents. Les activités d'intérêt incluent *cuisiner*, *manger*, *se relaxer*, *dormir*, *hygiène courte*, *hygiène longue*, *inactivité* ou *sortie*. Cette reconnaissance d'activités basique est réalisée via des ensembles de règles pouvant se reposer sur des hypothèses incomplètes. Ainsi, la reconnaissance effectuée n'est pas nécessairement précise et il ne nous est pas possible d'établir une estimation de cette erreur suffisante, notamment de par l'absence d'un étiquetage humain sérieux.

Au moment de la réalisation de notre travail, la base de données servant de stockage pour l'ensemble des observations réalisées accumulait approximativement 16 millions d'entrées brutes provenant des différents environnements connectés. Les 27 jeux de données que nous avons extraits sont tous composés d'un fichier comprenant une liste de tous les événements émis par les capteurs installés dans l'environnement ainsi que d'un fichier comprenant une liste de toutes les étiquettes générées par les algorithmes spécialisés instanciés pour la même installation. Les deux listes recoupant les informations et données émises sur une période de temps équivalente. Les Tableaux 5.1 et 5.2 exposent quelques exemples d'entrées appartenant à chaque jeu de données.

Tableau 5.1 : Exemples d'entrées d'un fichier comprenant la liste des événements émis par les capteurs d'une installation.

Identifiant	Protocole	Série	Sujet	Capteur	Modèle	Émission	État	Pièce	Maison
5fa1...da8d	zwave	027	entrance	battery	fgms001	1553094296466	100	entrance	openHABianPi06
5fa1...da8e	zwave	017	bathroom	luxmeter	fgms001	1553094299700	67	bathroom	openHABianPi06
5fa1...da90	zwave	017	bathroom	temperature	fgms001	1553094299860	26	bathroom	openHABianPi06
5fa1...da93	zwave	027	entrance	pirmotion	fgms001	1553094302132	true	entrance	openHABianPi06
5fa1...da96	zwave	017	bathroom	pirmotion	fgms001	1553094312749	true	bathroom	openHABianPi06
5fa1...bb4d	zwave	001	kitchen	pirmotion	fgms001	1552572970000	true	kitchen	openHABianPi06
5fa1...bb4e	zwave	001	kitchen	tamper	fgms001	1552572970028	true	kitchen	openHABianPi06
5fa1...db08	zwave	024	bedhead	temperature	fgms001	1553094910750	26	bedroom	openHABianPi06
5fa1...db09	zwave	024	bedhead	pirmotion	fgms001	1553094925195	true	bedroom	openHABianPi06
5fa1...df55	zwave	026	clothes2	contact	hsm02	1553107598427	CLOSED	bedroom	openHABianPi06
5fa1...df56	zwave	026	clothes2	contact	hsm02	1553107598470	CLOSED	bedroom	openHABianPi06
5fa1...e1c1	zwave	036	television	consumption_minvalue10	zw096	1553111591273	0	living	openHABianPi06
5fa1...e1c2	zwave	036	television	consumption_minvalue10	zw096	1553111592269	2	living	openHABianPi06
5fa1...59d8	zwave	005	plates2	battery	hsm02	1576689786375	80	kitchen	openHABianPi06
5fa1...59d9	zwave	005	plates2	contact	hsm02	1576689786573	OPEN	kitchen	openHABianPi06
5fa1...59e1	zwave	005	plates2	contact	hsm02	1576689799063	CLOSED	kitchen	openHABianPi06

5.1.2 TRAITEMENT DES DONNÉES

Afin de faciliter l'exploration des jeux de données que nous avons extraits de chaque environnement, nous avons réduit chaque paire de journaux précédemment collectés en journal d'instances. Chaque instance étant l'assemblage de différentes propriétés extraites d'un fenêtrage glissant sur l'ensemble des observations effectuées. Nous avons, de cette manière, généré pour chaque jeu de données, quatre journaux d'instances pour des fenêtres d'une durée de 5, 10, 15 et 20 minutes respectivement. L'activité la plus représentée au sein de chaque instance, parmi les différentes activités reconnues par les algorithmes experts dont nous avons déjà parlé, sert alors d'étiquette. Quand une fenêtre ne coïncide pas avec la reconnaissance d'une quelconque activité, ou qu'elle coïncide avec la reconnaissance d'une activité qui ne nous intéresse pas, elle est alors annotée comme une instance de la classe d'activité *autre*. Au final, les journaux d'instances ne peuvent contenir que des exemples de réalisation d'activités appartenant aux classes *dormir*, *manger*, *hygiène courte* ou *autre*. Ce choix est basé sur les activités communes entre les résidences du projet SAPA et l'ensemble de données Aruba de Cook (2010).

Tableau 5.2 : Exemples d'entrées d'un fichier comprenant la liste des étiquettes émises par les algorithmes d'une installation.

Identifiant	Début	Fin	Type	Pièce	Sujet	Maison
148491900	1552686929000	1552686977000	PRESENCE	kitchen	kitchen	openhabianpi08
147285655	1552686964000	1552686971000	USE	kitchen	plates	openhabianpi08
148690696	1552686964000	1552691870000	ACTIVITY		meal	openhabianpi08
148690696	1552686964000	1552691870000	ACTIVITY		meal	openhabianpi08
148491901	1552686977000	1552686985000	PRESENCE	dining	dining	openhabianpi08
147285656	1552686985000	1552686998000	USE	kitchen	fridge	openhabianpi08
147285658	1552692177000	1552692223000	USE	entrance	door	openhabianpi08
148491914	1552692272000	1552692313000	PRESENCE	living	living	openhabianpi08
148690697	1552692313000	1552697648000	ACTIVITY		rest	openhabianpi08
148690697	1552692313000	1552697648000	ACTIVITY		rest	openhabianpi08
148491915	1552692313000	1552697648000	PRESENCE	bedroom	bedroom	openhabianpi08
146931452	1552692322000	1552692337000	USE	bedroom	bedhead	openhabianpi08
146931454	1552692378000	1552692394000	USE	bedroom	bedhead	openhabianpi08
148491951	1552711327000	1552714156000	PRESENCE	dining	dining	openhabianpi08
148690701	1552711328000	1552714156000	ACTIVITY		inactivity	openhabianpi08
148690701	1552711328000	1552714156000	ACTIVITY		inactivity	openhabianpi08
148491952	1552714156000	1552714239000	PRESENCE	kitchen	kitchen	openhabianpi08
148690708	1552754341000	1552755044000	ACTIVITY		hygiene_long	openhabianpi08
148690708	1552754341000	1552755044000	ACTIVITY		hygiene_long	openhabianpi08
148491980	1552754341000	1552755044000	PRESENCE	bathroom	bathroom	openhabianpi08
148491981	1552755044000	1552755538000	PRESENCE	bedroom	bedroom	openhabianpi08

Les propriétés des instances sont donc des agrégations de statistiques relatives aux différents événements présents dans chaque fenêtre. Le Tableau 5.3 résume l'ensemble des propriétés calculées pour chaque fenêtre sur les observations. Les propriétés *presence*, *sensors :states*, *sensors :variety*, *normalized :sensors :states*, et *normalized :sensors :variety* sont calculées pour chaque type de pièces que contient la maison. Ces types de pièces peuvent être *cuisine*, *salon*, *salle à manger*, *chambre*, *salle de bain* ou *autre* pour les types de pièces plus exotiques. Dans certains des environnements observés, ces *pièces* ne sont pas toujours délimitées par des murs, et peuvent en réalité représenter différentes zones d'une même grande pièce. Ce sont donc plutôt des concepts logiques qui décrivent des espaces spécialisés et non la représentation d'une organisation physique stricte. La propriété *précédent* peut rapidement poser problème dans certaines configurations, elle existe dans les documents pour apprécier l'enchaînement et de la durée de chaque activité lors de l'apprentissage. Cependant, il est beau-

coup trop fréquent que l'étiquette de la fenêtre actuelle, et l'étiquette de la fenêtre précédente, coïncide entre elles, ce qui, en conséquence, génère un biais important. En effet, il suffit alors au modèle d'utiliser systématiquement l'étiquette précédente comme étiquette actuelle afin d'obtenir une justesse supérieure à 90%, les transitions entre activités étant bien plus rares que les continuations. À titre d'exemple, en exécutant un gain d'information sur chaque propriété précédemment citée, la propriété *précédent* est de loin la plus informative, avec un score de 0,7717. À titre informatif, nous avons calculé quelques résultats en utilisant cette propriété mais nous l'avons mise de côté pour la majorité des expérimentations.

Tableau 5.3 : Propriétés extraites pour chaque fenêtre glissante sur les observations.

Champ	Type	Description	Exemple
start	$[0, 24[$	Heure de début de la fenêtre	12.5, 22, 8.3
presence :room	$[0, 1]$	Pourcentage de temps passé dans la pièce	0.1, 0.5, 0, 1
sensors :variety :room	entier	Types de capteurs actifs dans la pièce	1, 5, 8
sensors :states :room	entier	Nombre d'événements émis dans la pièce	125
normalized :sensors :variety :room	réel	Rapport types actifs sur types existants	0.3, 0.8, 1
normalized :sensors :states :room	réel	Rapport états émis sur capteurs	0.8, 2.5, 10.6
précédent	énumération	Étiquette de la fenêtre précédente	other, eating, sleeping, hygiene_short
repeats	entier	Répétitions de la précédente étiquette	1, 3, 8
label	énumération	Étiquette de la fenêtre	other, eating, sleeping, hygiene_short

En dehors de la problématique liée à la propriété *précédent*, tous les journaux ainsi générés sont, de manière générale, plutôt déséquilibrés. En effet, dans chacun d'entre eux, les classes d'activité *autre* et *dormir* sont largement surreprésentées, a contrario de la classe d'activité *hygiène courte*, qui, elle, est largement sous-représentée. Ceci est explicable par le nombre d'itérations et par la durée moyenne de ces différents types d'activités. Alors qu'un individu peut dormir jusqu'à plus de 8 h par jour, une instance d'*hygiène courte* tient sur 5 à 10 minutes. Autrement dit, là où une instance de l'activité *dormir* occupera potentiellement 48 fenêtres de 10 minutes, une instance de l'activité *hygiène courte* tiendra, quant à elle, sur une seule. Pour éviter les biais lors de l'apprentissage, comme celui que nous avons précédemment présenté dans le cas de la propriété *précédent*, nous avons exclu les jeux de données les plus déséquilibrés et les plus inexploitables, n'en gardant ainsi que 11. En effet, sur les 27 jeux

initiaux, certains environnements intelligents étaient très récents, ne laissant ainsi que trop peu d'observations pour équilibrer les journaux d'instances. De plus, certains environnements n'étaient pas des maisons, mais des laboratoires, ce qui n'est pas approprié pour expérimenter sur des cas plus proches du réel.

Parmi 11 environnements que nous avons sélectionnés, les classes les plus sous-représentées peuvent cumuler un nombre d'instances compris entre 200 et 5.000. Afin de pouvoir tester tout de même l'impact du déséquilibrage, nous avons séparé chaque jeu final en deux versions, une équilibrée et une déséquilibrée. L'équilibrage consistant principalement à sélectionner un nombre d'instances de manière aléatoire dans toutes les instances disponibles de telle manière que les classes les plus représentées n'aient pas plus de 25% d'instances de plus que les classes les moins représentées. Bien qu'il soit possible d'affiner cette méthode d'équilibrage, cette dernière est largement suffisante pour nos premières explorations.

5.2 LOGIQUE FLOUE

La logique floue est un ensemble de modèles et de techniques basés sur la définition d'ensembles dits *flous*. Cette notion, introduite dans les travaux de Zadeh (1965), statue que, s'il existe un ensemble d'éléments Ω alors il est possible de définir sur cet ensemble, un ensemble dit *flou* $E \subseteq \Omega$ via une fonction d'appartenance $f_E : \Omega \to [0,1]$ où pour tout élément $e \in \Omega$, $f_E(e)$ indique le degré d'appartenance de e à l'ensemble E. Ainsi, pour tout élément $e \in \Omega$, il devient possible d'exprimer tout un spectre de relations entre la non-appartenance stricte d'un élément $f_E(e) = 0$ équivalente à la notation $e \notin E$ et l'appartenance stricte d'un élément $f_E(e) = 1$ équivalente à la notation $e \in E$. Dans l'article original, cette définition est notamment présentée comme une généralisation de la fonction caractéristique d'un ensemble.

C'est cette approche que nous avons choisie d'utiliser pour explorer les jeux de données

obtenus en Section 5.1.2, notamment parce qu'elle est particulièrement utile pour représenter les propriétés d'une instance de manière qualitative. Via la logique floue, il devient, par exemple, possible de remplacer la représentation d'un horaire précis, sujet à de nombreuses problématiques notamment liées à la position géographique, par une représentation plus abstraite. Ainsi, une instance d'activité démarrant à 10h00 peut alors être représentée comme une instance appartenant à 50% à l'ensemble « des instances débutant le matin » et à 50% à l'ensemble « des instances débutant le midi ». Avec cette approche plus qualitative que quantitative, il devient possible de transférer une logique apprise dans un certain créneau horaire, vers un autre, sans avoir à modifier la nature même des données. En généralisant cette idée, il devient aussi possible de trouver un standard dans lequel exprimer l'ensemble des instances extraites de divers habitats hétérogènes afin de construire et entraîner des modèles génériques applicables partout.

Aussi, si cette représentation est avantageuse pour encoder les propriétés d'une activité, elle peut l'être tout autant pour son annotation. Il devient alors possible d'exprimer qu'une instance particulière est un mélange entre l'activité *cuisiner* et l'activité *relaxer* si la personne prépare son repas en regardant la télévision. Cela permet aussi d'annoter d'autres éléments concernant la réalisation même de l'activité afin d'en compléter sa description. Ainsi, si une instance représente bien l'activité *cuisiner*, il devient alors possible d'étudier si l'exécution de cette dernière était cognitivement intensive, ou non. Par exemple, il semble plus difficile de préparer un repas complet en traitant l'ensemble des aliments et en surveillant plusieurs cuissons, qu'en faisant juste réchauffer un plat préparé à l'avance aux micro-ondes. Or, une tendance à réaliser de plus en plus d'activités simples pourrait être un élément annonciateur de la dégradation de l'état de santé d'un occupant. Finalement, si l'on ne chiffre pas spécialement des caractéristiques plus descriptives, il est à minima possible d'apprécier la confiance que l'on peut avoir dans une reconnaissance via ces mêmes degrés d'appartenance, ce qui peut

être important en fin de parcours pour modérer les résultats présentés dans un rapport.

La logique floue est plutôt expressive, notamment quand on ajoute à la définition des ensembles flous, celle des opérateurs et règles de logique. Cette simplicité peut alors grandement faciliter la communication de certains modèles. Par exemple, au lieu de présenter une instance d'activité comme une période de temps entre 11h45 et 12h12, avec une température moyenne de 25 degrés, une vitesse d'émission des capteurs de mouvements autour de 5 messages par minute et une consommation variant autour de 300 watts, il paraît bien plus simple de considérer qu'un algorithme a détecté une activité particulière parce qu'elle s'est déroulée aux alentours de midi, qu'elle a impliqué une forte chaleur, un fort usage d'électricité et une quantité moyenne de mouvements. Cette approche pouvant même être employée pour permettre à des professionnels de la santé de définir des requêtes ou des règles de reconnaissance simples.

Finalement, cette représentation nous permet aussi de profiter de certains effets de bord. Les degrés d'appartenance ne pouvant être exprimés que dans l'ensemble de valeur $[0, 1]$, ils normalisent ainsi par défaut les instances à classer, ce qui permet de ne pas biaiser certains algorithmes sensibles.

5.2.1 APPLICATION DANS LA RECONNAISSANCE D'ACTIVITÉS

Comme nous l'avons précédemment abordé dans la Section 3.2.4, les solutions de reconnaissance d'activités peuvent être divisées en deux catégories en fonction qu'elles tentent d'exploiter des données ou la connaissance des experts. Pour les approches basées sur les données, il est possible d'exploiter les modèles de Markov cachés comme dans l'étude de Martindale *et al.* (2019), les arbres de décision tels que décrit par Balli *et al.* (2019), les machines à vecteurs de support comme dans le travail de Chen *et al.* (2017), le modèle bayésien naïf tel que présenté par Jiménez & Seco (2018), les réseaux bayésiens comme dans

l'article de Liu *et al.* (2018) ou encore, les réseaux neuronaux tels qu'étudié dans le travail de Xia *et al.* (2020). Finalement, dans le cadre des approches basées sur la connaissance, il est régulièrement publié des articles se basant sur des ontologies comme dans le cadre des travaux de Chen *et al.* (2011) ou de Ye *et al.* (2015).

Malgré la popularité de ces approches, certains des premiers travaux réalisés dans le contexte de la maison intelligente exploitaient aussi la logique floue. On peut notamment citer le travail de Medjahed *et al.* (2009) où les auteurs utilisent des ensembles de règles *si-alors* floues rédigées par des experts en termes vagues, voire imprécis. Ces ensembles permettant, au final, de reconnaître des ADL comme *dormir*, *se lever* ou *faire la vaisselle*. La reconnaissance étant réalisée en floutant des données obtenues d'une simulation de capteurs infrarouges, de capteurs à états et de microphones.

Le Yaouanc & Poli (2012) proposent, quant à eux, un cadre de travail permettant de modéliser des relations spatio-temporelles entre des entités abstraites pouvant représenter des êtres humains comme des objets. Leur solution, basée sur la logique floue et les arborescences de situations telles que décrites dans le travail de Arens & Nagel (2003), leur permet de décrire de nombreuses situations où des objets se rapprochent, s'éloignent ou se tournent autour. Si leur modèle est intéressant pour définir des règles toujours plus complexes, celui-ci reste purement théorique et ne propose aucune expérimentation dans un contexte réel.

Finalement, le travail de Mahmoud *et al.* (2012) utilise la logique floue et l'analyse en composantes principales pour reconnaître des instances d'ADL aberrantes dans le cadre de trois études de cas. La différence entre une activité aberrante et une activité plus standard étant exprimée sous la forme d'un degré d'appartenance à des ensembles flous décrivant différents niveaux d'anormalité.

5.2.2 ENSEMBLES FLOUS SPÉCIFIQUES À NOTRE CONTEXTE

Pour décrire les activités de manière qualitative, nous avons défini cinq catégories d'ensembles flous. Les ensembles de chaque catégorie sont listés au sein du Tableau 5.4. Une instance d'activité particulière peut ainsi appartenir à différents ensembles flous d'une même catégorie en même temps, cependant, la somme de ses degrés d'appartenance à chacun de ces ensembles doit être égale à 1.

Tableau 5.4 : Ensembles flous par catégories.

Heure de la journée
Minuit, Matin, Midi, Après-Midi, Soirée
Durée
Fulgurante, Rapide, Courte, Moyenne, Longue, Étendue, En cours
Activité des capteurs
Silencieuse, Bruyante, Turbulente
Variété des capteurs
Aucune, Pauvre, Moyenne, Riche
Qualité
Irrégulière, Inconsistante, Ambigue, Fiable, Standard

HEURE DE LA JOURNÉE

L'heure de la journée est une information importante pour discriminer les activités car cette dernière peut être particulièrement corrélée à certains types de tâches, voire être liée à des habitudes. Par exemple, il est plus probable qu'une personne cuisine autour de ses heures de repas, comme à midi, que sur ses heures de sommeil, comme à minuit. Cependant, les habitudes ne sont pas universelles et peuvent largement changer d'une culture à l'autre. Certaines irrégularités peuvent aussi exister, et des instances d'activités particulières peuvent apparaître en dehors des heures usuelles. C'est pour ces raisons qu'il est intéressant de transformer l'approche quantitative utilisée habituellement pour représenter les heures de la journée en une approche plus qualitative via des ensembles flous.

DURÉE

La durée est aussi un facteur important permettant d'estimer la nature de certaines activités. Ainsi, nous sommes souvent plus enclins à dormir sur de longues périodes plutôt que sur de courtes. De la même manière, lors de déplacements d'une pièce à une autre, nous ne restons que brièvement dans chaque pièce intermédiaire. Ainsi, tout comme l'heure de la journée, la durée peut aussi largement être exploitée sous la forme d'ensembles flous.

ACTIVITÉ DES CAPTEURS

L'activité des capteurs, ici le nombre de messages reçus de leur part sur un certain intervalle de temps, est aussi un indice intéressant de la réalisation de certaines activités. Ceci est d'autant plus vrai quand les installations sont équipées d'un maillage de capteurs de mouvement. Ainsi, si l'occupant d'une maison dort, il semble peu probable qu'il provoque l'activation de beaucoup de capteurs. Les capteurs de contacts ne changeront donc pas d'état, l'individu ne pouvant interagir avec. Et pour les capteurs de mouvement, si un capteur peut éventuellement émettre un message parce qu'il observe de subtiles variations chez l'individu, la majorité des capteurs de la maison resteront en état de veille. Au contraire, quand un individu cuisine, une grande partie des capteurs présents dans la pièce vont commencer à émettre des observations régulières de ses différents mouvements et interactions avec le mobilier. Utiliser des ensembles pour représenter les moments de la journée où le niveau d'activité est faible ou importante permet ainsi de représenter ces concepts de manière plus qualitative.

VARIÉTÉ DES CAPTEURS

Finalement, la variété des capteurs peut aussi être un facteur important permettant de distinguer certaines activités. Plus précisément, le nombre de types de capteurs différents ayant émis des messages sur un intervalle de temps donné peut permettre de discriminer les tâches impliquant beaucoup d'interactions différentes. Ainsi, quand un individu cuisine, il peut provoquer l'activation de capteurs électriques (quand il interagit avec une cuisinière par exemple), de capteurs de mouvement (quand il se déplace dans la pièce) ou de capteurs de contacts (quand il ouvre et ferme des placards). Au contraire, les déplacements n'activeront certainement que des capteurs de mouvement.

QUALITÉ

La dernière catégorie d'ensembles flous que nous proposons représente la qualité. La qualité est un moyen d'exprimer à quel point nous estimons qu'une instance d'activité se conforme bien à la définition d'une certaine classe d'activités. Cette information est importante pour détecter des changements dans la réalisation des activités monitorées dans le temps. Dans les faits, un éloignement des différentes instances d'activité d'une classe *idéale* pourrait alors être l'indication d'une dégradation potentielle de l'état de santé d'un occupant. Comme nous l'avions déjà mentionné, nous supposons ici que si, par exemple, sur une période de temps donnée, un individu fait moins activement la cuisine, alors peut-être que ce dernier est dans un état ne lui permettant plus de s'investir comme il le faisait initialement dans son quotidien. Finalement, comme la qualité permet d'estimer à quel point une instance est proche d'une définition particulière, cette dernière permet aussi d'estimer la fiabilité d'un étiquetage, voire même, d'en relativiser la nature, en permettant à une instance de s'approcher de plusieurs activités en même temps.

5.2.3 FONCTIONS D'APPARTENANCE

Les ensembles flous n'indiquent pas, en tant que tel, le degré d'appartenance d'une certaine variable exprimée de manière quantitative. À cette fin, il faut accompagner chaque ensemble d'une fonction d'appartenance capable d'indiquer ce degré d'appartenance. On parle alors de *flouter* des variables. Dans cette section, nous allons présenter les différentes fonctions d'appartenance que nous allons utiliser par la suite. Ces fonctions, particulièrement simples, peuvent être altérées de bien des manières afin d'adapter la représentation à chaque situation.

HEURE DE LA JOURNÉE

Toute heure de la journée peut être floutée en degrés d'appartenance aux différents ensembles flous présentés au sein du Tableau 5.4 en associant à chacun d'eux une heure précise de la journée. Comme présenté au sein de la Figure 5.1, il suffit alors de comparer une heure précise à l'heure représentant chaque ensemble pour obtenir le degré d'appartenance.

Il est possible de généraliser cette approche à des périodes de temps données en intégrant et en normalisant les différentes fonctions d'appartenance. Cette opération est relativement simple dans la mesure où toute période couvrant une journée complète donne toujours le même résultat final.

DURÉE

Une durée, exprimée en millisecondes, peut être floutée en degrés d'appartenance aux ensembles décrits au sein du Tableau 5.4 en employant une méthode similaire à celle présentée pour les heures de la journée. Plus spécifiquement, en associant à chaque ensemble flou un

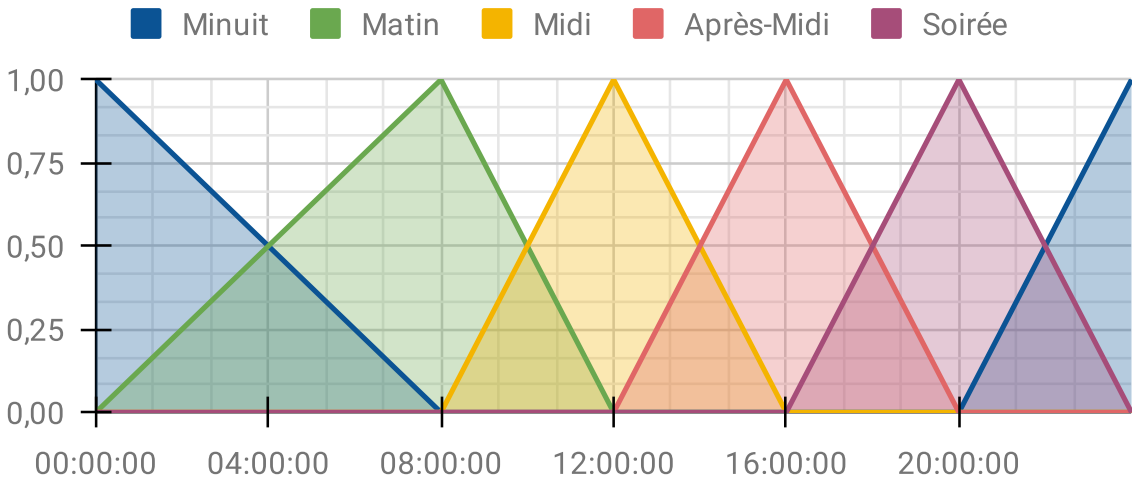

Figure 5.1 : Fonctions d'appartenance décrivant la relation entre différents moments de la journée représentés en abscisse et différents degrés d'appartenance aux ensembles flous précédemment définis représentés en ordonnée.

ensemble de durées valides. Le Tableau 5.5 présente cette configuration avec plus de détails. Chaque fonction d'appartenance résultant de cette approche est présentée au sein de la Figure 5.2, et est définie par la comparaison de la durée d'entrée avec les limites de l'ensemble flou qui lui est associé.

Tableau 5.5 : Configuration des ensembles flous représentant les durées.

Ensemble flou	Durée représentative	Millisecondes
Fulgurante	Entre 0 et 1 seconde	$[\![0, 1000]\!]$
Rapide	Entre 3 minutes et 5 minutes	$[\![180000, 300000]\!]$
Courte	15 minutes	$[\![900000, 900000]\!]$
Moyenne	30 minutes	$[\![1800000, 1800000]\!]$
Longue	Entre 1 heure et 3 heures	$[\![3600000, 10800000]\!]$
Étendue	12 heures et plus	$[\![43200000, +\infty[\![$
En-cours	Uniquement quand la durée n'est pas encore disponible	

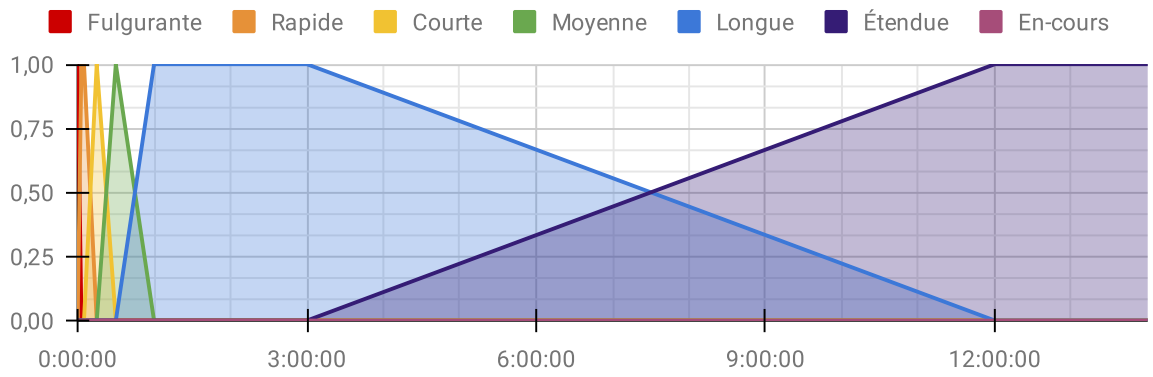

Figure 5.2 : Fonctions d'appartenance à chaque ensemble flou représentant une durée indiquant le degré d'appartenance représenté en ordonnée en fonction de la durée mesurée, représentée en abscisse.

ACTIVITÉ DES CAPTEURS

L'activité des capteurs, exprimée sous la forme du nombre d'événements émis par ces derniers toutes les 5 secondes sur une période de temps donnée, peut être floutée de la même manière que les durées. Les ensembles de valeurs valides sont plus amplement décrits au sein du Tableau 5.6 et les fonctions d'appartenance sont représentées au sein de la Figure 5.3. Il est aussi possible de retrouver chaque ensemble flou au sein du Tableau 5.4. Cette configuration peut être ajustée, si nécessaire, en fonction du nombre de capteurs présents dans l'environnement. Dans notre cas, chaque configuration a été selectionnée en se basant sur des statistiques extraites du jeu de données Aruba de CASAS, publié par l'équipe de Cook (2010).

Tableau 5.6 : Configuration des ensembles flous représentant l'activité des capteurs.

Ensemble flou	Description	Valeurs représentatives
Silencieuse	Aucun message	$\{0\}$
Bruyante	1 message et demi toutes les 5 secondes	$\{1.5\}$
Turbulente	Plus de 3 messages toutes les 5 secondes	$[3, +\infty[$

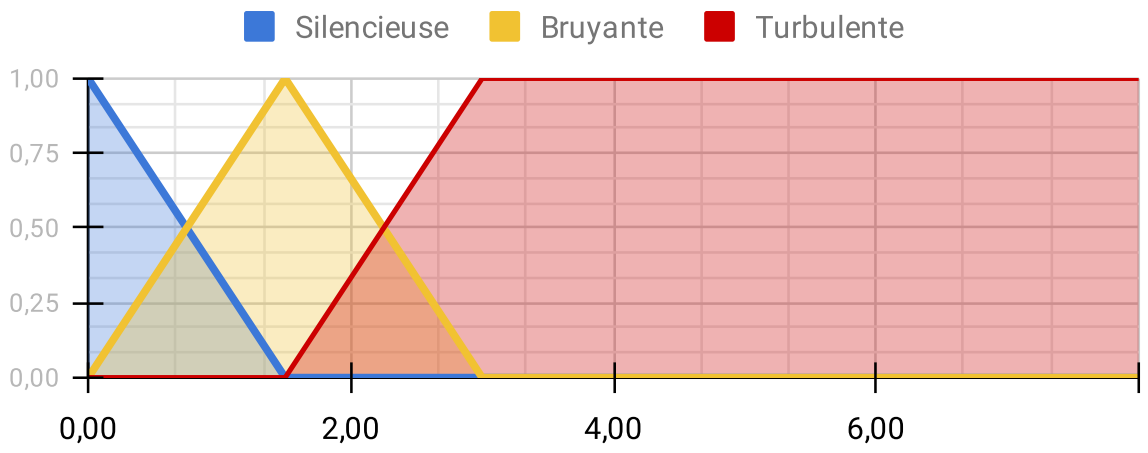

Figure 5.3 : Fonctions d'appartenance de chaque ensemble flou représentant l'activité des capteurs avec le degré d'appartenance en ordonnée en fonction du nombre d'événements reçus toutes les 5 secondes représenté en abscisse.

VARIÉTÉ DES CAPTEURS

Finalement, la variété des capteurs, exprimée sous la forme du nombre de types de capteurs différents ayant émis sur une période de temps donnée, peut aussi être floutée en employant la méthode déjà présentée pour les durées. Les ensembles de valeurs représentatives sont plus amplement décrits au sein du Tableau 5.7 et les fonctions d'appartenance sont représentées au sein de la Figure 5.4. Il est aussi possible de retrouver chaque ensemble flou au sein du Tableau 5.4.

Tableau 5.7 : Configuration des ensembles flous représentant les variétés de capteurs.

Ensemble flou	Description	Valeurs représentatives
Aucune	Aucun type de capteur	$\{0\}$
Pauvre	Un seul type de capteur	$\{1\}$
Moyenne	Entre deux et trois types de capteur	$[2,3]$
Riche	Plus de quatres types de capteur	$[4,+\infty[$

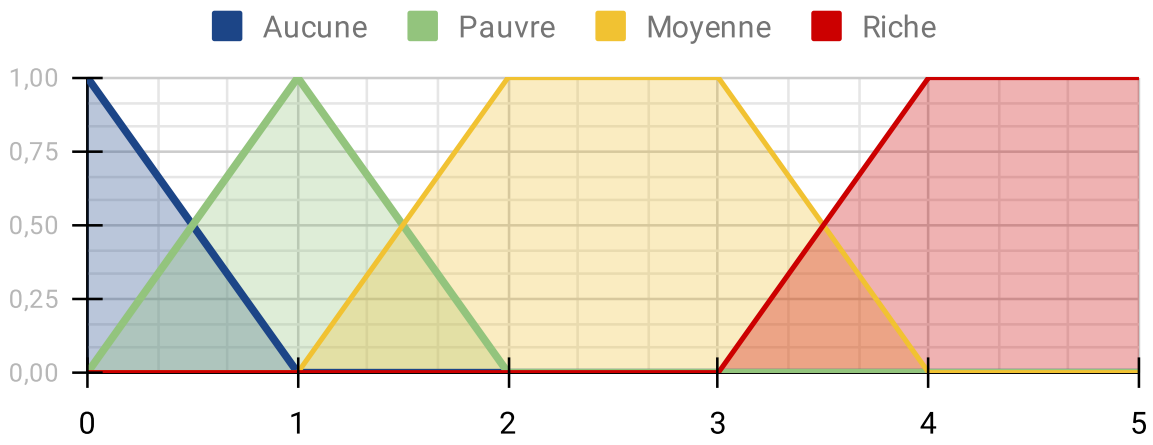

Figure 5.4 : Fonction d'appartenance de chaque ensemble flou représentant une variété de capteur avec le degré d'appartenance en ordonnée en fonction du nombre de types de capteurs différents ayant émis, en abscisse.

QUALITÉ

La qualité est un facteur dépendant de tous les autres ensembles flous déjà présentés. Les degrés d'appartenance sont donc, pour ces ensembles, déterminés d'une autre manière. Plus spécifiquement en utilisant des ensembles de règles de logique floue. Nous avons construit un ensemble par classe d'activité considérée dans notre expérimentation, c'est-à-dire un ensemble pour l'activité *dormir*, l'activité *se relaxer*, l'activité *cuisiner*, l'activité *manger* et l'activité *hygiène courte*. Il existe dans notre modèle une dernière classe *autre* qui n'a qu'un but utilitaire, c'est-à-dire représenter les périodes de temps où toutes les autres activités ne sont pas réalisées.

Chaque règle est alors exprimée sous la forme d'une conjonction d'ensembles flous formant une prémisse suivie d'une conclusion sous la forme d'un unique ensemble flou de sortie. Dans le cas de la qualité, cet ensemble de sortie est systématiquement une qualité telle que nous les avons définis au sein du Tableau 5.4. Ainsi, par exemple, avec les ensembles déjà

présentés, il est possible d'écrire la règle suivante au sujet de l'activité *dormir* :

$$Minuit \wedge Moyenne \wedge Silencieuse \wedge Pauvre \rightarrow Standard$$

Le degré d'appartenance final est calculé en utilisant l'opérateur de conjonction usuel de la logique floue, soit le produit de tous les degrés d'appartenance soit :

$$\forall n \in \mathbb{N} : n > 0, \forall A \in [0,1]^n, and(A) = \prod_{a \in A} a.$$

Pour tout vecteur de degrés d'appartenance A. Ce dernier est appliqué à l'ensemble des degrés d'appartenance d'une instance aux ensembles flous formant la prémisse de chaque règle, donnant ainsi plusieurs résultats intermédiaires concernant chaque conclusion. Le degré d'appartenance final à chaque ensemble représentant une qualité étant alors la somme de tous ces résultats intermédiaires groupés par ensemble flou.

Tous les ensembles de règles que nous avons définis peuvent être retrouvés dans le dépôt suivant : https://github.com/cedric-demongivert/goodtech-2020-rulesets.

5.2.4 SCORE DE QUALITÉ

Ainsi, pour ce qui est de la qualité, le résultat de notre processus quand il est appliqué à des instances d'activités est un vecteur à 5 composantes v tel que $v \in [0,1]^5$. Chacune de ces composantes représentant respectivement un degré d'appartenance aux différents ensembles flous *irrégulière*, *inconsistante*, *ambigue*, *fiable* et *standard*. Il est à noter que dans ce contexte précis, la somme de chaque composante de ce vecteur v est égale à 1.

Ce vecteur de degrés d'appartenance n'est pas évident à manipuler ou à analyser tel quel.

Dans un souci de simplification, nous avons donc traduit ce vecteur sous la forme d'un unique *score de qualité* exprimé dans l'ensemble de valeur $[0,1]$. Cette transformation consiste en un simple produit scalaire entre le vecteur v et le vecteur $\left(0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\right)$. D'autres méthodes peuvent être employées pour réaliser ce type de transformation vers un score et chacune d'elles a des conséquences sur le sens intrinsèque de ce score. La méthode que nous avons choisie implique que la gradation de l'appartenance à chaque ensemble flou est linéaire.

5.2.5 RÉSUMÉ DES ENSEMBLES DE DONNÉES GÉNÉRÉS

Le processus que nous avons suivi pour générer les jeux de données via les informations délivrées par les environnements du projet SAPA est résumé au sein de la Figure 5.5. Comme présenté sur l'image, ce procédé produit 792 fichiers finaux représentant différentes configurations possibles. Le Tableau 5.8 présente avec plus de détails ce nombre de documents en les divisant par configuration. Les jeux de données floutés ajoutent aux composantes de base de chaque instance les degrés d'appartenance aux différents ensembles flous que nous avons déjà présentés. Les jeux de données filtrés, quant à eux, éliminent les instances avec une qualité estimée inférieure à un certain seuil. Notez déjà qu'un jeu de données non flouté et filtré ne peut pas exister, dans la mesure où les degrés d'appartenances sont nécessaires pour estimer la qualité. Finalement, les jeux de données équilibrés sont une sélection d'instances telles que la classe la plus représentée ne dépasse pas 125% du nombre d'instances de la classe la moins représentée.

5.3 APPLICATION À L'ENSEMBLE CASAS

CASAS est un laboratoire reconnu de la School of Electrical Engineering and Computer Science de la Washington State University. Son équipe a, par ailleurs, publié plusieurs en-

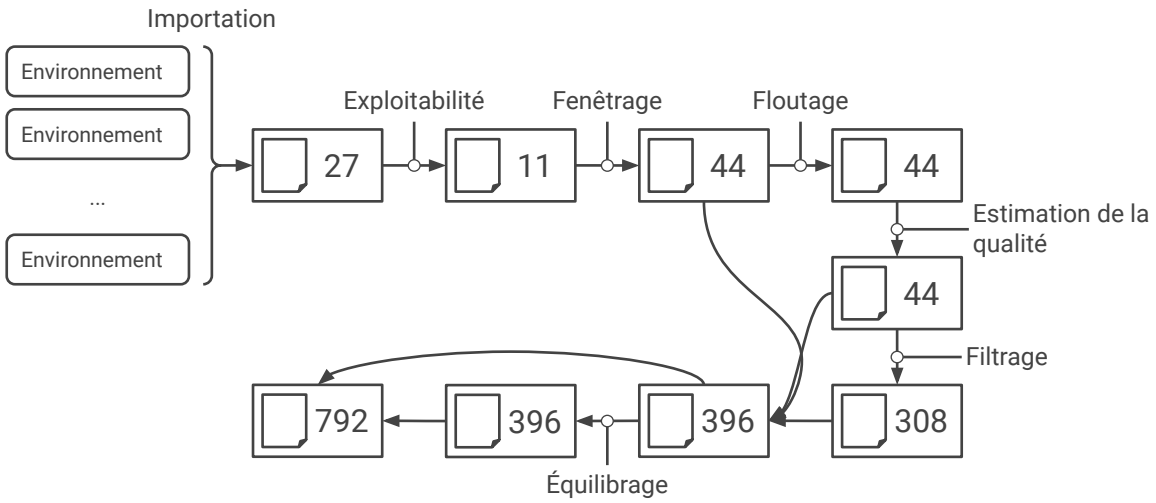

Figure 5.5 : Vue d'ensemble du processus de génération des données.

sembles de données collectées dans différentes maisons intelligentes parmi lesquelles on peut retrouver des jeux de données complètement annotés, c'est notamment le cas de certains jeux de l'article de Cook & Schmitter-Edgecombe (2009) et du travail de Cook (2012). Utiliser un jeu de données extérieur à notre environnement de travail est un bon moyen d'évaluer si notre modèle est suffisamment généralisable pour extraire de la connaissance d'un milieu et l'importer dans un second. Comme le projet SAPA regroupe des environnements extrêmement hétérogènes, il semble donc intéressant d'essayer de se baser sur un travail exogène afin de valider notre approche. Cette dernière devant par la suite être répétable pour nos propres environnements. Finalement, il est important de noter que l'absence d'une observation de terrain nous empêche tout simplement de construire et valider un tel modèle en utilisant directement les ensembles extraits de nos propres environnements. En effet, nos étiquettes sont produites directement par inférence logique, basée sur notre architecture présentée au Chapitre 4.

Pour ces raisons, nous avons choisi d'appliquer notre méthode sur le jeu de données

Tableau 5.8 : Jeux de donnés générés.

Fenêtrage	Floutage	Filtrage	Équilibrage	Quantité
5m	oui	oui	oui	77
5m	oui	oui	non	77
5m	oui	non	oui	11
5m	oui	non	non	11
5m	non	non	oui	11
5m	non	non	non	11
10m	oui	oui	oui	77
10m	oui	oui	non	77
10m	oui	non	oui	11
10m	oui	non	non	11
10m	non	non	oui	11
10m	non	non	non	11
15m	oui	oui	oui	77
15m	oui	oui	non	77
15m	oui	non	oui	11
15m	oui	non	non	11
15m	non	non	oui	11
15m	non	non	non	11
20m	oui	oui	oui	77
20m	oui	oui	non	77
20m	oui	non	oui	11
20m	oui	non	non	11
20m	non	non	oui	11
20m	non	non	non	11
total				792

étiquetées Aruba issu du travail de Cook (2010). Ce jeu de données contient des informations extraites d'un environnement intelligent. Le journal complet décrit la vie d'un adulte volontaire entre le 4 novembre 2010 et le 6 novembre 2011. La maison en elle-même est composée d'une cuisine, d'une salle à manger, d'un salon, de deux salles de bains, d'un bureau, de deux chambres et d'un garage. Au total, 43 capteurs y sont installés incluant 34 capteurs de mouvement, cinq capteurs de température et quatre capteurs pour la fermeture des portes. Finalement, l'étiquetage indique jusqu'à 11 types d'activités différentes incluant notamment *cuisiner*, *se relaxer*, *manger*, *travailler*, *dormir*, *faire la vaisselle*, *hygiène courte*, *entrer dans*

la maison, *quitter la maison*, *faire le ménage* et *prendre un médicament*.

Parmi ces 11 activités, nous avons choisi de n'en modéliser que cinq de sorte que les modèles soient à la fois assez génériques pour être transposés d'un environnement à l'autre, et aussi capable de répondre aux problématiques de nos propres environnements. Nous avons gardé l'activité *cuisiner* représentée par 1606 instances, l'activité *dormir* représentée par 401 instances, l'activité *manger* représentée par 257 instances, l'activité *relaxer* représentée par 2919 instances, et, finalement, l'activité *hygiène courte* représentée par 157 instances. Au total nous avons donc créé cinq ensembles de règles différentes puis nous les avons testées sur un ensemble de 5340 instances.

Si le traitement des données permettant d'estimer la qualité de chaque instance suit le processus que nous avons déjà mentionné précédemment, le prétraitement du jeu Aruba dévie cependant quelque peu. Plus précisément, nous avons procédé, avant toute chose, à une étape d'élimination des irrégularités principalement causées par les champs booléens pouvant être représentés par différents mots-clefs incluant, mais pas exclusivement, « CLOSED », « CLOSE », « OPEN », « ON », « OFF », « OFcF », « OcFF », « O » ou « OF ». Ceux-ci ont simplement été changés pour les représentations « TRUE » ou « FALSE ».

ANALYSE DES RÉSULTATS

Tableau 5.9 : Résultats obtenus sur Aruba via notre méthode.

		Moyenne des qualités					
Activité	**Instances**	**Irrégulière**	**Inconsistante**	**Ambigue**	**Fiable**	**Standard**	**Moyenne des qualités**
Dormir	401	01,59%	00,26%	03,92%	55,74%	38,48%	82,31%
Hygiène courte	157	18,57%	00,00%	00,00%	00,32%	81,11%	81,35%
Se relaxer	2919	00,16%	01,10%	47,37%	33,76%	17,61%	66,89%
Cuisiner	1606	92,07%	00,01%	00,00%	02,82%	05,10%	07,22%
Manger	257	03,00%	00,08%	29,82%	65,07%	02,03%	65,77%
Total	5340	28,59%	00,63%	27,63%	26,63%	16,53%	50,47%

Appliquer notre méthode sur le jeu de données Aruba nous a mené à un score de qualité moyen de 50,47%. Il est d'ailleurs notable que cette qualité moyenne monte à 74,08% si on supprime les étiquettes appartenant à la classe d'activité *cuisiner*. Les résultats sont plus amplement présentés au sein du Tableau 5.9.

Ainsi, les activités *dormir* et *hygiène courte* sont les plus compatibles avec nos règles. Les activités *se relaxer* et *manger* obtiennent, quant à elles, des résultats corrects bien qu'il semble qu'une partie importante des étiquettes soit irrégulière. Plus précisément, les classes d'étiquettes *se relaxer* et *manger* ont un degré d'appartenance moyen respectif de 47,37% et 29,82% à l'ensemble flou *ambigue*. Ce fait indique qu'il existe une certaine divergence entre notre propre définition de ces activités et celle utilisée par l'équipe de CASAS pour réaliser leur annotation.

Pour ce qui est des étiquettes représentant des instances de l'activité *cuisiner*, ces dernières obtiennent un score moyen de qualité de 7,22% et un degré d'appartenance moyen de 92,07% à l'ensemble *irrégulière*. Ceci est principalement la conséquence de l'absence d'une variété suffisante de capteur au sein du jeu de données Aruba par rapport aux nôtres. En effet, ce dernier est la résultante d'une infrastructure ne contenant que des capteurs de température, de mouvements et quelques capteurs de contact pour les portes extérieures. Loin donc de ce qu'il faudrait pour s'assurer qu'il existe des interactions avec les différents appareils électriques et meubles de la cuisine.

La variété est un facteur important dans notre ensemble de règles concernant l'activité *cuisiner*, bien plus importante même que la durée des étiquettes. À titre d'exemple, il est possible d'imaginer une variation de certaines activités *cuisiner* réalisées en fin de matinée telle que celles présentées au sein du Tableau 5.10. En appliquant notre méthode sur cette sélection, il est possible d'observer sur les quatre premiers résultats que le score de qualité

augmente drastiquement en passant simplement d'une variété pauvre à une variété moyenne. Cependant, le score de qualité ne varie pas aussi significativement si l'activité dure plus longtemps, comme le présente les deux dernières lignes. La durée ne joue donc un rôle que pour les activités impliquant une activité des capteurs plus faible.

Tableau 5.10 : Exemples d'activités *cuisiner* avec leur score de qualité respectifs.

Propriétés				
Heure de la journée	**Durée**	**Activité des capteurs**	**Variété des capteurs**	**Qualité**
Midi (80%) / Matin (20%)	Courte (80%) / Rapide (20%)	Bruyante (80%) / Turbulente (20%)	Pauvre (100%)	0%
Midi (80%) / Matin (20%)	Moyenne (80%) / Courte (20%)	Bruyante (80%) / Turbulente (20%)	Pauvre (100%)	0%
Midi (80%) / Matin (20%)	Courte (80%) / Rapide (20%)	Bruyante (80%) / Turbulente (20%)	Moyenne (100%)	100%
Midi (80%) / Matin (20%)	Moyenne (80%) / Courte (20%)	Bruyante (80%) / Turbulente (20%)	Moyenne (100%)	100%
Midi (80%) / Matin (20%)	Courte (80%) / Rapide (20%)	Bruyante (60%) / Silencieuse (40%)	Moyenne (100%)	80,50%
Midi (80%) / Matin (20%)	Moyenne (80%) / Courte (20%)	Bruyante (60%) / Silencieuse (40%)	Moyenne (100%)	87,50%

Si l'on procède à une analyse un peu plus détaillée, les activités *dormir* sont celles qui obtiennent les meilleurs scores de qualité. Cependant, 13 des 401 instances obtiennent un score inférieur à 50%. Chacune de ces 13 instances peut être en réalité identifiée comme un morceau appartenant à une activité plus large, chaque morceau étant séparé par un autre type d'activité très court. Via nos règles de logique floue, il est possible d'extraire les caractéristiques principales de chaque groupe d'activité. Ici, la durée joue un rôle majeur dans l'estimation de la qualité des instances de l'activité *dormir*, ce qui explique ce résultat.

Pour ce qui est de l'activité *se relaxer*, la majorité des étiquettes sont considérées comme ambigües. Plus précisément, environ 72% des étiquettes ont un score de qualité situé entre 50% et 75%. Avec notre méthode, il est possible d'identifier deux scénarios pour ce qui est de ce type d'étiquettes. Le premier scénario est celui où les instances décrivent un occupant qui se déplace beaucoup entraînant ainsi l'activation de nombreux capteurs de mouvements sur une courte période de temps. Cette situation représente un état de l'occupant bien plus actif en opposition avec ce que l'on peut imaginer obtenir pendant une activité de relaxation. Ces instances problématiques coupent en général des instances d'activités plus complexes, comme des instances de l'activité *cuisiner*, et semblent être en réalité l'annotation de pauses entre la

réalisation de deux tâches complexes appartenant à une même activité. Le second scénario quant à lui regroupe des séquences d'étiquettes *se relaxer* séparées par de très courtes activités *autres*. Ces séquences, a contrario de celles appartenant au premier scénario, peuvent aisément être fusionnées pour former une seule activité *se relaxer* à la fois bien plus longue et de bien meilleure qualité.

Les instances représentant des activités *hygiène courte* sont celles qui obtiennent le second meilleur score moyen de qualité avec une valeur de 81,35%. Cette activité représente simplement les déplacements entre la chambre et la salle de bain associée. Il est facile de voir ici que ces étiquettes associent une courte durée à une bonne activité des capteurs, ce qui est en accord avec notre définition de ce type d'activité.

Les activités *cuisiner* sont celles qui obtiennent le pire score de qualité moyen avec une valeur de 7,22%. Comme nous l'avons mentionné plus tôt, ce score est majoritairement dû à une variété de capteurs bien trop pauvre dans Aruba, ce qui est un problème majeur vis-à-vis de notre propre définition. Cependant, il n'existe pas qu'un problème de variété des capteurs pour ces étiquettes, il est aussi intéressant de constater que toutes les instances sont majoritairement courtes, s'étendant ainsi sur quelques minutes seulement. Si certaines instances souffrent du problème de coupure par intermittence décrit dans le cadre des activités *se relaxer*, la majorité d'entre elles sont juste très rapides. Cette information suggère ici que l'occupant de la maison ne se prépare pas des repas très complexes, mais plutôt des plats tout faits ne requérant pas beaucoup d'investissement en terme de temps.

Finalement, les instances représentant l'activité *manger* obtiennent un score de qualité moyen de 65,77% avec une quantité notable d'instances *fiables*. La variation de la qualité des instances représentant ce type d'activité est principalement explicable par leur durée. Les activités les plus irrégulières sont celles qui sont sujettes à des coupures telles que nous les

avons présentées dans le cadre des activités *se relaxer*. Cependant, la plupart des instances sont justes courtes, et décrivent simplement un comportement qui diverge de notre modèle, comme pour les activités *cuisiner*. Ce point appuie, par ailleurs, un peu plus l'hypothèse de la consommation de plats déjà préparés.

5.4 CONCLUSION

Nous avons commencé ce chapitre en traitant en Section 5.1 de la nature des données que nous avons à explorer. Nous avons vu comment ces dernières ont été collectées puis transformées afin d'être analysables. Nous avons aussi traité des problématiques propres à notre contexte de recherche et plus particulièrement de la nature hétérogène des différents environnements que nous avons à gérer ainsi que de l'absence notable d'étiquetage humain permettant d'entraîner des modèles déjà existants.

Dans un second temps, en Section 5.2, nous avons traité d'un modèle basé sur la logique floue pour transférer de la connaissance de jeux de données étiquetés vers les nôtres. Nous avons commencé par introduire très brièvement les éléments fondamentaux de cet outil, puis, nous avons commenté des applications de ce dernier à la reconnaissance d'activités, notamment en analysant des travaux de la littérature.

Finalement, en Section 5.3, nous avons présenté une application de notre approche à un jeu de données publié par le laboratoire CASAS. Nous avons montré comment ce jeu de données nous a permis de produire nos ensembles de règles ainsi que les résultats que nous avons obtenus après leurs applications.

Ainsi, nous pouvons dire que l'utilisation de la logique floue pour analyser des jeux de données étiquetés permet d'augmenter la quantité d'informations extraites de l'annotation originelle. Ceci est, par ailleurs, rendu possible par l'évaluation de caractéristiques addition-

nelles liées à la réalisation de chaque instance d'activité. Ces caractéristiques, par l'utilisation d'ensembles de règles floues *si-alors*, permettent d'établir un modèle formel nous allouant d'estimer la qualité de chaque annotation. Le haut niveau d'abstraction offert par la logique floue ayant la particularité de créer des règles très génériques, potentiellement applicables à d'autres environnements sans avoir à les adapter directement.

Dans notre traitement du jeu de données de CASAS, nous nous sommes principalement focalisés sur l'estimation de la qualité de chaque étiquette. Il semble évident que bien plus de caractéristiques que celles que nous avons proposées peuvent être définies et extraites des différentes instances du jeu de données. Ces caractéristiques peuvent, par ailleurs, servir à produire d'autres métriques permettant de mesurer d'autres aspects de chaque activité, comme le niveau d'implication d'un individu ou le niveau d'originalité d'une instance par rapport à celles déjà observées par exemple.

Reste alors deux tâches importantes couvertes dans le Chapitre 6. La première consistant à établir si ce modèle peut être appliqué à nos environnements, ces derniers étant très différents de ceux opérés par CASAS. Puis, la seconde, visant à s'assurer que l'application de nos règles permet bien de transférer de la connaissance déjà acquise ailleurs. L'application de notre méthode sur le jeu de données Aruba du laboratoire CASAS a fait l'objet d'une publication évaluée par un comité de pairs par Demongivert *et al.* (2020).

CHAPITRE VI

TRANSFÉRABILITÉ DE NOTRE MÉTHODE AU PROJET SAPA

Dans une première partie du Chapitre 5, nous avons fait le lien entre notre infrastructure, présentée dans le Chapitre 4 de cette thèse, et les jeux de données que nous avons à explorer. Nous avons alors décrit la méthode utilisée pour collecter les jeux de données, les transformations qu'ils ont subites ainsi que leurs limites. Et, c'est en étudiant les contraintes inhérentes à la nature de notre problématique que nous avons explicité pourquoi les solutions de la littérature scientifique ne pouvaient pas être directement exploitées. Ces solutions ont été plus amplement traitées dans le Chapitre 3.

Dans une seconde partie du Chapitre 5, nous avons présenté un modèle basé sur la logique floue permettant de représenter des classes d'activités sous la forme d'ensembles de règles floues. Nous avons ainsi présenté les différentes composantes de ces règles, c'est-à-dire des ensembles et des fonctions d'appartenance. Puis, nous avons montré comment il était possible d'utiliser ces règles pour extraire un score de qualité pour chaque instance d'un jeu de données quelconque.

Finalement, dans une dernière partie du Chapitre 5, nous avons appliqué notre approche au jeu de données Aruba de CASAS afin d'en extraire des règles pour des activités très génériques. Nous avons présenté les résultats, une analyse plus approfondie de ceux-ci et les conclusions qu'ils apportent au sujet de ce jeu de données extérieur. Ainsi, nous nous retrouvons avec des définitions pour les activités *dormir* , *hygiène courte* , *se relaxer* , *cuisiner* et *manger* .

Reste alors à explorer nos propres jeux de données et à démontrer, ou infirmer, la capacité de ces définitions à transférer de la connaissance d'un environnement hétérogène à un

autre. Sujet que nous allons aborder dans ce chapitre au sein de deux grandes sections. Dans la première nous présenterons une exploration non supervisée de nos jeux de données permettant d'en établir un profil. Puis, dans la seconde, nous nous pencherons sur une application de notre méthode dans le contexte d'une classification supervisée.

6.1 EXPLORATION NON SUPERVISÉE DES DONNÉES

Avant d'appliquer notre méthode, nous nous proposons d'explorer nos jeux de données par le biais de méthodes non supervisées. Nos étiquetages, résultant de l'application d'algorithmes basés sur des ensembles de règles logiques fonctionnant sans vérité de terrain, ne sont pas fiables, il est intéressant d'essayer de complémenter ceux-ci d'une analyse des éventuels motifs présents dans nos jeux de données. Dans les faits, l'absence de telles structures peut rapidement poser problème. Le manque de motifs ayant tendance à rendre difficile la classification des éléments d'une quelconque manière, en indiquant un éventuel problème dans l'étiquetage ou dans le pré-traitement de nos données.

6.1.1 MESURES D'ÉVALUATION POUR L'APPRENTISSAGE NON SUPERVISÉ

Évaluer la pertinence d'une approche supervisée est un problème complexe pouvant faire l'objet d'un sujet à part entière, comme dans le cadre de l'étude de Zhu & Liu (2021). Or, dans ce type d'approche, on possède toujours un ensemble d'exemples issus d'une observation du terrain, ensemble définissant clairement un objectif à atteindre auquel on peut aisément se comparer. On peut donc imaginer qu'apprécier la qualité d'une partition produite par une approche non supervisée, où cette vérité de terrain n'est pas disponible, peut être particulièrement ardu.

Les conséquences de cette absence d'objectif clair peuvent facilement être observées

en constatant la grande variété des mesures utilisées au sein de la littérature récente. On peut penser, plus particulièrement, à la dizaine de mesures présentées dans chacun des travaux de Benabdellah *et al.* (2019) (indice CH, indice Gamma, indice de Ball-Hall), de Saxena *et al.* (2017b) (somme des erreurs quadratiques, indice de Condorcet, critère de dispersion) et de Xu & Tian (2015) (indice de Rand, indicateur de Jaccard, indice de Fowlkes-Mallows), tous incluant leurs propres avantages et inconvénients. Si l'absence d'une définition unique et évidente du problème explique en partie cette profusion d'approches, il ne faut pas oublier que c'est aussi la conséquence d'une variété importante d'algorithmes produisant chacun des sorties respectant des contraintes bien différentes qu'il faut bien prendre en compte d'une manière ou d'une autre, comme précisé dans l'article de Halkidi *et al.* (2001).

Heureusement, la plupart des travaux récents sur le sujet, comme ceux de Hassan & Rashid (2021) ou de Forte *et al.* (2019), utilisent une catégorisation de ces mesures de validation afin de se positionner plus facilement en fonction des objectifs visés. Si l'on se base sur les articles de Hämäläinen *et al.* (2017) et de Rendón *et al.* (2011), il est donc possible de définir 3 grandes catégories de mesures : celles basées sur des critères externes, celles basées sur des critères internes et celles basées sur des critères relatifs.

Les méthodes basées sur des *critères internes* regroupent toutes les solutions qui n'utilisent que des informations déjà disponibles dans le jeu de données exploré pour évaluer la qualité d'une agrégation. On peut ainsi, par exemple, vouloir mesurer à quel point chaque regroupement est compact tout en étant éloigné de tous les autres regroupements aussi produits. C'est la définition de l'indice de Davies-Bouldin aussi présenté dans le travail de Benabdellah *et al.* (2019) cité plus haut.

Les méthodes basées sur des *critères externes* regroupent toutes les solutions qui utilisent des informations obtenues au préalable, informations donc *extérieures* au jeu de données.

On peut, par exemple, penser à une classification réalisée par un expert. Dans ce cas précis, on peut vouloir mesurer à quel point chaque regroupement effectué est pur, c'est-à-dire, ne contenant que des éléments appartenant à une seule et même classe. C'est le principe derrière la mesure de pureté qui sera traitée dans quelques lignes.

Finalement, les méthodes basées sur des *critères relatifs* vont, quant à elles, se baser sur des comparaisons entre les résultats de plusieurs algorithmes d'agrégation, ou du même algorithme mais exécuté avec différentes configurations de paramètres. C'est par exemple la catégorie de la méthode visant à extraire une partition en comparant les étages d'un regroupement hiérarchique.

Dans le cadre de notre travail, l'objectif consiste à pouvoir mesurer la qualité de notre étiquetage. Ainsi, les méthodes basées sur des *critères externes*, critères qui seront plus amplement décrits plus bas, semblent les meilleures dans la mesure où ces dernières nous permettraient de comparer les résultats de différents algorithmes à notre étiquetage. Un désaccord entre ce dernier et le résultat d'une agrégation automatique pouvant potentiellement indiquer un problème dans notre annotation, ou du moins, dans le choix des propriétés de nos instances (heure de la journée, durée, variété des capteurs, etc.).

PURETÉ

Si l'on se base sur les articles que nous avons déjà présentés et sur le travail de Amigó *et al.* (2009), l'indice de pureté peut être défini comme une valeur indiquant à quel point chaque regroupement effectué ne contient que des instances d'une seule et unique classe. En bref, plus chaque agrégation est bruitée, c'est-à-dire, plus elle contient d'instances de différentes classes, plus l'indice de pureté est faible.

Plus formellement, si l'on considère un ensemble d'instances Ω, un ensembles d'activités $A = \{a_1, a_2, ..., a_n\}$, une classification $C : \Omega \to A$ et P_Ω une partition de Ω à évaluer. Alors, l'indice de pureté peut être défini à l'aide de l'équation (6.1).

$$\text{Pureté} = \sum_{p \in P_\Omega} \frac{\text{Card}(p)}{\text{Card}(\Omega)} \left(\max_{a \in A} \frac{\text{Card}(\{x \in p : C(x) = a\})}{\text{Card}(p)} \right) \tag{6.1}$$

Évidemment, l'indice de pureté connait des limites. Pour citer directement Amigó *et al.* (2009), cet indice pénalise les ensembles bruités, mais ne privilégie pas les situations où toutes les instances d'une même classe sont regroupées dans un même ensemble. Ainsi, il est trivial de constater qu'au sein de l'équation (6.1), si chaque ensemble $p \in P_\Omega$ ne contient qu'un unique élément, autrement dit que $\text{Card}(p) = 1$, alors l'indice de pureté est aussi égal à 1.

Un déséquilibre dans la représentation de chaque classe peut aussi impacter l'indice de pureté car il devient alors plus probable qu'une partition donnée contienne une classe particulière plutôt qu'une autre. Si ce déséquilibre est trop grand, de nombreuses partitions risquent ainsi d'êtres pures par défaut impliquant une valeur plancher au score de pureté.

Dans notre contexte d'utilisation, ces deux problématiques peuvent advenir en fonction des algorithmes utilisés et des jeux de données explorés. Cependant, cet indice est intéressant pour observer un éventuel défaut dans l'annotation ou dans le choix de propriétés, une telle situation entraînant inévitablement la production d'agrégats comprenant des instances appartenant à différentes classes, et donc, une baisse de l'indice pureté.

INDICE DE RAND

L'indice de Rand (1971), du nom de son auteur, est une mesure de la similarité entre deux partitions d'un ensemble. Si l'on se base sur le travail plus récent de Steinley & Brusco

(2018), cet indice représente le rapport entre les paires d'objets pour lesquelles les deux regroupements comparés adoptent un comportement équivalent, sur l'ensemble de toutes les paires d'objets possibles.

Plus spécifiquement, si l'on considère un ensemble d'instances Ω, et deux partitions X et Y de ce même ensemble, alors, pour toutes paires d'objets (a,b) parmi l'ensemble des $\binom{\text{Card}(\Omega)}{2}$ paires d'objets de Ω possibles (ici, la notation $\binom{n}{k}$ désigne le coefficient binomial), il existe quatres scénarios possibles :

1. Les objets a et b sont présents dans un même groupe au sein de la partition X et dans un même groupe au sein de la partition Y. Parlons ici de *Vrais Positifs* (VP).
2. Les objets a et b sont présents dans un même groupe au sein de la partition X et dans deux groupes différents au sein de la partition Y. Parlons ici de *Faux Positifs* (FP).
3. Les objets a et b sont présents dans deux groupes différents au sein de la partition X et dans un même groupe au sein de la partition Y. Parlons ici de *Faux Négatifs* (FN).
4. Les objets a et b sont présents dans deux groupes différents au sein de la partition X et dans deux groupes différents au sein de la partition Y. Parlons ici de *Vrais Négatifs* (VN).

Ainsi, dans les scénarios 1 et 4 de la liste précédente, les deux partitions X et Y adoptent des comportements similaires, soit en regroupant les deux objets, soit en les séparant. Il devient donc possible de définir l'indice de Rand avec l'équation (6.2).

$$R = \frac{VP + VN}{VP + FP + VN + FN} \tag{6.2}$$

Comme précisé dans l'étude de Kayaalp & Erdogmus (2020), l'indice de Rand peut être influencé par la quantité de données et de regroupements testés. Toutefois, dans notre contexte,

comme imaginé par l'auteur original, notre nombre de regroupements devrait rester limité en regard du nombre d'instances (une dizaine d'étiquettes pour plusieurs milliers d'exemples) ce qui ne devrait pas trop déséquilibrer les résultats.

INFORMATION MUTUELLE NORMALISÉE

L'information mutuelle normalisée est une autre mesure permettant d'établir la similarité entre deux partitions d'un même ensemble. Comme présenté au sein du travail de Newman *et al.* (2020), c'est une mesure qui va se baser sur des principes propres à la théorie de l'information pour tenter d'établir à quel point, la connaissance d'une des partitions d'une paire, peut permettre d'en déduire l'autre.

Le calcul de ce score va s'effectuer en deux temps : premièrement, il faut calculer ce que l'on appelle l'information mutuelle, puis, dans un second temps, choisir une méthode pour normaliser ce résultat. Il est bon de noter qu'il existe différentes manières de normaliser l'information mutuelle. L'article de Vinh *et al.* (2010) procède à une description plus détaillée de ce processus. Nous n'expliquerons pas ici les détails concernant cette dernière étape pour des raisons de concision.

Si l'on considère un ensemble d'instances Ω, et deux partitions X et Y de ce même ensemble, l'information mutuelle est alors la différence entre l'entropie de la partition X et l'entropie conditionnelle de X sachant Y. Autrement dit, le gain d'information que l'on obtient en *logits* si l'on doit décrire Y à un interlocuteur connaissant déjà X. Si ce gain est nul, alors les deux ensembles sont dit indépendants, et n'ont rien à voir entre eux. Si ce gain est maximal, alors les deux ensembles sont totalement équivalents.

Plus précisément, si l'on peut définir $\mathbb{P}(a)$ la probabilité qu'un élément $\omega \in \Omega$ appar-

tienne à l'ensemble a d'une des partitions X ou Y et $\mathbb{P}(x,y)$ la probabilité qu'un élément $\omega \in \Omega$ appartienne à l'ensemble $x \in X$ et à l'ensemble $y \in Y$, alors, selon Newman *et al.* (2020), l'information mutuelle est déterminée par l'équation (6.3).

$$I(X,Y) = \sum_{x \in X, y \in Y} \mathbb{P}(x,y) \log \frac{\mathbb{P}(x,y)}{\mathbb{P}(x)\mathbb{P}(y)} \tag{6.3}$$

De manière équivalente aux mesures précédentes, l'information mutuelle normalisée trouve ses limites dans le cas extrême où tous les ensembles d'une partition ne contiennent qu'un unique objet. Finalement, c'est une mesure qui a l'intérêt d'être parfaitement symesure, c'est-à-dire que $I(X,Y) = I(Y,X)$.

INDICE DE FOWLKES-MALLOWS

L'indice de Fowlkes-Mallows, du nom des deux auteurs de l'article original sur le sujet Fowlkes & Mallows (1983), est une autre mesure de la similarité entre deux partitions d'un ensemble que nous nous proposons d'utiliser. Contrairement aux mesures de similarité précédentes, cet indice a été introduit comme une mesure de la similarité entre deux agrégations hiérarchiques. Cependant, l'indice de Fowlkes-Mallows peut être facilement généralisé pour évaluer des agrégations simples, comme précisé dans l'article de Bihari *et al.* (2019).

Selon Wagner & Wagner (2007), l'indice de Fowlkes-Mallows peut être vu comme la moyenne géomesure entre la précision et le rappel d'un jeu de données annoté. Plus précisément, si l'on considère un ensemble d'instances Ω, et deux partitions X et Y de ce même ensemble, il est alors possible de calculer l'indice de Fowlkes-Mallows avec l'équation (6.4).

$$FM(X,Y) = \frac{T(X,Y)}{\sqrt{P(X,Y)Q(X,Y)}} \tag{6.4}$$

où

$$T(X,Y) = \sum_{x \in X} \sum_{y \in Y} m_{xy}^2 - \text{Card}(\Omega)$$

$$P(X,Y) = \sum_{x \in X} \left(\sum_{y \in Y} m_{xy} \right)^2 - \text{Card}(\Omega)$$

$$Q(X,Y) = \sum_{y \in Y} \left(\sum_{x \in X} m_{xy} \right)^2 - \text{Card}(\Omega)$$

et $m_{xy}, \forall (x,y) \in X \times Y$ représente le nombre d'éléments en commun entre l'ensemble x de la partition X et l'ensemble y de la partition Y.

Comme précisé dans l'article de Wagner & Wagner (2007), l'indice de Fowlkes-Mallows a le désavantage notable de tendre vers 1 quand le nombre de groupes comparés est faible. Ce désavantage doit être pris en compte ici dans la mesure où nos ensembles sont annotés avec peu de classes. De plus, certaines étant bien plus présentes que d'autres, ce déséquilibre peut aussi induire une amélioration notable de ce *score*.

6.1.2 ALGORITHMES POUR L'EXPLORATION

Pour cette analyse non supervisée de nos jeux de données, nous avons choisi d'appliquer 4 algorithmes d'agrégations différents, tous couramment utilisés dans la littérature scientifique.

Le premier d'entre eux, K-Moyennes, est un algorithme très commun pour l'étude du regroupement que l'on peut notamment retrouver dans le travail de Yuan & Yang (2019). L'objectif de cet algorithme est tout simplement de minimiser la somme des distances au carré entre chaque instance du jeu de données exploré et k points représentant chacun un

regroupement à produire. Pour y parvenir, l'algorithme divise le jeu d'instances en k-groupes de points en fonction de leur proximité avec les centroïdes. Ensuite, il calcule le nouveau centre de masse de chaque groupe et met à jour chacun des k points en conséquence. Au gré des itérations, les k centroïdes dérivent alors en direction des masses d'instances les plus importantes. K-Moyennes a le désavantage notable de requérir le nombre d'agrégations à produire à l'avance.

Le second algorithme que nous nous proposons d'utiliser est *Mean-Shift*, plus amplement décrit dans le travail de Wu & Yang (2007). L'objectif de cet algorithme est d'extraire les modes, ou valeurs dominantes, d'une distribution d'instances en se basant sur une estimation par noyau de la densité de probabilité de ce dernier. Comme le précise l'auteur, trouver les valeurs dominantes d'une densité de probabilité revient à extraire les positions les plus peuplées d'un ensemble d'instances. Ces positions pouvant alors être considérées comme des estimations des centroïdes de différents agrégats.

Le troisième algorithme que nous souhaitons utiliser est DBSCAN, notamment présenté dans le travail de Hahsler *et al.* (2019). DBSCAN est une méthode d'agrégation basée sur la densité de points. Plus spécifiquement, pour chaque instance d'un ensemble à explorer, cet algorithme définit un ensemble de points dits voisins comprenant tous les points présents dans un certain rayon autour du centre considéré. Si ce nombre de voisins est supérieur ou égal à un certain seuil, le point est alors considéré comme appartenant au corps d'un agrégat. Si un point ne possède pas assez de voisins, mais est dans le voisinage d'une instance appartenant à un agrégat, il est alors considéré comme une bordure. Finalement, tous les autres cas sont considérés comme du bruit. En se basant sur ce modèle, DBSCAN construit les agrégats en naviguant à travers chaque point par le réseau de voisins. Une fois la procédure achevée, les points non assignés à un agrégat sont considérés comme du bruit.

Finalement, le dernier algorithme que nous souhaitons utiliser est OPTICS, plus amplement présenté dans l'article de Subudhi & Panigrahi (2019). OPTICS fonctionne sur le même modèle que DBSCAN mais ne produit pas des agrégats à proprement parler. Si l'on reprend l'algorithme précédent, pour produire une agrégation, il faut préciser un nombre minimal de points voisins et une distance maximale au-delà de laquelle deux points donnés ne sont plus considérés comme voisins. L'objectif d'OPTICS est de produire une liste de points ordonnée de telle manière qu'il devient possible de générer rapidement toutes les configurations d'agrégations pour chaque distance de voisinage possible, en utilisant seulement le nombre de points voisins minimum à respecter pour qu'un point appartienne au corps d'un agrégat.

6.1.3 EXPLORATION DES DONNÉES BRUTES DE SAPA

Maintenant que nous avons présenté les mesures et les algorithmes, nous pouvons discuter de l'exploration de nos jeux de données. Cette dernière va s'effectuer en plusieurs étapes distinctes comprenant chacune des résultats intermédiaires et une analyse conjointe de ceux-ci.

La première étape de ce processus consiste simplement à agréger les jeux de données sans composante floue, sans filtrage et sans équilibrage. L'objectif ici est d'évaluer les différentes mesures présentées en Section 6.1.1 afin d'avoir une idée de la compatibilité entre les motifs présents dans les données et notre propre étiquetage sans l'apport d'informations complémentaires par notre modèle. Les mesures sont calculées en comparant directement les partitions résultantes de l'application des différents algorithmes présentés en Section 6.1.2 et l'étiquetage initial obtenu via le processus décrit en Section 5.1.2.

Pour complémenter notre analyse, nous avons aussi choisi de produire différentes visualisations à la fois de notre propre classification et des différentes agrégations obtenues.

Si les mesures permettent d'évaluer globalement s'il existe une compatibilité, ou non, entre les motifs inhérents à nos données et notre propre classification alors les visualisations, quant à elles, sont là pour permettre d'approfondir l'analyse par une observation plus directe des données. Ainsi, pour chaque jeu traité, nous avons décidé de générer à la fois une projection en deux dimensions de chaque configuration via une *Uniform Manifold Approximation and Projection for Dimension Reduction* (UMAP), plus amplement présentée dans l'article de McInnes *et al.* (2018), et un graphique pair à pair de certaines propriétés des instances.

Finalement, nous avons limité notre analyse à un fenêtrage de 20 minutes. Les fenêtrages plus courts provoquent des problèmes avec certaines des implémentations que nous avons utilisées et ne permettent pas de produire des visualisations suffisamment lisibles. En effet, réduire la durée des fenêtres d'un facteur revient à augmenter le nombre d'instances de chaque jeu de données d'autant. Certaines des implémentations utilisées ne sont justes pas assez optimisées pour supporter une telle augmentation de la taille des jeux de données et provoquent des problèmes mémoire. Pour ce qui est des graphiques, l'augmentation du nombre de points ne permet pas de les analyser correctement. Ainsi, en se basant sur le Tableau 5.8, cela revient à traiter 11 jeux de données.

Tableau 6.1 : Moyennes des mesures d'évaluation par algorithme.

Algorithme	agrégats	Pureté	Rand	IMN	IFM
K-Moyenne	4	32,16%	58,33%	4,73%	38,26%
Mean-Shift	4	32,49%	46,80%	3,22%	52,18%
DBSCAN	146	39,66%	52,37%	6,34%	37,88%
OPTICS	792	45,76%	51,97%	8,86%	38,55%

Le Tableau 6.1 résume l'ensemble des mesures obtenues après la réalisation de cette première étape d'exploration. Le nombre d'agrégats affichés est la partie entière de la moyenne du nombre de groupes générés par les algorithmes sur l'ensemble des jeux de données explorés. Dans le cas de l'algorithme des K-Moyennes nous avons fixé à 4 la quantité de regroupement

à produire en accordance avec le nombre de classes de notre propre étiquetage.

Il est aisé de remarquer que la plupart des mesures effectuées n'indiquent pas de bonnes concordances entre les résultats produits par les algorithmes et notre annotation. La pureté, par exemple, est toujours inférieure à 50% indiquant ainsi que les regroupements extraits de manière supervisée contiennent ainsi des instances de différentes classes et non d'un seul et unique type.

L'indice de Rand est un peu meilleur de manière générale mais il n'excède pas 60%. Cela indique une certaine cohérence entre notre propre classification et les agrégations produites par les algorithmes. En effet, cet indice indique le rapport de paires d'objets classés ensemble dans les deux situations, ou classés dans différents groupes dans les deux situations. Il faut cependant noter que les jeux de données groupés sont ici particulièrement déséquilibrés avec une surreprésentation des activités de la classe *autre* comme observable au sein de la Figure 6.1. Cette situation influence grandement l'indice de Rand qui a ainsi plus de chances de trouver des concordances entre les instances des activités *autres* dont le nombre permet de minimiser les discordances.

L'information mutuelle normalisée est systématiquement inférieure à 10% ce qui donne une idée claire du niveau de ressemblance entre notre étiquetage et l'agrégation automatique. Ce dernier est en effet pratiquement nul, l'information que l'on obtient de l'un ne permet pas réellement de décrire l'autre facilement.

Finalement l'indice de Fowlkes-Mallows vient compléter ce portrait en indiquant une dernière fois que le rapport entre les motifs extraits et notre propre classification est douteux au plus. Ce score étant strictement inférieur à 40% dans la plupart des cas, exception faite des résultats obtenus via l'algorithme *Mean-Shift* où il vaut environ 52%.

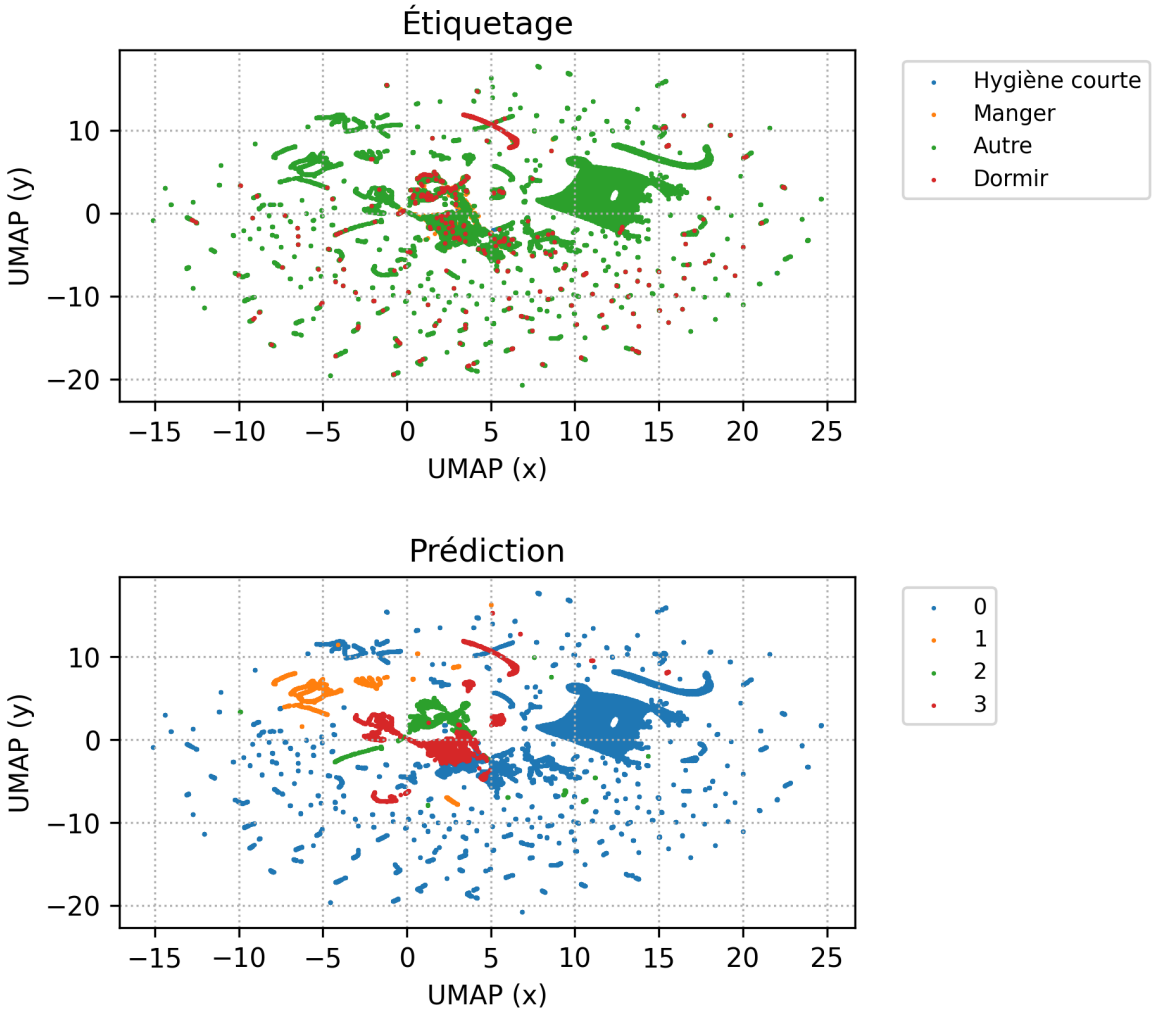

Figure 6.1 : Visualisation via UMAP des résultats obtenus via l'algorithme des K-Moyennes sur le jeu de données de l'environnement numéro 1.

Les graphiques des Figures 6.1 et 6.2 permettent d'observer plus directement notre étiquetage et le résultat d'une application de l'algorithme des K-Moyennes. Premièrement, il est assez évident que notre étiquetage est bruité incluant ainsi un mélange de nombreuses instances difficilement discernables. En particulier, pour les deux environnements, il est possible d'observer de nombreuses tâches vertes ou rouges écrasant d'autres nuages de points représentant les activités de la classe *Manger* ou *Hygiène courte*. Ce sont ces masses de

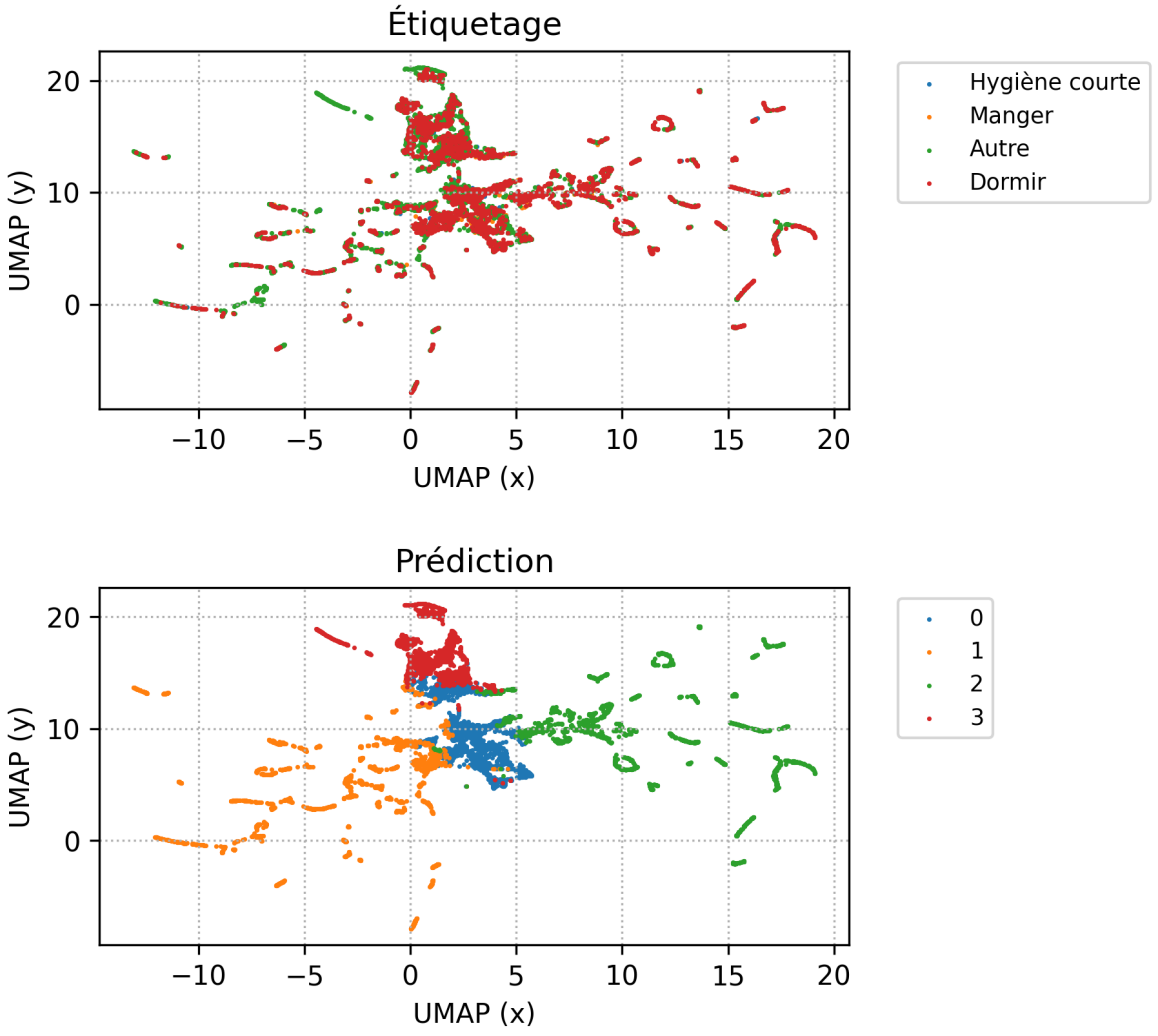

Figure 6.2 : Visualisation via UMAP des résultats obtenus via l'algorithme des K-Moyennes sur le jeu de données de l'environnement numéro 3.

points qui expliquent largement les scores de pureté atteints, ainsi que la valeur des indices de Rand, leur présence occultant toutes les erreurs issues de la mauvaise classification des instances d'autres activités, et leur surreprésentation permettant d'avoir, par défaut, des groupes relativement purs.

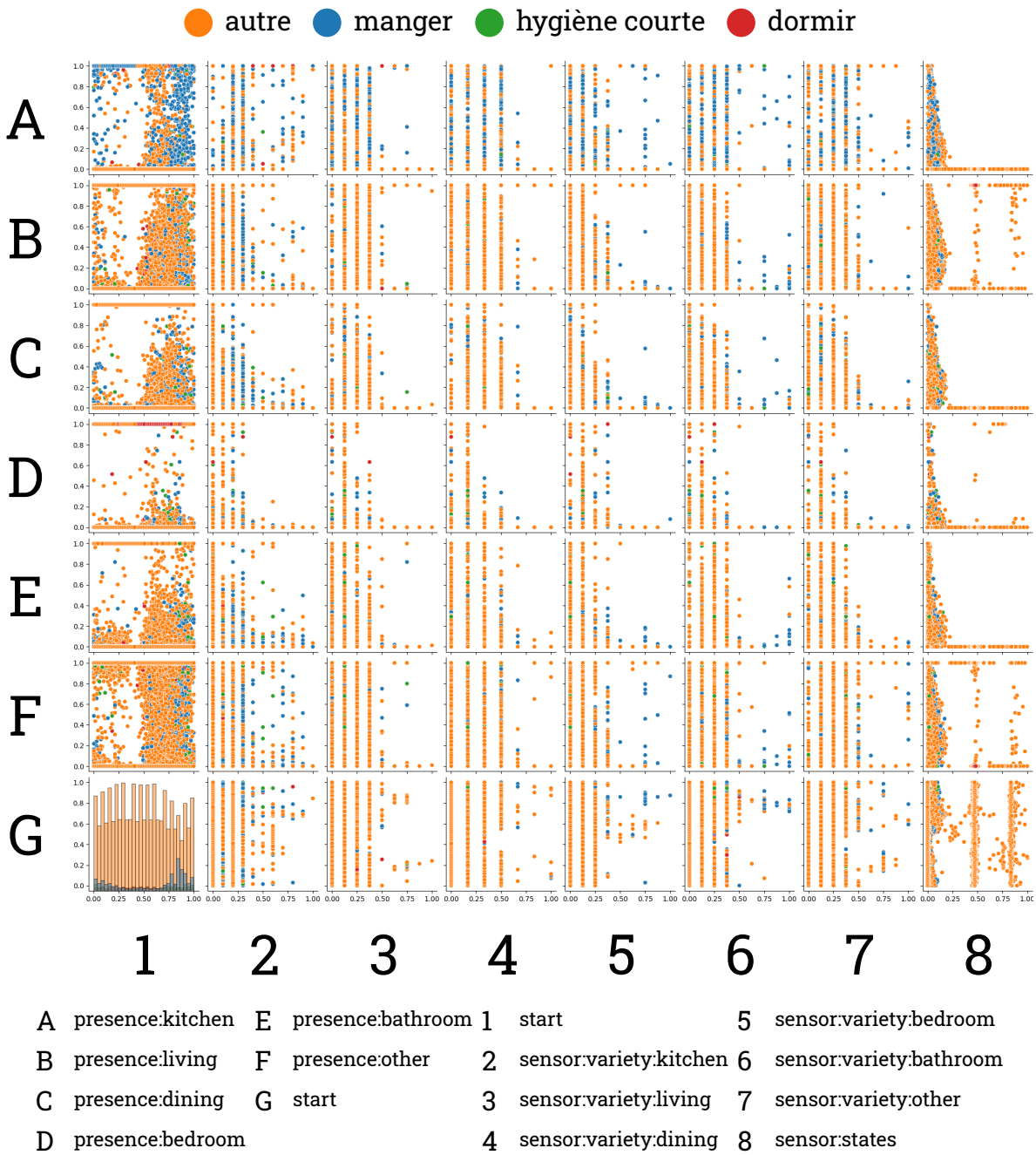

Figure 6.3 : Visualisation pair à pair des instances de l'environnement numéro 1. Chaque colonne et chaque ligne correspondent à une propriété particulière de chaque instance. Chaque cellule de la grille est une représentation en deux dimensions de la distribution des instances en fonction des propriétés de sa colonne et de sa ligne.

Le graphique de la Figure 6.3 est une représentation pair à pair de la distribution

des instances de l'environnement 17. Toutes les paires de propriétés possibles ne sont pas présentées au sein de cette visualisation pour des raisons de concision. À la verticale, il est possible d'y retrouver l'heure de la journée représentée sur une échelle de 0 à 1. À cette temporalité s'ajoutent différentes lignes pour chaque présence dans différents types de pièces représentés sous la forme d'un pourcentage de l'instance. À l'horizontale, chaque colonne représente une information relative aux états des capteurs et à leur variété. Cette approche permet d'apporter un éclairage différent sur les données en y observant de potentielles relations entre nos étiquettes et certains paramètres. Ce que l'on peut déjà observer ici, comme on pouvait l'observer au sein de la figure 6.1, c'est la surreprésentation des activités *autres* prenant la forme de points oranges. Il semble bien que ces instances recouvrent une très grande partie de l'espace disponible pouvant correspondre soit à une éventuelle quantité de *faux négatifs* dans nos jeux de données, soit à une superposition d'instances semblables sur différents jours mais ne représentant pas les mêmes activités. Dans ce second scénario, il y aurait alors l'absence de potentielles activités récurrentes.

Le graphique de la Figure 6.4 est une représentation du même environnement que celui de la Figure 6.3 mais sans les instances de l'activité *autre*. Pour des raisons de lisibilité, et pour pousser un peu plus loin l'analyse, nous éliminerons ces mêmes instances dans les autres graphiques à venir. En observant la colonne 1 de ce graphique, il est possible de remarquer une claire relation entre les instances de l'activité *manger* et la fin de journée. En effet, ces dernières, représentées par des points bleus, sont souvent amassées à droite des cellules. Cette dernière activité semble cependant pas uniquement corrélée à une présence dans la cuisine, les points bleus dans la cellule A1 étant répartis sur toute la hauteur.

Le graphique de la Figure 6.5 est une représentation similaire des instances de l'envi-

ronnement 17. Cette fois-ci, les instances de la classe d'activité *dormir* semblent très liées à une heure précise de la journée (amat de points jaunes). Cette heure, en milieu de journée, ne semble clairement pas s'aligner avec notre modèle. Aussi, les instances de l'activité *dormir* ne semblent pas liées à une pièce particulière, étant éparpillées sur la hauteur de chaque cellule de chaque colonne. Cette dernière situation est elle aussi plutôt étrange, certaines instances indiquant que l'occupant dort en milieu de journée dans la cuisine. Il ne semble cependant pas ressortir, parmi ces représentations pair à pair, d'autres relations évidentes entre les classes d'activités et les propriétés de chaque instance.

Finalement, le graphique de la Figure 6.6 est une représentation pair à pair de la distribution des instances de l'environnement 23. Dans ce cas très précis, on retrouve une nouvelle fois un comportement notable des instances de la classe d'activité *dormir* représentée, cette fois-ci, en vert. Ce qui est particulièrement intéressant dans cette représentation, c'est surtout l'absence notable d'information concernant la ou les pièces pouvant servir de salle à manger. Ce constat pouvant être tiré des droites formées par les points de la ligne C et les points de la colonne 4. On est certainement, ici, dans une situation où une pièce est soit complètement absente, soit observée par un capteur brisé, soit existante mais annotée avec un autre type, cette dernière situation pouvant advenir si une pièce est à la fois une salle à manger et une cuisine par exemple.

Cette exploration des données brutes nous permet donc d'établir un premier état des lieux. Ainsi, les différentes mesures montrent qu'il n'existe pas, ou peu, de rapport entre les motifs inhérents aux données et notre étiquetage initial. Les visualisations permettent, quant à elles, d'aller un peu plus loin, en observant à la fois le bruit ambiant de notre étiquetage et les raisons pour lesquelles certains scores atteignent des seuils. On peut ainsi remarquer que certaines classes d'activités, comme *dormir*, semblent être facilement discernable dans certaines situations et pas d'en d'autres. Finalement, on peut observer l'absence de certaines

observations au sein de certains jeux de données.

6.1.4 EXPLORATION DES ENSEMBLES AVEC CARACTÉRISTIQUES FLOUES

Nous avons, dans la section précédente, exploré les données brutes. Il est donc temps de s'intéresser aux résultats après l'ajout des propriétés liées à la logique floue. La procédure est ici similaire à celle décrite précédemment à l'exception près que pour cette seconde analyse nous allons nous pencher sur les jeux de données comprenant les degrés d'appartenance de chaque instance aux ensembles flous dont nous avons parlés en Section 5.2.2. Ainsi, en se basant sur le Tableau 5.8, cela revient à traiter les 11 jeux de données non filtrés et non équilibrés disposant d'un fenêtrage de 20 minutes.

Tableau 6.2 : Différences entre les moyennes des mesures d'évaluation par algorithme obtenues avec les propriétés floues, et les moyennes obtenues sans les propriétés floues.

Algorithme	Agrégats	Pureté	Rand	IMN	IFM
K-Moyenne	0	2,19%	-0,80%	1,17%	-1,70%
Mean-Shift	56	1,91%	-9,73%	-0,41%	10,31%
DBSCAN	-3	-1,85%	-6,96%	-3,22%	10,37%
OPTICS	-32	4,65%	-0,76%	1,39%	1,65%

Le Tableau 6.2 présente la différence entre la moyenne des résultats obtenus lors de cette seconde analyse et les résultats obtenus lors de la première. Ces derniers sont eux présentés au sein du Tableau 6.1 de la section précédente. On peut observer ici une légère réduction du nombre d'agrégats moyens pour chaque algorithme à l'exception de *Mean-Shift*. K-Moyennes n'est évidemment pas impacté, le nombre d'agrégats produits étant choisi au départ.

Pour ce qui est des mesures, la pureté augmente d'environ deux points de pourcentage, à l'exception de l'agrégation par DBSCAN où elle perd deux points de pourcentage. L'indice de Rand subit une dégradation comprise entre 1 et 10 points de pourcentage en fonction des

algorithmes. L'information mutuelle normalisée reste plutôt stable, toujours à l'exception de l'agrégation via DBSCAN où l'indice perd 3 points de pourcentage. Finalement, l'indice de Fowlkes-Mallows semble augmenter d'environ 10 points de pourcentage avec les algorithmes DBSCAN et *Mean-Shift* mais reste stable dans les autres situations. Ainsi en dehors de l'indice de Rand et de l'indice de Fowlkes-Mallows, l'ajout des propriétés floues semble ne pas impacter tant les résultats. Reste alors à expliquer les modifications concernant l'indice de Rand et l'indice de Fowlkes-Mallows.

Le graphique de la Figure 6.7 est une visualisation de notre premier ensemble de données après l'ajout des degrés d'appartenance aux ensembles flous présentés en Section 5.2.2. En comparant cette projection à celle obtenue lors de l'étape précédente, c'est-à-dire celle présentée dans le graphique de la Figure 6.1, il est possible d'observer un changement notable du nombre de groupes d'instances visibles. En effet, dans cette situation, il semble que certains regroupements du graphique de la Figure 6.1 semblent avoir été subdivisés en plusieurs plus petits amas de points. Notamment, le regroupement visible en haut à gauche du graphique de la Figure 6.1.

Cette augmentation du nombre d'amas peut être observée sur la quasi-totalité des jeux de données, y compris le jeu de l'environnement numéro 3 représenté au sein du graphique de la Figure 6.8 et précédemment au sein du graphique de la Figure 6.2. Cette subdivision en plus petits agrégats augmente le nombre de paires d'instances séparées dans des groupes différents par les algorithmes alors qu'ils sont classés ensemble dans le cadre de notre propre annotation. C'est cette augmentation du nombre de faux négatifs qui permettrait d'expliquer les changements de l'indice de Rand et de Fowlkes-Mallows. Ces deux scores étant basés sur le regroupement ou la séparation de paires d'instances entre deux configurations. L'augmentation du nombre de regroupements dans le cadre de *Mean-Shift* corrobore aussi cette hypothèse.

Le graphique de la Figure 6.9 est une représentation pair à pair des instances de l'environnement numéro 17. Cette représentation est similaire à celle adoptée au sein du graphique de la Figure 6.5 mais, cette fois-ci, en ce focalisant uniquement sur les degrés d'appartenance flous. On peut toujours observer, en H3, H4, I3, I4 et potentiellement en H5, H10, I5 et I10, des importants amas de points jaunes représentant des instances de l'activité *dormir* et donc une potentielle relation entre ces instances et nos propriétés. Cependant, il est plus notable de remarquer que certaines propriétés sont tout simplement systématiquement nulles, notamment celles présentées au sein de la ligne D et E en partant du haut du graphique. Ce sont ici des degrés d'appartenance de durées représentant les instances courtes. Notre fenêtrage étant d'une durée de 20 minutes, aucune instance n'est corrélée de près ou de loin à ces ensembles flous. L'utilisation de fenêtres trop grandes peut donc avoir un impact notable sur la qualité des regroupements, ces dernières pouvant entraîner la désactivation pure et simple de certains ensembles flous.

Il est bon de noter que toutes les paires possibles de propriétés ne sont pas représentées au sein des graphiques pair à pair présentés jusqu'ici, et ce, pour des raisons simples d'espace disponible. Ces graphiques ne permettent donc que de se forger une vague idée d'une partie des dynamiques à l'oeuvre. Les graphiques pair à pair ne permettent pas non plus d'évaluer ce que pourrait donner un regroupement de trois propriétés ou plus. Cependant, c'est une méthode qui permet de visualiser les données sans le filtre d'une transformation plus ou moins complexe telle que l'analyse en composantes principales, ou UMAP, projection utilisée plus haut.

On peut aussi se poser la question du nombre optimal de centroïdes à configurer pour maximiser l'ensemble des scores que nous utilisons via l'algorithme des K-Moyennes. Si on utilise la méthode du *coude* telle que présentée dans le travail de Yuan & Yang (2019), le

graphique présenté à la Figure 6.10 semble indiquer que ce nombre idéal soit compris entre 15 et 20. Il semble donc que les jeux mettent en exergue un nombre de motifs plus important que l'ensemble des classes d'activités que nous traitons. Cette configuration *optimale* ne produit cependant pas des scores très élevés, avec une pureté moyenne aux alentours de 40%, une information mutuelle normalisée aux alentours de 15% et un indice de Fowlkes-Mallows autour de 20%. Seul l'Indice de Rand peut atteindre, au mieux, une valeur de 60%.

Il semble donc, en conclusion de cette seconde étape, que l'ajout des composantes floues subdivise un peu plus les masses d'instances provoquant une modification notable de certains scores. Dans l'ensemble, les données restent très bruitées avec des confusions importantes entrainant, même dans les meilleures situations, un plafonnement notable des scores que nous calculons.

6.1.5 CLUSTERING PAR QUALITÉ

Nous avons exploré nos jeux de données sans composantes floues au sein de la Section 6.1.3, puis, avec les composantes floues au sein de la Section 6.1.4. Ces deux premières étapes nous ont permis d'établir un premier profil de nos ensembles de données et, notamment, d'observer le décalage entre les motifs qu'ils contiennent et leur étiquetage. On peut alors se demander si ce décalage, prenant la forme d'un bruit ambiant, n'est pas la conséquence d'une annotation permissive embarquant dans chaque groupe un bon nombre d'instances problématiques. Dans cette dernière étape d'exploration non supervisée, nous allons tenter d'utiliser le score de qualité que nous avons défini au sein de la Section 5.2.3 afin d'établir s'il est possible de bonifier les agrégations non supervisées des données en transformant les instances problématiques.

Pour filtrer les instances, nous allons commencer par utiliser leurs composantes floues

afin de calculer leur score de qualité. Une fois ce score obtenu, il devient alors possible de considérer des scénarios où les instances obtenant un score inférieur à un seuil respectif de 10%, 20%, 30%, 40%, 50%, 60% et 70% seront transformées. On se retrouve alors dans la situation d'un fenêtrage de 20 minutes avec floutage et filtrage, mais sans équilibrage, soit avec 77 jeux de données à analyser si l'on se réfère au Tableau 5.8.

Nous allons considérer dans cette dernière étape deux transformations. La première d'entre elles consiste simplement à regrouper toutes les instances peu qualitatives du jeu de données dans une seule et nouvelle classe d'activités *peu fiables*. La seconde transformation, quant à elle, consiste à regrouper les instances peu qualitatives d'une même classe d'activités au sein d'une nouvelle classe prévue à cet effet, ainsi, dans ce dernier cas, les instances peu qualitatives de l'activité *cuisiner* se verront attribuer la classe *cuisiner peu fiable*. Évidemment, ce type de modification n'impactera pas les nombreuses instances classifiées comme *autre* ayant elles aussi des conséquences notables sur les indices.

Les Tableaux 6.3 et 6.4 présentent un résumé des résultats obtenus pour chaque algorithme d'agrégation considéré après filtrage. Il est possible de noter que le score de pureté est en général dégradé par le processus et perd entre 0,4 et 7 points de pourcentage. Ce résultat n'est pas très surprenant, l'ajout de nouvelles classes ayant tendance à augmenter le bruit dans les regroupements effectués. Cette situation s'inverse cependant quand l'on utilise DBSCAN ou OPTICS et que l'on filtre en utilisant plusieurs classes peu fiables (6.4). Ce comportement peut être expliqué par le nombre d'agrégats importants générés par ces méthodes. Ainsi, il y a plus de chances que certains d'entre eux se voient attribuer une seule et unique classe d'instance.

Dans l'ensemble, l'indice de Rand bouge peu, à l'exception des résultats obtenus pour les algorithmes des K-Moyennes et *Mean-Shift* dans le cadre de l'utilisation d'une seule

Tableau 6.3 : Différences entre les moyennes des mesures d'évaluation après filtrage dans le cas où une seule classe peu fiable est utilisée et la moyenne des mesures obtenues avec les propriétés floues.

	K-Moyennes				**DBSCAN**			
Filtrage	**Pureté**	**Rand**	**IMN**	**IFM**	**Pureté**	**Rand**	**IMN**	**IFM**
10%	-2,18%	0,80%	0,60%	-0,66%	-1,03%	0,24%	0,59%	-0,93%
20%	-2,94%	0,83%	0,63%	-0,81%	-1,03%	0,14%	0,59%	-1,01%
30%	-2,48%	1,05%	1,14%	-0,67%	-1,13%	0,03%	0,60%	-1,13%
40%	-3,13%	1,03%	0,70%	-1,08%	-1,23%	-0,12%	0,63%	-1,36%
50%	-3,48%	1,46%	1,50%	-1,16%	-1,54%	-0,55%	0,63%	-2,05%
60%	-5,78%	2,89%	1,13%	-2,95%	-4,12%	-2,08%	0,82%	-4,74%
70%	-6,77%	3,20%	1,33%	-2,93%	-3,66%	-2,87%	1,03%	-5,16%
	Mean-Shift				**OPTICS**			
Filtrage	**Pureté**	**Rand**	**IMN**	**IFM**	**Pureté**	**Rand**	**IMN**	**IFM**
10%	-0,44%	-1,88%	0,00%	-1,28%	-0,95%	0,53%	0,76%	-0,90%
20%	-0,45%	-1,98%	0,00%	-1,35%	-0,96%	0,51%	0,80%	-1,00%
30%	-0,57%	-2,11%	0,00%	-1,47%	-1,00%	0,49%	0,88%	-1,12%
40%	-0,67%	-2,39%	0,00%	-1,71%	-1,06%	0,51%	1,05%	-1,33%
50%	-0,99%	-3,10%	0,00%	-2,33%	-1,34%	0,52%	1,33%	-1,89%
60%	-2,97%	-6,49%	0,00%	-5,42%	-4,05%	1,06%	2,10%	-3,88%
70%	-2,98%	-6,92%	0,00%	-5,74%	-4,86%	1,00%	2,86%	-3,92%

et unique classe. La dégradation ou l'amélioration des résultats s'explique largement par les conséquences de l'ajout d'une ou plusieurs classes supplémentaires, provoquant ainsi le changement de certaines paires d'instances autrefois classées ensembles et maintenant classées séparément. Comme nous l'avons décrit dans les sections précédentes, les résultats obtenus via les algorithmes d'agrégation étant très différents de notre propre étiquetage, la séparation de paires d'instances à toutes les raisons de lui être bénéfique si l'algorithme d'agrégation génère peu de groupes.

L'information mutuelle normalisée s'améliore systématiquement entre 0,05 et 3 points de pourcentage. Cette modification reste très faible et ne fait atteindre un score absolu qu'aux alentours de 10%, ce qui est bien peu pour tirer une quelconque conclusion. Dans le cadre de

Mean-Shift, il est possible d'observer strictement aucune modification du score. Ceci est dû au fait que ce dernier était déjà très proche de zéro dans cette situation et l'est toujours après filtrage.

Finalement, l'indice Fowlkes-Mallows varie lui aussi subtilement et régulièrement à la baisse. Comme nous l'avons déjà décrit, cet indice fonctionne sur les mêmes bases que l'indice de Rand et la séparation de nombreuses paires d'instances peut largement expliquer ces subtiles modifications.

Dans l'ensemble, les scores ne semblent pas réellement changer de manière significative dans les deux scénarios proposés. Les nombreuses instances étiquetées *autre* peuvent être notamment responsables d'une réduction de l'amplitude des changements, provoquant un seuil ou un plafond impassable pour chaque score. Finalement, le bruit ambiant et l'absence de motifs clairs dans les données, que nous avons pu observer dans les sections précédentes, n'aident clairement pas à juger de l'efficacité de cette méthode.

6.1.6 DISCUSSION

Nous avons donc exploré nos jeux de données en 3 étapes distinctes. Dans un premier temps, nous avons testé un choix d'algorithmes sur nos jeux de base, n'intégrant aucune composante floue. Les résultats de ces méthodes ont ensuite été évalués au travers d'un ensemble de mesures décrit en Section 6.1.1. Finalement, nous avons complété cette évaluation par une analyse plus directe des résultats via différentes représentations de nos ensembles de données.

Cette première étape permet d'établir un constat de base sur nos jeux de données afin de s'assurer de la nature exacte des impacts de notre approche sur ces derniers. Nous avons pu constater que, dans l'ensemble, les motifs trouvés par les approches non supervisées étaient

Tableau 6.4 : Différences entre les moyennes des mesures d'évaluation après filtrage dans le cas où plusieurs classes peu fiables sont utilisées et la moyenne des mesures obtenues avec les propriétés floues.

	K-Moyennes				**DBSCAN**			
Filtrage	**Pureté**	**Rand**	**IMN**	**IFM**	**Pureté**	**Rand**	**IMN**	**IFM**
10%	-1,94%	0,28%	0,40%	0,99%	-0,79%	0,74%	1,40%	-1,59%
20%	-2,75%	0,24%	0,38%	1,06%	-0,79%	0,74%	1,40%	-1,58%
30%	-1,97%	0,28%	0,40%	0,98%	2,63%	1,47%	2,04%	-2,93%
40%	-2,10%	0,20%	0,61%	1,76%	2,63%	1,43%	2,01%	-2,96%
50%	-2,19%	0,29%	0,56%	1,66%	2,53%	1,28%	1,92%	-3,19%
60%	-4,30%	1,03%	0,27%	0,98%	1,13%	0,01%	1,78%	-4,73%
70%	-5,57%	1,30%	0,05%	0,66%	2,41%	-0,65%	1,57%	-4,89%
	Mean-Shift				**OPTICS**			
Filtrage	**Pureté**	**Rand**	**IMN**	**IFM**	**Pureté**	**Rand**	**IMN**	**IFM**
10%	-3,11%	0,52%	0,00%	0,58%	-1,13%	-0,38%	0,24%	1,12%
20%	-3,13%	0,52%	0,00%	0,58%	-1,13%	-0,38%	0,23%	1,12%
30%	-3,21%	0,51%	0,00%	0,58%	2,38%	0,30%	0,82%	0,45%
40%	-1,44%	1,41%	0,00%	1,24%	2,39%	0,30%	0,84%	0,43%
50%	-1,62%	1,21%	0,00%	1,06%	2,26%	0,29%	0,91%	0,28%
60%	-2,87%	-0,42%	0,00%	-0,33%	0,73%	0,29%	0,98%	-0,76%
70%	-1,40%	-1,00%	0,00%	-0,76%	-1,33%	0,64%	1,09%	-0,76%

bien différents des motifs représentés par notre étiquetage. La surreprésentation des activités *autre* n'aide pas non plus à améliorer les choses, ces dernières se confondant régulièrement avec les autres classes d'activités. Si ces activités sont mises de côté, nous avons cependant pu observer qu'il existait bien des corrélations entre certaines de nos propriétés et des classes d'activités spécifiques. Évidemment, ces résultats sont loin d'être génériques et peuvent varier en particulier à cause de l'hétérogénéité des environnements, impliquant des habitudes comportementales différentes et parfois même l'absence de certaines pièces.

Dans un second temps, nous avons procédé de la même manière mais, cette fois-ci, en intégrant les composantes floues. L'objectif étant ici d'observer l'impact des éventuels apports d'informations liés aux degrés d'appartenance. Nous avons pu observer que si la variation des

différentes mesures n'était pas si importante, il y a bien visuellement un changement notable de la configuration spatiale des instances, ces dernières formant plus de petits agrégats après projection. Nous avons aussi pu remarquer qu'il existait toujours des corrélations importantes entre nos propriétés floues et certaines classes d'instances. Nous avons aussi remarqué que tous les degrés d'appartenance n'étaient pas actifs, dû principalement à la durée des fenêtres d'observation utilisées, impliquant alors de possibles biais dans les résultats présentés. Nous avons finalement conclu cette seconde étape avec une analyse plus poussée des résultats obtenus via l'algorithme des K-Moyennes en cherchant le nombre optimal d'agrégats via la méthode du *coude* . Nous avons alors pu extraire les plafonds et seuils de chacun des scores que nous avons utilisés, ainsi que leurs maximums. Ces résultats ont montré que même dans les meilleures situations, les scores ne pouvaient qu'atteindre des valeurs relativement basses.

Finalement, nous avons procédé à un certain type de filtrage afin de voir si notre score de qualité pouvait avoir un impact significatif sur les résultats obtenus via chacune des approches non supervisées que nous avons sélectionnées. Les résultats montrent des changements de faible amplitude, principalement mitigés par la quantité importante d'activités *autre*.

Cette analyse préliminaire permet d'établir que nos ensembles de données vont être particulièrement difficiles à traiter, dans la mesure où chaque situation est particulièrement hétérogène. Aussi, aucun motif ne se dégage naturellement de nos ensembles de données, et ce, peu importe l'approche employée pour les traiter. Nous avons aussi pu visualiser que ces derniers sont particulièrement bruités. Finalement, la taille de nos fenêtres est un facteur important pouvant limiter l'expressivité de certaines composantes floues. De la même manière, il est très possible que ce bruit ambiant soit lié au peu de caractéristiques que nous utilisons, ou parfois, à l'absence de données relatives à certaines pièces au sein d'une partie des environnements.

6.2 CLASSIFICATION

Afin de conclure sur les impacts des propriétés floues et de notre estimation de la qualité, nous avons choisi de finir ce chapitre par une analyse des résultats obtenus après plusieurs classifications de nos jeux de données par des algorithmes supervisés. Évidemment, cette analyse sera à considérer à l'aube des problématiques déjà observées au sein des sections précédentes, mais aussi à l'absence d'une vérité de terrain fiable. En effet, comme nous l'avons déjà précisé, notre étiquetage est le résultat d'un ensemble d'automates développés par une autre partie de l'équipe de recherche et pour lesquels nous n'avons aucun moyen de certifier leur véracité. Cependant, c'est bien cet étiquetage qui servira de base pour valider notre classification, en supposant donc qu'il soit relativement bon, hypothèse faible en regard de l'exploration non supervisée que nous avons réalisée précédemment.

Cette classification supervisée a pour objectif d'évaluer à quel point nos propriétés se généralisent correctement entre différents environnements et à quel point le score de qualité peut aider à évaluer les étiquettes d'un jeu de données. Nous n'espérons évidemment pas, à la lumière des résultats déjà exposés, aboutir à des précisions importantes sur les ensembles testés.

Pour ce qui est de la classification en elle-même, le Tableau 6.5 présente les moyennes des scores de justesse et de F1 obtenues après une validation croisée à 10 plis des implémentations des algorithmes C4.5 (J48), Forêt Aléatoire, Bayes Naïf et K plus proches voisins de la version 3.8.5 de Weka [31]. Cet outil est, par ailleurs, plus amplement présenté au sein du document de H Witten *et al.* (2017).

Le premier élément remarquable est évidemment la différence majeure entre les jeux de données équilibrés et déséquilibrés. Cette tendance générale peut être simplement expliquée

31. Voir https://www.cs.waikato.ac.nz/ml/index.html.

Tableau 6.5 : Meilleures moyennes de la justesse et du score F1 après une validation croisée à 10 plis d'une application des algorithmes C4.5 (J48), Forêt Aléatoire, K Plus Proches Voisins, et Bayes Naïf sur tous les jeux de données disponibles via Weka 3.8.5. Les valeurs soulignées sont les scores maximums de chaque catégorie.

	Équilibré					
	Base		**Avec flou**		**Filtré**	
	Just.	**F1**	**Just.**	**F1**	**Just.**	**F1**
C4.5	54,57%	54,00%	65,40%	65,00%	65,76%	66,00%
Forêt aléatoire	56,07%	55,49%	68,51%	68,47%	68,93%	68,85%
K Plus Proches Voisins	50,04%	49,16%	65,51%	65,22%	65,49%	65,20%
Bayes Naïf	40,83%	38,28%	61,35%	59,71%	61,70%	60,05%
Moyenne	50,38%	49,23%	65,19%	64,60%	65,47%	65,03%

	Déséquilibré					
	Base		**Avec Flou**		**Filtré**	
	Just.	**F1**	**Just.**	**F1**	**Just.**	**F1**
C4.5	74,86%	74,00%	76,18%	76,00%	77,08%	76,00%
Forêt aléatoire	75,34%	73,68%	77,10%	76,12%	77,98%	76,90%
K Plus Proches Voisins	65,56%	68,50%	70,95%	69,40%	71,25%	69,70%
Bayes Naïf	53,37%	50,78%	54,47%	58,42%	54,45%	58,44%
Moyenne	67,28%	66,74%	69,67%	69,98%	70,19%	70,26%

par l'impact de la distribution des différentes classes d'instances. En effet, la surreprésentation de certaines classes, comme nous l'avons déjà fait remarquer lors de notre analyse non supervisée, entraîne un coût moindre quand l'algorithme se trompe sur des activités moins fréquentes et un coût bien plus important quand il se trompe sur des activités surreprésentées. En guise d'exemple, si une activité représente plus de 80% des instances, partir du principe que toutes les instances appartiennent à cette activité permet d'obtenir un maximum de 80% de réussite. Bien que certains algorithmes soient moins impactés par ce type de situation, dans tous les cas, la surreprésentation des instances d'*autre* activités induit, pour chaque ensemble non équilibré, une précision minimale de 50%. Évidemment, cette tendance ne fera qu'empirer si l'on filtre, éliminant ainsi des instances supplémentaires appartenant aux autres classes

d'activités, c'est-à-dire aux classes *manger*, *dormir* et *hygiène courte*.

Pour ce qui est des ensembles équilibrés, on peut observer un gain d'environ 15% de justesse une fois que les degrés d'appartenance flous sont ajoutés aux instances. Cet effet est aussi observable, dans une moindre mesure, pour les jeux déséquilibrés. Dans les faits, les algorithmes basés sur des arborescences tendent à prioriser les propriétés floues pour discriminer les instances ce qui indique probablement que ces propriétés sont informatives malgré le fait qu'elles aient été conçues à partir d'un contexte expérimental totalement différent, celui de CASAS. De plus, si les scores sont plutôt faibles, ils tendent tout de même à démontrer qu'il est possible d'améliorer l'optimisation des modèles de classification en utilisant directement de la connaissance générique extraite des jeux de CASAS. Nous pensons donc que les règles floues et les ensembles que nous avons définis peuvent en effet aider à transférer de la connaissance dans de futurs travaux.

Finalement, il semblerait que la réduction de la taille des fenêtres impacte aussi la justesse globale, comme montré à la Figure 6.11. Notre hypothèse au sujet de cette différence de résultat est relativement simple : la réduction de la durée des fenêtres provoque simplement une augmentation notable du nombre d'instances dans un jeu de données. Ainsi, si la durée des fenêtres est divisée par 4, comme quand l'on passe de fenêtres de 20 minutes à des fenêtres de 5 minutes, le nombre d'instances en résultant est lui-même multiplié par 4. Ce phénomène peut aussi être couplé à une meilleure appréciation des activités courtes comme les activités d'*hygiène courte* ayant, elles, tendance à être mieux représentées par des fenêtres de 5 minutes que par des fenêtres de 20 minutes. Pour les jeux déséquilibrés, cette sévère augmentation du nombre d'instances amplifie grandement les effets présentés au début de cette section, cette augmentation suivant la distribution sous-jacente des différentes classes d'activités.

Tableau 6.6 : Différences des résultats obtenus après filtrage.

	Justesse		
	Meilleure	**Pire**	**Moyenne**
J48	+1,87%	-2,63%	+0,25%
Forêt Aléatoire	+2,09%	-2,67%	+0,25%
K Plus Proches Voisins	+3,23%	-2,99%	+0,11%
Bayes Naïf	+1,56%	-11,68%	-1,72%
	F-Score		
	Meilleur	**Pire**	**Moyen**
J48	+2,00%	-3,00%	+0,34%
Forêt Aléatoire	+1,55%	-2,77%	+0,08%
K Plus Proches Voisins	+2,32%	-2,85%	+0,20%
Bayes Naïf	+1,28%	-11,05%	-1,81%

6.2.1 FILTRAGE EN TERMES DE QUALITÉ

Les résultats concernant le filtrage par qualité ne permettent pas, quant à eux, de tirer des conclusions. Le Tableau 6.6 présente les meilleures, pires et différences moyennes entre les scores obtenus avant et après filtrage des données. Si les meilleurs scores sont obtenus après filtrage, la différence de résultat reste minime.

De plus, malgré les résultats globaux, les gains et pertes de justesse restent inconsistants d'un jeu de données à l'autre. Ici, les différents processus d'apprentissage supervisé pourraient très bien être capables d'apprendre les instances moins fiables et considérer que parfois la même activité est juste réalisée différemment. Ceci représente bien évidemment une limite majeure pour décider de l'efficacité d'une telle approche, car dans cette situation, supprimer des instances de faible qualité ne peut avoir qu'un impact mineur sur la capacité d'apprentissage des différents modèles. De plus, le filtrage réduit un peu plus le nombre d'instances ce qui pourrait largement nous amener à une situation où certaines classes finissent par être sous-représentées. Quoi qu'il en soit, les résultats restent encourageants si l'on tient compte des limitations auxquelles nous devons faire face pour les exploiter. Les conséquences de l'absence

d'une vérité de terrain fiable étant inévitables.

6.3 CONCLUSION

Dans ce dernier chapitre, nous avons étudié la capacité de notre approche à transférer de la connaissance d'un contexte expérimental très différent, celui de CASAS, à notre propre contexte constitué d'un ensemble de maisons intelligentes hétérogènes.

Dans une première partie, en Section 6.1, nous avons appliqué différentes méthodes d'apprentissage non supervisé afin de réaliser un premier état des lieux concernant notre étiquetage et nos différents environnements. Par le biais de mesures et représentations, nous avons ainsi pu explorer les motifs et limites de nos jeux de données. En Section 6.1.3, nous avons pu faire le constat que les ensembles étaient particulièrement bruités et que les motifs qui s'en dégageaient naturellement restaient très éloignés de notre propre étiquetage malgré que, dans certaines situations, il soit possible d'observer des corrélations entre notre étiquetage et certaines propriétés des instances. Nous avons aussi pu observer que certaines observations types pouvaient être totalement absentes de certains jeux, en conséquence de leur hétérogénéité. En Section 6.1.4, nous avons pu observer que si l'ajout des propriétés floues n'améliorait pas la concordance entre notre étiquetage propre et les motifs inhérents à nos jeux de données, cette dernière avait la particularité de fragmenter un peu plus les projections et de former plus de petits agrégats de points remarquables. Nous avons aussi établi dans cette section la configuration de l'algorithme des K-Moyennes donnant les meilleurs scores et ainsi observer qu'il semble exister, en effet, bien plus de classes d'activités que celles que nous considérons. Finalement, en Section 6.1.5, nous avons étudié les impacts de deux méthodes de filtrage sur les résultats des regroupements. Nous avons finalement conclu qu'entre la magnitude des changements et la nature de nos mesures, il était relativement difficile d'extraire une conclusion pour ce qui est de l'application de notre score de qualité aux différents ensembles

considérés.

Nous avons conclu ce chapitre par la Section 6.2 présentant les résultats obtenus via différentes approches de classification supervisées. Cette section comprenant une première analyse générale des résultats obtenus en fonction de différentes configurations des ensembles explorés et se concluant sur une analyse des résultats obtenus si l'on filtre les données avec notre mesure de la qualité. Cette dernière section nous a permis de mettre en exergue un potentiel apport d'information par nos propriétés floues. Un apport pouvant potentiellement indiquer un transfert de connaissances des environnements de CASAS vers les nôtres. Malgré cela, il reste difficile de conclure pour ce qui est de l'estimation de la qualité, les impacts étant mitigés bien que globalement positifs.

Les résultats de ces deux sections nous permettent donc d'être optimistes quant à la possibilité de transférer des connaissances d'un environnement à l'autre en employant des modèles flous. En effet, ces derniers ont tendance à impacter grandement les ensembles d'instances de tous les environnements hétérogènes que nous explorons et à améliorer notablement les performances des différentes approches de classification que nous avons utilisées. Pour ce qui est de l'appréciation de la qualité, il reste très difficile d'émettre un avis pertinent, nos deux tentatives n'ayant guère impacté les résultats de chaque approche. Il est très possible que pour cette dernière mesure, nous nous heurtions ici aux limites de notre propre étiquetage, qui pourrait s'avérer comme le laisse présager la première partie de ce chapitre, incomplet si ce n'est peu fiable dans son ensemble. Il est aussi largement possible que nous soyons sujets à un manque d'expressivité dû à la généricité de notre propre modèle, ou au manque de capteurs dans les différents environnements.

Il reste important de mentionner que les travaux présentés dans ce chapitre sur la classification ont fait l'objet d'une publication revue par les pairs par Demongivert *et al.*

(2021b).

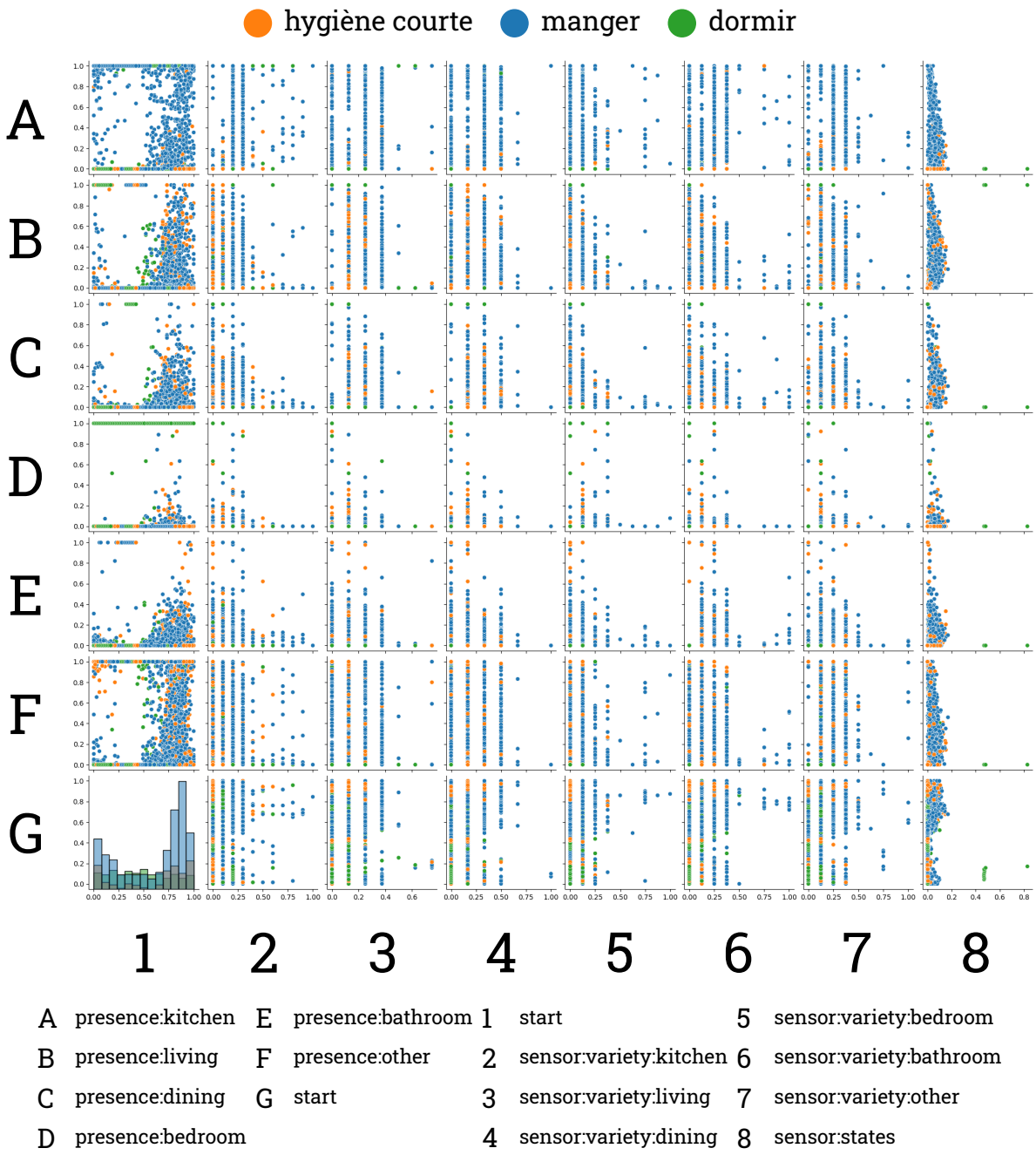

Figure 6.4 : Visualisation pair à pair des instances de l'environnement 1. Chaque colonne et chaque ligne correspondent à une propriété particulière de chaque instance. Chaque cellule de la grille est une représentation en deux dimensions de la distribution des instances en fonction des propriétés de sa colonne et de sa ligne. Les instances de l'activité *autre* ont été éliminées de cette représentation.

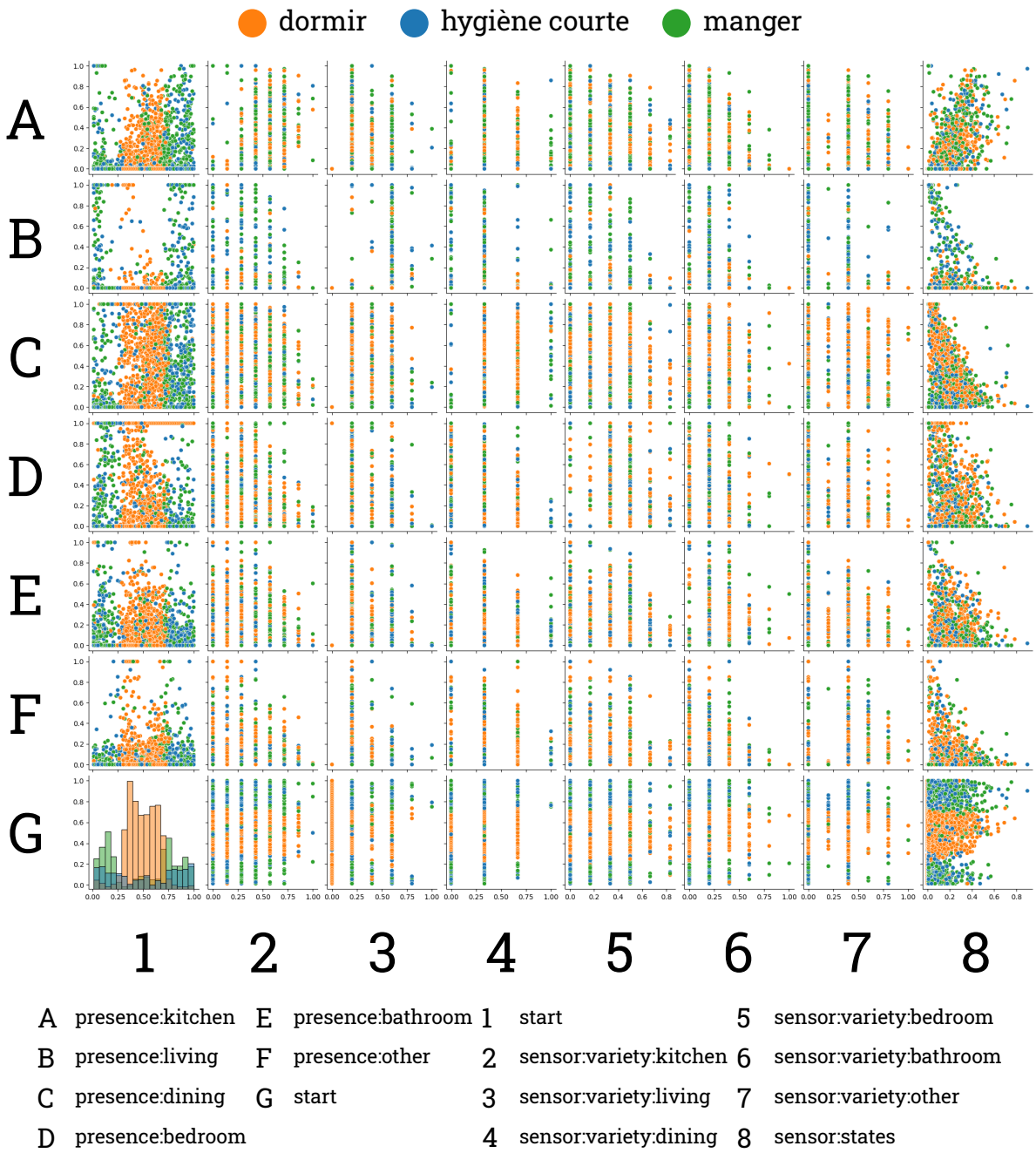

Figure 6.5 : Visualisation pair à pair des instances de l'environnement 17. Chaque colonne et chaque ligne correspondent à une propriété particulière de chaque instance. Chaque cellule de la grille est une représentation en deux dimensions de la distribution des instances en fonction des propriétés de sa colonne et de sa ligne. Les instances de l'activité *autre* ont été éliminées de cette représentation.

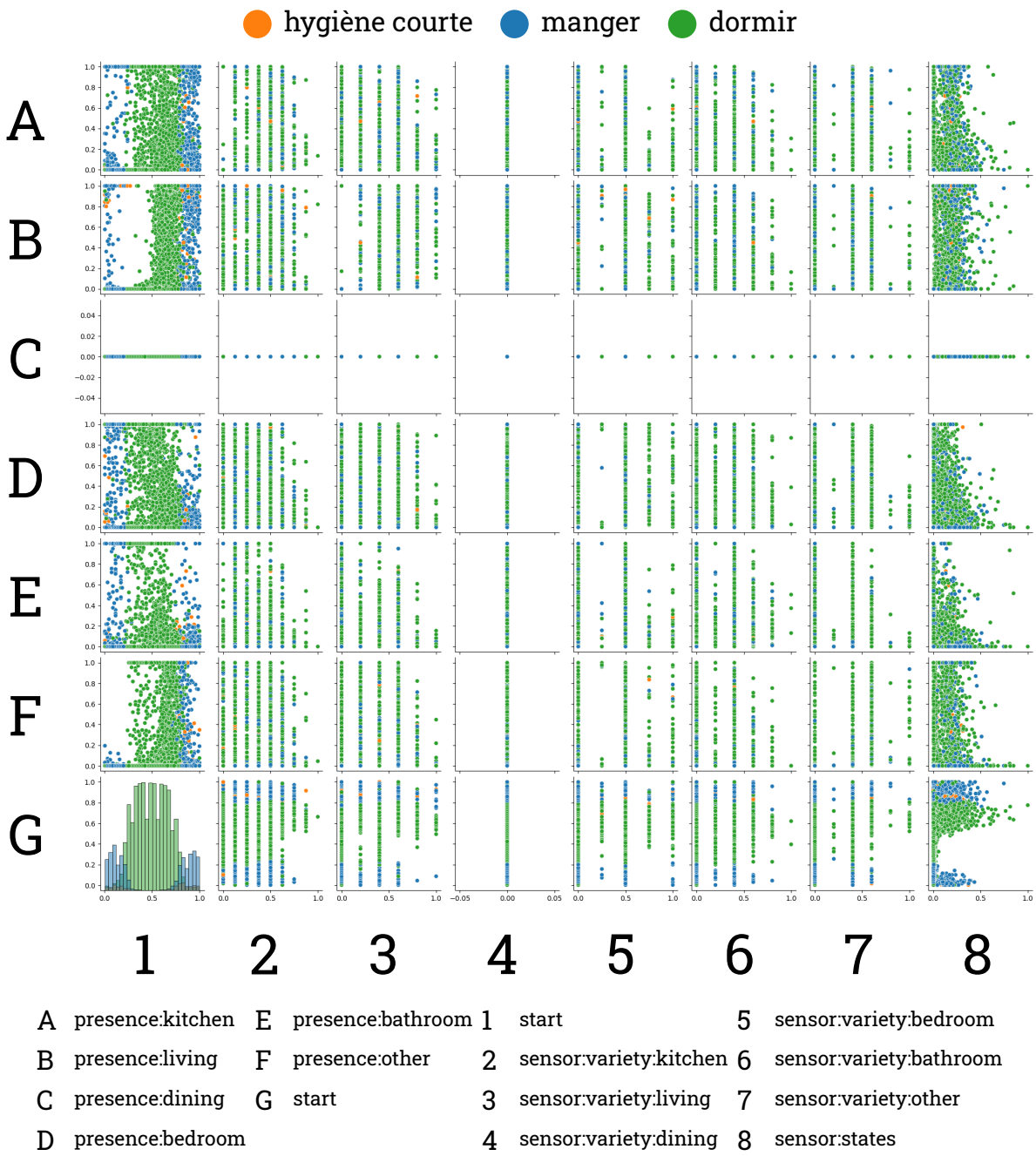

Figure 6.6 : Visualisation pair à pair des instances de l'environnement 23. Chaque colonne et chaque ligne correspondent à une propriété particulière de chaque instance. Chaque cellule de la grille est une représentation en deux dimensions de la distribution des instances en fonction des propriétés de sa colonne et de sa ligne. Les instances de l'activité *autre* ont été éliminées de cette représentation.

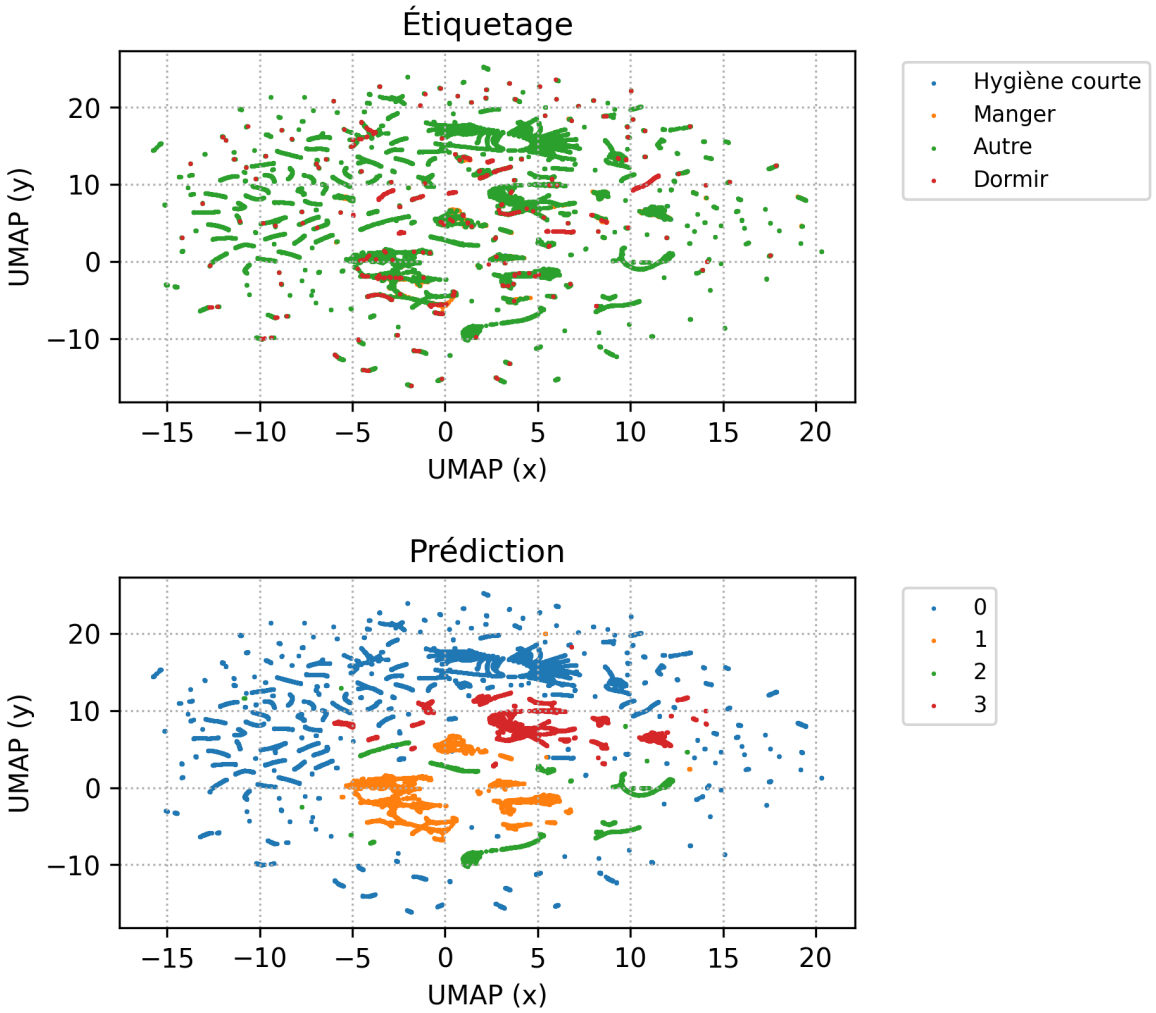

Figure 6.7 : Visualisation via UMAP des résultats obtenus via l'algorithme des K-Moyennes sur le jeu de données de l'environnement 1 comprenant les degrés d'appartenance à nos ensembles flous.

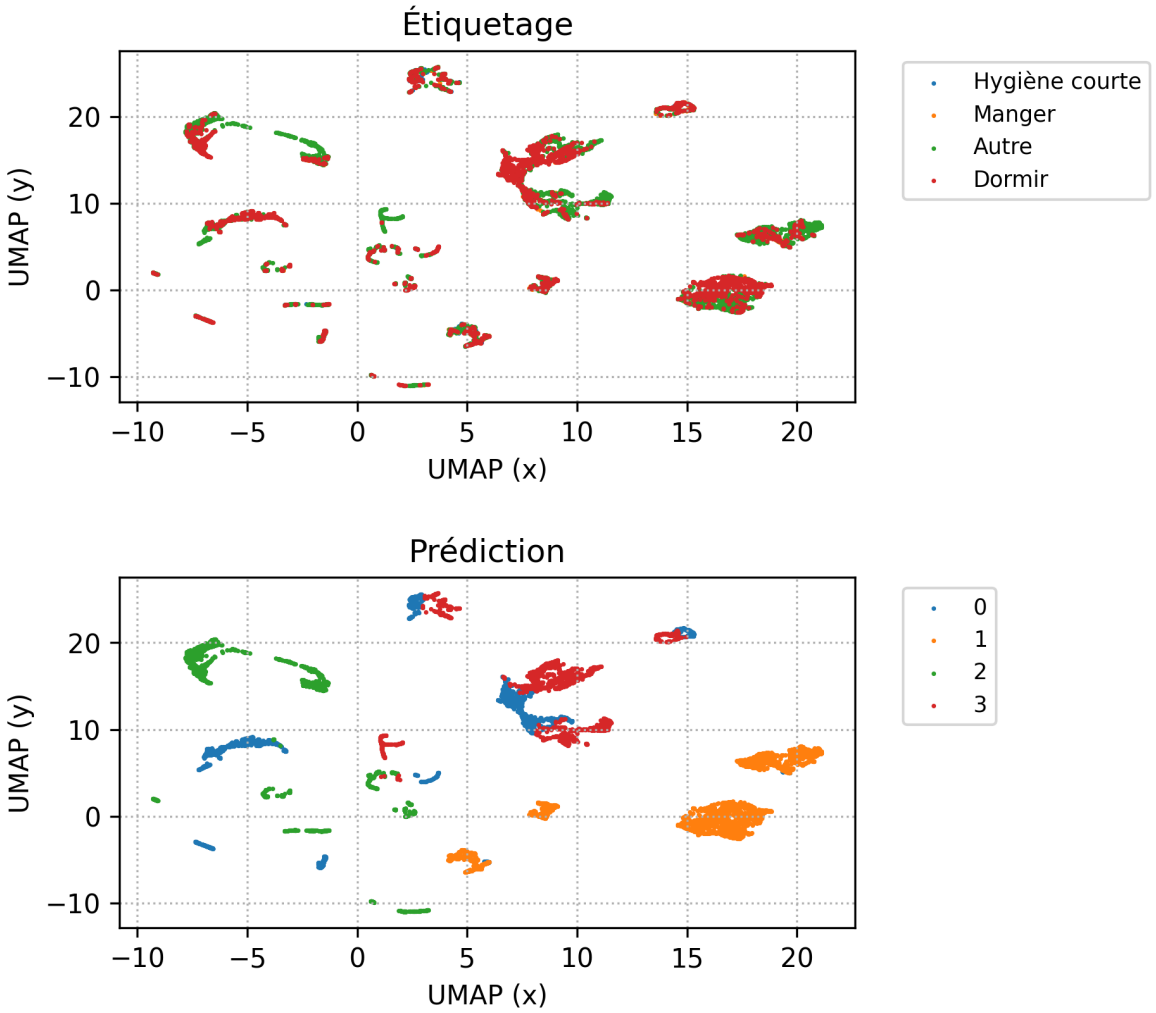

Figure 6.8 : Visualisation via UMAP des résultats obtenus via l'algorithme des K-Moyennes sur le jeu de données de l'environnement 3 comprenant les degrés d'appartenance à nos ensembles flous.

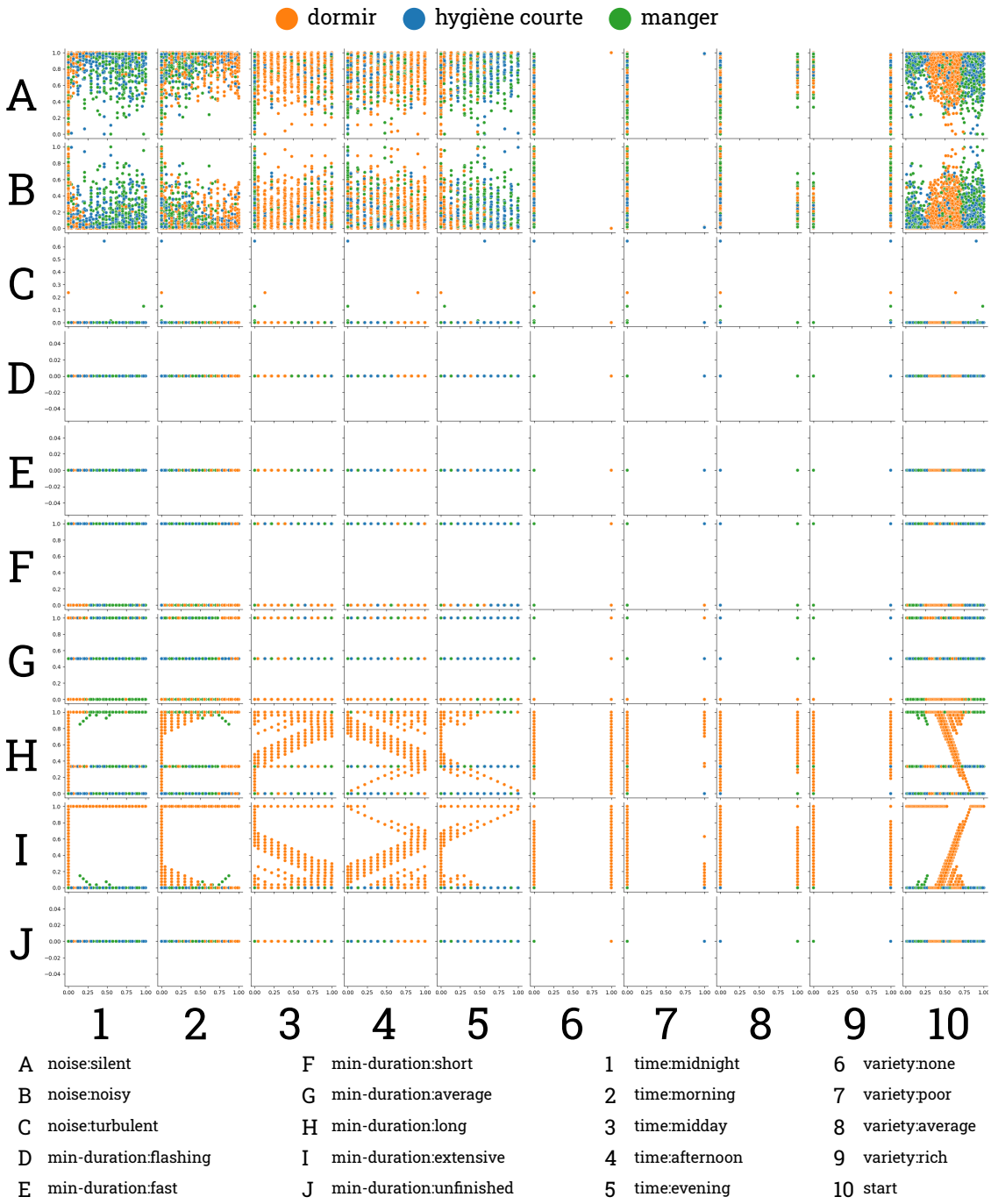

Figure 6.9 : Visualisation pair à pair des instances de l'environnement 17. Chaque colonne et chaque ligne correspondent à une propriété particulière de chaque instance. Chaque cellule de la grille est une représentation en deux dimensions de la distribution des instances en fonction des propriétés de sa colonne et de sa ligne. Les instances de l'activité *autre* ont été éliminées de cette représentation.

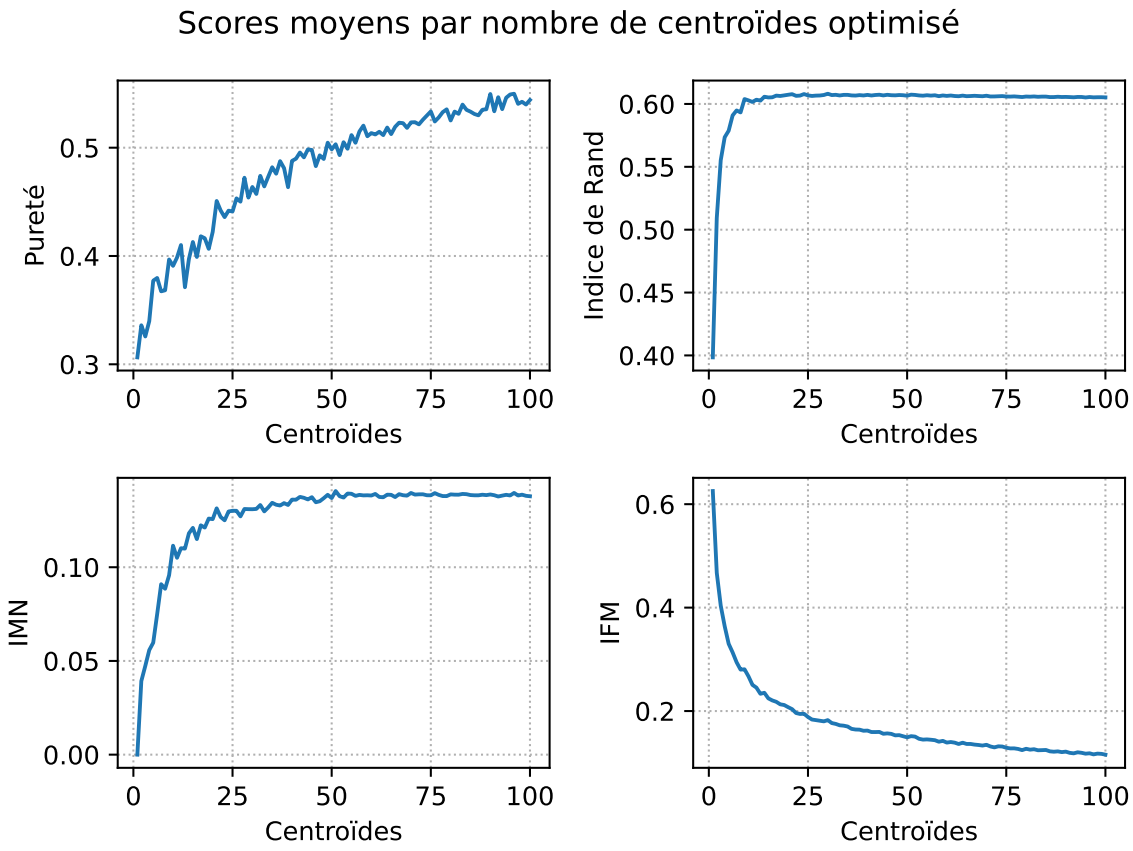

Figure 6.10 : Scores moyens par nombre de centroïde optimisé via l'algorithme des k-moyennes.

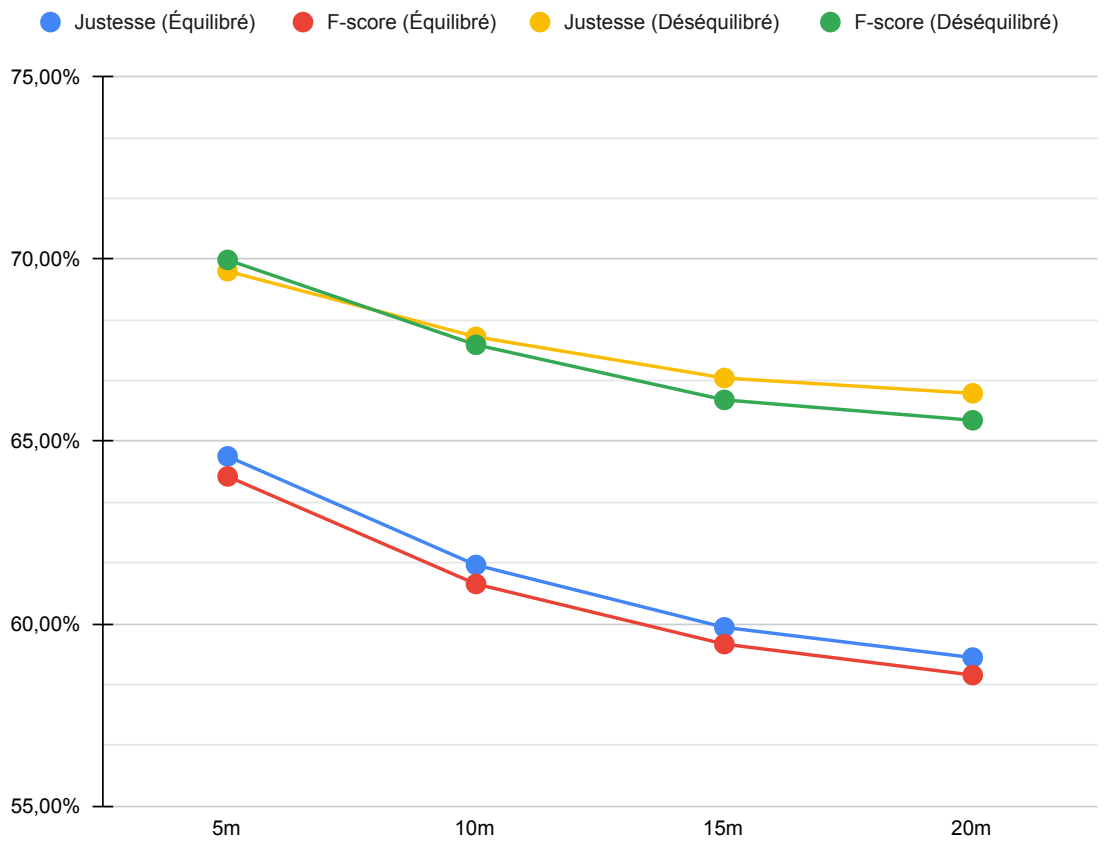

Figure 6.11 : Évolution de la justesse et du F-score moyen en fonction de la taille du fenêtrage adopté pour chaque jeux de données flouté.

CHAPITRE VII
CONCLUSION GÉNÉRALE

Dans cette thèse, nous avons traité de notre approche visant à mettre au point un assistant ambiant personnalisable basé sur les environnements intelligents. L'objectif de cet outil étant double avec, premièrement, la capacité à permettre à des individus atteints de maladies neurodégénératives de conserver leur autonomie le plus longtemps possible et, ensuite, de permettre aux cliniciens de suivre leurs patients à distance en monitorant les environnements dans lesquels ces derniers évoluent.

La recherche présentée ici, ont été réalisés dans le cadre d'un projet plus vaste, le projet SAPA, sans lequel il aurait été impossible de parvenir aux résultats que nous avons exposés. En effet, ce projet interdisciplinaire nous a permis de profiter de l'expérience d'experts de nombreux domaines (médical, social ou informatique). Il nous a aussi apporté l'aide de nombreux professionnels indispensables à l'établissement et à la maintenance d'un parc d'environnements de cette ampleur. Et finalement, il nous a offert l'apport critique et la coopération nécessaire au développement d'un projet de recherche solide.

Dans le Chapitre 1, nous avons exposé notre contexte, nos enjeux et notre problématique. Cette introduction a fait suite au Chapitre 2 au sein duquel nous avons présenté plusieurs environnements intelligents de la littérature scientifique. Cette revue des différentes solutions existantes nous a finalement permis de conclure sur une représentation plus générique basée sur les traits communs de toutes ces architectures. Finalement, nous avons conclu cette revue de littérature par le Chapitre 3 dans lequel nous avons présenté ce qu'était la reconnaissance d'activité, et quelles étaient les approches permettant de la mettre en oeuvre. Nous avons conclu ce chapitre sur un questionnement relatif à la nature même de ce que produit la reconnaissance

d'activité, et comment il serait possible d'aller plus loin, en faisant plus que seulement détecter la présence, ou non, d'une activité sur une période d'observation donnée.

Suite à cette première partie, nous avons scindé la présentation de notre travail de recherche au sein de trois nouveaux chapitres. Ainsi, dans le Chapitre 4, nous avons présenté une infrastructure capable d'extraire, transformer et déplacer de l'information depuis plusieurs environnements intelligents hétérogènes. En se basant sur le fonctionnement des environnements intelligents présentés au sein du Chapitre 2, nous avons donc dérivé quelque chose de nouveau, capable d'utiliser ces différents environnements existants tout en répondant à nos attentes.

Au sein du Chapitre 5 nous avons à la fois présenté les données recueillies grâce à notre infrastructure et un modèle, basé sur la logique floue, permettant de définir des activités. Nous avons, finalement, appliqué notre approche à l'environnement Aruba de CASAS afin d'en extraire des définitions d'activités, et d'en explorer les potentielles limites.

Finalement, au sein du Chapitre 6, nous avons exploré nos jeux de données en utilisant des approches non supervisées. Cette exploration nous a permis d'évaluer globalement la qualité intrinsèque de notre étiquetage et ses limites. Finalement, en employant des méthodes de classification supervisées, nous avons tenté d'utiliser notre modèle afin de transférer de la connaissance d'un environnement annoté, Aruba, vers nos propres environnements. Ces deux derniers chapitres faisant finalement écho aux idées que nous avions développées au sein du Chapitre 3.

Ce long développement basé sur trois de nos travaux ayant fait l'objet d'autant de publications revues par les pairs, nous amène donc à ce dernier chapitre de conclusion générale, visant à établir notre progression en regard de nos objectifs initiaux, les apports personnels de cette thèse et une ouverture, sur de potentiels travaux futurs liés à notre problématique.

7.1 RÉALISATION DES OBJECTIFS

Nous avons exposé nos objectifs au sein de la Section 1.5 du Chapitre 1 de ce document. Ces derniers sont au nombre de 5 et, à la lumière des développements des trois derniers chapitres, il est possible d'établir dans quelle mesure ces objectifs ont été atteints.

Le premier des objectifs que nous nous étions fixés visait principalement à gérer l'hétérogénéité des environnements intelligents. Si l'on se base sur l'aspect architectural de notre thèse, c'est-à-dire sur le contenu du Chapitre 4 et sur la Section 5.1, nous pouvons considérer que cet objectif a été largement atteint. En effet, d'une part, notre architecture permet de gérer des environnements très hétérogènes, partiellement présentés au sein de l'Image 4.1. D'autre part, le nombre d'environnements intégrés au projet SAPA au fil des années parle de lui-même, c'est-à-dire 37 environnements au total, tous gérés en temps réel.

Reste alors le second aspect de cet objectif, c'est-à-dire, l'aspect lié au traitement des jeux de données produits. En regard des développements présentés au sein des Chapitres 6 et 5, notre réponse à cet aspect de la problématique reste relativement mitigé. Si le modèle que nous avons proposé nous permet bien de traiter tous les environnements de manière uniforme, il ne nous a pas été possible de conclure sur l'efficacité de notre méthode de validation des étiquetages. Nous avons donc, en somme, un modèle qui permet de traiter des environnements hétérogènes, mais à partir duquel nous n'avons pas encore développé de méthode de reconnaissance d'activités totalement fiable.

Le deuxième objectif que nous nous étions fixé concernait l'ajout, au fil du temps, de nouvelles maisons à suivre au sein de l'architecture. En effet, la plupart des architectures de maisons intelligentes de la littérature ne s'occupant de gérer qu'un unique environnement, cette problématique était relativement nouvelle dans notre situation. Dans les faits, l'utilisation d'un système réactif basé sur des agents autonomes tel que celui décrit dans le Chapitre 4

nous a permis de facilement installer de nouveaux environnements, ces derniers n'ayant qu'à communiquer par eux-mêmes avec un des noeuds d'échange de notre système. Encore une fois, le nombre d'environnements installés et leur diversité témoignent en notre faveur de l'efficacité de notre solution vis-à-vis de cette problématique spécifique.

Le troisième objectif auquel il nous fallait répondre était d'assurer la fiabilité des données reçues. Nous avons traité de cet aspect au sein du Chapitre 4 et du Chapitre 6 de ce document. D'une part d'un point de vue du processus de génération des données, c'est-à-dire comment s'assurer qu'un flux de données entrant valide produise systématiquement un flux de données sortant valide. D'autre part du point de vue de l'annotation en elle-même, avec une estimation de sa qualité par l'utilisation d'un modèle flou. Il est bon de noter que dans le cadre de cette problématique, au-delà des réserves que l'on peut émettre sur l'estimation de la qualité des instances et des opérations de filtrage que nous avons tenté de mettre en place, la gestion de la fiabilité des données reçues est en réalité une problématique dépassant de loin notre propre périmètre de recherche.

En effet, si au sein du projet SAPA nous ne nous sommes occupés seulement du traitement intermédiaire des données, les sources d'erreurs au sujet de la fiabilité des données ne sont pas cantonnés qu'à cette unique partie du flux. Des erreurs peuvent aisément subvenir en amont de ce dernier, liés à des problèmes lors de l'installation, des problèmes relatifs à l'usure du matériel, des problèmes de paramétrage, ou, tout simplement, des problèmes réseaux. Des problématiques peuvent aussi advenir en aval du flux, avec, en particulier, des problèmes de présentation des données récupérées par les applications, ou des problèmes liés à d'autres traitements intermédiaires des données réalisées par les clients. Évidemment dans le cadre de notre recherche ces problématiques situées en amont et en aval sont autant de complexités supplémentaires à gérer quand il s'agit de trouver l'origine d'un problème. Ce dernier pouvant relever de l'étiquetage automatique, ou de notre propre traitement de ces

résultats.

Ainsi, cette problématique spécifique, en dehors des propositions que nous avons pu faire, mériterait un traitement particulier et beaucoup plus systématique. La collecte, et la production de données erronées étant une catastrophe quand il s'agit d'établir des conclusions solides, et nécessitant un temps considérable si l'on ne peut pas garantir la fiabilité de tous les éléments de l'infrastructure utilisée. Dans l'idéal, la source d'une erreur ne devrait provenir que du dernier maillon de la chaîne de traitement, simplifiant ainsi grandement le processus d'analyse. Reste à déterminer, évidemment, comment valider de manière efficace des éléments de nature si différente.

Le quatrième objectif que nous nous sommes fixé était de traiter des jeux d'informations non annotés. Pour répondre à cette problématique spécifique, nous avons proposé une approche de transfert des connaissances basée sur la logique floue. C'est le sujet des Chapitres 5 et 6 de ce document. Dans les faits, en employant notre modélisation nous avons largement été capables d'explorer de manière non supervisée et supervisée nos ensembles de données, et ce malgré leur hétérogénéité. Ces deux étapes ont été largement décrites dans la première et seconde partie du Chapitre 6. En cela nous pouvons considérer que cet objectif a été atteint, malgré les réserves que l'on peut exprimer au sujet de la mesure de la qualité des annotations et de la tentative de transfert de connaissances du jeu Aruba vers les nôtres.

Le dernier objectif que nous nous étions fixé visait à mesurer la qualité d'une annotation automatique donnée. C'est la problématique que nous avons tenté de traiter en définissant notre modèle au sein du Chapitre 5 et en l'appliquant dans la seconde partie du Chapitre 6. Dans les faits, si nous avons plutôt réussi à utiliser notre proposition pour modéliser des activités depuis Aruba, il ne nous a pas été possible de prouver, ni d'infirmer définitivement que ces modèles nous permettaient bien de transférer les connaissances acquises sur ce jeu de

données externes pour mesurer la qualité de notre propre étiquetage automatique. C'est donc sur cette problématique que nous allons ouvrir cette conclusion.

7.2 ÉVALUATION DES TRAVAUX

Dans cette section, nous nous proposons d'évaluer les différents travaux présentés au sein des 3 derniers chapitres de cette thèse. Le premier d'entre eux étant, évidemment, l'architecture que nous avons proposée pour gérer plusieurs habitats intelligents hétérogènes.

Cette architecture spécifique a été proposé sous la forme d'un monolithe rassemblant deux aspects distincts afin de gagner du temps. Le premier aspect concernant la génération des connaissances, et le second concernant l'accès en lecture des données stockées. Dans l'ensemble, l'utilisation d'une approche classique d'accès aux données a été l'un des plus gros problèmes à gérer. En effet, cette approche provoque un stress important de la mémoire lié à la mise en cache des différents objets utilisés pour manipuler les données. Il s'avère aussi que l'architecture standard des bases de données n'est pas parfaite pour gérer des séries temporelles, ces dernières pouvant profiter d'une approche différente des B-Tree classiques quand le nombre de données stockées augmente grandement. Dans l'ensemble, il eut été intéressant de proposer une approche basée sur la manipulation de BLOBs comme nous l'avons présenté en Section 2.2.1 ainsi qu'en se basant sur des bases de données adaptées au stockage des séries temporelles.

Pour ce qui est de l'architecture, la gestion des données n'est pas la seule problématique. Le logiciel en lui-même est une plateforme complexe et certainement difficile à manipuler et à installer. Avec du recul, il eut aussi été intéressant de se demander s'il n'était pas possible d'adopter une approche plus minimaliste afin de proposer un outil plus simple à déployer et à étendre, peut-être en se basant sur une approche basée sur le développement plutôt que

la configuration, notamment en utilisant des technologies comme Groovy, si l'on privilégie toujours les outils gravitant autour de la Machine Virtuelle Java.

Viens finalement la question de la validation du code. Il s'avère que nous avons adopté une approche basée sur les tests afin de valider notre infrastructure. Cette approche nous a permis de couvrir l'ensemble de la plateforme, mais pas de maintenir efficacement les tests avec son évolution. Se pose alors la question de la simplification de ces derniers, profiter d'un système de manipulation des données basées sur les BLOB, comme il peut en exister en python par ailleurs, aurait été d'une grande utilité ici. La mise en place d'un ensemble d'outils simplificateurs, pour la définition d'architectures et de séries temporelles, aurait aussi largement permis d'aider à l'élaboration plus rapide de tests plus lisibles.

Nous avons exposé trois grands points ici qui se résument tous à une problématique bien plus générale. Comment produire une infrastructure de cette ampleur sans que la demande d'investissement dans l'outil ne dépasse nos propres capacités de développement en terme de temps et de main-d'oeuvre ? Il semble clair que dans l'ensemble, l'utilisation des grandes architectures disponibles était une approche bien trop ambitieuse, et qu'il eût fallu se rabattre sur quelque chose de plus minimaliste, au prix malheureux de l'absence de telles solutions dans les dépôts de librairies de l'époque.

Le deuxième travail que nous avons présenté regroupait à la fois la définition d'un modèle basé sur la logique floue, et d'un ensemble de règles permettant de définir certaines classes d'activités en se basant sur le jeu de données Aruba de CASAS. Si le modèle que nous avons proposé nous a permis d'analyser le jeu de données de CASAS, il faut avouer qu'avec du recul les ensembles flous relatifs au nombre d'évènements émis par les capteurs et à leur variété sont peut-être les plus faibles du modèle. En effet, dans le cadre d'Aruba, l'installation est principalement constituée d'une grille de capteurs de mouvements, entraînant par le fait

même un nombre important de messages, mais une variété plutôt faible de capteurs. Dans notre cas, les environnements sont plutôt un mélange de divers capteurs, pouvant entraîner dans certaines situations moins de messages, mais surtout systématiquement une variété de capteurs plus importante. Il aurait peut-être été préférable de proposer une approche plus sémantique, avec des ensembles permettant d'exprimer une relation entre certains éléments et nos activités directement par exemple. Dans cette situation, la manipulation d'une casserole peut être liée à l'activité *faire la cuisine* et *faire la vaisselle* à hauteur d'un certain degré d'appartenance. Ou peut-être, des éléments plus abstraits, comme l'utilisation d'eau, ou la manipulation d'objets complexes.

Évidemment, se tenir à cette modélisation nous a permis d'analyser manuellement l'ensemble de données de CASAS tout en définissant des ensembles de règles à échelle humaine. Il eut été intéressant cependant, dans cette seconde étape, de proposer une méthode automatisée de création des règles floues. Peut-être en dérivant une approche basée sur les arbres de décision, ou en calculant des poids directement en comptant le nombre de relations existantes entre certains éléments et certaines étiquettes. De même, avec une méthode automatisée, il aurait certainement été possible de définir les règles floues en employant plusieurs jeux annotés différents, limitant ainsi certainement le sur-apprentissage et l'adaptation de nos modèles à des spécificités trop locales. Ce type d'approche est cependant toujours impacté par la fiabilité de l'étiquetage utilisé, et dans le cadre du jeu de données que nous avons exploré, nous avons pu mettre en exergue des choix et parfois quelques problématiques indiquant bien évidemment qu'un étiquetage humain n'est jamais réellement exempt de biais ou d'erreurs pouvant avoir des impacts importants lors de l'optimisation d'un modèle.

Le troisième et dernier travail que nous avons présenté, c'est lui aussi réalisé en deux étapes. La première consistant en une exploration non supervisée de nos données, et la seconde prenant la forme d'une analyse supervisée de notre étiquetage automatique. Pour ce qui est de

l'ensemble, nous avons bien évidemment été confronté aux problématiques liées à l'absence d'un étiquetage humain fiable permettant de vérifier nos modèles. À ce premier problème, qui justifie la direction prise dans notre recherche, c'est ajouté la problématique de fiabilité de nos propres données. Comme précisé précédemment, quand une infrastructure atteint une telle ampleur, être capable de garantir le bon fonctionnement de tous les éléments est une étape indispensable pour garantir que notre analyse se base sur des fondations solides. Plus précisément, que l'on observe bien des motifs réels et non des motifs provoqués par d'éventuelles erreurs provoquées par les éléments matériels ou logiciels.

En dehors de cette problématique liée à la fiabilité des données, on peut aussi se pencher sur les limites liées à leur abondance. Un des points forts de notre approche étant la quantité d'observations collectées depuis différents environnements intelligents, il nous a fallu, notamment pour les situations où l'on utilisait un fenêtrage plus réduit, traiter d'importants volumes de données. Cette charge rend les outils usuels d'observation et de traitement difficile d'utilisation, voire inopérant. Ainsi, il s'avère que Weka et Scikit-Learn, les plateformes que nous avons utilisées, trouvent vite leurs limites quand il s'agit de stocker d'importants volumes de données en mémoire. C'est le cas notamment à cause de la représentation utilisée, provoquant une forme d'inflation du poids des données une fois ces dernières chargées. Quand cette problématique ne se pose pas, ce sont les outils d'observation eux-mêmes qui peuvent soit produire une erreur, soit être illisible. Il faut bien avouer, que les graphiques présentés dans la dernière partie de cette thèse sont plus difficiles à interpréter, certainement à cause du choix réalisé en terme de représentation, mais aussi à cause du volume important de points.

Si Weka nous a servi pour la réalisation de l'exploration supervisée, c'est Scikit-Learn que nous avons utilisé pour l'exploration non supervisée. L'utilisation de librairie comme numpy permet de répondre notamment à la problématique du stockage des données une fois ces dernières chargées, particulièrement grâce à leur approche basée sur des BLOB. Ceci

étant, si le chargement et le stockage fonctionnent, le traitement lui, peut réserver son lot de surprises, avec des implémentations d'algorithmes peu optimisés pour traiter des gros volumes de données. Nous n'avons pas été confrontés, lors de nos travaux, à des situations où les données ne pouvaient pas être toutes chargées d'une traite, et l'on peut se demander quels outils il aurait fallu trouver pour pouvoir produire des représentations et des analyses concluantes en chargeant les sources par blocs. Dans tous les cas, notre travail s'est retrouvé nécessairement limité aussi, par ces problématiques d'ordre purement technique.

Finalement, on pourrait conclure l'autocritique de ce dernier travail sur le choix des représentations. Nous avons utilisé des projections et des grilles de figures pour pouvoir observer plus directement nos données. Ces éléments ne permettent cependant pas de tirer des conclusions solides, soient par ce que ces dernières sont basées sur une transformation pouvant altérer notre perception de la problématique dans le cas des projections, soit par ce qu'elles sont basées sur un espace trop spécifique, dans le cas des grilles par exemple. Il est évident que cet aspect précis de notre travail pourrait faire l'objet d'une thèse à part entière, notamment s'il s'agissait d'explorer des représentations intéressantes exploitant les nouvelles technologies comme la réalité virtuelle ou augmentée, la 3D, ou l'exploration automatisée.

Dans l'ensemble, il nous est possible de dire que ce travail de thèse, de par l'ampleur que lui permet d'atteindre son inscription dans le projet plus vaste qu'est le projet SAPA, pose réellement la question du développement d'outils plus spécialisés, plus ergonomiques et plus performant à destination uniquement du travail de recherche. Que ce soit pour stocker facilement, explorer rapidement ou représenter de manière automatique des ensembles de données bien plus vastes et proches d'une réalité de terrain toujours plus complexe que celle théorisée dans les travaux de laboratoire classique. L'environnement de la recherche rend par ailleurs le développement de ce type d'infrastructure difficile car il impose à ces dernières d'être suffisamment adaptable pour pouvoir mettre en place très rapidement tout type

d'expérimentation. Il s'avère, ceci étant, que ces outils soient un travail en cours, travail dans lequel s'inscrit cette thèse et nécessitant des apports de plusieurs domaines de compétences différents.

7.3 TRAVAUX FUTURS

En guise de conclusion, nous allons nous pencher sur les possibles travaux futurs et questions ouvertes par les différents travaux présentés au sein de cette thèse.

Pour ce qui est de l'architecture que nous avons proposée, présentée au sein du Chapitre 4, son évaluation présentée dans la Section 7.2 est déjà riche en possibilités d'amélioration. Parmi ces dernières, on peut considérer tout naturellement d'évaluer le gain en termes de temps apporté par l'utilisation d'une base de données plus spécialisées afin de stocker et récupérer les informations reçues. Il est aussi possible de mener un travail important au sujet de l'ergonomie du système dans son ensemble, que ce soit dans la facilitation de son installation, de sa manipulation et de sa maintenance. Cette étape peut d'ailleurs mener à repenser l'approche pour quitter la configuration et passer à un système totalement configurable via du code. On peut aussi imaginer, comme explicité dans l'autocritique, travailler à l'amélioration de la validation de la plateforme en facilitant la rédaction et l'exécution des tests.

En dehors de ces aspects liés à l'évaluation que nous avons proposée, on peut aussi se poser la question de l'intégration d'agents apprenants. Le processus d'apprentissage étant certainement un peu plus complexe à mettre en oeuvre dans notre situation, ce dernier nécessitant de s'adapter en fonction de l'ajout, la suppression ou la mutation du jeu de données dans le temps. Évidemment, si certaines approches comme les K plus proches voisins ou les arbres de décision peuvent s'avouer assez évidents à adapter, les réseaux neuronaux, peuvent eux, vite s'avérer plutôt complexes.

Pour finir, nous avons proposé ici une plateforme monolithique prenant en charge les deux aspects d'écriture et de lecture en même temps. Il pourrait être particulièrement intéressant de travailler à diviser cet ensemble en deux solutions séparées, voir peut-être même trois, afin de pouvoir changer ou se passer d'interface pour requêter les données si nécessaire.

En ce qui concerne notre second travail, présenté au sein du Chapitre 5, il serait intéressant d'explorer de nouvelles caractéristiques floues pour remplacer certaines de nos propositions, peut-être moins facilement transférables. On pourrait ainsi imaginer ajouter une composante pour l'intéraction avec des appareils électriques ou chercher à améliorer les composantes relatives à la variété des capteurs et à la quantitée de messages reçus. Il pourrait aussi être intéressant, pour ce qui est des règles, de proposer une méthode basée sur l'aggrégation non supervisée d'instances afin de pouvoir sortir ces définitions de manière automatique. Finalement, pour ce qui est de la proposition faite au sujet de l'estimation de la qualité des instances, il pourrait être pertinent d'étendre cette approche en calculant d'autres caractérisations. La ressemblance d'une instance vis-à-vis de celles déjà observées, un score permettant de caractériser les habitudes et les exceptions ou un score pour apprécier la relation entre certaines instances et la réponse à certains besoins pourrait être trois options intéressantes à explorer.

S'il fallait remettre le fond de ce travail en question, on pourrait aussi proposer de quitter cette idée des scores ou d'un étiquetage très formel pour tenter de caractériser des espaces de temps à l'aide de notions sémantiques. Ce pour quoi est faite la logique floue et ce qui nous rapprocherait un peu plus d'une forme d'analyse de texte, où les événements forment une liste de symboles permettant de donner du sens à certains espaces de temps, devenant des mots. Reste alors à tenter d'interpréter le sens du texte, comme cela peut se faire dans d'autres travaux usant de ce type de modèle. On pourrait alors considérer que les événements sont plus ou moins liés à différentes idées, des classes d'activités, et tenter de faire notre reconnaissance

en évaluant à quel point un ensemble d'événements se rapproche d'une notion plutôt qu'une autre.

Reste alors nos derniers travaux d'exploration présentés au sein du Chapitre 6. Il est évident qu'à l'aube des problématiques que nous y avons rencontrées, ce travail pose réellement la question de la validation des procédés et des données quand ces derniers s'inscrivent dans l'évolution d'un système complexe.

En dehors de la problématique de la fiabilité, il serait intéressant de déterminer dans quelle mesure il serait possible de convertir cette approche en pure méthode de reconnaissance d'activités. On peut aussi chercher à déterminer pour de bon si l'on peut utiliser notre proposition pour filtrer et bonifier des jeux de données, et donc valider en quelque sorte son étiquetage au regard de connaissances que l'on possède déjà. Si possible, il pourrait être intéressant que cette validation soit double, entraînant ainsi aussi une amélioration des ensembles de règles eux même.

Pour terminer, comme le laissait entendre la fin de notre section d'évaluation, il pourrait être intéressant de creuser des sujets plus proches de l'outillage d'analyse scientifique. En proposant des outils plus performants, comme ceux accessibles à l'heure actuelle pour l'apprentissage profond.

7.4 APPORTS PERSONNELS

Cette thèse est l'aboutissement de quatre années d'investissement. Emprunter la voie de ce doctorat, c'était entreprendre un long voyage dont le crépuscule offre l'opportunité de se poser un instant pour apprécier le chemin parcouru dans son ensemble. C'est donc l'occasion de faire le point sur ce que ces quatre années m'ont apporté, que ce soit du côté technique comme du côté social.

Pour ce qui est de la technique d'abord, cette thèse fut évidemment l'occasion de partager du contenu avec la communauté scientifique, permettant ainsi de parfaire sa maîtrise de la langue anglaise, ses compétences rédactionnelles et sa rigueur intellectuelle. Ce fut aussi l'occasion de s'investir dans un projet d'ampleur en partant de zéro, et ainsi d'expérimenter et comprendre les tenants et aboutissants de la gestion du temps, des contraintes et surtout des revers quand ils se présentent. Pour ce qui est du sujet en lui-même, ce fut l'occasion d'aiguiser mes connaissances pratiques de l'informatique, en particulier du domaine du génie logiciel, des systèmes d'exploitation et aussi de l'aspect matériel. Ce fut aussi l'occasion d'approfondir mes connaissances théoriques, en particulier liées au domaine de l'apprentissage machine, que ce soit relatif à l'optimisation, aux probabilités, à l'algorithmique ou à l'algèbre. Plus spécifiquement, ce fut finalement l'opportunité de découvrir l'apprentissage profond, le forage de données et finalement le coeur de ce travail, la reconnaissance d'activités.

J'ai fait le point sur les savoir-faire, il est donc temps de parler de l'aspect plus humain de ces quatre dernières années. Réaliser un projet de cette ampleur, c'est tout d'abord apprendre la patience, une capacité indispensable à la bonne conduite de nombreuses idées. C'est aussi travailler en équipe, apprendre à réaliser des compromis, à échanger de manière constructive et à entretenir des rapports humains. Cette thèse fut l'occasion d'aiguiser ma curiosité de tout, mais en comprenant bien l'investissement nécessaire pour obtenir une réelle expertise sur un sujet particulier. Ce fut, de plus, l'occasion de former mon esprit et, en particulier, mon esprit critique, en me confrontant à la rigueur scientifique. Quatre années au Québec, c'est aussi le temps d'apprendre à vivre à l'étranger, à connaître une autre culture, une autre histoire et un autre territoire.

Finalement, on peut dire que ces quatre années furent l'occasion d'apprendre, d'apprendre encore, d'apprendre beaucoup, mais d'apprendre surtout que le travail de formation n'est pas un moment éphémère, mais bien l'entreprise d'une vie entière.

BIBLIOGRAPHIE

Abdallah, Z. S., Gaber, M. M., Srinivasan, B. & Krishnaswamy, S. (2018). Activity recognition with evolving data streams : A review. *ACM Computing Surveys*, *51*(4).

Abidine, B. M., Fergani, L., Fergani, B. & Oussalah, M. (2018). The joint use of sequence features combination and modified weighted SVM for improving daily activity recognition. *Pattern Analysis and Applications*, *21*(1), 119–138.

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A. & Arshad, H. (2018). State-of-the-art in artificial neural network applications : A survey. *Heliyon*, *4*(11), e00938.

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Umar, A. M., Linus, O. U., Arshad, H., Kazaure, A. A., Gana, U. & Kiru, M. U. (2019). Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, *7*, 158820–158846.

Aggarwal, J. & Ryoo, M. (2011). Human activity analysis : A review. *ACM Computing Surveys*, *43*(3).

Agrawal, R. & Srikant, R. (1994). Fast algorithms for mining association rules. Dans *Proceedings of the 20th International Conference of Very Large Data Bases, VLDB*, Vol. 1215, pp. 487–499.

Albuquerque, P. U. B., Ohi, D. K. d. A., Pereira, N. S., Prata, B. d. A. & Barroso, G. C. (2018). Proposed architecture for energy efficiency and comfort optimization in smart homes. *Journal of Control, Automation and Electrical Systems*, *29*(6), 718–730.

Alzheimer's Disease International (2015). *World Alzheimer report 2015*.

Amigó, E., Gonzalo, J., Artiles, J. & Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, *12*(4), 461–486.

Anguita, D., Ghio, A., Oneto, L., Parra, X. & Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. Dans *International Workshop on Ambient Assisted Living*, pp. 216–223. Springer.

Arduino (2005). *Arduino*. Repéré le 29 Juin 2020, à http://web.archive.org/web/20200629/https://www.arduino.cc/

Arens, M. & Nagel, H.-H. (2003). Behavioral knowledge representation for the understanding and creation of video sequences. Dans *Annual Conference on Artificial Intelligence*, pp. 149–163. Springer.

Arthur, J. & Azadegan, S. (2005). Spring framework for rapid open source J2EE web application development : a case study. Dans *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pp. 90–95.

Asghari, P., Soleimani, E. & Nazerfard, E. (2020). Online human activity recognition employing hierarchical hidden Markov models. *Journal of Ambient Intelligence and Humanized Computing*, *11*(3), 1141–1152.

Augusto, J., Giménez-Manuel, J., Quinde, M., Oguego, C., Ali, M. & James-Reynolds, C. (2020). A Smart Environments Architecture (SEArch). *Applied Artificial Intelligence*, *34*(2), 155–186.

Bae, I.-H. (2014). An ontology-based approach to ADL recognition in smart homes. *Future Generation Computer Systems*, *33*, 32 – 41.

Bainomugisha, E., Carreton, A. L., Cutsem, T. v., Mostinckx, S. & Meuter, W. d. (2013). A survey on reactive programming. *ACM Computing Surveys (CSUR)*, *45*(4), 1–34.

Balli, S., Sağbaş, E. A. & Peker, M. (2019). Human activity recognition from smart watch sensor data using a hybrid of principal component analysis and random forest algorithm. *Measurement and Control*, *52*(1-2), 37–45.

Banos, O., Galvez, J.-M., Damas, M., Pomares, H. & Rojas, I. (2014). Window size impact in human activity recognition. *Sensors*, *14*(4), 6474–6499.

Bao, K., Mauser, I., Kochanneck, S., Xu, H. & Schmeck, H. (2016). A microservice architecture for the intranet of things and energy in smart buildings. Dans *Proceedings of the 1st International Workshop on Mashups of Things and APIs*, pp. 1–6.

Bao, L. & Intille, S. S. (2004). Activity recognition from user-annotated acceleration data. Dans *International Conference on Pervasive Computing*, pp. 1–17. Springer.

Baumgartner, R. N., Wayne, S. J., Waters, D. L., Janssen, I., Gallagher, D. & Morley, J. E. (2004). Sarcopenic obesity predicts instrumental activities of daily living disability in the elderly. *Obesity Research*, *12*(12), 1995–2004.

Bayes, T. (1763). LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. *Philosophical Transactions of the Royal Society of London*, *53*, 370–418.

Ben-Kiki, O., Evans, C. & Ingerson, B. (2004). *YAML Ain't Markup Language (YAML™) 1.0 final draft*. YAML. https ://yaml.org/spec/1.0/.

Ben-Kiki, O., Evans, C. & Ingy döt Net (2009). *YAML Ain't Markup Language (YAML™) version 1.2 final draft*. YAML. https ://yaml.org/spec/1.2/spec.html.

Benabdellah, A. C., Benghabrit, A. & Bouhaddou, I. (2019). A survey of clustering algorithms for an industrial context. *Procedia Computer Science*, *148*, 291–302.

Berry, M. W., Kogan, J. & on Data Mining., S. I. C. (2010). *Text mining : Applications and theory*. Wiley.

Bettadapura, V., Schindler, G., Plötz, T. & Essa, I. (2013). Augmenting bag-of-words : Data-driven discovery of temporal and structural information for activity recognition. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2619–2626.

Beynon-Davies, P. (1996). *Database systems*. Macmillan Education, Limited.

Bihari, A., Tripathi, S. & Deepak, A. (2019). Gene expression analysis using clustering techniques and evaluation indices. Dans *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE)*.

Blalock, D., Ortiz, J. J. G., Frankle, J. & Guttag, J. (2020). What is the state of neural

network pruning ? *arXiv preprint arXiv :2003.03033*.

Bonner, R. E. (1964). On some clustering techniques. *IBM Journal of Research and Development*, *8*(1), 22–32.

Bouchard, K., Bouzouane, A. & Bouchard, B. (2013). Discovery of topological relations for spatial activity recognition. Dans *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 73–80. IEEE.

Boukhers, Z., Wete, D. & Staab, S. (2020). LaHAR : Latent Human Activity Recognition using LDA. *arXiv preprint arXiv :2011.11151*.

Brand, M., Oliver, N. & Pentland, A. (1997). Coupled hidden Markov models for complex action recognition. Dans *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 994–999. IEEE.

Bray, T. (2017). *The JavaScript Object Notation (JSON) data interchange format* publication n° 90. RFC Editor.

Breiman, L., Friedman, J., Stone, C. J. & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Buczak, A. L. & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, *18*(2), 1153–1176.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F. & Li, H. (2007). Learning to rank : From pairwise approach to listwise approach. Dans *Proceedings of the 24th International Conference on Machine Learning*, pp. 129–136.

Castro, D., Coral, W., Rodriguez, C., Cabra, J. & Colorado, J. (2017). Wearable-based human activity recognition using an iot approach. *Journal of Sensor and Actuator Networks*, *6*(4), 28.

Cendrowska, J. (1987). PRISM : An algorithm for inducing modular rules. *International*

Journal of Man-Machine Studies, *27*(4), 349–370. 349.

Chan, N. H. (2010). Introduction. Dans *Time Series : Applications to Finance with R and S-Plus®, Second Edition* (pp. 1–14). John Wiley & Sons, Ltd.

Chen, A. I., Balter, M. L., Maguire, T. J. & Yarmush, M. L. (2020). Deep learning robotic guidance for autonomous vascular access. *Nature Machine Intelligence*, *2*(2), 104–115.

Chen, C., Das, B. & Cook, D. J. (2010). A data mining framework for activity recognition in smart environments. Dans *2010 Sixth International Conference on Intelligent Environments*, pp. 80–83.

Chen, L., Hoey, J., Nugent, C. D., Cook, D. J. & Yu, Z. (2012). Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(6), 790–808.

Chen, L., Nugent, C. D. & Wang, H. (2011). A knowledge-driven approach to activity recognition in smart homes. *IEEE Transactions on Knowledge and Data Engineering*, *24*(6), 961–974.

Chen, L. C. s. & Nugent, C. D. (2019). *Human activity recognition and behaviour analysis : For cyber-physical systems in smart environments*. Springer.

Chen, M., Hao, Y., Hwang, K., Wang, L. & Wang, L. (2017). Disease prediction by machine learning Over big data from healthcare communities. *IEEE Access*, *5*, 8869–8879.

Chen, Z., Zhang, L., Cao, Z. & Guo, J. (2018). Distilling the knowledge from handcrafted features for human activity recognition. *IEEE Transactions on Industrial Informatics*, *14*(10), 4334–4342.

Chen, Z., Zhu, Q., Soh, Y. C. & Zhang, L. (2017). Robust human activity recognition using smartphone sensors via CT-PCA and online SVM. *IEEE Transactions on Industrial Informatics*, *13*(6), 3070–3080.

Cherfi, A., Nouira, K. & Ferchichi, A. (2018). Very fast C4. 5 decision tree algorithm.

Applied Artificial Intelligence, *32*(2), 119–137.

Chikhaoui, B., Wang, S. & Pigot, H. (2011). A frequent pattern mining approach for ADLs recognition in smart environments. Dans *2011 IEEE International Conference on Advanced Information Networking and Applications*, pp. 248–255. IEEE.

Chowdhary, C. L., Mittal, M., Pattanaik, P. & Marszalek, Z. (2020). An efficient segmentation and classification system in medical images using intuitionist possibilistic fuzzy C-mean clustering and fuzzy SVM algorithm. *Sensors*, *20*(14), 3903.

Cios, K. J. (2007). *Data mining : A knowledge discovery approach*.

Civitarese, G., Bettini, C., Sztyler, T., Riboni, D. & Stuckenschmidt, H. (2018). NECTAR : Knowledge-based collaborative active learning for activity recognition. Dans *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10. IEEE.

Conti, G., Bratus, S., Shubina, A., Sangster, B., Ragsdale, R., Supan, M., Lichtenberg, A. & Perez-Alemany, R. (2010). Automated mapping of large binary objects using primitive fragment type classification. *Digital Investigation*, *7*, S3–S12.

Cook, D. (2012). Learning setting-generalized activity models for smart spaces. *IEEE Intelligent Systems*, *27*(1), 32–38.

Cook, D., Crandall, A., Thomas, B. & Krishnan, N. (2013). *CASAS : A smart home in a box*.

Cook, D. & Das, S. (2005). *Smart environments : Technology, protocols and applications*. Wiley-Interscience.

Cook, D. J. (2010). Learning setting-generalized activity models for smart spaces. *IEEE Intelligent Systems*, *2010*(99), 1.

Cook, D. J. (2015). *Activity learning : Discovering, recognizing, and predicting human behavior from sensor data*. Wiley.

Cook, D. J., Krishnan, N. C. & Rashidi, P. (2013). Activity discovery and activity recognition : A new partnership. *IEEE Transactions on Cybernetics*, *43*(3), 820–828.

Cook, D. J. & Schmitter-Edgecombe, M. (2009). Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, *48*(05), 480–485.

Cormack, R. M. (1971). A review of classification. *Journal of the Royal Statistical Society : Series A (General)*, *134*(3), 321–353.

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, *EC-14*(3), 326–334.

Covinsky, K. E., Hilton, J., Lindquist, K. & Dudley, R. A. (2006). Development and validation of an index to predict activity of daily living dependence in community-dwelling elders. *Medical Care*, *44*(2), 149–157.

Crandall, A. & Cook, D. J. (2010). Learning activity models for multiple agents in a smart space. Dans *Handbook of Ambient Intelligence and Smart Environments* pp. 751–769. Springer.

Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D. & Ho, S. (2020). Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv :2006.11287*.

de développement de GraphQL, É. (2018). *GraphQL*. Facebook. http ://spec.graphql.org/June2018.

de Sousa Santos, I., de Castro Andrade, R. M., Rocha, L. S., Matalonga, S., de Oliveira, K. M. & Travassos, G. H. (2017). Test case design for context-aware applications : Are we there yet ? *Information and Software Technology*, *88*, 1–16.

Demongivert, C., Bouchard, K., Gaboury, S., Bouchard, B., Lussier, M., Parenteau, M., Laliberté, C., Couture, M., Bier, N. & Giroux, S. (2021). A distributable event-oriented architecture for activity recognition in smart homes. *Journal of Reliable Intelligent Environments*, pp. 1–17.

Demongivert, C., Bouchard, K., Gaboury, S., Lussier, M., Kenfack-Ngankam, H., Couture, M., Bier, N. & Giroux, S. (2020). Fuzzy features for quality estimation of activity instances in a dataset. Dans *Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good*, pp. 54–59.

Demongivert, C., Bouchard, K., Gaboury, S., Lussier, M., Kenfack-Ngankam, H., Couture, M., Bier, N. & Giroux, S. (2021). Handling of labeling uncertainty in smart homes using generalizable fuzzy features. Dans *Proceedings of the Conference on Information Technology for Social Good*, GoodIT '21, p. 248–253. ACM.

Di Giorgio, A. & Pimpinella, L. (2012). An event driven smart home controller enabling consumer economic saving and automated demand side management. *Applied Energy*, *96*, 92–103.

Domingos, P. & Pazzani, M. (1996). Beyond independence : Conditions for the optimality of the simple bayesian classifier. Dans *Proceedings of the 13th International Conference on Machine Learning*, pp. 105–112.

Domingos, P. & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, *29*(2-3), 103–130.

Dong, G. & Pei, J. C. s. (2007). *Sequence data mining*. Springer.

Doruk, H., Mas, M., Yıldız, C., Sonmez, A. & Kýrdemir, V. (2004). The effect of the timing of hip fracture surgery on the activity of daily living and mortality in elderly. *Archives of Gerontology and Geriatrics*, *39*(2), 179 – 185.

Du, W. & Zhan, Z. (2002). Building decision tree classifier on private data. *Electrical Engineering and Computer Science*, *8*.

Équipe de Kafka (2020). *Apache Kafka 2.6*. Repéré le 28 Octobre 2020, à http://web.archive.org/web/20201028020408/https://kafka.apache.org/documentation

Équipe de Kafka (2020). *Apache Kafka protocol*. Repéré le 28 Octobre 2020, à http://web.archive.org/web/20201028020408/https://kafka.apache.org/protocol

Équipe de RabbitMQ (2020). *RabbitMQ 3.8.4*. Repéré le 28 Octobre 2020, à http://web.archive.org/web/20201028020408/https://www.rabbitmq.com/documentation.html

Équipe de REDIS (2020). *Documentation REDIS 6.0.9*. Repéré le 28 Octobre 2020, à http://web.archive.org/web/20201028020408/https://redis.io/documentation

Équipe du Benjamin Rose Hospital (1959). Multidisciplinary studies of illness in aged persons : II. A new classification of functional status in activities of daily living. *Journal of Chronic Diseases*, *9*(1), 55 – 62.

Ercolano, G., Riccio, D. & Rossi, S. (2017). Two deep approaches for ADL recognition : A multi-scale LSTM and a CNN-LSTM with a 3D matrix skeleton representation. Dans *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 877–882.

European association for standardizing information and communication systems (1999). *ECMAScript language specification*. ECMA. http ://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262,%203rd%20edition,%20December%201999.pdf.

European association for standardizing information and communication systems (2017). *The JSON data interchange syntax*. ECMA. http ://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

Everitt, B. S., Landau, S., Leese, M. & Stahl, D. (2011). *Cluster analysis 5th ed*.

Fatima, I., Fahim, M. & Lee, S. (2013). Analysis and effects of smart home dataset characteristics for daily life activity recognition. *The Journal of Supercomputing*, *66*.

Ferdoush, S. & Li, X. (2014). Wireless sensor network system design using Raspberry Pi and Arduino for environmental monitoring applications. *Procedia Computer Science*, *34*, 103 – 110.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. (Thèse).

Forgy, E. W. (1965). Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, *21*, 768–769.

Forte, J. C., Perner, A. & van der Horst, I. C. (2019). The use of clustering algorithms in critical care research to unravel patient heterogeneity. *Intensive Care Medicine*, *45*(7), 1025–1028.

Fowlkes, E. B. & Mallows, C. L. (1983). A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, *78*(383), 553–569.

Freed, N., Klensin, D. J. C. & Hansen, T. (2013). *Media type specifications and registration procedures*. RFC 6838.

Freedman, R., Jung, H.-T. & Zilberstein, S. (2014). Plan and activity recognition from a topic modeling perspective. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 24.

Galbrun, E., Cellier, P., Tatti, N., Termier, A. & Crémilleux, B. (2018). Mining periodic patterns with a MDL criterion. Dans *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 535–551. Springer.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. M. (1994). *Design patterns : Elements of reusable object-oriented software* (1). Addison-Wesley Professional.

Gantz, J. & Reinsel, D. (2011). *Extracting value from chaos*. International Data Corporation.

Garrod, R., Bestall, J., Paul, E., Wedzicha, J. & Jones, P. (2000). Development and validation of a standardized measure of activity of daily living in patients with severe COPD : The London Chest Activity of Daily Living scale (LCADL). *Respiratory Medicine*, *94*(6), 589 – 596.

Gill, K., Yang, S., Yao, F. & Lu, X. (2009). A Zigbee-based home automation system. *IEEE Transactions on Consumer Electronics*, *55*(2), 422–430.

Gislason, D. (2008). *Zigbee wireless networking*. Newnes.

Gjoreski, H. & Roggen, D. (2017). Unsupervised online activity discovery using temporal behaviour assumption. Dans *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, pp. 42–49.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, *10*(1), 1–309.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT Press. Repéré le 26 Juillet 2022, à http://www.deeplearningbook.org

Gordon, A. D. (1999). *Classification* (2). Chapman and Hall/CRC.

Grünwald, P. D. & Grunwald, A. (2007). *The minimum description length principle*. MIT press.

Gu, F., Chung, M.-H., Chignell, M., Valaee, S., Zhou, B. & Liu, X. (2021). A survey on deep learning for human activity recognition. *ACM Computing Surveys (CSUR)*, *54*(8), 1–34.

Gu, S., Kelly, B. & Xiu, D. (2021). Autoencoder asset pricing models. *Journal of Econometrics*, *222*(1), 429–450.

Gudgin, M., Lafon, Y., Nielsen, H. F., Mendelsohn, N., Karmarkar, A., Moreau, J.-J. & Hadley, M. (2007). *SOAP version 1.2 part 2 : Adjuncts (second edition)*. W3C. https ://www.w3.org/TR/2007/REC-soap12-part2-20070427/.

Gutierrez, F. (2019). Web applications with Spring Boot. Dans *Pro Spring Boot 2 : An Authoritative Guide to Building Microservices, Web and Enterprise Applications, and Best Practices* (pp. 87–125). Apress.

H Witten, I., Frank, E., A Hall, M. & J Pal, C. (2017). *Data mining practical machine learning tools and techniques*.

Hahsler, M., Piekenbrock, M. & Doran, D. (2019). DBScan : Fast density-based clustering with R. *Journal of Statistical Software*, *91*(1), 1–30.

Halkidi, M., Batistakis, Y. & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, *17*(2), 107–145.

Hallé, S. (2017). From complex event processing to simple event processing. *arXiv preprint arXiv :1702.08051*.

Hämäläinen, J., Jauhiainen, S. & Kärkkäinen, T. (2017). Comparison of internal clustering validation indices for prototype-based clustering. *Algorithms*, *10*(3), 105.

Han, J., Pei, J. & Kamber, M. (2011). *Data mining : concepts and techniques*. Elsevier.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. & Hsu, M. (2001). Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. Dans *Proceedings of the 17th International Conference on Data Engineering*, pp. 215–224. Citeseer.

Hassan, B. A. & Rashid, T. A. (2021). A multidisciplinary ensemble algorithm for clustering heterogeneous datasets. *Neural Computing and Applications*, pp. 1–24.

Hassan, M. M., Uddin, M. Z., Mohamed, A. & Almogren, A. (2018). A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems*, *81*, 307 – 313.

He, Z. & Jin, L. (2009). Activity recognition from acceleration data based on discrete consine transform and SVM. Dans *2009 IEEE International Conference on Systems, Man and Cybernetics*, pp. 5041–5044. IEEE.

Helal, S., Bose, R., Pickles, S., Elzabadani, H., King, J. & Kaddourah, Y. (2008). The Gator Tech smart house : A programmable pervasive space. Dans *The engineering handbook of smart technology for aging, disability, and independence* (pp. 693–709). John Wiley & Sons, Ltd.

Helbostad, J. L., Vereijken, B., Becker, C., Todd, C., Taraldsen, K., Pijnappels, M., Aminian, K. & Mellone, S. (2017). Mobile health applications to promote active and healthy ageing. *Sensors*, *17*(3).

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hssina, B., Merbouha, A., Ezzikouri, H. & Erritali, M. (2014). A comparative study of decision tree ID3 and C4. 5. *International Journal of Advanced Computer Science and Applications*, *4*(2), 13–19.

Huang, C. & Townshend, J. (2003). A stepwise regression tree for nonlinear approximation : Applications to estimating subpixel land cover. *International Journal of Remote Sensing*, *24*(1), 75–90.

Hussain, Z., Sheng, M. & Zhang, W. E. (2019). Different approaches for human activity recognition : A survey. *arXiv preprint arXiv :1906.05074*.

Huynh, T., Fritz, M. & Schiele, B. (2008). Discovery of activity patterns using topic models. Dans *Proceedings of the 10th International Conference on Ubiquitous Computing*, pp. 10–19.

ISO (1986). *Information processing – Text and office systems – Standard Generalized Markup Language (SGML).* International Organization for Standardization.

Japkowicz, N. (2001). Supervised versus unsupervised binary-learning by feedforward neural networks. *Machine Learning*, *42*(1-2), 97–122.

Jarraya, A., Ramoly, N., Bouzeghoub, A., Arour, K., Borgi, A. & Finance, B. (2016). A fuzzy semantic CEP model for situation identification in smart homes. Dans *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pp. 1678–1679.

Jefferson, A. L., Paul, R. H., Ozonoff, A. & Cohen, R. A. (2006). Evaluating elements of executive functioning as predictors of Instrumental Activities of Daily Living (IADLs). *Archives of Clinical Neuropsychology*, *21*(4), 311–320.

Jiménez, A. R. & Seco, F. (2018). Multi-event naive Bayes classifier for activity recognition in the UCAmI Cup. Dans *Multidisciplinary Digital Publishing Institute Proceedings*, Vol. 2, p. 1264.

Judge, J. O., Schechtman, K., Cress, E. & Group, F. (1996). The relationship between physical performance measures and independence in instrumental activities of daily living. *Journal of the American Geriatrics Society*, *44*(11), 1332–1341.

Kaewkasi, C. & Gurd, J. R. (2008). Groovy AOP : A dynamic AOP system for a JVM-based language. Dans *Proceedings of the 2008 AOSD Workshop on Software Engineering Properties of Languages and Aspect Technologies*, SPLAT '08, pp. 3 :1–3 :6. ACM.

Kantety, R. V., La Rota, M., Matthews, D. E. & Sorrells, M. E. (2002). Data mining for simple sequence repeats in expressed sequence tags from barley, maize, rice, sorghum and wheat. *Plant Molecular Biology*, *48*(5), 501–510.

Karabatak, M. (2015). A new classifier for breast cancer detection based on naïve bayesian. *Measurement*, *72*, 32 – 36.

Kassahun, Y., Yu, B., Tibebu, A., Stoyanov, D., Giannarou, S., Metzen, J. & Vander Poorten, E. (2015). Surgical robotics beyond enhanced dexterity instrumentation : A survey of machine learning techniques and their role in intelligent and autonomous surgical actions. *International Journal of Computer Assisted Radiology and Surgery*, *11*.

Katz, S., Ford, A. B., Moskowitz, R. W., Jackson, B. A. & Jaffe, M. W. (1963). Studies of illness in the aged : The index of ADL : A standardized measure of biological and psychosocial function. *Jama*, *185*(12), 914–919.

Kayaalp, F. & Erdogmus, P. (2020). Benchmarking the clustering performances of evolutionary algorithms : A case study on varying data size. *IRBM*, *41*(5), 267–275.

Ke, S.-R., Thuc, H. L. U., Lee, Y.-J., Hwang, J.-N., Yoo, J.-H. & Choi, K.-H. (2013). A review on video-based human activity recognition. *Computers*, *2*(2), 88–131.

Kientz, J. A., Patel, S. N., Jones, B., Price, E., Mynatt, E. D. & Abowd, G. D. (2008). The Georgia tech aware home. Dans *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, p. 3675–3680. ACM.

Kim, E., Helal, S. & Cook, D. (2010). Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, *9*(1), 48–53.

Koushik, J. (2016). Understanding convolutional neural networks. *arXiv preprint arXiv :1605.09081*.

Kreuzer, K. (2017). *OpenHAB 2 has arrived!* Repéré le July 7, 2015, à https://www.openhab.org/blog/2017-01-22-openhab2.html

Krishnan, N. C. & Cook, D. J. (2014). Activity recognition on streaming sensor data. *Pervasive and Mobile Computing*, *10*, 138 – 154.

Laguna, J. O., Olaya, A. G. & Borrajo, D. (2011). A dynamic sliding window approach for activity recognition. Dans *International Conference on User Modeling, Adaptation, and Personalization*, pp. 219–230. Springer.

Lam, H.-K. (2018). A review on stability analysis of continuous-time fuzzy-model-based control systems : From membership-function-independent to membership-function-dependent analysis. *Engineering Applications of Artificial Intelligence*, *67*, 390–408.

Latte Panda (2016). *Latte Panda*. Repéré le 29 Juin 2020, à http://web.archive.org/web/20200629/https://www.lattepanda.com/

Lawton, M. P. & Brody, E. M. (1969). Assessment of older people : Self-maintaining and instrumental activities of daily living. *The Gerontologist*, *9*(3_Part_1), 179–186.

Le Yaouanc, J.-M. & Poli, J.-P. (2012). A fuzzy spatio-temporal-based approach for activity recognition. Dans *International Conference on Conceptual Modeling*, pp. 314–323. Springer.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. & Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, *2*.

Lee, Y.-S. & Cho, S.-B. (2011). Activity recognition using hierarchical hidden Markov models on a smartphone with 3D accelerometer. Dans *International Conference on Hybrid Artificial Intelligence Systems*, pp. 460–467. Springer.

Lehrner, J., Kalchmayr, R., Serles, W., Olbrich, A., Pataraia, E., Aull, S., Bacher, J., Leutmezer, F., Gröppel, G., Deecke, L. & Baumgartner, C. (1999). Health-Related Quality of Life (HRQOL), Activity of Daily Living (ADL) and depressive mood disorder in temporal lobe epilepsy patients. *Seizure*, *8*(2), 88 – 92.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. Dans *Soviet physics doklady*, Vol. 10.8, pp. 707–710. Soviet Union.

Lewis, D. D. (1998). Naive (Bayes) at forty : The independence assumption in information retrieval. Dans *European Conference on Machine Learning*, pp. 4–15. Springer.

Lhomme, S., Rice, D. & Bunkus, M. (2020). *Extensible binary meta language*. RFC 8794.

Li, J.-H., Tian, L., Wang, H., An, Y., Wang, K. & Yu, L. (2019). Segmentation and recognition of basic and transitional activities for continuous physical human activity. *IEEE Access*, *7*, 42565–42576.

Li, Q., Jin, Y., He, T. & Xu, H. (2013). Smart home services based on event matching. Dans *2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pp. 762–766. IEEE.

Li, X., Zhang, Y., Marsic, I., Sarcevic, A. & Burd, R. S. (2016). Deep learning for RFID-based activity recognition. Dans *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pp. 164–175.

Li, Z., Dey, N., Ashour, A. S., Cao, L., Wang, Y., Wang, D., McCauley, P., Balas, V. E., Shi, K. & Shi, F. (2017). Convolutional neural network based clustering and manifold learning method for diabetic plantar pressure imaging dataset. *Journal of Medical Imaging and Health Informatics*, *7*(3), 639–652.

Liciotti, D., Bernardini, M., Romeo, L. & Frontoni, E. (2020). A sequential deep learning application for recognising human activities in smart homes. *Neurocomputing*, *396*, 501–513.

Liu, L., Wang, S., Hu, B., Qiong, Q., Wen, J. & Rosenblum, D. S. (2018). Learning structures of interval-based Bayesian Networks in probabilistic generative model for human

complex activity recognition. *Pattern Recognition*, *81*, 545–561.

Liu, L., Wang, S., Su, G., Huang, Z.-G. & Liu, M. (2017). Towards complex activity recognition using a bayesian network-based probabilistic generative framework. *Pattern Recognition*, *68*, 295 – 309.

Liu, Y., Nie, L., Liu, L. & Rosenblum, D. S. (2016). From action to activity : Sensor-based activity recognition. *Neurocomputing*, *181*, 108–115.

Lopez Medina, M. A., Espinilla, M., Paggeti, C. & Medina Quero, J. (2019). Activity recognition for IoT devices using fuzzy spatio-temporal features as environmental sensor fusion. *Sensors*, *19*(16), 3512.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., Brinkkemper, S. & Zowghi, D. (2017). Behavior-driven requirements traceability via automated acceptance tests. Dans *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pp. 431–434. IEEE.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Dans *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1.14, pp. 281–297. Oakland, CA, USA.

Mahmoud, S. M., Lotfi, A. & Langensiepen, C. (2012). User activities outlier detection system using principal component analysis and fuzzy rule-based system. Dans *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*, pp. 1–8.

Malazi, H. T. & Davari, M. (2018). Combining emerging patterns with random forest for complex activity recognition in smart homes. *Applied Intelligence*, *48*(2), 315–330.

Maler, E., Bray, T., Sperberg-McQueen, M., Yergeau, F. & Paoli, J. (2008). *Extensible markup language (XML) 1.0 (fifth edition)*. W3C. https ://www.w3.org/TR/2008/REC-xml-20081126/.

Manning, C. D., Raghavan, P. & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Markov, A. A. (1906). Extension of the law of large numbers to dependent quantities. *Bulletin of the Society of the Physics Mathematics*, *2*, 155–156.

Martindale, C. F., Sprager, S. & Eskofier, B. M. (2019). Hidden Markov model-based smart annotation for benchmark cyclic activity recognition database using wearables. *Sensors*, *19*(8), 1820.

Martinez-Arroyo, M. & Sucar, L. E. (2006). Learning an optimal naive bayes classifier. Dans *18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 3, pp. 1236–1239. IEEE.

Maurer, U., Smailagic, A., Siewiorek, D. P. & Deisher, M. (2006). Activity recognition and monitoring using multiple sensors on different body positions. Dans *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, pp. 4–pp. IEEE.

McInnes, L., Healy, J. & Melville, J. (2018). UMAP : Uniform Manifold Approximation and Projection for dimension reduction. *arXiv preprint arXiv :1802.03426*.

Medjahed, H., Istrate, D., Boudy, J. & Dorizzi, B. (2009). Human activities of daily living recognition using fuzzy logic for elderly home monitoring. Dans *2009 IEEE International Conference on Fuzzy Systems*, pp. 2001–2006. IEEE.

Mehta, M., Rissanen, J. & Agrawal, R. (1995). MDL-Based decision tree pruning. Dans *KDD*, Vol. 21.2, pp. 216–221.

Mendelsohn, N., Beech, D., Thompson, H., Maloney, M., Gao, S. & Sperberg-McQueen, M. (2012). *W3C XML Schema Definition language (XSD) 1.1 Part 1 : Structures*. W3C. https ://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/.

Mendelsohn, N., Hadley, M., Gudgin, M., Karmarkar, A., Nielsen, H. F., Lafon, Y. & Moreau, J.-J. (2007). *SOAP version 1.2 part 1 : messaging framework (second edition)*. W3C. https ://www.w3.org/TR/2007/REC-soap12-part1-20070427/.

Michie, D. (1968). "Memo" functions and machine learning. *Nature*, *218*(5136), 19–22.

Millán-Calenti, J. C., Tubío, J., Pita-Fernández, S., González-Abraldes, I., Lorenzo, T., Fernández-Arruty, T. & Maseda, A. (2010). Prevalence of functional disability in Activities of Daily Living (ADL), Instrumental Activities of Daily Living (IADL) and associated factors, as predictors of morbidity and mortality. *Archives of Gerontology and Geriatrics*, *50*(3), 306–310.

Mitchell, T. M. (2004). The role of unlabeled data in supervised learning. Dans *Language, Knowledge, and Representation* pp. 103–111. Springer.

Mo, L., Li, F., Zhu, Y. & Huang, A. (2016). Human physical activity recognition based on computer vision with deep learning model. Dans *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pp. 1–6.

Mohan, P., Padmanabhan, V. N. & Ramjee, R. (2008). Nericell : Rich monitoring of road and traffic conditions using mobile smartphones. Dans *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, p. 323–336. ACM.

Mongiello, M., Patrono, L., Di Noia, T., Nocera, F., Parchitelli, A., Sergi, I. & Rametta, P. (2017). A complex event processing based smart aid system for fire and danger management. Dans *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, pp. 44–49. IEEE.

Mosa, A., Yoo, I. & Sheets, L. (2012). A systematic review of healthcare applications for smartphones. *BMC Medical Informatics and Decision Making*, *12*, 67.

Mozer, M. C. (1999). An intelligent environment must be adaptive. *IEEE Intelligent Systems and their Applications*, *14*(2), 11–13.

Murray, J. D. & VanRyper, W. (1996). *Encyclopedia of graphics file formats (second edition)*. OŔeilly & Associates, Inc.

Mylonas, A., Meletiadis, V., Mitrou, L. & Gritzalis, D. (2013). Smartphone sensor data as digital evidence. *Computers & Security*, *38*, 51 – 75.

Natarajan, P. & Nevatia, R. (2007). Coupled hidden semi Markov models for activity recognition. Dans *2007 IEEE Workshop on Motion and Video Computing (WMVC'07)*, pp.

10–10. IEEE.

Nef, T., Urwyler, P., Büchler, M., Tarnanas, I., Stucki, R., Cazzoli, D., Müri, R. & Mosimann, U. (2015). Evaluation of three state-of-the-art classifiers for recognition of activities of daily living from smart home ambient data. *Sensors (Basel, Switzerland)*, *15*(5), 11725–11740.

Nelson, J. (2016). *Mastering Redis*. Packt Publishing Ltd.

Newman, M. E., Cantwell, G. T. & Young, J.-G. (2020). Improved mutual information measure for clustering, classification, and community detection. *Physical Review E*, *101*(4), 042304.

Nguyen, N. (Jun 2018). Hidden Markov model for stock trading. *International Journal of Financial Studies*, *6*(2).

Nguyen, N. T., Phung, D. Q., Venkatesh, S. & Bui, H. (2005). Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. Dans *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2, pp. 955–960. IEEE.

Ni, Q., Cleland, I., Nugent, C., Hernando, A. B. G. & de la Cruz, I. P. (2019). Design and assessment of the data analysis process for a wrist-worn smart object to detect atomic activities in the smart home. *Pervasive and Mobile Computing*, *56*, 57–70.

Nicolae, B., Antoniu, G. & Bougé, L. (2009). BlobSeer : How to enable efficient versioning for large object storage under heavy access concurrency. Dans *Proceedings of the 2009 EDBT/ICDT Workshops*, pp. 18–25.

Noor, M. H. M., Salcic, Z., Kevin, I. & Wang, K. (2017). Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer. *Pervasive and Mobile Computing*, *38*, 41–59.

Noura, M., Atiquzzaman, M. & Gaedke, M. (2019). Interoperability in internet of things : Taxonomies and open challenges. *Mobile Networks and Applications*, *24*(3), 796–809. 796.

Novac, P.-E., Castagnetti, A., Russo, A., Miramond, B., Pegatoquet, A. & Verdier, F. (2020). Toward unsupervised human activity recognition on microcontroller units. Dans *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pp. 542–550. IEEE.

OASIS Message Queuing Telemetry Transport (MQTT) Technical Committee (2019). *MQTT version 5.0*. OASIS. https ://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

Organisation des Nations Unies (2013). *World population ageing 2013*.

Organisation des Nations Unies (2015). *World population ageing 2015*.

Organisation des Nations Unies (2017). *World population ageing 2017*.

Organisation des Nations Unies (2019). *World population ageing 2019*.

Otter, D. W., Medina, J. R. & Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(2), 604–624.

Pallotta, V., Bruegger, P. & Hirsbrunner, B. (2008). Smart heating systems : Optimizing heating systems by kinetic-awareness. Dans *2008 Third International Conference on Digital Information Management*, pp. 887–892.

Paulraj, P. (2005). The database approach. Dans *Database design and development* (pp. 1–35). John Wiley & Sons, Ltd.

Perumal, T., Sulaiman, M. N., Sharif, K. Y., Ramli, A. R. & Leong, C. Y. (2013). Development of an embedded smart home management scheme. *International Journal of Smart Home*, *7*(2), 15–26.

Plantevin, V., Bouzouane, A. & Gaboury, S. (2017). The light node communication framework : A new way to communicate inside smart homes. *Sensors*, *17*(10).

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C. & Iyengar, S. S. (2018). A survey on deep learning : Algorithms, techniques, and applications.

ACM Computing Surveys (CSUR), *51*(5), 1–36.

Pradhan, S. K., Parhi, D. R. & Panda, A. K. (2009). Fuzzy logic techniques for navigation of several mobile robots. *Applied Soft Computing*, *9*(1), 290 – 304.

Prossegger, M. & Bouchachia, A. (2014). Multi-resident activity recognition using incremental decision trees. Dans *International Conference on Adaptive and Intelligent Systems*, pp. 182–191. Springer.

Qiu, X., Zhang, L., Ren, Y., Suganthan, P. N. & Amaratunga, G. (2014). Ensemble deep learning for regression and time series forecasting. Dans *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)*, pp. 1–6. IEEE.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (1987). Generating production rules from decision trees. Dans *IJCAI*, Vol. 87, pp. 304–307. Citeseer.

Quinlan, J. R. (1993). *C4.5 : Programs for machine learning*. Morgan Kaufmann Publishers Inc.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Raileanu, L. E. & Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, *41*(1), 77–93.

Ramasamy Ramamurthy, S. & Roy, N. (2018). Recent trends in machine learning for human activity recognition—A survey. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, *8*(4), e1254.

Ranasinghe, S., Machot, F. A. & Mayr, H. C. (2016). A review on applications of activity recognition systems with regard to performance and evaluation. *International Journal of Distributed Sensor Networks*, *12*(8), 1550147716665520.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*(336), 846–850.

Rasbperry Pi (2012). *Rasbperry Pi*. Repéré le 29 Juin 2020, à http://web.archive.org/web/20200629/https://www.raspberrypi.org/

Rashidi, P. & Cook, D. J. (2013). COM : A method for mining and monitoring human activity patterns in home-based health monitoring systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *4*(4), 1–20.

Rashidi, P., Cook, D. J., Holder, L. B. & Schmitter-Edgecombe, M. (2010). Discovering activities to recognize and track in a smart environment. *IEEE Transactions on Knowledge and Data Engineering*, *23*(4), 527–539.

Redux (2020). *Redux*. Repéré le 7 Mai 2020, à https://redux.js.org/introduction/getting-started

Rendón, E., Abundez, I., Arizmendi, A. & Quiroz, E. M. (2011). Internal versus external cluster validation indexes. *International Journal of Computers and Communications*, *5*(1), 27–34.

Restrepo, A. (1999). Diagnostic tests : The Katz activities of daily living scale. *The American Journal of Nursing*, *99*(1), 24BB–24DD.

Riboni, D. & Murtas, M. (2019). Sensor-based activity recognition : One picture is worth a thousand words. *Future Generation Computer Systems*, *101*, 709–722.

Rish, I. (2001). An empirical study of the naive Bayes classifier. Dans *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, Vol. 3.22, pp. 41–46.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*(5), 465–471.

Romero, D., Hermosillo, G., Taherkordi, A., Nzekwa, R., Rouvoy, R. & Eliassen, F. (2010). RESTful integration of heterogeneous devices in pervasive environments. Dans *IFIP International Conference on Distributed Applications and Interoperable Systems*, pp. 1–14.

Springer.

Ronao, C. A. & Cho, S.-B. (2014). Human activity recognition using smartphone sensors with two-stage continuous hidden Markov models. Dans *2014 10th International Conference on Natural Computation (ICNC)*, pp. 681–686. IEEE.

Ronao, C. A. & Cho, S.-B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, *59*, 235–244.

Rosati, S., Balestra, G. & Knaflitz, M. (2018). Comparison of different sets of features for human activity recognition by wearable sensors. *Sensors*, *18*(12), 4189.

Roy, N., Misra, A. & Cook, D. (2016). Ambient and smartphone sensor assisted ADL recognition in multi-inhabitant smart environments. *Journal of Ambient Intelligence and Humanized Computing*, *7*(1), 1–19.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C. & Li, F. (2014). ImageNet large scale visual recognition challenge. *CoRR*, *abs/1409.0575*.

Salton, G., Wong, A. & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

San-Segundo, R., Montero, J. M., Moreno-Pimentel, J. & Pardo, J. M. (2016). HMM adaptation for improving a human activity recognition system. *Algorithms*, *9*(3), 60.

Santafe, G., Inza, I. & Lozano, J. A. (2015). Dealing with the evaluation of supervised classification algorithms. *Artificial Intelligence Review*, *44*(4), 467–508.

Sarkar, A. J., Lee, Y.-K. & Lee, S. (2010). A smoothed naive bayes-based classifier for activity recognition. *IETE Technical Review*, *27*(2), 107–119.

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W. & Lin, C.-T. (2017). A review of clustering techniques and developments. *Neurocomputing*, *267*, 664–681.

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W. & Lin, C.-T. (2017). A review of clustering techniques and developments. *Neurocomputing*, *267*, 664–681.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J. & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471.

Schwarz, A. M., Arias, O. S., Zviniene, A., Rudolph, H. P., Eckardt, S., Koettl, J., Immervoll, H. & Abels, M. (2014). *The inverting pyramid : Pension systems facing demographic challenges in europe and central asia*. The World Bank.

Shahad, R. A., Saad, M. H. M. & Hussain, A. (2018). Activity recognition for smart building application using complex event processing approach. *International Journal on Advanced Science, Engineering and Information Technology*, *8*(2), 315–322.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423.

Shapiro, M. & Miller, E. (1999). Managing databases with binary large objects. Dans *16th IEEE Symposium on Mass Storage Systems in cooperation with the 7th NASA Goddard Conference on Mass Storage Systems and Technologies (Category Number 99CB37098)*, pp. 185–193. IEEE.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D : Nonlinear Phenomena*, *404*, 132306.

Shoaib, M., Bosch, S., Incel, O. D., Scholten, H. & Havinga, P. J. (2016). Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, *16*(4), 426.

Shou, Z. & Di, X. (2018). Similarity analysis of frequent sequential activity pattern mining. *Transportation Research Part C : Emerging Technologies*, *96*, 122–143.

Shumway, R. H. & Stoffer, D. S. (2017). Characteristics of time series. Dans *Time Series Analysis and Its Applications : With R Examples* (pp. 1–44). Springer International Publishing.

Siirtola, P. & Röning, J. (2013). Ready-to-use activity recognition for smartphones. Dans *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 59–64. IEEE.

Singh, D., Merdivan, E., Psychoula, I., Kropf, J., Hanke, S., Geist, M. & Holzinger, A. (2017). Human activity recognition using recurrent neural networks. Dans *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pp. 267–274. Springer.

Skocir, P., Krivic, P., Tomeljak, M., Kusek, M. & Jezic, G. (2016). Activity detection in smart home environment. *Procedia Computer Science*, *96*, 672–681.

Smirek, L., Zimmermann, G. & Ziegler, D. (2014). *Towards universally usable smart homes – How can MyUI, URC and OpenHAB contribute to an adaptive user interface platform ?*

Soda, S., Nakamura, M., Matsumoto, S., Izumi, S., Kawaguchi, H. & Yoshimoto, M. (2012). Implementing virtual agent as an interface for smart home voice control. Dans *2012 19th Asia-Pacific Software Engineering Conference*, Vol. 1, pp. 342–345.

Soliman, M., Abiodun, T., Hamouda, T., Zhou, J. & Lung, C.-H. (2013). Smart home : Integrating internet of things with web services and cloud computing. Dans *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, Vol. 2, pp. 317–320. IEEE.

Solis, C. & Wang, X. (2011). A study of the characteristics of behaviour driven development. Dans *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 383–387.

Song, Y.-Y. & Ying, L. (2015). Decision tree methods : Applications for classification and prediction. *Shanghai Archives of Psychiatry*, *27*(2), 130.

Sousa, W., Souto, E., Rodrigres, J., Sadarc, P., Jalali, R. & El-Khatib, K. (2017). A comparative analysis of the impact of features on human activity recognition with smartphone sensors. Dans *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, pp. 397–404.

Spector, W. D. & Fleishman, J. A. (1998). Combining activities of daily living with instrumental activities of daily living to measure functional disability. *The Journals of Gerontology : Series B*, *53B*(1), S46–S57.

Sperberg-McQueen, M., Bray, T. & Paoli, J. (1998). *XML 1.0 recommendation*. W3C. https ://www.w3.org/TR/1998/REC-xml-19980210.

Spock (2019). Repéré le 22 Mai 2019, à http://spockframework.org

Srivastava, N., Mansimov, E. & Salakhudinov, R. (2015). Unsupervised learning of video representations using LSTMs. Dans *International Conference on Machine Learning*, pp. 843–852.

Steinley, D. & Brusco, M. J. (2018). A note on the expected value of the Rand index. *British Journal of Mathematical and Statistical Psychology*, *71*(2), 287–299.

Stikic, M., Huynh, T., Van Laerhoven, K. & Schiele, B. (2008). ADL recognition based on the combination of RFID and accelerometer sensing. Dans *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*, pp. 258–263.

Stoean, C. & Stoean, R. (2014). *Support vector machines and evolutionary algorithms for classification*. Springer.

Stuart Russell, P. N. (2009). *Artificial intelligence : A mordern approach* (3). Pearson.

Stuart Russell, P. N. (2011). Problem-solving. Dans *Artificial intelligence : A modern approach* (3, pp. 34–63). Pearson.

Subudhi, S. & Panigrahi, S. (2019). Application of OPTICS and ensemble learning for database intrusion detection. *Journal of King Saud University-Computer and Information*

Sciences.

T. van Kasteren, T. & Krose, B. (2007). Bayesian activity recognition in residence for elders. *IET Conference Proceedings*, pp. 209–212(3).

Tang, S., Kalavally, V., Ng, K. Y. & Parkkinen, J. (2017). Development of a prototype smart home intelligent lighting control architecture using sensors onboard a mobile computing system. *Energy and Buildings*, *138*, 368 – 376.

Torrisi, M., Pollastri, G. & Le, Q. (2020). Deep learning methods in protein structure prediction. *Computational and Structural Biotechnology Journal*, *18*, 1301–1310.

Turing, A. (1950). Computing machinery and intelligence. *Mind*, *59*(236), 433–460.

Valerio, L., Passarella, A. & Conti, M. (2017). A communication efficient distributed learning framework for smart environments. *Pervasive and Mobile Computing*, *41*, 46–68.

Vapnik, V. (1982). *Estimation of dependences based on empirical data berlin*. Springer-Verlag.

Vapnik, V. & Chervonenkis, A. (1974). *Theory of pattern recognition*.

Vapnik, V. N. & Chervonenkis, A. Y. (1968). The uniform convergence of frequencies of the appearance of events to their probabilities. Dans *Doklady Akademii Nauk*, Vol. 181.4, pp. 781–783. Russian Academy of Sciences.

Verma, P. & Sood, S. (2018). Fog assisted-IoT enabled patient health monitoring in smart homes. *IEEE Internet of Things Journal*, *PP*, 1–1.

Vinh, N. X., Epps, J. & Bailey, J. (2010). Information theoretic measures for clusterings comparison : Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, *11*, 2837–2854.

Voulodimos, A., Doulamis, N., Doulamis, A. & Protopapadakis, E. (2018). Deep learning for computer vision : A brief review. *Computational Intelligence and Neuroscience*, *2018*.

Vreeken, J., Van Leeuwen, M. & Siebes, A. (2011). Krimp : Mining itemsets that compress. *Data Mining and Knowledge Discovery*, *23*(1), 169–214.

Vujović, V. & Maksimović, M. (2015). Raspberry Pi as a sensor web node for home automation. *Computers & Electrical Engineering*, *44*, 153 – 171.

Wagner, S. & Wagner, D. (2007). *Comparing clusterings : An overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe.

Wan, J., O'grady, M. J. & O'Hare, G. M. (2015). Dynamic sensor event segmentation for real-time activity recognition in a smart home context. *Personal and Ubiquitous Computing*, *19*(2), 287–301.

Wan, S., Qi, L., Xu, X., Tong, C. & Gu, Z. (2020). Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications*, *25*(2), 743–755.

Wang, J., Chen, Y., Hao, S., Peng, X. & Hu, L. (2019). Deep learning for sensor-based activity recognition : A survey. *Pattern Recognition Letters*, *119*, 3 – 11. Deep Learning for Pattern Recognition.

Wang, L. (2016). Recognition of human activities using continuous autoencoders with wearable sensors. *Sensors*, *16*(2).

Wang, L. & Liu, R. (2020). Human activity recognition based on wearable sensor using hierarchical deep LSTM networks. *Circuits, Systems, and Signal Processing*, *39*(2), 837–856.

Wang, Y., Jiang, H., Drew, M. S., Li, Z.-N. & Mori, G. (2006). Unsupervised discovery of action classes. Dans *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2, pp. 1654–1661. IEEE.

Weilkiens, T., Lamm, J. G., Roth, S. & Walker, M. (2015). The layered perspective. Dans *Model-based system architecture* (pp. 100–102). John Wiley & Sons, Inc.

Weiser, M. (1999). The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Review*, *3*(3), 3–11.

Weiyao Lin, Ming-Ting Sun, Poovandran, R. & Zhengyou Zhang (2008). Human activity recognition for video surveillance. Dans *2008 IEEE International Symposium on Circuits and Systems*, pp. 2737–2740.

Wen, J., Zhong, M. & Wang, Z. (2015). Activity recognition with weighted frequent patterns mining in smart environments. *Expert Systems with Applications*, *42*(17-18), 6423–6432.

Wennie Huang, W.-N., Perera, S., VanSwearingen, J. & Studenski, S. (2010). Performance measures predict onset of activity of daily living difficulty in community-dwelling older adults. *Journal of the American Geriatrics Society*, *58*(5), 844–852.

Wielenga, G. (2015). Testing and code quality. Dans *Beginning NetBeans IDE : for Java developers* (pp. 165–180). Apress.

Wu, K.-L. & Yang, M.-S. (2007). Mean shift-based clustering. *Pattern Recognition*, *40*(11), 3035–3052. Repéré le 26 Juillet 2022, à https://www.sciencedirect.com/science/article/pii/S0031320307000921

Xia, K., Huang, J. & Wang, H. (2020). LSTM-CNN architecture for human activity recognition. *IEEE Access*, *8*, 56855–56866.

Xia, X., Togneri, R., Sohel, F. & Huang, D. (2018). Random forest classification based acoustic event detection utilizing contextual-information and bottleneck features. *Pattern Recognition*, *81*, 1 – 13.

Xu, D. & Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, *2*(2), 165–193.

Yadav, S. S. & Jadhav, S. M. (2019). Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, *6*(1), 1–18.

Yao, L., Sheng, Q. Z., Benatallah, B., Dustdar, S., Wang, X., Shemshadi, A. & Kanhere, S. S. (2018). WITS : An IoT-endowed computational framework for activity recognition in personalized smart homes. *Computing*, *100*(4), 369–385.

Ye, J., Stevenson, G. & Dobson, S. (2015). KCAR : A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing*, *19*, 47–70.

Yin, J., Yang, Q. & Pan, J. J. (2008). Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering*, *20*(8), 1082–1090.

Youngblood, G., Cook, D. & Holder, L. (2004). *The MavHome architecture* publication n° CSE-2004-18. Department of Computer Science and Engineering of University of Texas at Arlington.

Yuan, C. & Yang, H. (2019). Research on K-value selection method of K-means clustering algorithm. *Multidisciplinary Scientific Journal*, *2*(2), 226–235.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353. Repéré le 26 Juillet 2022, à https://www.sciencedirect.com/science/article/pii/S001999586590241X

Zappi, P., Stiefmeier, T., Farella, E., Roggen, D., Benini, L. & Troster, G. (2007). Activity recognition from on-body sensors by classifier fusion : Sensor scalability and robustness. Dans *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, pp. 281–286. IEEE.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X. & Metaxas, D. N. (2018). Stackgan++ : Realistic image synthesis with stacked generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(8), 1947–1962.

Zhang, Z., Beck, M. W., Winkler, D. A., Huang, B., Sibanda, W. & Goyal, H. (2018). Opening the black box of neural networks : Methods for interpreting neural network models in clinical applications. *Annals of Translational Medicine*, *6*(11).

Zhang, Z., Geiger, J., Pohjalainen, J., Mousa, A. E.-D., Jin, W. & Schuller, B. (2018). Deep learning for environmentally robust speech recognition : An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *9*(5),

1–28.

Zheng, Y., Kell, S., Bulej, L., Sun, H. & Binder, W. (2016). Comprehensive multiplatform dynamic program analysis for Java and Android. *IEEE Software*, *33*(4), 55–63.

Zhiwen Yu, Xingshe Zhou, Zhiyong Yu, Daqing Zhang & Chung-Yau Chin (2006). An OSGi-based infrastructure for context-aware multimedia services. *IEEE Communications Magazine*, *44*(10), 136–142.

Zhu, B. & Liu, Y. (2021). General approximate cross validation for model selection : Supervised, semi-supervised and pairwise learning. Dans *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 5281–5289.

Zia, T., Bruckner, D. & Zaidi, A. (2011). A hidden Markov model based procedure for identifying household electric loads. Dans *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*, pp. 3218–3223. IEEE.