UNIONE EUROPEA
FONDO EUROPEO DI SVILUPPO REGIONALE

Ministero Sviluppo Economico

REGIONE PUGLIA
AREA POLITICHE PER LO SVILUPPO ECONOMICO
IL LAVORO E L'INNOVAZIONE

Regolamento regionale della Puglia per gli aiuti in esenzione n. 17 del 30/09/2014

Titolo II – Capo 1 "Aiuti ai programmi di investimento delle Grandi imprese"

POR PUGLIA – Fondo Europeo di Sviluppo Regionale 2014 – 2020

CONTRATTO DI PROGRAMMA

"Digital Future"

CODICE PROGETTO: CP12PA6

OR 2 – Piattaforma Digital Future: Metodi e Tecniche

Attività 2.1 – Metodi e tecniche per il Middleware

Deliverable 2.1.B – Progettazione di soluzioni Edge Intelligence per la piattaforma Digital Future

Diritti di autore e riservatezza

Questo documento è proprietà esclusiva della società Exprivia S.p.A. e non può essere riprodotto, anche in forma parziale, senza un'autorizzazione scritta della società stessa.

Indice dei contenuti

INDICE DEI CONTENUTI **3**

INDICE DELLE FIGURE **5**

INDICE DELLE TABELLE **6**

1. INTRODUZIONE **7**

2. ARCHITETTURA DI RIFERIMENTO **9**

2.1. ARCHITETTURE A MICROSERVIZI 10

2.2. OSMOTIC COMPUTING 15

2.3. COMPONENTI INTEGRATI 21

2.3.1. Docker *24*

2.3.2. Balena *25*

2.3.3. TensorFlow *26*

2.3.4. Keras *26*

2.3.5. Apache Kafka *27*

2.3.6. Redis *28*

2.3.7. SQLite *29*

2.4. BENEFICI ATTESI E PROBLEMI APERTI 30

2.4.1. Deployment di microservizi *30*

2.4.2. Sicurezza *31*

2.4.3. Edge computing e Intelligent Edge *32*

2.4.4. Interferenze e contese di risorse tra microservizi *32*

2.4.5. Orchestrazione ed elasticità *33*

2.5. BIBLIOGRAFIA 33

3. FRAMEWORK PER EDGE INTELLIGENCE **35**

3.1. IDENTIFICAZIONE DI EVENTI BASATA SU RAPPRESENTAZIONE DELLA CONOSCENZA 36

3.2. BENEFICI ATTESI E PROBLEMI APERTI 42

3.3. BIBLIOGRAFIA 47

4. PROGETTAZIONE DI UN FRAMEWORK DI CONTEXT AWARENESS A LIVELLO EDGE **48**

4.1. MODELLO DI CONTEXT AWARENESS 49

4.2. INTEGRAZIONE DELLA CONOSCENZA A LIVELLO EDGE 51

4.3. BENEFICI ATTESI E PROBLEMI APERTI 57

4.4. BIBLIOGRAFIA 59

5. PROGETTAZIONE DI UN FRAMEWORK PER LA VALUTAZIONE DI UN GRAFO ARGOMENTATIVO BASATO SU RAGIONAMENTO AUTOMATICO 61

5.1. VALUTAZIONE DEL PESO DI SUPPORTO 63

5.2. VALUTAZIONE DEL PESO DI ATTACCO 64

5.3. PROCESSO DI ARGOMENTAZIONE 66

5.4. BENEFICI ATTESI E PROBLEMI APERTI 68

5.5. BIBLIOGRAFIA 70

6. CONCLUSIONI 72

Indice delle figure

Figura 2.1 - Architettura di riferimento della piattaforma Digital Future 9
Figura 2.2 – Modelli architetturali applicabili ad una piattaforma software 10
Figura 2.3 – Componenti di una tipica piattaforma a microservizi 13
Figura 2.4 – Edge e cloud computing per l'IoT [2.2] 16
Figura 2.5 - Idea di base dell'osmotic computing [2.2] 18
Figura 2.6 – Osmotic computing in datacenter cloud ed edge [2.2] 19
Figura 2.7 – Componenti integrati nell'architettura proposta 22
Figura 3.1 – Architettura di riferimento per l'identificazione di eventi 37
Figura 3.2 – Esempio di modellazione di una ontologia 37
Figura 3.3 – Esempio di modellazione attraverso OWL Annotation Property 38
Figura 3.4 – Algoritmo di costruzione della matrice delle occorrenze 39
Figura 3.5 – Esempio di annotazione di un evento 44
Figura 3.6 – Esempio di classificazione realizzata con alberi decisionali 45
Figura 3.7 – Esempio di classificazione realizzata con reti neurali 46
Figura 4.1 – Costruzione di un pacchetto di Relay 51
Figura 4.2 – Flowchart relativo al Concept Integration Problem 53
Figura 4.3 – Diagramma operativo per la generazione del campo *M'* 53
Figura 4.4 – Diagramma operativo per la generazione del campo *C'* 54
Figura 4.5 – Diagramma operativo per la generazione del campo *X'* 55
Figura 4.6 – Diagramma operativo per la generazione del campo *E'* 56
Figura 5.1 – Algoritmo di definizione del tipo di relazione tra coppie di argomenti 62
Figura 5.2 – Esempio di relazione di supporto 63
Figura 5.3 – Esempio di relazione di attacco 64

Indice delle tabelle

Tabella 2.1 - Componenti, responsabilità e strumenti nell'architettura Digital Future 23
Tabella 3.1 - Esempio di matrice di confusione..... 42
Tabella 4.1 – Descrizione dei campi di un pacchetto semantico..... 49
Tabella 4.2 – Comparazione tra alcuni approcci cognitivi in letteratura e soluzione di CIP proposta..... 58

1. Introduzione

La crescente disponibilità di dispositivi elettronici portatili ed interconnessi ha contribuito a rendere sempre più concrete le grandi potenzialità delle architetture distribuite e dei sistemi pervasivi. Un paradigma di sempre maggior rilevanza è l'**edge computing**, che caratterizza tutte le fasi di gestione dell'informazione nei sistemi pervasivi, dalla raccolta ed immagazzinamento dei dati fino alla loro elaborazione e trasmissione in cloud. La disponibilità sempre crescente di questi dispositivi low-cost per l'Internet of Things (IoT), aventi capacità computazionali non trascurabili e dimensioni ridotte, consente ormai di interagire simultaneamente con numerosi apparecchi, anche diversi tra loro, ed estrarre dati da oggetti smart che popolano l'ambiente durante le normali attività quotidiane al fine di individuare possibili eventi o interazione in atto. I dati, raccolti tramite sensori eterogenei, generano un flusso in real-time da cui è possibile ottenere informazioni di interesse da usare per il supporto alle decisioni o per procedure di controllo automatizzato.

Per realizzare una visione di questo tipo è indispensabile adottare tecniche per l'apprendimento automatico e algoritmi di machine learning, una branca dell'Intelligenza Artificiale che si occupa dell'implementazione di sistemi che permettono ai calcolatori di sviluppare dei modelli decisionali e riconoscere in maniera automatica eventi anche complessi. In questo modo, anche un dataset di grandi dimensioni può essere trasformato in conoscenza fruibile per diversi scopi. Per migliorare ulteriormente tali processi, è chiara l'opportunità di descrivere la grande quantità di dati grezzi a disposizione in maniera strutturata e machine-understandable, ovvero interpretabile da un elaboratore. Tale visione può essere raggiunta annotando gli oggetti del mondo reale, i dati raccolti e gli ambienti in cui sono immersi, con descrizioni concise, strutturate e semanticamente ricche. La combinazione dell'IoT con i modelli e le tecnologie del Web semantico ha portato in questi anni alla definizione del ***Semantic Web of Things*** (SWoT), un nuovo paradigma che mira all'introduzione di nuove classi di applicazioni e servizi intelligenti basati sulla rappresentazione della conoscenza (KR), sfruttando le interferenze automatiche basate su semantica per ricavare informazioni implicite da un evento esplicito e dal rilevamento del contesto. Mediante metadati espressi in linguaggi logici formali, l'informazione è resa ricca di significato e adatta per poter un'elaborazione ed interpretazione automatica. Allo stesso tempo, associando una descrizione strutturata e comprensibile alla macchina, definita a partire da linguaggi standard originariamente pensati per il Semantic Web, ogni output di classificazione potrebbe avere un significato inequivocabile. Inoltre, le capacità di interpretazione dei dati su base semantica consentono una crescente fiducia degli utilizzatori rispetto ai risultati del sistema. Se i micro-dispositivi pervasivi sono in grado di elaborare in modo efficiente

i dati recuperati localmente, possono descrivere sé stessi e il contesto in cui si trovano inviando successivamente queste informazioni verso dispositivi e applicazioni esterni. Ciò migliora l'interoperabilità tra elementi della rete e incrementa l'autonomia dei sistemi pervasivi basati sulla conoscenza, non ancora completamente integrata nelle tipiche infrastrutture IoT.

Nell'edge computing, tali tecnologie, sia legate al *machine learning* che al Semantic Web, devono però essere necessariamente riadattate in modo da tener conto del contesto pervasivo in cui andranno ad operare. All'interno del lavoro proposto è stato realizzato lo studio e lo sviluppo di un framework per *Edge Intelligence* in grado di supportare algoritmi di machine learning su base semantica, specifici per contesti di elaborazione pervasiva, in grado di analizzare un flusso di dati grezzi da cui estrarre una descrizione strutturata di osservazioni di interesse. Tali descrizioni saranno poi utilizzate per realizzare una fase di classificazione di eventi complessi e per supportare un approccio innovativo che combini servizi di ragionamento automatico con tecniche di ***argumentation***. L'obiettivo primario è superare i limiti degli approcci esistenti e poter realizzare un framework che possa essere applicato anche in un contesto caratterizzato da limitazioni in termini di risorse computazionali e fisiche. L'architettura proposta rappresenta il punto di partenza per le successive fasi di progetto in cui la soluzione realizzata sarà implementata e testata in differenti scenari applicativi.

Di seguito sarà presentata una panoramica generale relativa all'architettura di riferimento ed ai principali paradigmi utilizzati. Saranno inoltre presentati gli algoritmi adottati per l'analisi dei dati, dettagliando tutte le fasi dalla raccolta delle informazioni fino all'identificazione di un evento.

2. Architettura di riferimento

L'architettura generale disegnata per la piattaforma Digital Future è mostrata in Figura 2.1. Essa è stata concepita come un'*architettura a microservizi*, e in particolare segue i principi e dell'*osmotic computing*, paradigma innovativo per applicazioni edge/cloud ibride con elevata scalabilità e dinamicità.

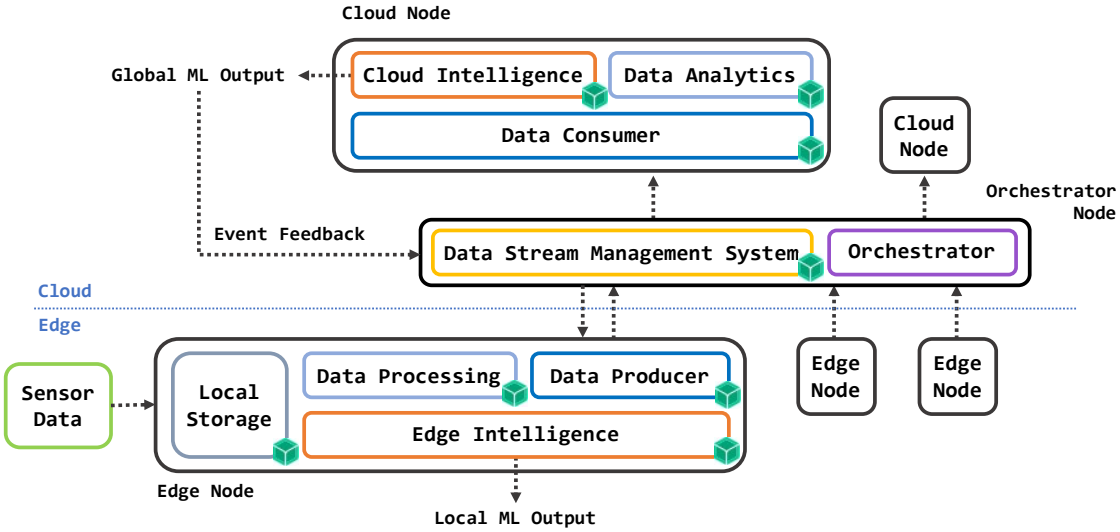

Figura 2.1 - Architettura di riferimento della piattaforma Digital Future

I componenti fondamentali del framework di riferimento sono:

- uno o più **nodi edge** in grado di acquisire dati grezzi da differenti sensori (presenti a bordo o collegati direttamente) ed elaborarli localmente utilizzando reti neurali per operazioni di classificazione/regressione;
- uno o più **nodi cloud** in grado di ricevere dati aggregati dai nodi edge ed effettuare operazioni di classificazione/regressione operando su un insieme più ampio e articolato di dati;

- un **sistema di stream management** in grado di veicolare i dati provenienti dall'edge della rete verso le componenti cloud, che può inoltre fornire supporto per le operazioni di storage dei dati;
- un **orchestratore di container**, in grado di gestire i diversi container *Docker*[1] presenti nella rete in cui saranno implementate le funzionalità richieste come microservizi.

Nei paragrafi successivi si approfondiscono i principi teorici e tecnologici delle architetture a microservizi e dell'Osmotic Computing che giustificano la definizione dell'architettura. Si fornisce inoltre una proposta concreta di integrazione di componenti COTS (commercial off the shelf) per la piattaforma Digital Future, descrivendone le caratteristiche che hanno indotto a selezionarli.

2.1. Architetture a microservizi

In generale, nello sviluppo di una soluzione software tre sono i differenti approcci architetturali applicabili; come mostrato in Figura 2.2.

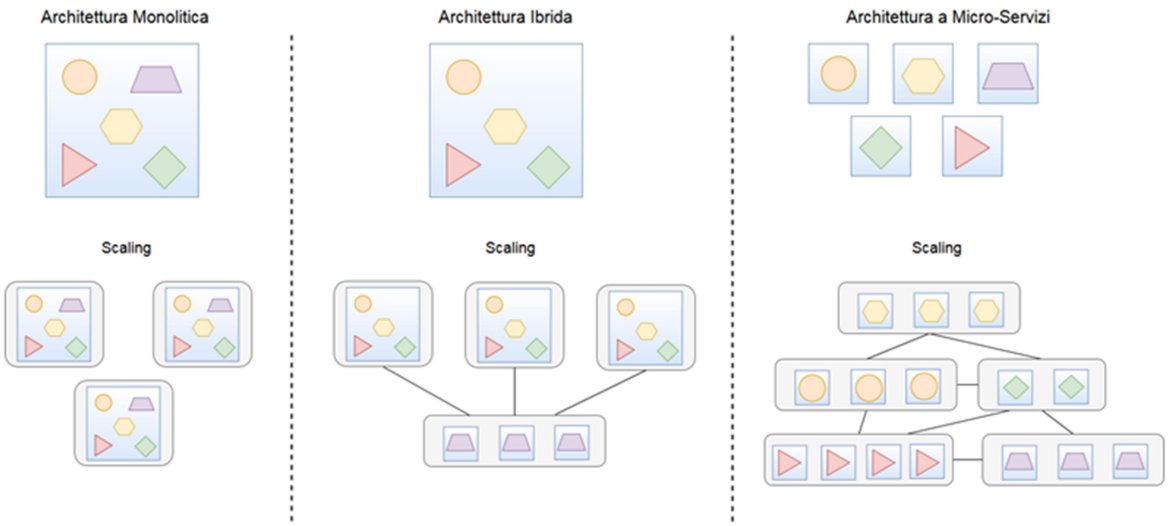

Figura 2.2 – Modelli architetturali applicabili ad una piattaforma software

[1] Docker: https://www.docker.com/

Una soluzione di tipo monolitico è ovviamente più semplice da implementare ma difficile da scalare. Un'architettura di tipo ibrido offre un compromesso tra le caratteristiche di flessibilità/scalabilità e complessità nello sviluppo. Il vantaggio di una architettura a microservizi è proprio quello che ciascun servizio può essere sviluppato, testato e schierato su cloud in maniera indipendente dagli altri; garantendo una maggiore capacità di scalare ma anche una maggiore manutenibilità in caso di sistemi complessi e di grandi dimensioni.

In particolare, la soluzione proposta segue un approccio avanzato in cui i microservizi sono opportunisticamente distribuiti in componenti virtuali, chiamati **container** (contenitori). Le tecnologie di virtualizzazione basate su container (ad esempio, *Linux Containers*[2], *Docker*, *Preboot Execution Environment* [2.3], *Google Container*[3] e *Amazon Web Services Fargate*[4]) sono emerse come alternativa leggera agli approcci basati su hypervisor (come *Xen* e *Microsoft Hyper-V*). Un container consente di incapsulare solo componenti software ben definiti (come un database server), il che porta a una significativa riduzione dell'overhead di distribuzione e una densità di istanze su singolo dispositivo molto più elevata rispetto a un hypervisor. Pertanto, i nuovi approcci basati su container consentono l'implementazione di microservizi leggeri su dispositivi intelligenti resource-constrained e programmabili ai margini della rete come gateway (ad esempio *Raspberry Pi*), switch di rete (*HP OpenFlow*) e router (come *Cisco IOx*), ma aumentano anche le prestazioni nella gestione dinamica dei microservizi nei datacenter cloud.

I principali benefici di un'architettura a microservizi containerizzati possono essere così riassunti:

- il singolo microservizio è eseguito in un processo (container) distinto che può comunicare con gli altri processi (container) mediante protocolli di trasporto standard (HTTP/HTTPS, WebSocket, AMQP);
- ciascun microservizio implementa una specifica funzionalità di business e può essere sviluppato in modo autonomo ed indipendente, utilizzando qualsiasi linguaggio di programmazione;
- un microservizio può definire in modo completamente decentralizzato il data model privato e la logica di business e può integrare tecnologie differenti per lo storage;

[2] Linux Containers: https://linuxcontainers.org/

[3] Google Cloud Containers: https://cloud.google.com/containers

[4] Amazon AWS Fargate: https://aws.amazon.com/it/fargate/

- le applicazioni possono essere composte in modo flessibile e granulare, combinando molteplici servizi installati singolarmente;
- il load balancer può applicare una politica granulare alla scalabilità, incrementando le istanze dei microservizi per le quali è richiesta maggiore capacità computazionale o banda di rete.

Ovviamente, una soluzione di questo tipo può introdurre alcune complessità:

- occorre gestire le connessioni necessarie alla comunicazione tra container garantendo il rispetto delle policy di sicurezza;
- un malfunzionamento parziale, che riguarda uno o più container, deve essere opportunamente gestito al fine di non compromettere la consistenza dei dati;
- occorre gestire l'overhead della rete per coordinare lo scambio di messaggi tra i vari container;
- l'eventuale refactoring di componenti legacy esistenti potrebbe essere complicato.

Lo sviluppo di un'architettura a microservizi prevede la decomposizione di ogni singola funzionalità esposta dal sistema di partenza in uno o più microservizi. Come mostrato in Figura 2.3, l'accesso a tali servizi è tipicamente gestito mediante un opportuno insieme di API Gateway e attraverso l'adozione di alcuni paradigmi standard tipici di tali architetture. L'obiettivo di tale componente è quello di fornire un *single entry point* che svolga i seguenti task:

- inoltrare le richieste al microservizio appropriato attraverso una funzione di proxy/router;
- implementare policy di sicurezza (Identify Management);
- comporre i servizi offerti da uno o più microservizi per generare l'output a seguito di una richiesta complessa.

Molto spesso in architetture di questo tipo il numero dei servizi coinvolti risulta essere elevato. Per tale motivo, un altro aspetto da considerare è l'utilizzo di un componente che svolga il ruolo di Service Registry, al fine di memorizzare le informazioni sulle istanze dei vari microservizi attivi; specificandone i rispettivi endpoint e le policy di accesso. Le istanze sono registrate all'avvio di ciascun microservizio e sono rimosse quando il servizio è disattivato. I diversi "consumer" (o gli API Gateway) possono ricercare i microservizi disponibili e i rispettivi endpoint interrogando il Service Registry attraverso un opportuno meccanismo di Service Discovery. Due sono i possibili approcci per il service discovery:

- Client-side Discovery. Il client o l'API Gateway ottiene l'endpoint di una istanza di servizio interrogando direttamente il Service Registry.

- Server-side Discovery. In questo caso il client o l'API Gateway contatta un componente dedicato, in ascolto su un endpoint pubblico, che si occupa di interrogare il Service Registry al fine di recuperare l'endpoint di un microservizio.

All'interno di un'architettura a microservizi, un ulteriore pattern architetturale che spesso è necessario considerare è quello che prevede l'utilizzo di un componente denominato Integration Server; questo in caso risulti essere necessario gestire processi di business che prevedono l'interazione con sistemi legacy.

Un altro aspetto riguarda l'integrazione tra i vari servizi, che può avvenire attraverso l'utilizzo di un componente denominato Message Broker; basato sullo scambio di messaggi asincroni. Tale strumento ha come obiettivo quello di disaccoppiare la logica di inoltro dei messaggi dai singoli servizi, delegando al componente l'inoltro delle informazioni ai vari microservizi. Oltre all'invio dei dati, il broker può gestire aspetti legati alla sicurezza, alla priorità e all'ordinamento delle code di inoltro.

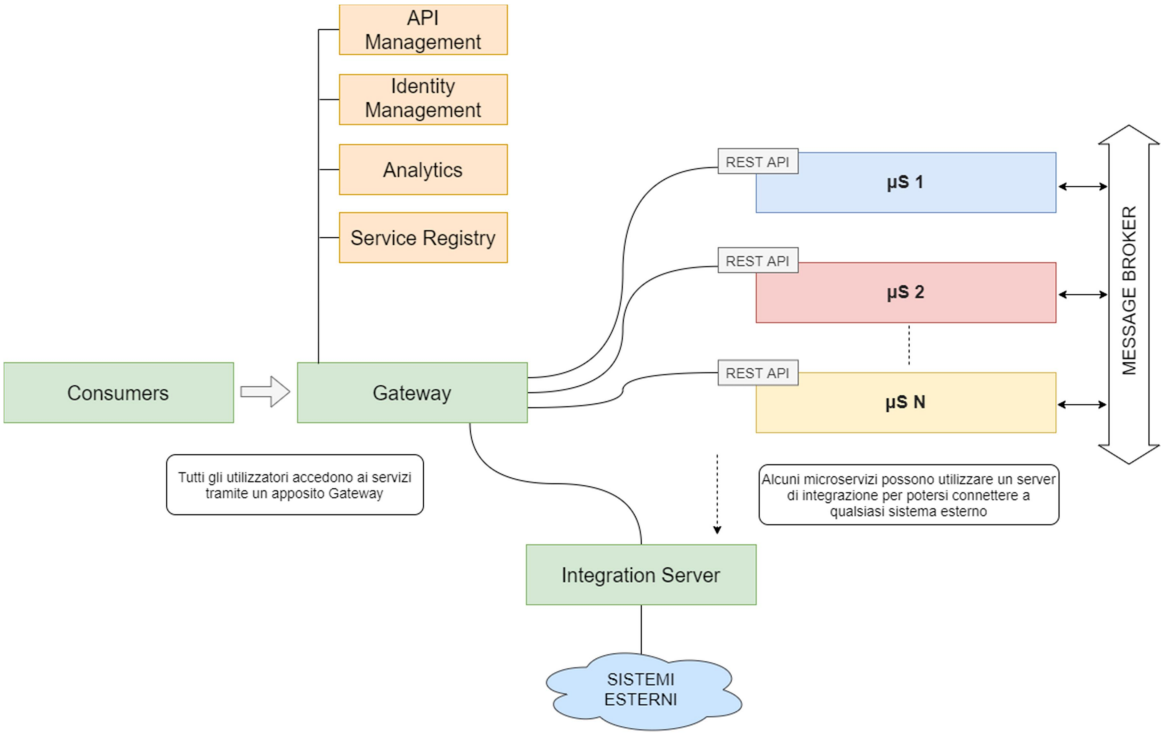

Figura 2.3 – Componenti di una tipica piattaforma a microservizi

Sono previsti i seguenti paradigmi di interazione:

- Publish/subscribe: i messaggi pubblicati sul canale sono inviati ai microservizi in base a sottoscrizioni verso specifici topic. Sono supportati diversi scenari di inoltro (ad esempio, uno a molti o molti a molti).

- Queuing: il messaggio di richiesta è inoltrato su una coda specifica, la risposta è inviata sulla medesima coda.
- Store/forward: il broker memorizza temporaneamente i messaggi, che sono inoltrati ai destinatari interessati.

Una delle difficoltà maggiori nell'utilizzare una gestione distribuita delle informazioni è quella di definire in maniera ben precisa i confini di ciascun microservizio che costituisce l'architettura di sistema. Ogni microservizio deve rappresentare un frammento dell'applicazione complessiva ed essere completamente autonomo, con tutti i vantaggi e gli svantaggi che ne derivano. Esistono differenti modelli di gestione dei dati che possono essere adoperati per ottenere una migliore efficienza:

- Database Per Service. Ciascun microservizio gestisce i propri dati, nessun altro microservizio può accedere direttamente ad essi. La comunicazione o lo scambio di informazioni può avvenire solo utilizzando un insieme di API ben definite. Tuttavia, molto spesso i microservizi necessitano di interagire tra loro ed hanno bisogno di aggregare i rispettivi dati per poter implementare la propria logica di business. Per un'applicazione monolitica molto grande risulta complesso separare ed isolare tra loro i dati necessari al funzionamento di ciascun microservizio. Un'altra difficoltà è quella di implementare transazioni di business che prevedono l'interazione e la composizione di più microservizi. Tuttavia, se implementato correttamente, il principale vantaggio di tale approccio è quello di avere microservizi debolmente accoppiati tra loro. In questo modo è possibile:
    - sviluppare, installare e scalare microservizi in maniera indipendente;
    - ciascun servizio può utilizzare la tipologia di database che meglio si adatta alle proprie esigenze;
- Shared Database. Rappresenta una soluzione praticabile nel caso in cui l'operazione di disaccoppiamento tra i vari microservizi risulta essere troppo complessa da gestire. Tale pattern prevede un approccio meno restrittivo e si basa sull'utilizzo di un database condiviso a cui possono accedere più microservizi (sono garantite le proprietà ACID dei sistemi transazionali per mantenere consistenza delle informazioni). Tuttavia, tale approccio rimuove gran parte dei benefici di un'architettura a microservizi: gli sviluppatori di ciascun microservizio necessitano di coordinarsi in caso di modifiche allo schema del database condiviso.

Una soluzione decentralizzata produce microservizi completamente disaccoppiati e offre la libertà di scegliere tecniche di gestione dei dati diverse per ciascun servizio (SQL, NoSQL, ecc.) Ovviamente, per transazioni complesse che prevedono l'interazione tra più microservizi, le proprietà transazionali devono essere implementate utilizzando le API offerte da ciascun servizio e la logica di business può trovarsi o a

livello client o a livello intermedio (Gateway). In questi casi è necessario adoperare alcuni pattern funzionali; tra questi ci sono:

- Saga Pattern. È la soluzione da adoperare per implementare transazioni logiche che coinvolgono più microservizi. Una 'saga' rappresenta fondamentalmente una sequenza di transazioni locali. Per ciascuna transazione eseguita all'interno di una saga, il servizio che la esegue pubblica un evento (avvalendosi dei servizi di un message broker). La pubblicazione di tale evento comporta l'attivazione della transazione successiva. Se una delle transazioni nella catena fallisce, sono eseguite una serie di transazioni di riparazione per annullare i risultati di tutte le operazioni precedenti.
- API Composition. Questo modello rappresenta la soluzione diretta al problema dell'implementazione di query complesse in un'architettura a microservizi. In tale approccio, un componente noto come API Composer: (i) invoca i vari microservizi nell'ordine richiesto per recuperare le informazioni necessarie ad eseguire una transazione complessa, (ii) esegue un in-memory join dei dati raccolti, (iii) trasmette il risultato finale al consumer che ne ha fatto richiesta. Come evidente, lo svantaggio di tale approccio è l'utilizzo di operazioni di integrazione dei dati in memoria, che possono risultare inefficienti in caso di dataset di grandi dimensioni.
- CQRS (Command Query Responsability Segregation). Si tratta di un'altra soluzione per aggregare dati provenienti da microservizi differenti che sfrutta il cosiddetto Materialized View Pattern. In tale approccio si genera una vista (o view, una tabella a sola lettura) che contiene dati generati da più microservizi a seguito di una query complessa (la tabella ha una struttura conforme alle esigenze dell'applicazione client). Questa soluzione non solo risolve il problema di partenza (come eseguire query complesse e join SQL che coinvolgono informazioni gestite autonomamente da più microservizi), ma migliora anche le prestazioni in maniera considerevole; poiché le informazioni necessarie all'applicazione sono già presenti nella view costruita a priori. Questo ovviamente comporta uno sforzo di sviluppo ulteriore per la generazione delle view necessarie e la gestione della consistenza dei dati.

2.2. Osmotic computing

La sempre crescente disponibilità di dispositivi IoT di complessità e tipo variabile, insieme ai grandi volumi di dati che tali dispositivi (potenzialmente) possono generare, possono avere un impatto significativo sulla nostra vita, alimentando lo sviluppo di servizi e applicazioni critici di nuova generazione in un varietà di domini applicativi (assistenza sanitaria, finanza, gestione delle catastrofi e così via). Comprendere come i

dati provenienti da tali dispositivi possano essere analizzati in modo più efficiente rimane una sfida, per via della dipendenza da sistemi di cloud computing su larga scala che, nel tempo, diventano colli di bottiglia. Il trasferimento di grandi flussi di dati in tali ambienti di data center cloud centralizzati, in modo tempestivo e affidabile, è una limitazione fondamentale dei modelli di programmazione IoT incentrati sul cloud (come Amazon IoT e il dataflow cloud di Google). Questi modelli di programmazione IoT sono considerati inadeguati nel contesto delle applicazioni IoT emergenti per il motivo principale che presumono che l'intelligenza e la capacità delle risorse necessarie per l'elaborazione dei dati risiedano prevalentemente nel data center cloud.

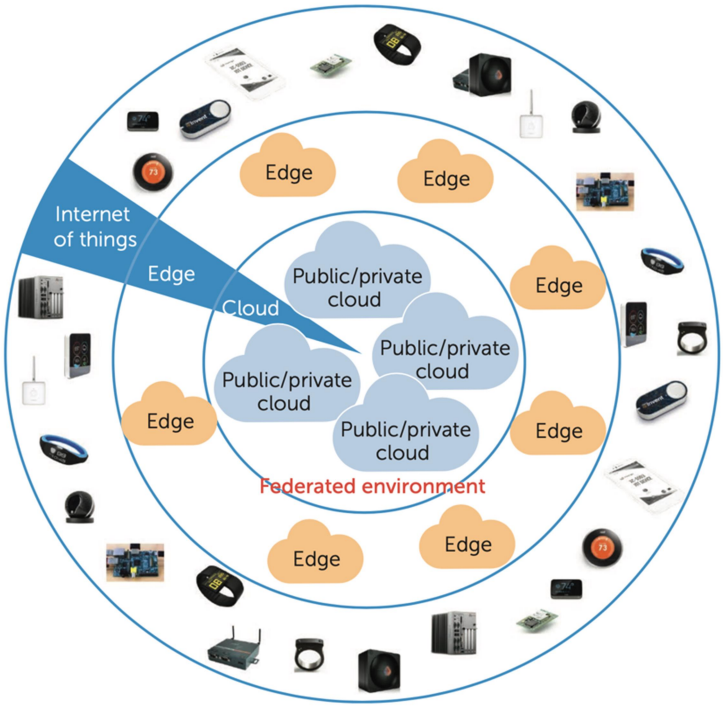

Figura 2.4 – Edge e cloud computing per l'IoT [2.2]

Pertanto, per implementare complessi sistemi di elaborazione orientati all'IoT, le risorse cloud ed edge dovrebbero essere entrambe sfruttate nell'impostazione di un'infrastruttura virtuale ibrida, come mostra la **Errore. L'origine riferimento non è stata trovata.**. I data center cloud e edge dovrebbero essere gestiti in un ambiente federato, in cui diversi provider condividono le loro risorse a favore dei servizi IoT e del supporto alle applicazioni. L'onere del caricamento dei dati verso i datacenter porta a un uso inefficiente della larghezza di banda della rete e ad un consumo elevato di energia, e un recente studio di Cisco [2.1] mostra che il traffico totale dei datacenter triplicherà entro il 2019, peggiorando ulteriormente la situazione. Gli approcci di *store-and-process-later*, che possono ridurre l'utilizzo di risorse di rete, minano il

processo decisionale in tempo reale, che è spesso un requisito necessario per le applicazioni IoT in settori come la gestione delle catastrofi e l'assistenza sanitaria. Al contrario, l'edge computing mira a spostare la computazione sui dispositivi edge con risorse limitate, come mostra la **Errore. L'origine riferimento non è stata trovata.**. Le applicazioni Edge sono estremamente time-sensitive (ad esempio, le applicazioni di avvertimento di pericoli per condizioni ambientali come tempeste, frane e inondazioni) perché eseguono istantaneamente l'analisi o forniscono in tempo reale la risposta ai dati sensoriali raccolti.

Tuttavia, anche se i modelli di programmazione basati su cloud non sono in grado di supportare il livello desiderato di reattività per le applicazioni IoT, possono aumentare notevolmente la disponibilità di calcolo e archiviazione ogni volta che è necessario. Di conseguenza, il più diffuso modello di programmazione IoT incentrato sul cloud deve essere rivisto al fine di essere più adattabile e decentralizzato per soddisfare le esigenze delle applicazioni IoT emergenti.

L'**osmotic computing** [2.2] mira a scomporre le applicazioni in microservizi ed eseguire la personalizzazione dinamica dei microservizi in ambienti intelligenti sfruttando le risorse delle infrastrutture edge e cloud. Il delivery delle applicazioni segue un comportamento "osmotico", in cui i container dei microservizi sono distribuiti *opportunisticamente* nei sistemi cloud e edge. Come l'osmosi nel contesto della chimica – cioè il movimento delle molecole di solvente attraverso una membrana semipermeabile in una regione con una concentrazione di soluto più elevata per equalizzare le concentrazioni di soluto sui due lati della membrana – così nell'osmotic computing la gestione dinamica delle risorse nei datacenter cloud e edge si evolve verso il deployment equilibrato di microservizi che soddisfa ben definiti vincoli di basso livello ed esigenze di alto livello, come mostra la **Errore. L'origine riferimento non è stata trovata.**. Tuttavia, diversamente dal processo chimico, l'osmotic computing consente la configurazione del coinvolgimento delle risorse, seguendo la loro disponibilità ed i requisiti dell'applicazione (si veda la **Errore. L'origine riferimento non è stata trovata.**). Questa è una distinzione importante: la differenza di configurazione (molto dipendente dall'infrastruttura e dall'applicazione) può determinare se i microservizi devono migrare da cloud a edge o viceversa.

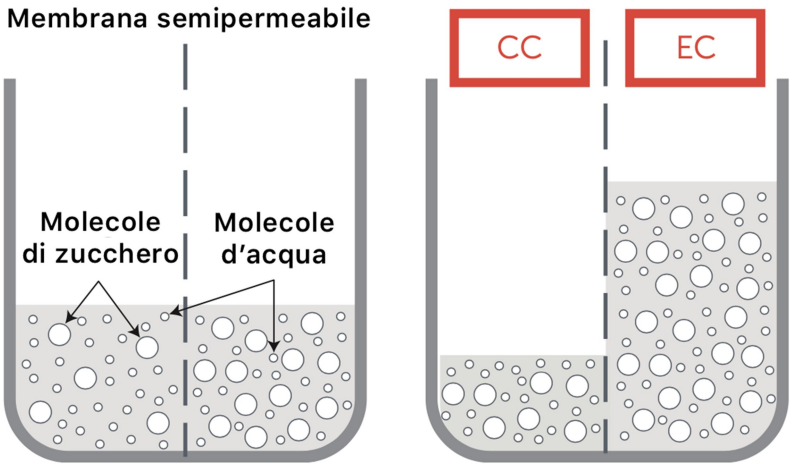

Figura 2.5 - Idea di base dell'osmotic computing [2.2]

L'osmotic computing va oltre la semplice gestione elastica delle risorse distribuite, poiché le strategie di distribuzione sono correlate ai requisiti sia dell'infrastruttura (come load balancing, affidabilità e disponibilità) sia delle applicazioni (come capacità di sensing e attivazione, consapevolezza del contesto, prossimità e QoS) e possono anche cambiare nel tempo. A causa dell'elevate eterogeneità delle risorse fisiche, l'attività di distribuzione dei microservizi deve adattare l'ambiente virtuale alle apparecchiature hardware interessate. Pertanto, deve essere gestito un flusso bidirezionale di microservizi adattati da cloud a edge (e viceversa). Inoltre, la migrazione dei microservizi nel sistema edge/cloud implica la necessità di una gestione dinamica ed efficiente dei problemi della rete virtuale per evitare il fallimento delle applicazioni o il degrado del QoS.

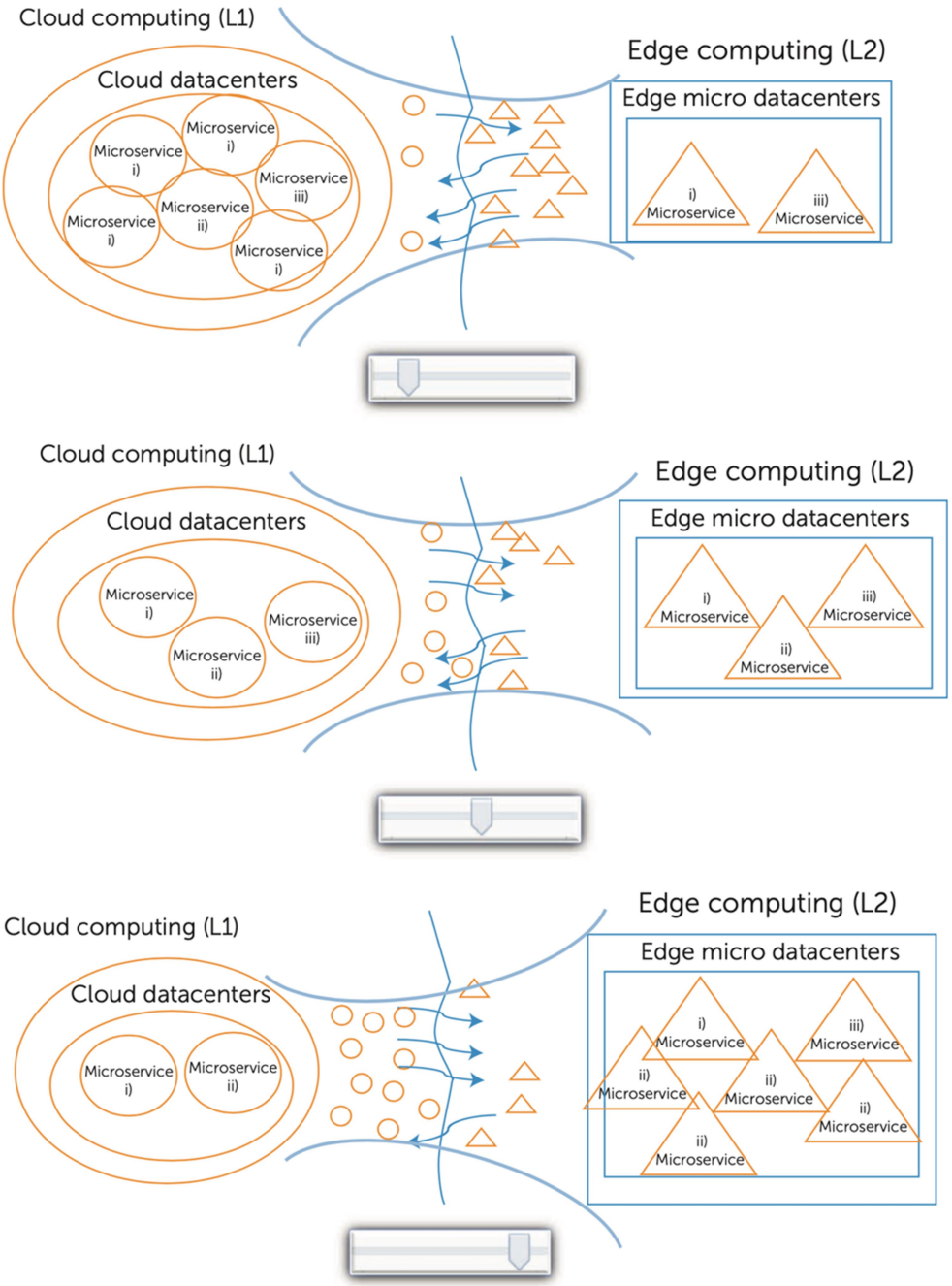

Figura 2.6 – Osmotic computing in datacenter cloud ed edge [2.2]

Un approccio innovativo per affrontare questi problemi è quello di separare la gestione di dati e applicazioni utente dalla gestione dei servizi di rete e di sicurezza. L'osmotic computing si muove in questa

direzione, fornendo un'infrastruttura flessibile tramite la distribuzione automatica e sicura di microservizi. In particolare, l'osmotic computing si basa su un innovativo approccio application-agnostic, che sfrutta tecnologie di virtualizzazione basate su container leggeri (come Docker e Kubernetes[5]) per l'implementazione di microservizi in data center eterogenei di edge e cloud.

Come mostrato in **Errore. L'origine riferimento non è stata trovata.**, l'osmotic computing si estende su due livelli infrastrutturali principali. Il livello **L1** è costituito da datacenter cloud, che offrono diversi tipi di servizi e microservizi. Ai fini dell'osmotic computing, a questo livello i microservizi sono composti in base ai requisiti di alto livello degli utenti. Il livello **L2** identifica l'ambiente di edge computing, che include punti di acquisizione dati e nodi gateway, in grado di eseguire operazioni (di aggregazione su base statistica, ma non solo) su dati locali. Questi dispositivi acquisiscono dati con una frequenza predefinita (spesso dettata dalla velocità di variazione del fenomeno osservato), a seconda della capacità del dispositivo di registrare o raccogliere dati e dei requisiti di sistema specifici che devono essere soddisfatti. I dispositivi L2 possono eseguire varie operazioni più avanzate sui dati grezzi raccolti nell'ambiente, come la crittografia di un flusso di dati in entrata o operazioni di codifica / transcodifica prima di inoltrare questi dati per successive analisi a L1. A causa delle diverse proprietà dei sistemi in L1 e L2, ha senso immaginare un cloud eterogeneo distribuito composto da diversi tipi di risorse situate in ciascuno dei due strati. Comprendere come un microservizio ospitato su un cloud in L1 possa interagire e coordinarsi con un microservizio in L2 è una sfida chiave per la ricerca in tali sistemi. Ogni livello ha le sue funzionalità oggettive che influenzano i tipi di operazioni eseguite. Ad esempio, L2 è generalmente costituito da dispositivi a risorse limitate (batteria limitata, capacità della rete, ecc.) ed elementi di rete, che devono eseguire attività senza sovraccaricare le risorse disponibili.

I datacenter L1 e i microdatacenter L2 possono appartenere a provider diversi. Tuttavia, in uno scenario federato, i fornitori possono stabilire relazioni e cooperare per condividere risorse e servizi, aumentando così le loro opportunità commerciali. In questo scenario, un framework di osmotic computing è application-agnostic, ed offre alle applicazioni utente ambienti di runtime che funzionano in modo distribuito e sicuro. Pertanto, i principali tipi di microservizi che il framework di osmotic computing deve orchestrare e distribuire nell'infrastruttura cloud e edge sono:

- microservizi di uso generale, strettamente correlati allo specifico obiettivo applicativo;

[5] Kubernetes: https://kubernetes.io/

- microservizi per la gestione della rete per la creazione di reti virtuali tra microservizi distribuiti nel sistema cloud/edge distribuito e federato;
- microservizi per la gestione della sicurezza per supportare lo sviluppo multipiattaforma di microservizi sicuri.

La soluzione di provisioning dei microservizi può trarre vantaggio dall'aggregazione di diversi tipi di risorse negli ambienti di distribuzione L1 e L2. Comprendere come questi sistemi possano essere aggregati per supportare i requisiti dell'applicazione (in particolare i requisiti non funzionali, come latenza, throughput, sicurezza e budget) rimane una sfida importante.

L'osmotic computing si propone di caratterizzare il modo in cui i microservizi composti devono essere adattati automaticamente ai siti di distribuzione, considerando la posizione e il contesto di distribuzione, poiché i container sono strettamente correlati alle capacità dell'host fisico. Inoltre, un decisore deve mappare i microservizi nella posizione pertinente. Tale decisione è influenzata dai vincoli identificati dall'applicazione specifica e dal fornitore dell'infrastruttura, come l'utilizzo di risorse specializzate (come un cluster GPU), il miglioramento dei ricavi o la riduzione delle spese generali di gestione (ad esempio amministrazione del sistema e/o costi energetici). L'adattamento dei microservizi alle fluttuazioni nell'ambiente di computazione deve essere eseguito nel tempo, durante l'esecuzione. Pertanto, è necessaria un'orchestrazione dinamica basata su feedback per rilevare i cambiamenti nelle prestazioni dell'infrastruttura e nelle metriche di QoS.

2.3. Componenti integrati

Lo stile architetturale a microservizi adottato ha come ulteriore beneficio la possibilità di integrare componenti Commercial-Off-The-Shelf (COTS) già disponibili sul mercato per svolgere le funzionalità di livello middleware e per i moduli applicativi di base. Ciò riduce i costi e tempi di sviluppo della piattaforma, permettendo di concentrare l'effort nello sviluppo delle funzionalità a valore aggiunto.

La Figura 2.7 illustra il mapping dell'architettura proposta in Figura 2.1 con il diagramma dei componenti: per ciascun tipo di microservizio, è specificato lo strumento selezionato per l'integrazione nella piattaforma. A questo scopo è stata condotta un'ampia e approfondita ricerca di mercato, valutando le funzionalità, le caratteristiche tecnologiche, i costi, le licenze e i requisiti hardware e software per ciascun componente candidato ad essere incluso nella piattaforma. La scelta è infine ricaduta su componenti COTS open source di ampio utilizzo nell'industria, con comprovate caratteristiche di affidabilità, sicurezza e prestazioni, nonché garanzie di compatibilità con la containerizzazione per la gestione tramite Docker. La

selezione è stata inoltre guidata da alcuni requisiti innovativi di tipo funzionale e architetturale specifici della piattaforma Digital Future.

Innanzi tutto, un nodo edge dovrà essere in grado di addestrare un modello di machine learning per l'inferenza statistica sui flussi di dati grezzi, e non solo eseguirne uno già creato offline o in ambito cloud. Per questo motivo lato edge non possono essere utilizzate versioni ridotte delle principali librerie di machine learning, come Core ML[6] o TensorFlow Lite[7], tipicamente adottate su dispositivi mobili ed embedded.

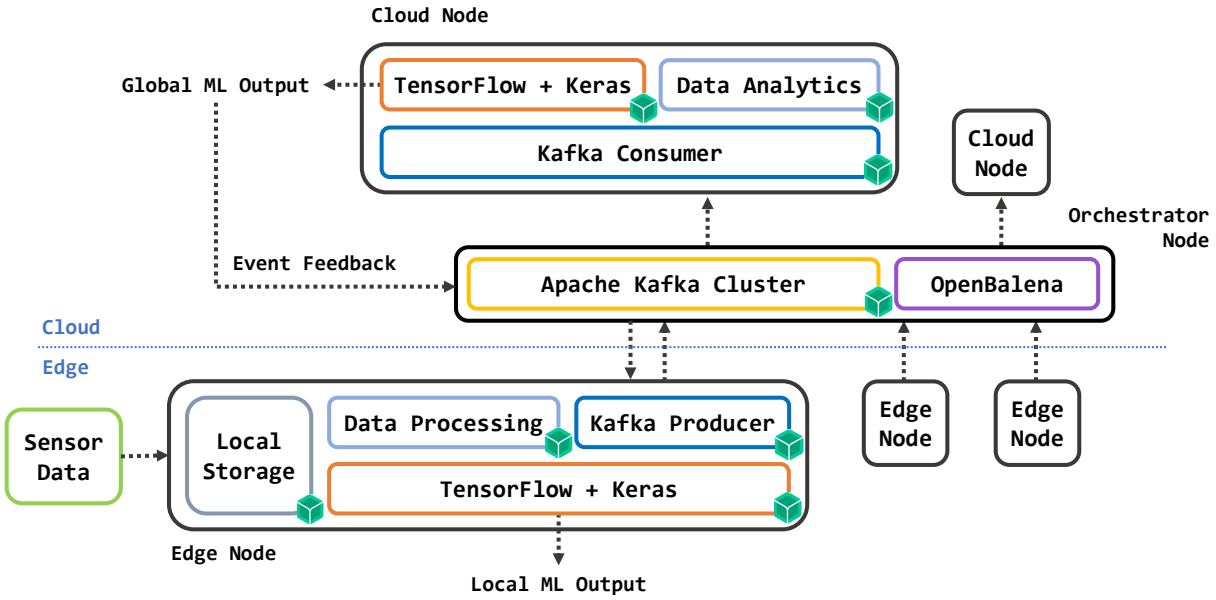

Figura 2.7 – Componenti integrati nell'architettura proposta

Inoltre il data stream management system (DSMS) deve permettere ad un nodo edge di comunicare non solo con i nodi cloud, ma anche con altri nodi edge all'interno dell'infrastruttura. Questo è un fattore importante poiché permette la condivisione di conoscenza su base logica e/o argomentativa tra nodi omologhi a livello edge, rendendo le applicazioni intelligenti quanto più possibile vicine ai dati e capaci di sfruttarne le proprietà di località.

[6] Core ML: https://developer.apple.com/documentation/coreml

[7] TensorFlow Lite: https://www.tensorflow.org/lite

È possibile prevedere che il modello addestrato sul cloud per l'inferenza statistica e logica possa essere utilizzato per aggiornare il modello locale dei singoli nodi edge. Sarà opportunamente valutato, nell'ambito dello sviluppo sperimentale di dimostratori dell'architettura proposta in scenari di riferimento selezionati, se implementare questa funzionalità oppure permettere ai nodi cloud di inviare verso l'edge un più semplice feedback basato sull'analisi globale dei dati. La natura di questo feedback dipenderà strettamente dallo scenario specifico e potrà generare un'azione o un cambiamento di stato in uno o più nodi edge.

La Tabella 2.1 riassume, per ciascun componente, le responsabilità principali e lo strumento individuato. Nei seguenti sottoparagrafi sono descritte le caratteristiche dei COTS selezionati.

Componente	Responsabilità	Strumento proposto
Local Storage	*Memorizzare localmente e temporaneamente i dati rilevati da utilizzare per le successive operazioni di processing.*	*Redis in-memory database o SQLite, in accordo con la piattaforma HW di riferimento.*
Data Processing	*Aggregare i dati da inviare sul cloud. Utilizzato per ridurre la mole di dati ed i tempi di trasmissione.*	*N.D.: implementazione specifica dipendente dallo scenario applicativo.*
Data Producer	*Inviare i dati verso la piattaforma cloud.*	*Apache Kafka Producer client.*
Edge Intelligence	*Eseguire algoritmi di machine learning (training del modello + evaluation) per classificazione e regressione a partire dai dati presenti nello storage locale. Può anche integrare tecniche di inferenza logica e argomentativa.*	*Keras con TensorFlow.*
Data Consumer	*Recuperare i dati inviati dai nodi edge e presenti nella piattaforma di management*	*Apache Kafka Consumer client*
Cloud Intelligence	*Eseguire algoritmi di machine learning (training del modello + evaluation) per classificazione e regressione elaborando una quantità maggiore di dati. Può anche integrare tecniche di inferenza logica e argomentativa.*	*Keras con TensorFlow.*
Data Analytics	*Effettuare ulteriori analisi di business su grandi moli di dati. Può integrare strumenti di dashboarding e reportistica.*	*N.D.: dipendente dallo scenario applicativo.*
Data Stream Management System	*Gestire i flussi di dati e di eventi da edge a cloud e viceversa, nonché tra nodi cloud e tra nodi edge.*	*Apache Kafka cluster.*
Orchestrator	*Deployment e gestione dei container dei microservizi su nodi edge e cloud. Migrazione dei container da edge e cloud e opzionalmente da cloud a edge.*	*OpenBalena con Docker.*

Tabella 2.1 - Componenti, responsabilità e strumenti nell'architettura Digital Future

2.3.1. *Docker*

Docker è un insieme di prodotti *Platform as a Service* (PaaS) che usa la virtualizzazione a livello di sistema operativo per il deployment delle applicazioni in pacchetti detti *contenitori* (container). I container sono isolati l'uno dall'altro e raggruppano i propri software, librerie e file di configurazione; possono comunicare tra loro solo attraverso canali ben definiti. Tutti i contenitori in esecuzione su una macchina host fisica sono gestiti da un singolo kernel del sistema operativo e risultano quindi più leggeri delle macchine virtuali.

L'approccio di virtualizzazione a livello di sistema operativo adottato da Docker utilizza le funzionalità di isolamento delle risorse del kernel Linux, come ad esempio *cgroups* (control groups) e *namespace*, per consentire a container indipendenti di coesistere sulla stessa istanza di Linux. I namespace del kernel Linux isolano ciò che l'applicazione può vedere dell'ambiente operativo, incluso l'albero dei processi, la rete, gli ID utente e i file system montati, mentre i cgroups garantiscono l'isolamento e la limitazione dell'accesso alle risorse, inclusi la CPU, la memoria, i dispositivi di I/O a blocchi e la rete.

Docker è un progetto open source rilasciato con licenza Apache ed è sviluppato principalmente in linguaggio Go. Include due strumenti specifici per il deployment e provisioning di applicazioni distribuite, utilizzabili anche nelle architetture a microservizi:

- **Docker Compose**: strumento per la definizione e l'esecuzione di applicazioni Docker multi-container. Utilizza file in formato YAML per configurare i servizi dell'applicazione ed esegue il processo di creazione e avvio di tutti i container con un singolo comando. Docker Compose consente agli utenti di eseguire comandi su più container contemporaneamente, ad esempio la creazione di immagini, il ridimensionamento dei container, l'esecuzione di container che sono stati arrestati e altro ancora.
- **Docker Swarm**: fornisce funzionalità di clustering nativo per container Docker, che trasforma un cluster di nodi Docker in un singolo nodo Docker virtuale. Consente agli utenti di eseguire vari comandi per gestire i nodi in un cluster, ad esempio elencare, aggiornare e rimuovere i nodi. L'elaborazione distribuita tra nodi di un cluster Docker Swarm si avvale dell'algoritmo di consenso *Raft* [2.4].

Il sito Web ***Docker Hub***[8] fornisce un ampio catalogo di container con (micro)servizi pronti da scaricare e usare, che include i più popolari pacchetti software open source.

2.3.2. *Balena*

OpenBalena[9] è una piattaforma di distribuzione e gestione di dispositivi connessi in rete. Tali dispositivi utilizzano il sistema operativo balenaOS, progettato per l'esecuzione di container su dispositivi IoT, e sono gestiti tramite l'interfaccia da riga di comando *balena*, che è possibile utilizzare per configurare i container dell'applicazione, inviare aggiornamenti, controllare lo stato dei dispositivi, visualizzare log e così via. I servizi di back-end di OpenBalena, composti da componenti già utilizzati in produzione su *balenaCloud*, possono archiviare le informazioni dei dispositivi in modo sicuro e affidabile, consentire la gestione remota tramite un servizio VPN integrato e distribuire in modo efficiente le immagini dei container sui dispositivi.

BalenaOS[10] è un sistema operativo con kernel Linux ottimizzato per l'esecuzione di container Docker su dispositivi embedded, con particolare attenzione all'affidabilità per lunghi periodi di funzionamento ed alla semplicità inerente al workflow di sviluppo di applicazioni. L'idea di base dietro balenaOS è che i container Linux offrono un percorso pratico per l'utilizzo della virtualizzazione su dispositivi embedded. Le macchine virtuali e gli hypervisor hanno portato a enormi passi avanti nella produttività e nell'automazione per le distribuzioni cloud, ma la loro astrazione dell'hardware, l'overhead delle risorse e la mancanza di supporto hardware non sono adatti per scenari integrati. Con la virtualizzazione a livello di sistema operativo, come implementata per i container Linux, entrambe queste problematiche vengono meno, rendendo l'approccio molto conveniente su dispositivi IoT.

BalenaOS è un sistema operativo creato per essere facilmente portabile su più tipi di dispositivi (tramite il framework *Yocto*) ed è ottimizzato per i container Linux, in particolare per Docker. I pacchetti di userspace di balenaOS forniscono solo gli elementi essenziali per l'esecuzione di container: l'idea di base è che il software e i servizi devono essere forniti dal container, a meno che non siano genericamente utili a tutti i

[8] Docker Hub: https://hub.docker.com/

[9] OpenBalena: https://www.balena.io/open/

[10] BalenaOS: https://www.balena.io/os/

container o non possano esistere in un container. Tale filosofia si sposa bene con un modello architetturale a microservizi.

2.3.3. *TensorFlow*

TensorFlow[11] è una piattaforma open source end-to-end per l'apprendimento automatico. Dispone di un ecosistema completo e flessibile di strumenti, librerie e risorse della comunità orientate sia alla ricerca nel campo del machine learning, che allo sviluppo di applicazioni basate su machine learning. Esso consente agli sviluppatori di creare "dataflow graph", strutture che descrivono il modo in cui i dati si spostano attraverso un grafo o una sequenza di nodi di elaborazione. Ogni nodo nel grafo rappresenta un'operazione matematica e ogni connessione tra i nodi è un *tensore*, cioè un array di dati multidimensionale.

TensorFlow è in grado di addestrare ed eseguire reti neurali profonde per la classificazione di cifre scritte a mano, riconoscimento di immagini, word embedding, reti neurali ricorrenti, modelli sequence-to-sequence per la traduzione automatica, elaborazione del linguaggio naturale e simulazioni basate su equazioni differenziali alle derivate parziali.

TensorFlow è stato originariamente sviluppato da ricercatori e ingegneri del team di Google Brain all'interno della Google Machine Intelligence Organization per condurre ricerca nell'ambito dell'apprendimento automatico e delle reti neurali profonde. Il sistema è abbastanza generale per essere applicabile anche in un'ampia varietà di altri domini. TensorFlow fornisce API stabili in Python e C++, nonché API instabili per numerosi altri linguaggi. È distribuito con licenza Apache.

2.3.4. *Keras*

Keras[12] è una libreria open source per reti neurali scritta in Python e distribuita con licenza MIT. Progettato per consentire una rapida sperimentazione con reti neurali profonde, si concentra sull'essere intuitivo, modulare ed estensibile. È in grado di funzionare utilizzando diversi backend per il calcolo, come

[11] TensorFlow: https://www.tensorflow.org/

[12] Keras: https://keras.io/

TensorFlow, Microsoft Cognitive Toolkit[13], R[14], Theano[15] o PlaidML[16]. È stato sviluppato come parte del progetto di ricerca ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) ed il suo principale autore e manutentore è François Chollet, un ingegnere di Google.

Keras contiene numerose implementazioni di componenti comunemente utilizzati nelle reti neurali, come livelli, obiettivi, funzioni di attivazione, ottimizzatori e una serie di strumenti per facilitare il lavoro con i dati di immagine e testo, e per semplificare il codice necessario per programmare reti neurali profonde. Oltre alle reti neurali standard, Keras supporta le reti neurali convoluzionali e ricorrenti. La libreria supporta altre utilità di uso frequente come dropout, normalizzazione batch e pooling.

Keras consente agli utenti di produrre modelli profondi su smartphone (iOS e Android), sul Web o su Java Virtual Machine. Consente inoltre l'addestramento distribuito di modelli di deep learning su cluster di Graphics Processing Unit (GPU) e Tensor Processing Unit (TPU), principalmente tramite CUDA[17].

2.3.5. *Apache Kafka*

Apache Kafka[18] è una piattaforma software open source di stream processing sviluppata originariamente da LinkedIn e donata alla Apache Software Foundation, scritta in Scala e Java. Il progetto mira a fornire una piattaforma unificata, ad alto throughput e bassa latenza, per la gestione di flussi di dati in tempo reale. Kafka può connettersi a sistemi esterni per l'importazione/esportazione di dati tramite *Kafka Connect* e fornisce una libreria Java di elaborazione di stream chiamata *Kafka Streams*. Kafka utilizza un protocollo binario basato su TCP che è ottimizzato per l'efficienza e si basa sull'astrazione del "set di messaggi", ossia in un raggruppamento dei messaggi atto a ridurre l'overhead del roundtrip di rete. Ciò implica pacchetti di rete più grandi, operazioni sequenziali più grandi su disco e blocchi di memoria contigui, che consentono a Kafka di trasformare un flusso di scritture casuali in scritture sequenziali.

---

[13] Microsoft Cognitive Toolkit: https://docs.microsoft.com/en-us/cognitive-toolkit/

[14] The R Project for Statistical Computing: https://www.r-project.org/

[15] Theano: http://deeplearning.net/software/theano/

[16] PlaidML: https://github.com/plaidml/plaidml

[17] Compute Unified Device Architecture: https://developer.nvidia.com/cuda-zone

[18] Apache Kafka: https://kafka.apache.org/

Kafka memorizza messaggi in forma chiave-valore provenienti da diversi processi chiamati *produttori*. Altri processi, chiamati *consumatori*, possono leggere i messaggi generati dai produttori. Kafka funziona su un cluster di uno o più server (chiamati *broker*), in modo da poter distribuire anche grandi flussi di messaggi in maniera fault-tolerant. A partire dalla versione 0.11.0.0, Kafka offre scritture transazionali tramite l'API *Streams*. Kafka supporta due tipi di topic per i messaggi: *normale (regular)* e *compatto*. I topic normali possono essere configurati con un tempo massimo di conservazione o un limite di spazio. Se esistono record più vecchi del tempo di conservazione specificato o se viene superato il limite di spazio, Kafka può cancellare i vecchi dati per liberare spazio di archiviazione. Per impostazione predefinita, gli argomenti sono configurati con un tempo di conservazione di 7 giorni, ma è anche possibile archiviare i dati a tempo indeterminato. Per i topic compatti, i record non scadono in base al tempo o ai limiti di spazio, bensì Kafka tratta i messaggi successivi come aggiornamenti a messaggi più vecchi con la stessa chiave e garantisce di non eliminare mai l'ultimo messaggio per ciascuna chiave. Gli utenti possono eliminare completamente i messaggi per una chiave specifica scrivendo un cosiddetto messaggio *tombstone* con valore null.

Ci sono quattro principali API in Kafka:

- **Producer API:** consente ad un'applicazione di pubblicare stream di record.
- **Consumer API:** consente ad un'applicazione di iscriversi a topic per ricevere ed elaborare stream di record.
- **Connect API:** API di comunicazione di alto livello costruita in cima alle API producer e consumer.
- **Stream API:** elabora i flussi di input in modo da ottenere output e produrre risultati.

Le API producer e consumer si basano sul protocollo di messaggistica Kafka e offrono un'implementazione di riferimento in Java per i client consumer e producer. Il protocollo di messaggistica sottostante è un protocollo binario che gli sviluppatori possono utilizzare per programmare i propri client consumer o producer in qualunque linguaggio di programmazione. Questo svincola Kafka dall'ecosistema Java.

2.3.6. *Redis*

Redis è un DBMS NoSQL open source residente in memoria, privo di schema e con persistenza facoltativa. Il nome deriva da REmote DIctionary Server. Si tratta, infatti, di un database di tipo *key-value store*: il modello dei dati è costituito da dizionari, insiemi di coppie chiave-valore. Supporta diversi tipi di strutture di dati astratte, come stringhe, liste, mappe, set, set ordinati, bitmap, stream e indici geospaziali, con il supporto a query basate su centro e raggio.

Il deployment può avvenire su singolo nodo o su cluster di nodi con repliche sincronizzate. Redis è utilizzato non solo come per memorizzare basi di dati in senso convenzionale, ma spesso è adoperato come cache in memoria – associato ad esempio a un Web server o ad un application server – e può essere utilizzato anche come message broker, poiché supporta il paradigma publish/subscribe.

Redis è disponibile come microservizio su container Docker e fornisce prestazioni elevate, rivelandosi adeguato anche a contesti di edge computing con single-board computer come Raspberry Pi[19] per diversi motivi:

- i requisiti di memoria e CPU sono contenuti; Redis può essere eseguito anche su dispositivi piccoli ed economici come il Raspberry Pi Zero[20];
- le strutture di dati previste sono generalmente adatte alle applicazioni IoT ed embedded, ad esempio per accumulare serie temporali di dati, ricevere e mettere in coda comandi da eseguire o risposte da inviare ai server remoti;
- la versatilità della modellazione dei dati in un database Redis può risultare efficace per applicazioni in cui le decisioni sono prese direttamente al livello edge;
- Redis può essere utilizzato come sistema di comunicazione interprocesso tra i processi/servizi in esecuzione nel dispositivo;
- le funzionalità di persistenza su file, con archiviazione di tipo append-only, sono adatte per le memorie elettroniche come schede SD (Secure Digital) e dischi SSD (solid-state drive).

Il creatore e principale sviluppatore di Redis è il siciliano Salvatore Sanfilippo; il software è distribuito con licenza BSD.

2.3.7. SQLite

SQLite è una libreria software scritta in linguaggio C che implementa un DBMS SQL autoconsistente, serverless, senza necessità di configurazione, incorporabile all'interno di applicazioni. Permette di creare una base di dati (comprese tabelle, indici, viste, trigger) incorporata in un unico file. SQLite non è un

[19] Redis on ARM: https://redis.io/topics/ARM

[20] CPU ARM11 a 1 GHz, 512 MB RAM, dimensioni fisiche 65 x 30 × 5 mm, prezzo indicativo 5 €.

processo standalone utilizzabile di per sé, ma può essere incorporato all'interno di un altro programma come libreria. È utilizzabile con il linguaggio C/C++, ma esistono binding anche per altri linguaggi.

Sono numerose le caratteristiche che rendono SQLite un'opzione favorevole in contesti IoT/embedded e in soluzioni per l'edge computing.

- SQLite è una libreria compatta: con tutte le funzionalità abilitate, può risultare anche più piccola di 600 kB in base alla piattaforma target. Le prestazioni, inoltre, sono buone anche in ambienti con poca memoria disponibile.
- Offre un'implementazione molto vasta di SQL, con dimensioni dello schema e della base dati pressoché illimitate, supporto a query nidificate e a transazioni ACID (atomicità, consistenza, isolamento, durabilità) anche nidificate, viste e trigger in quantità illimitata, una libreria di funzioni SQL standard, ricerca completa all'interno di stringhe, ricerca per pattern matching, gestione di campi in formato JSON, ed altro ancora.
- Il formato di file SQLite è standardizzato, stabile, multipiattaforma e retrocompatibile. Per questa ragione i database SQLite sono spesso usati non solo per lo storage locale dei dati, ma anche per trasferire dati strutturati tra sistemi e applicazioni e per l'archiviazione a lungo termine.
- Compatibilità con numerose piattaforme a 32 e 64 bit, anche per dispositivi mobili ed embedded.
- Elevata affidabilità contro errori applicativi, errori di sistema, input o basi di dati corrotti o manipolati in modo malevolo.

Il codice sorgente di SQLlite è stato rilasciato nel pubblico dominio, rendendolo utilizzabile senza alcuna restrizione. Tuttavia non è possibile a chiunque contribuire modifiche al codice: lo sviluppo è condotto da un piccolo team di esperti seguendo rigorose metodologie di progettazione, implementazione e test ispirate al software safety-critical per l'avionica. Le API nativamente sviluppate sono per i linguaggi C/C++ e Tcl, ma può essere utilizzato con numerosi altri linguaggi di programmazione mediante API dedicate (è il caso di Python) oppure interfacce standard per DBMS (come l'interfaccia JDBC – Java Data Base Connectivity – disponibile per Java).

2.4. Benefici attesi e problemi aperti

2.4.1. *Deployment di microservizi*

L'osmotic computing può beneficiare dell'estensione della funzionalità di virtualizzazione ai dispositivi IoT basata su container, come descritto nei paragrafi 2.3.1 e 2.3.2. Un framework di osmotic computing

dovrebbe fornire un *microservice engine*, che consente a utenti e sviluppatori di distribuire container che eseguono microservizi su dispositivi IoT e edge, consentendo l'esecuzione e la distribuzione di microservizi. L'innovazione offerta da questo approccio faciliterà la creazione di un mercato di applicazioni virtuali basate su IoT. I meccanismi di adattamento e versione del software consentiranno ai provider di piattaforme edge e cloud di distribuire microservizi costituiti da un pool eterogeneo di dispositivi fisici. I vantaggi attesi includono il deployment di microservizi orientati all'IoT distribuiti, il consolidamento del software e l'ottimizzazione del servizio.

Uno dei precursori del paradigma dell'osmotic computing per il deployment di servizi in una soluzione PaaS è DRACO [2.5]. Esso permette di creare sistemi software distribuiti auto-configuranti, auto-controllanti, auto-rigeneranti, auto-protetti e auto-ottimizzanti (in breve, *auto-gestiti* o *auto-**). A differenza dei sistemi auto-organizzanti che si evolvono in base a considerazioni probabilistiche, questo approccio si concentra sull'incapsulamento, la replica e l'esecuzione di attività distribuite e gestite che sono specificate con precisione.

2.4.2. *Sicurezza*

Esiste una letteratura significativa sul supporto della sicurezza nei sistemi Cloud e IoT, ma anche in contesti orientati al bordo della rete viene spesso utilizzato un approccio incentrato sul cloud [2.6]. Le sfide dell'integrazione dei dispositivi IoT con i sistemi cloud sono descritte in [2.7]: è discusso un modello di business per tale tipologia di architetture, nonché i limiti e le problematiche relative alla sicurezza dei dispositivi IoT. Il framework proposto in questo report necessita di una politica di sicurezza che deve essere supportata coerentemente sia all'interno di un data center cloud sia in un ambiente edge computing, per consentire l'esecuzione e la migrazione dei microservizi. Garantire il rispetto delle stesse considerazioni di sicurezza, per un particolare microservizio, in entrambi questi ambienti rimane una sfida.

Le funzionalità di sicurezza consentiranno processi di autoidentificazione che renderanno più semplice e sicura la distribuzione di microservizi all'interno dei dispositivi cloud e edge, facilitando anche l'adozione ampia della tecnologia proposta. Inoltre, un altro obiettivo è quello di aggiungere funzionalità di sicurezza al microservice engine, al fine di consentire il deployment sicuro di container di microservizi su dispositivi IoT. Più specificamente, il framework dovrà consentire agli sviluppatori di costruire catene di fiducia che coinvolgano sia dispositivi edge che sistemi cloud.

2.4.3. *Edge computing e Intelligent Edge*

I recenti sforzi nel creare un "IoT Cloud" open source con approcci quali sensors-as-a-service [2.8], Fog Computing e Cloudlet Computing indicano un interesse significativo in questo settore da parte della comunità accademica, come dimostra il progetto europeo OpenIoT[21]. Contemporaneamente, le offerte commerciali discusse nel deliverable 1.2.A denotano forti interessi e un grado di maturità tecnologica in rapida crescita.

L'approccio proposto non è un'alternativa a tali sforzi, ma si concentra sul trasferimento/migrazione e integrazione di funzionalità basate sull'AI. Esso suggerisce la necessità di abilitare l'off-load di elaborazioni inizialmente effettuate all'interno di un datacenter su un dispositivo a bordo rete. Come già discusso in precedenti documenti di progetto, ciò consente di eseguire il calcolo più vicino al fenomeno misurato, superando le latenze e i costi di trasferimento dei dati. La valutazione dell'approccio e dell'architettura proposti si concentrerà sulla comprensione dei tipi di funzionalità di AI più rilevanti da eseguire come microservizi lato edge, piuttosto che all'interno di un ambiente cloud, e viceversa.

2.4.4. *Interferenze e contese di risorse tra microservizi*

I microservizi distribuiti su datacenter cloud o edge possono portare a problemi di contesa di risorse computazionali e di rete, che possono influire sulla Quality of Service (QoS). Durante la distribuzione di microservizi, le tecniche di orchestrazione devono considerare quali microservizi devono essere combinati su una risorsa cloud, per ridurre al minimo la contesa di risorse a causa di interferenze tra i carichi di lavoro. Il consumo di risorse dei workload e la QoS non sono additivi, quindi comprendere la natura della loro composizione è fondamentale per decidere quali microservizi possono essere schierati insieme.

Recenti lavori hanno studiato diversi approcci per ridurre al minimo l'impatto dell'interferenza del carico di lavoro sul QoS delle applicazioni ospitate sui datacenter cloud. In [2.9] è stato sviluppato uno schema per quantificare gli effetti della contesa della cache tra carichi di lavoro consolidati. Tuttavia, le tecniche sopra menzionate si concentrano su problemi di contesa di un solo tipo di risorsa hardware (cache) ignorandone altri. Nessuno degli approcci accademici esistenti né i framework di orchestrazione dei container come OpenShift Origin, Amazon EC2 Container Service e Kubernetes sono in grado di rilevare e gestire

[21] OpenIoT: http://www.openiot.eu

automaticamente la contesa di risorse tra microservizi distribuiti tra cloud e edge. Dovrebbe essere studiata una tecnica basata su software che apprende la contesa tra microservizi tramite simulazione e benchmarking, quindi applicare le informazioni acquisite in tempo reale per rilevare la contesa tra microservizi tramite monitoraggio attivo e previsione. Su questa linea la ricerca dovrebbe concentrarsi su nuove tecniche di consolidamento dei microservizi in grado di rilevare e risolvere dinamicamente la contesa di risorse tramite la caratterizzazione delle prestazioni dei microservizi, la definizione delle priorità dei carichi di lavoro e il deployment coordinato.

2.4.5. *Orchestrazione ed elasticità*

L'orchestrazione a runtime di microservizi in un sistema edge/cloud scalabile è un problema complesso a causa della difficile stima del comportamento del carico di lavoro dei microservizi in termini di volume di dati da analizzare, velocità di arrivo dei dati, tipi di query, distribuzioni dei tempi di elaborazione dei dati, distribuzione dei tempi di elaborazione delle query, comportamento dei sottosistema di I/O e numero di utenti che si connettono a tipi diversi e combinazioni differenti di microservizi. Senza conoscere i comportamenti del carico di lavoro dei microservizi, è difficile prendere decisioni sui tipi e le dimensioni delle risorse dei nodi cloud e edge da sottoporre a provisioning per i microservizi in un dato momento.

In piattaforme basate sull'architettura proposta, la tradizionale nozione di controllo e riconfigurazione di runtime che considera solo le risorse ospitate nei datacenter cloud va estesa alle risorse distribuite e disponibili al bordo della rete. Mediante tecniche di machine learning possono essere sviluppati modelli predittivi per prevedere il carico di lavoro e le metriche delle prestazioni su più microservizi raggruppati sulle risorse lato cloud e lato edge. Sulla base di tali modelli, inoltre, potranno essere sviluppati algoritmi di orchestrazione delle risorse intelligenti, che tengano conto della QoS e delle contese.

2.5. Bibliografia

[2.1] Cisco Global Cloud Index: Forecast and Methodology, 2016–2021, White Paper. [Online]. Archiviato su: https://web.archive.org/web/20200319153349/https://virtualization.network/Resources/Whitepapers/0b75cf2e-0c53-4891-918e-b542a5d364c5_white-paper-c11-738085.pdf

[2.2] M. Villari, M. Fazio, S. Dustdar, O. Rana, R. Ranjan, Osmotic computing: A new paradigm for edge/cloud integration. *IEEE Cloud Computing*, 3(6): 76-83, 2016.

[2.3] Intel Corp. & SystemSoft, Preboot Execution Environment (PXE) Specification, Version 2.1, 1999. [Online]. Archiviato su: https://web.archive.org/web/20110524083740/http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf

[2.4] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm. In 2014 USENIX Annual Technical Conference, pp. 305-319, 2014.

[2.5] A. Celesti, N. Peditto, F. Verboso, M. Villari, A. Puliafito, DRACO PaaS: A distributed resilient adaptable cloud oriented platform. In IEEE 27th International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), pp. 1490–1497, 2013.

[2.6] K. Lee, D. Kim, D. Ha, U. Rajput, H. Oh, On security and privacy issues of Fog Computing supported Internet of Things environment. In: 6th International Conference on the Network of the Future (NOF), pp. 1–3, 2015.

[2.7] P. Parwekar, From Internet of Things towards Cloud of Things. In 2nd International Conference on Computer and Communication Technology (ICCCT-2011), pp. 329-333, IEEE, 2011.

[2.8] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, D. Georgakopoulos, Sensor search techniques for sensing as a service architecture for the internet of things. IEEE Sensors Journal, 14(2): 406-420, 2013.

[2.9] S. Govindan, J. Liu, A. Kansal, A. Sivasubramaniam, Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines. In Proceedings of the 2Nd ACM Symposium on Cloud Computing (SOCC), pp. 22:1–22:14, ACM, 2011.

3. Framework per Edge Intelligence

Negli ultimi anni, il paradigma dell'edge computing sta emergendo notevolmente attraverso l'adozione diffusa di sensori micro e nano-dispositivi a basso costo immersi in ambienti quotidiani e interconnessi in reti. La quantità dei dispositivi pervasivi aumenta progressivamente e di conseguenza anche la quantità di dati grezzi disponibili per l'elaborazione e l'analisi aumenta esponenzialmente. Per tale motivo, sono necessari metodi efficaci per trattare i flussi di dati con l'obiettivo finale di fornire un'interpretazione significativa delle informazioni recuperate. I questi scenari si utilizza spesso il termine *"Big Data"* per indicare la ricerca e lo sviluppo di tecniche di analisi dei dati e infrastrutture di gestione per affrontare i problemi di "volume, velocità, varietà e veridicità" delle informazioni [3.1] che emergono quando si materializzano grandi quantità di informazioni da manipolare. Attualmente, l'approccio più diffuso è basato sull'utilizzo di algoritmo di apprendimento automatico adottati per classificare i dati grezzi e fare previsioni orientate al supporto decisionale e all'automazione. I progressi negli algoritmi e nell'ottimizzazione delle tecniche di machine learning procedono in parallelo con i progressi nelle tecnologie pervasive e nelle architetture di gestione dei dati in cloud. Infatti numeri e indiscutibili benefici sono stati introdotti nell'analisi dei dati. Tuttavia, sono ancora evidenti alcune debolezze non trascurabili rispetto alla crescente complessità ed eterogeneità degli scenari di elaborazione pervasiva. In particolare, la mancanza di una caratterizzazione machine-understandable degli output è un limite importante delle tecniche attuale di machine learning che presentano in modo sempre più evidente l'utilizzo semplicistico delle informazioni acquisite ed una mancanza di espressività dei risultati rendendo tali approcci spesso inadatti per un possibile utilizzo in scenari applicativi completamente autonomi.

Il lavoro svolto nell'ambito di questo progetto ha portato allo sviluppo di un framework per machine learning su base semantica all'interno del quale un tipico problema di classificazione è risolto utilizzando approcci inizialmente sviluppati per il resource discovery semantico. Nel framework proposto si vuole migliorare l'analisi sui flussi dei dati raccolti da sensori eterogenei, proponendo non una classificazione semplicistica degli stessi ma associando ai dati reali descrizioni ricche dal punto di vista semantico. L'approccio utilizza servizi di inferenza non standard [3.2] per ottenere una caratterizzazione, annotata rispetto ad una ontologia di riferimento, delle osservazioni raccolte in modo tale da poter effettuare un'analisi dettagliata dell'evento da individuare. La proposta sfrutta sia la teoria generale che le tecnologie dei Pervasive Knowledge-Based Systems (PKBS), intesi come KBS i cui i singoli individui (che compongono la conoscenza asserzionale) sono fisicamente legati ad oggetti disseminati in un determinato ambiente, senza coordinamento centralizzato. Ogni annotazione si riferisce ad una ontologia che fornisce la

concettualizzazione e il vocabolario per il particolare dominio della conoscenza in modo da abilitare l'utilizzo di servizi avanzati di matchmaking che possano operare sui metadati memorizzati nei singoli dispositivi. La proposta non prevede basi di conoscenza centralizzate, al contrario le attività inferenziali sono distribuite tra i dispositivi che forniscono capacità di calcolo minime. Nella visione proposta, metodi di analisi innovativi applicati ai dati estratti da dispositivi low-cost possono fornire risultati utili nel riconoscimento di eventi senza richiedere grandi risorse computazionali superando i limiti di elaborazione imposti dall'hardware minimale utilizzato.

3.1. Identificazione di eventi basata su rappresentazione della conoscenza

L'approccio proposto, mostrato in Figura 3.1, presenta il tipico workflow utilizzato nelle principali tecniche di data mining e machine learning a cui si aggiunge il supporto alla semantica che va a modificare il modo in cui ciascuno step viene eseguito. In particolare, sono state analizzate le seguenti fasi:

- acquisizione e modellazione dei dati grezzi;
- addestramento del modello tramite un training set di riferimento;
- classificazione ed elaborazione delle previsioni;
- valutazione dei risultati.

Nella prima fase di acquisizione delle informazioni, i dati grezzi privi di struttura semantica sono raccolti da sensori distribuiti nell'ambiente da monitorare, ognuno per il rilevamento di differenti parametri di misura, detti *features*, per un totale di *m* features differenti. In questa fase si utilizza un'ontologia $\mathcal{T}$ per supportare l'annotazione e l'interpretazione dei dati. L'ontologia modella la concettualizzazione del dominio attraverso pattern propriamente definiti, inoltre occorre che $\mathcal{T}$ sia aciclica ed espressa in $\mathcal{ALN}$ [3.3]. Come evidenziato in Figura 3.2, ogni parametro è rappresentato da una o più relazioni gerarchiche con le possibili sottoclassi al fine di modellare tutte le configurazioni significative che il parametro stesso può assumere nel dominio di interesse. La profondità della gerarchia e l'ampiezza di ciascun livello sono proporzionali alla risoluzione, all'intervallo di misura dei sensori utilizzati, così come al grado di dettaglio necessario per una corretta rappresentazione dei dati.

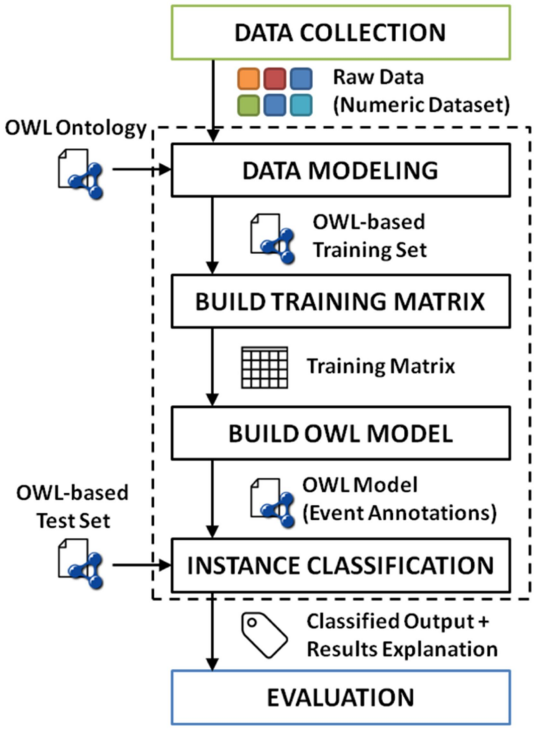

Figura 3.1 – Architettura di riferimento per l'identificazione di eventi

Figura 3.2 – Esempio di modellazione di una ontologia

Ciascuna classe foglia definita dell'ontologia è associata inoltre ad una serie di OWL Annotation Property, proprietà che possono essere associate a qualsiasi elemento di una ontologia per fornire informazioni

aggiuntive. Nell'approccio proposto si sfruttano opportunamente due annotation property (*minValue* e *maxValue*) tramite cui valutare l'appartenenza di una rilevazione al range definito per la sottoclasse. Un esempio riferito all'ontologia mostrata in precedenza è mostrato in Figura 3.3. Questa soluzione è stata adottata per modellare le sottoclassi afferenti ad uno stesso parametro senza generare inconsistenze. In tal modo, inoltre, non si è obbligati ad utilizzare soltanto valori interi non negativi, come nel caso di modellazioni differenti basate su restrizioni di cardinalità.

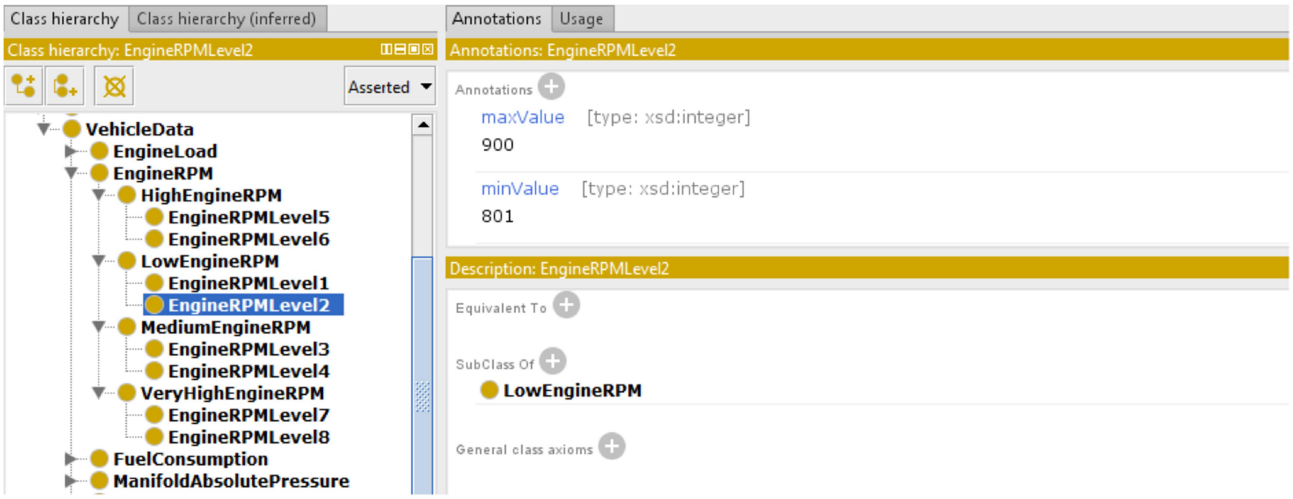

Figura 3.3 – Esempio di modellazione attraverso OWL Annotation Property

Dopo la fase di acquisizione è possibile effettuare la costruzione di un'annotazione semantica per ogni possibile classe di output, che vada a connotare il fenomeno o l'evento osservato. Si parte da un set di dati di training *S*, contenente *n* campioni (o rilevazioni) di addestramento, ciascuno caratterizzato da *m* features (le proprietà della rilevazione). Per ogni rilevazione, in una finestra temporale definita, si effettua il parsing dei dati: a ciascun valore assunto da ogni proprietà nella rilevazione in esame viene attribuito il concetto più adatto; ovvero ogni valore della feature è mappato al corrispettivo concetto più specifico presente nell'ontologia $\mathcal{T}$. Il training set, oltre a contenere i valori rilevati dai sensori, definisce anche *w* distinte classi di output indicati dall'operatore umano. Ognuna delle *n* rilevazioni è dunque composta da:

- k componenti dei concetti ($C_{i,1}$, …, $C_{i,k}$) che annotano le features (con $k \leq m$, dato che ogni singolo campione potrebbe risultare incompleto a causa della diversa frequenza di rilevazione dei sensori);
- uno o più output osservati O_i, etichettati da una classe dell'ontologia.

I campioni vengono processati in sequenza per costruire la *Matrice delle Occorrenze M*. Essa è una matrice di dimensione massima $(w + 1) \times (k + 1)$ che riporta sulla prima colonna tutti i differenti output, sulla prima

riga le distinte componenti dei concetti e in ogni cella il numero di occorrenze della componente del concetto nel campione avente il corrispettivo output.

Require:

- L Logica Descrittiva;
- TBox aciclica T;
- training set $S = \{ S_1, S_2, \dots S_n \}$, con $S_i = (C_{i,1}, \dots\dots, C_{i,m}, O_i)$ $\forall$ i = {1, …, n}, dove tutti i $C_{i,j}$ e gli O_i sono espressi in L e soddisfacibili in T.

Ensure:

- M : $(w + 1) \times (k + 1)$ è la matrice delle occorrenze dei concetti per ogni output osservato, dove k è il numero totale dei concetti distinti che compaiono in S.

```
M := 0   // matrice (1 × 1)
r := 1, c := 1
for i := 1 to |S| do
        ur := findConceptIndex (Oi, M(: , 1))
        if ur == null then
                aggiungere una riga a M
                r := r + 1
                ur := r
                M(ur, 1) := Oi
                inizializzare M(ur, 2 : c) a zero
        end if
        for j := 1 a m do
        uc := findConceptIndex(Ci,j, M(1, :))
                if uc == null then
                        aggiungere una colonna a M
                        c := c + 1
                        uc := c
                        M(1, uc) := Ci,j
                        inizializzare M(2 : r, uc) a zero
                end if
                M(ur , uc) = M(ur , uc) + 1          // modificare le occorrenze
        end for
end for
return M
```

Figura 3.4 – Algoritmo di costruzione della matrice delle occorrenze

L'algoritmo di costruzione della matrice, descritto in Figura 3.4, si compone delle seguenti operazioni:

- viene analizzato un campione per volta da cui è estratto l'output (o gli output) di riferimento verificando se già inserito nella matrice;
- sono poi considerati i concetti nell'osservazioni legati all'output, andando ad aggiungere una colonna per ogni concetto non ancora inserito;

- per ogni corrispondenza output/concetto, viene inizializzato ad 1 il numero delle occorrenze se il concetto non è mai stato individuato in precedenza altrimenti il valore esistente viene incrementato di una unità.

Al termine di questo algoritmo, la matrice *M* contiene un'immagine completa del training set opportunamente associato all'ontologia di riferimento. A partire da questa rappresentazione, ogni classe di output può essere definita come congiunzione dei concetti atomici che meglio lo rappresentano. Occorre quindi definire un valore-soglia TS che permetta di discriminare i concetti utili rispetto ad ogni output. Tale valore rappresenta il valore minimo numero di occorrenze che un singolo concetto deve almeno eguagliare per poter essere considerato significativo rispetto ad un particolare output. La struttura numerica e schematica di *M* dà anche la possibilità di definire differenti valori soglia per ogni output e per ogni feature calcolate attraverso un processo di normalizzazione sul dataset in input. L'uso di soglie differenti consente di focalizzare la sensibilità dell'analisi sugli attributi con più alta variazione e su quegli output sui quali è più difficile fare previsioni. Si utilizza una fase di normalizzazione dei valori nella matrice per considerare la diversa frequenza di sensing/capturing di ciascun sensore associato allo specifico attributo. Il fattore di normalizzazione $fn_{i,j}$ associato al j-mo attributo dell'i-mo evento è calcolato con la seguente formula:

$$fn_{i,j} = max\ (\sum SC_j(occorrenza_{i,j})) / \sum SC_j(occorrenza_{i,j})$$

La normalizzazione viene effettuata considerando, per ogni evento, il valore massimo di occorrenze di un parametro nella matrice ottenuto sommando tutte le sue sottoclassi. Successivamente, per ciascun gruppo di classi associate ad un singolo attributo, è normalizzato il numero di occorrenze rapportandolo al valore massimo attraverso il fattore $fn_{i,j}$. Per ogni concetto, il nuovo valore normalizzato sarà calcolato utilizzando la formula seguente:

$$occorrenza_norm_{i,j} = occorrenza_{i,j} * fn_{i,j}$$

Inoltre, per ogni classe di output è individuato il valore soglia da utilizzare per selezionare gli attributi che comporranno la propria caratterizzazione. Il valore soglia TS è calcolato (per ogni output) come prodotto tra una percentuale di sogliatura impostata come parametro dell'algoritmo e il valore medio derivante dalle occorrenze normalizzate:

$$TS_i = PS * avg(occ_norm_{i,j})$$

Le classi selezionate dopo avere definito i valori di soglia saranno utilizzate per costruire l'annotazione di ciascun output. Tali output sono ora dotati di una descrizione formale, dalla semantica ricca, machine-understandable utilizzabile in successive elaborazioni di alto livello. Sotto questo aspetto, l'approccio appartiene alla categoria delle tecniche di "*instance-based learning for class-membership prediction*" [3.4]:

un approccio basato sull'apprendimento istance-based che ha l'obiettivo di prevedere la classe di appartenenza di un individuo basandosi sul calcolo di una misura di similarità o distanza semantica. Questo approccio di addestramento produce una base di conoscenza con conoscenza concettuale (TBox) modellata da operatori umani e una conoscenza fattuale (ABox) creata automaticamente dal flusso di dati disponibili, con istanze che rappresentano gli eventi di interesse, che il sistema è ora in grado di riconoscere.

Per verificare le prestazioni del classificatore, vengono classificati i campioni del testset opportunamente annotati come fatto precedentemente per i campioni del training set. L'operazione di classificazione utilizza un processo di matchmaking semantico basato sui seguenti servizi di inferenza non standard:

- Concept Abduction (CAP) [3.2];
- Concept Contraction (CCP) [3.2].

Data un'ontologia $\mathcal{T}$ e due annotazioni *A* e *B* modellate rispetto ad essa che presentano caratteristiche in conflitto, l'algoritmo di *Concept Contraction* determina un'espressione *G* (*Give Up*) che indica quanto *A* non sia compatibile con *B* e restituisce un valore di penalità (indicato con penalty_c) rappresentativo della distanza semantica associata a *G*. Dualmente, se *A* è invece compatibile con *B*, ma non presentano descrizioni completamente identiche, il servizio di *Concept Abduction* calcola un'espressione *H* (*Hypothesis*) che rappresenta cosa dovrebbe essere ipotizzato in *B* per soddisfare completamente *A* restituendo anche in questo caso il relativo valore di distanza, indicato con penalty_a. I due valori di penalty ottenuti dal matchmaking sono opportunamente combinati per calcolare la distanza semantica tra le istanze annotate in input e le descrizioni delle classi di output generate durante la fase di addestramento. La classe predetta o riconosciuta sarà quella con distanza minore.

Utilizzando questo approccio è possibile programmare una fase di valutazione dei risultati ottenuti dalla classificazione, realizzata per quantificare statisticamente la correttezza tra la classe ottenuta attraverso l'algoritmo di predizione e la classe reale associata a ciascuna istanza. Per ottenere una misura della precisione e dell'accuratezza della classificazione effettuata si calcola la *confusion matrix*, che mostra le relazioni tra le classi predette e le classi reali. Ha dimensioni L × L, dove L indica il numero delle diverse classi di output coinvolte nella predizione. Il numero di istanze classificate correttamente è dato dalla somma dei valori sulla diagonale, mentre le altre istanze sono classificate in modo errato.

In Tabella 3.1 è mostrato un esempio di matrice di confusione 2×2 che include i seguenti valori:

a. numero di correct negative predictions;
b. numero di incorrect positive predictions;
c. numero di incorrect negative predictions;

d. numero di correct positive predictions.

Output ***Reale / Predetto***	***False***	***True***
False	*a*	*b*
True	*c*	*d*

Tabella 3.1 - Esempio di matrice di confusione

Utilizzando la matrice di confusione, è possibile individuare i seguenti indici di riferimento:

- **accuracy**, percentuale di istanze, sia positive che negative, classificate correttamente:

$$(a + d) / (a + b + c + d)$$

- **recall**, percentuale di istanze positive classificate corettamente:

$$d / (c + d)$$

- **precision,** percentuale di predizioni positive classificate correttamente:

$$d / (b + d)$$

Per ogni data set ottenuto, è inoltre effettuata un'operazione di *cross-validation* che mira a verificare la bontà delle previsioni ottenute. Tale processo consiste nel suddividere il dataset in porzioni. Ciascuna parte, opportunamente combinata con le altre, sarà destinata a comporre il training set o il test set generando così insiemi di dati distinti tra loro e facendo in modo che la singola osservazione non vada ad influenzare unicamente la fase di addestramento o di test, essendo stata utilizzata in momenti diversi per entrambi gli scopi. Il classificatore può inoltre essere testato più volte (pari al numero delle combinazioni create) sullo stesso dataset per una validazione più completa delle prestazioni ottenute.

3.2. Benefici attesi e problemi aperti

Il framework proposto utilizza un approccio di machine learning su base semantica basato su tecnologie e paradigmi di analisi di dati, applicati a scenari di edge computing in cui l'intelligenza e la conoscenza sono incorporate in oggetti di uso comune e in micro-dispositivi eterogenei che trasmettono informazioni utili e che contribuiscono a caratterizzare un dato evento/scenario. Tale flusso di informazioni, raccolto in real-time, dovrà essere correttamente strutturato per essere processato automaticamente attraverso strumenti

software. L'approccio presentato è in grado analizzare flussi di dati grezzi dall'ambiente, associando ad essi descrizioni annotate su base semantica, al fine di riconoscere gli eventi caratterizzanti il contesto al quale l'analisi si riferisce. Con l'approccio proposto si vuole superare le attuali limitazioni che caratterizzano i classici scenari edge in cui le informazioni vengono raccolte attraverso micro-dispositivi collegati a elementi comuni o distribuiti in determinati ambienti e interconnessi in modalità wireless. Fondamentalmente, a causa delle loro dimensioni ridotte, tali oggetti hanno capacità di elaborazione minime, ridotte capacità di archiviazione e capacità di comunicazione contenute. Inoltre, producono continuamente dati grezzi che dovranno essere inviati ed elaborati da infrastrutture in cloud. Le tecniche classiche di machine learning sono state ampiamente utilizzate per questo, ma spesso le rappresentazioni degli eventi rilevati non sono completamente gestibili in applicazioni pratiche: ciò è dovuto principalmente alla difficoltà di generare descrizioni interoperabili rispetto ai vocabolari condivisi. Inoltre, le tecniche comunemente adottate presentano due inconvenienti principali: i) spesso svolgono una semplice classificazione partendo dagli input ricevuti; ii) la loro precisione aumenta se applicate su grandi quantità di dati, quindi le analisi online difficilmente raggiungono alte prestazioni in scenari IoT aventi dataset ridotti a causa di requisiti computazionali e di archiviazione. Questi fattori impediscono ancora la possibilità di realizzare oggetti realmente smart, in grado di prendere decisioni e intraprendere azioni a livello locale dopo la fase di rilevamento delle informazioni. Al contrario, l'approccio proposto permetta ai micro-dispositivi pervasivi di poter descrivere se stessi e il contesto in cui si trovano ed elaborare tale annotazione a bordo oppure inviarla verso dispositivi e applicazioni esterne, andando a migliorare l'interoperabilità e la flessibilità degli stessi. Questa visione consente l'autonomia dei sistemi pervasivi basati sulla conoscenza, non ancora consentita dalle tipiche infrastrutture per edge computing. Ne derivano due importanti conseguenze.

- L'interazione uomo-macchina risulta migliorata, riducendo lo sforzo dell'utente necessario per beneficiare dei sistemi di elaborazione. Nei paradigmi edge classici, spesso occorre un'interazione esplicita con un dispositivo per volta per eseguire un'attività. Al contrario, agenti intelligenti eseguiti su dispositivi mobili dovrebbero essere in grado di interagire simultaneamente con molti microcomponenti, fornendo agli utenti attività personalizzate rispetto al contesto e supporto alle decisioni.
- In secondo luogo, anche se le attuali tecniche, algoritmi e strumenti di machine learning hanno consentito nuove classi di analisi (particolarmente utili in ambito Big Data), l'utilizzo di strategie di discovery basate sulla logica ed in grado di identificare anche match parziali proposte in questo progetto compensano possibili errori o anomalie nell'acquisizione dei dati, dovute alla volatilità e mobilità dei dispositivi e all'affidabilità delle comunicazioni wireless. In questo modo sarà possibile sviluppare soluzioni di Edge Intelligence innovative, resilienti e versatili.

Oltre all'adeguatezza in termini di accuratezza della classificazione e sostenibilità computazionale, uno degli obiettivi principali dell'approccio proposto è migliorare l'interpretabilità dei risultati rispetto alle tecniche attualmente disponibili. Si considerino come riferimento, oltre all'approccio proposto, due grandi categorie di algoritmi di classificazione di eventi: alberi decisionali, k-Nearest Neighbors (k-NN) e reti neurali. I modelli generati da questi algoritmi di machine learning sono di seguito confrontati in una valutazione qualitativa.

La Figura 3.5 mostra il modello, ovvero un evento annotato rispetto all'ontologia riportata in Figura 3.2, prodotto attraverso l'approccio proposto. Ogni evento non è semplicemente etichettato, ma è rappresentato attraverso annotazioni elaborabili automaticamente e facilmente leggibili. Come detto, l'utilizzo del matchmaking semantico consente anche una spiegazione dei risultati basata sulla logica oltre a garantire i risultati della classificazione. Inoltre, l'applicazione di servizi di inferenza non standard e le corrispondenze approssimate aumentano la resilienza del modello contro letture dei sensori mancanti o informazioni spurie nei campioni di test durante l'uso del sistema. La Figura 3.6 mostra il modello prodotto attraverso un algoritmo basato su alberi decisionali (per motivi di leggibilità, la figura riporta solo una parte dell'intero modello).

```
Description: UnevenCondition
Types
  (hasAltitudeVariation only
     (LightClimb
      and LightSlope))
   and (hasAverageVehicleSpeed only
     LowVSALevel)
   and (hasEngineCoolantTemperature only
     MediumECTLevel)
   and (hasEngineLoad only
     (EngineLoadLevel1
      and EngineLoadLevel2))
   and (hasEngineRPM only
     (EngineRPMLevel3
      and EngineRPMLevel4))
   and (hasIntakeAirTemperature only
     MediumIATLevel)
   and (hasLongitudinalAcceleration only
     (LowNegativeLongitudinalAcceleration
      and LowPositiveLongitudinalAcceleration))
   and (hasManifoldAbsolutePressure only
     LowMAPLevel)
   and (hasMassAirFlow only
     (MAFLevel1
      and MAFLevel2))
   and (hasVehicleSpeedVariation only
     (SoftVehicleAcceleration
      and SoftVehicleBreaking))
```

Figura 3.5 – Esempio di annotazione di un evento

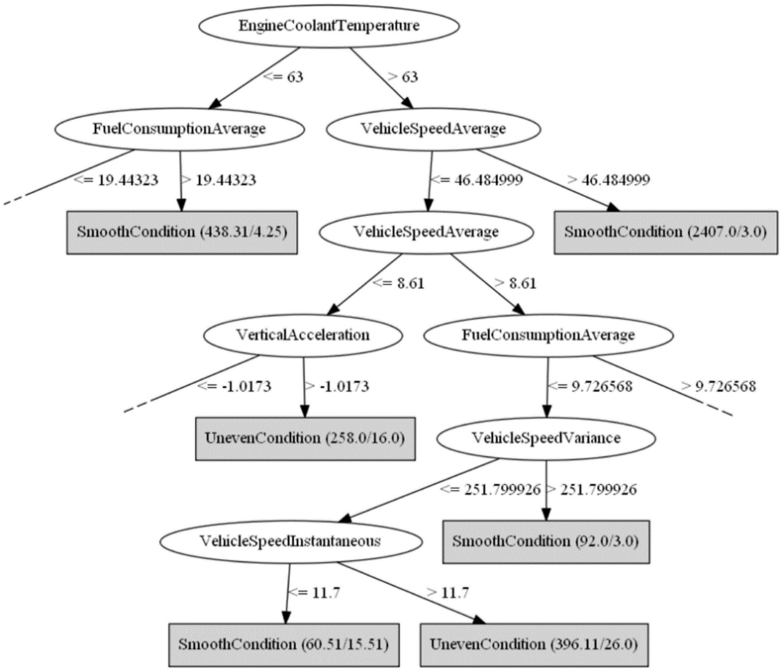

Figura 3.6 – Esempio di classificazione realizzata con alberi decisionali

È possibile notare come ogni nodo verifichi una funzione attraverso una soglia specifica: i rami definiscono i percorsi che conducono ai nodi foglia, che rappresentano gli output della classificazione. Anche in questo caso, il modello è facilmente leggibile sia dagli umani che dai sistemi software: ogni classe è sostanzialmente rappresentata da una clausola in *Disjunctive Normal Form*. Tuttavia, l'uso di soglie numeriche negli atomi proposizionali può rendere l'approccio vulnerabile a lievi variazioni dei dati del campione, ad esempio a causa di problemi di misurazione.

I modelli generati da algoritmi k-NN sono fondamentalmente costituiti da nuvole di punti dove ogni elemento rappresenta un campione di addestramento in uno spazio n-dimensionale, dove n è il numero di features.

Infine, un possibile modello generato attraverso una semplice rete neurale è rappresentato nella Figura 3.7. Gli attributi di input sono riportati in verde, le classi di output in giallo, i neuroni sono organizzati in strati connessi tra loro. Tali modelli sono praticamente delle “black box”, sia per un’interpretazione da parte di utilizzatori che per elaborazioni in sistemi automatici, perché la relazione tra le caratteristiche del campione di input e le etichette della classe di output è codificata solo in soglie di nodo e pesi delle connessioni.

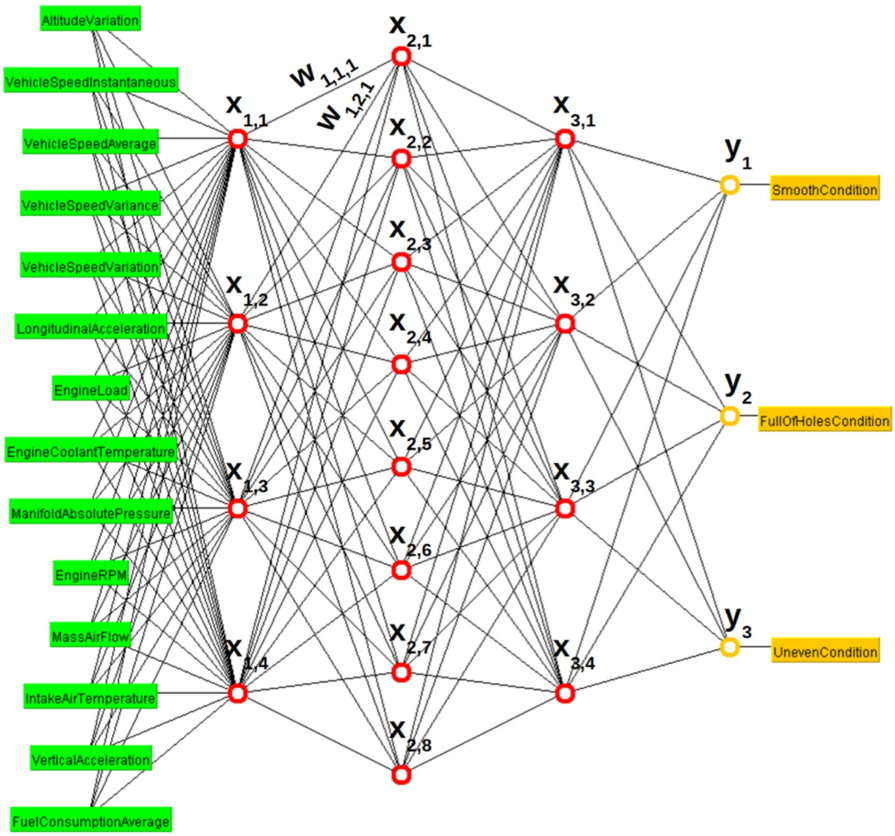

Figura 3.7 – Esempio di classificazione realizzata con reti neurali

Ad esempio, il primo nodo del secondo livello nascosto è modellato come combinazioni dei pesi corrispondenti attraverso una funzione di attivazione, parametrizzata dal valore di soglia del nodo relativo. Di conseguenza, una descrizione significativa non può essere associata alle etichette della classe di output. Inoltre, alcune tecniche basate su reti neurali profonde sono particolarmente vulnerabili alla perturbazione dei dati di input e in presenza di piccole variazioni nei dati possono produrre output ampiamente diversi in modi imprevedibili. Questa mancanza di affidabilità e adattabilità impedisce l'adozione di tali modelli in molte applicazioni di interesse per l'Edge Intelligence. Sono sempre più diffusi e significativi gli sforzi di ricerca per identificare nuovi algoritmi basati su reti neurali che siano più solidi, interpretabili e meno sensibili alle variazioni accidentali o intenzionali delle informazioni, tuttavia presentano problemi simili di utilizzabilità del modello.

Nonostante gli avanzamenti introdotti applicando l'approccio proposto, è possibile identificare possibili sviluppi futuri del lavoro da svolgere per:

- individuare e studiare metodi alternativi per la modellazione della base di conoscenza;
- definire nuovi algoritmi per la generazione delle descrizioni degli eventi;

- ottimizzare gli algoritmi e le strutture dati utilizzate per ridurre ulteriormente i requisiti di elaborazione e la memoria richiesta;
- effettuare un tuning dell'algoritmo su dispositivi reali (single-board computer o smartphone) all'interno di specifici scenari di progetto.

3.3. Bibliografia

[3.1] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, D. Barton, Big data. The management revolution. *Harvard Business Review*, 90(10): 61–67, 2012.

[3.2] F. Scioscia, M. Ruta, G. Loseto, F. Gramegna, S. Ieva, A. Pinto, E. Di Sciascio, Mini-ME matchmaker and reasoner for the Semantic Web of Things. In: Innovations, Developments, and Applications of Semantic Web and Information Systems, pp. 262-294, IGI Global, 2018.

[3.3] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi, *The Description Logic Handbook: Theory, implementation and applications*, Cambridge University Press, 2003.

[3.4] A. Rettinger, U. Lösch, V. Tresp, C. d'Amato, N. Fanizzi, Mining the Semantic Web, *Data Mining and Knowledge Discovery*, 24(3): 613–662, 2012.

4. Progettazione di un framework di context awareness a livello edge

I sistemi di Intelligenza Artificiale capaci di effettuare task complessi su grandi volumi di dati eterogenei si suddividono fondamentalmente in due categorie: (i) quelli basati sulla rappresentazione e manipolazione della conoscenza (KRR), (ii) quelli che utilizzano tecniche di machine learning (come ad esempio le reti neurali). Poiché molte applicazioni lavorano su enormi flussi di dati che richiedono allo stesso tempo elevata affidabilità e tempi di computazione rapidi, spesso risulta necessario adottare delle tecniche di fusione dell'informazione, in maniera tale da ridurre la mole di dati da elaborare.

Per questi motivi, l'enorme quantità di informazione proveniente dai dispositivi IoT vicini alle sorgenti dati può essere elaborata da un insieme di nodi edge che fungono da collettori, ai fini di controllo, coordinamento e cooperazione. In particolare, sfruttando le tecniche e le tecnologie alla base della rappresentazione della conoscenza e del ragionamento automatico è possibile modellare tale informazione basandosi sulla interpretazione formale e rigorosa del suo significato. Questo consente non solo una maggiore interoperabilità tra differenti piattaforme hardware e software, ma anche di applicare delle procedure di ragionamento che consentono di rendere esplicita quella conoscenza che implicitamente è modellata all'interno di una base di conoscenza (KB).

Per le operazioni di raccolta ed elaborazione di dati contestuali, in letteratura sono stati proposti degli approcci che utilizzano formalismi e modelli custom su cui tutte le entità in gioco devono necessariamente accordarsi. Al contrario, l'utilizzo di linguaggi formali standard garantisce un certo livello di astrazione e interoperabilità, adottando vocabolari non ambigui e riutilizzabili (ontologie) finalizzati alla modellazione del contesto e a supportare servizi di inferenza automatizzati per l'implementazione di processi cognitivi di alto livello.

In questo paragrafo è descritto un framework di context awareness che sfrutta la collaborazione tra diversi nodi edge (che fungono da gateway) per l'implementazione di un servizio di inferenza di tipo non-monotono per la fusione dell'informazione raccolta che:

- consente la risoluzione di eventuali inconsistenze;
- garantisce robustezza in caso di informazione spuria.

In scenari operativi in cui è necessario raccogliere e fondere tra loro grandi quantità di dati, una distinzione di base esiste tra le tecniche di High-Level Information Fusion (HLIF) e quelle di Low-Level Information Fusion (LLIF) [4.1]. LLIF tipicamente fa riferimento ad algoritmi di fusione dati applicati a livello sensoristico

che hanno a che fare con una informazione numerica e il cui obiettivo è quello di classificare, identificare e tracciare. Invece, gli algoritmi di HLIF hanno come input una informazione astratta simbolica e mirano ad ottenere una descrizione di alto livello dei fenomeni ed al raffinamento progressivo delle informazioni in conoscenza, cioè informazione disponibile per la decisione o l'azione. Il framework descritto in questo paragrafo delinea un algoritmo che rientra nella categoria HLIF. Inoltre, l'approccio proposto si basa sull'assunzione di mondo aperto (OWA: Open World Assumption), che lo rende adatto anche per l'elaborazione di informazioni incomplete.

4.1. Modello di context awareness

In questo paragrafo è descritto in modo operativo ed intuitivo il modello di context awareness proposto per la fusione di informazioni provenienti da dispositivi IoT. Il punto di partenza è che, a livello generale, un insieme di agenti indipendenti (i vari nodi edge presenti all'interno dell'infrastruttura) scambiano tra loro pacchetti di dati arricchiti semanticamente. Ogni pacchetto contiene le informazioni aggiornate che ciascun nodo possiede sul contesto, espresse in un linguaggio logico derivante dalle Description Logics (DL) e facendo riferimento ad una ontologia condivisa che descrive il particolare dominio o scenario di interesse. In particolare, ciascun pacchetto contiene una quadrupla di informazioni, ognuna caratterizzata da un ben preciso punto di vista sul contesto, come mostrato in Tabella 4.1.

Campo	Descrizione
C (Confirmed)	*Elementi che sono stati osservati sia dal mittente che dagli altri agenti*
X (Clash)	*Rappresenta gli aspetti osservati dal mittente, che sono in contrasto con le informazioni raccolte dagli altri nodi*
M (My)	*Raggruppa tutti gli elementi osservati dal mittente, ma non dagli altri agenti*
E (External)	*Elementi osservati dagli altri agenti, ma non dal nodo mittente*

Tabella 4.1 – Descrizione dei campi di un pacchetto semantico

Nella gestione di tutta l'informazione raccolta ed elaborata, l'approccio proposto segue l'assunzione di mondo aperto, che afferma che l'assenza di informazione non rappresenta un vincolo di negazione, ma semplicemente rappresenta informazione mancante perché irrilevante o sconosciuta. Questo implica, ad esempio, che il campo *X* contiene solo frammenti di conoscenza esplicitamente in conflitto, mentre un

agente considera gli elementi dell'insieme *E* come possibile informazione valida, anche se tale informazione non può essere confermata direttamente. Al contrario, molti sistemi di gestione di basi di dati adottano l'assunzione di mondo chiudo (CWA: Closed World Assumption), in cui l'informazione non memorizzata in maniera esplicita è considerata essere falsa.

Ogni volta che un agente completa un ciclo di raccolta ed annotazione dei dati, esso genera una descrizione semantica *N* che rappresenta l'informazione rilevata e aggiornata sul contesto. Il framework proposto adopera un protocollo di tipo collaborativo per la disseminazione di queste informazioni, che sfrutta algoritmi di integrazione della conoscenza al fine di risolvere eventuali incongruenze. Durante l'arrivo dei pacchetti e la loro memorizzazione in cache, all'interno di un nodo tre operazioni di base possono essere realizzate:

1. Generation: avviene se un agente genera un'annotazione *N* ma non possiede alcun pacchetto valido nella propria cache.
2. Relay: avviene se un agente ha un pacchetto *P* nella propria cache e non è in grado di produrre una propria annotazione semantica.
3. Integration and relay: accade quando un agente possiede almeno un pacchetto *P* valido in cache, la cui informazione deve essere integrata con quella prodotta localmente *N* prima di poter propagare il nuovo pacchetto così generato.

In ogni caso, il risultato delle operazioni precedenti è un nuovo pacchetto *P'* da trasmettere in broadcast agli altri nodi.

Nel caso di "Generation", il pacchetto *P'* risultante è così formato:

$$P' = < C \equiv \top; X \equiv \top; M \equiv N; E \equiv \top >$$

Il risultato mostra come l'annotazione *N* (informazione propria di un nodo) sia inserita nel campo *M* del pacchetto (l'osservazione del contesto da parte del mittente del pacchetto), mentre gli altri campi sono valorizzati a ⊤ (Top o Thing) che rappresenta l'osservazione più generica possibile.

In caso di "Relay", l'agente inoltrerà un pacchetto *P* ricevuto in cache, quindi il nuovo pacchetto *P'* conterrà esclusivamente le informazioni prodotte da altri nodi e sarà così fatto:

$$\square' = < \square \equiv \top; \square \equiv \top; \square \equiv \top; \square \equiv \square_{\square} \sqcap \square_{\square} \sqcap \square_{\square} \sqcap \square_{\square} >$$

Come mostrato in Figura 4.1, il campo *E* è popolato con la congiunzione delle quattro annotazioni presenti in *P*; gli altri sono impostati al valore generico ⊤.

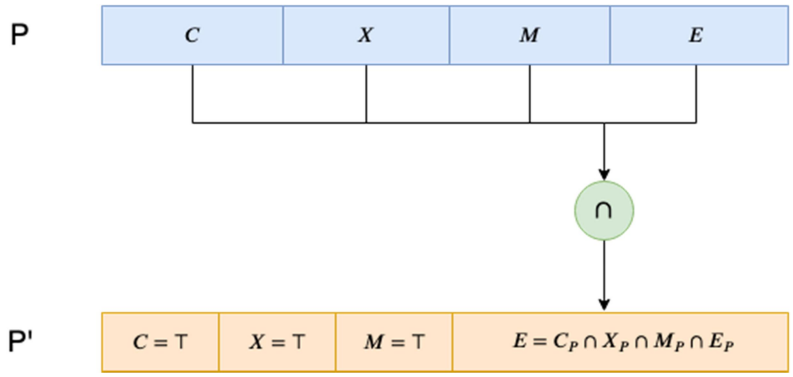

Figura 4.1 – Costruzione di un pacchetto di Relay

È bene evidenziare che, per ciascun pacchetto scambiato, le quattro annotazioni presenti all'interno sono semanticamente compatibili, cosicché la loro congiunzione risulta sempre essere soddisfacibile man mano che nuove osservazioni sono generate dai vari nodi. Tale proprietà è garantita anche per il caso "Integration and relay", che richiede la fusione delle informazioni ricevute nei diversi pacchetti. A tale scopo è stato sviluppato un complesso servizio di inferenza, in grado di preservare il significato dei campi presenti nei pacchetti sia in ingresso che in uscita. Tale algoritmo, dal nome di Concept Integration, è descritto nel paragrafo successivo.

4.2. Integrazione della conoscenza a livello edge

Il problema di Concept Integration è formalizzato nel modo seguente: sia $\mathcal{L}$ un linguaggio logico appartenente alla famiglia delle DL, $\mathcal{T}$ un insieme di assiomi in $\mathcal{L}$, N una descrizione semantica in $\mathcal{L}$ e **P** = (P_1, P_2, …, P_n) un insieme cronologicamente ordinato (dal più al meno recente) di quadruple di espressioni logiche P_i = <C_i, X_i, M_i, E_i> in $\mathcal{L}$, dove N rappresenta la conoscenza attuale dell'agente A, P_i è la conoscenza accumulata dall'agente A_i; N, C_i, X_i, M_i, E_i sono soddisfacibili in $\mathcal{T}$ ∀ i. Risolvere un *Concept Integration Problem* (CIP), identificato da <$\mathcal{L}$, N, P, $\mathcal{T}$>, significa trovare una quadrupla di espressioni logiche P' = <C', X', M', E'>, che è il risultato dell'integrazione di N e P e rappresenta la conoscenza accumulata dall'agente A.

Il caso più semplice di CIP si ha quando la cardinalità dell'insieme **P** è pari ad uno (|**P**| = 1). La procedura utilizzata per la risoluzione del problema in tale casistica è descritta in [4.3] e sfrutta i servizi non-standard di inferenza non monotona di Concept Contraction, Concept Abduction, Difference e Bonus. L'algoritmo per la risoluzione di un CIP nel caso generale (|P| ≥ 1) è mostrato in Figura 4.2. In particolare, gli operatori Contract, Abduce, Bonus e Difference (anche denotato come -) fanno riferimento alle procedure di Concept

Contraction, Concept Abduction, Bonus e Difference rispettivamente. L'algoritmo sviluppato ha come base le seguenti scelte progettuali:

- un congiunto logico può essere confermato se tutti gli altri nodi della rete non hanno rilevato annotazioni in contrato con esso;
- un congiunto logico non è confermato se in contrasto con le annotazioni rilevate da almeno un altro nodo della rete;
- un congiunto logico è un'ipotesi se nessun altro nodo l'ha rilevato.

In particolare, come mostrato nel flowchart di Figura 4.2, dopo aver effettuato il calcolo di variabili ausiliarie (S_i, Q_i, K_i, G_i, H_i), il campo M' del nuovo pacchetto è calcolato andando ad eliminare iterativamente le osservazioni più datate, presenti all'interno del nodo in cui l'algoritmo è applicato, da quelle più recenti.

In Figura 4.3 sono mostrate in dettaglio le operazioni necessarie al calcolo del campo M' di un nuovo pacchetto semantico, nel caso in cui $|\mathbf{P}| = 2$. In questo caso N rappresenta l'informazione posseduta dal nodo Edge che esegue l'algoritmo per la risoluzione del CIP, mentre P_1 e P_2 rappresentano i pacchetti in cache. In dettaglio la procedura realizza i seguenti step:

1. calcola S_1 che rappresenta la congiunzione logica dei campi C_1, X_1, M_1 e E_1;
2. calcola S_2 che rappresenta la congiunzione logica dei campi C_2, X_2, M_2 e E_2;
3. effettua l'operazione di Concept Contraction tra N e la descrizione S_1, generando in output le descrizioni K_1 (Keep) e G_1 (Give-Up);
4. effettua l'operazione di Concept Contraction tra N e la descrizione S_2, generando in output le descrizioni K_2 e G_2;
5. effettua l'operazione di Concept Abduction tra K_1 e S_1, generando in output la descrizione H_1 (Hypothesis);
6. effettua l'operazione di Concept Abduction tra K_2 e S_2, generando in output la descrizione H_2;
7. il nuovo campo M' è il risultato dell'applicazione di una operazione di Double Difference tra le descrizioni H_1 e H_2.

L'operatore di Double Difference prende in input due operandi A e B ed effettua la seguente operazione: A - (A - B). In pratica, data l'espressività del linguaggio logico utilizzato per implementare i diversi servizi di inferenza, tale operazione emula il comportamento dell'intersezione insiemistica (condizione necessaria per la sua applicabilità è che i due operandi non siano in contrasto tra loro).

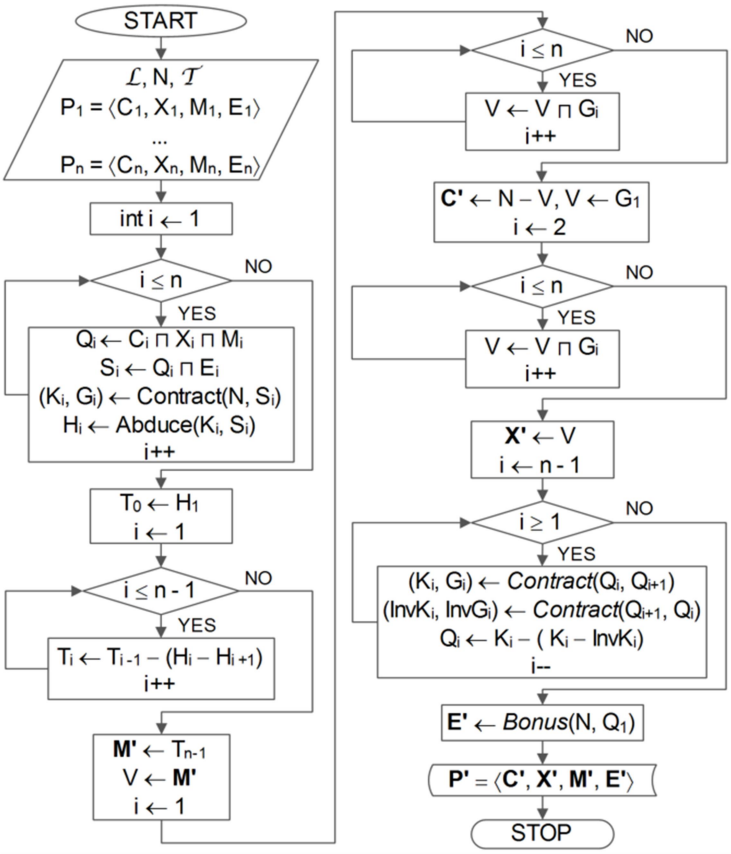

Figura 4.2 – Flowchart relativo al Concept Integration Problem

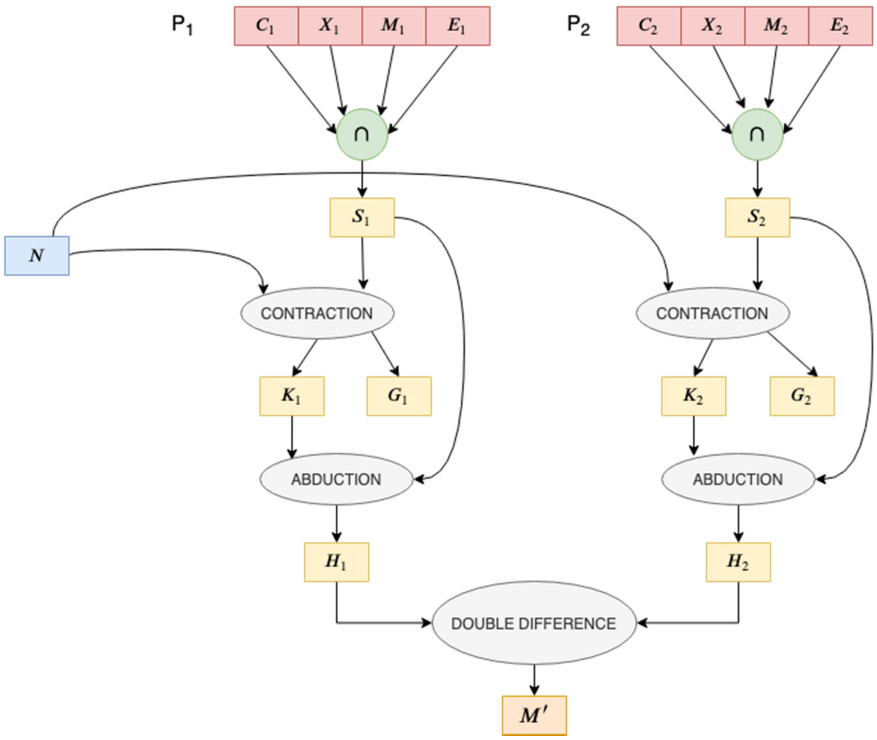

Figura 4.3 – Diagramma operativo per la generazione del campo *M'*

Successivamente, sempre con riferimento al diagramma di flusso di Figura 4.2, è calcolato il nuovo campo *C'* andando a rimuovere dalle informazioni presenti in *N* tutti quegli atomi descrittivi che sono non confermati dagli altri agenti (e quindi presenti nel campo *M'* precedentemente calcolato) oppure semanticamente inconsistenti con le osservazioni pregresse. Per completezza e chiarezza, in Figura 4.4 sono mostrati i dettagli delle operazioni necessarie al calcolo di *C'* nel caso in cui |**P**| = 2. Di seguito i passi eseguiti dall'algoritmo:

1. calcola S_1 che rappresenta la congiunzione logica dei campi C_1, X_1, M_1 e E_1;
2. calcola S_2 che rappresenta la congiunzione logica dei campi C_2, X_2, M_2 e E_2;
3. effettua l'operazione di Concept Contraction tra *N* e la descrizione S_1, generando in output le descrizioni K_1 e G_1;
4. effettua l'operazione di Concept Contraction tra *N* e la descrizione S_2, generando in output le descrizioni K_2 e G_2;
5. crea la variabile ausiliaria *V* mettendo congiungendo il contenuto delle descrizioni *M'*, G_1 e G_2;
6. il nuovo campo *C'* è il risultato dell'operazione di Difference effettuata tra *N* e *V*.

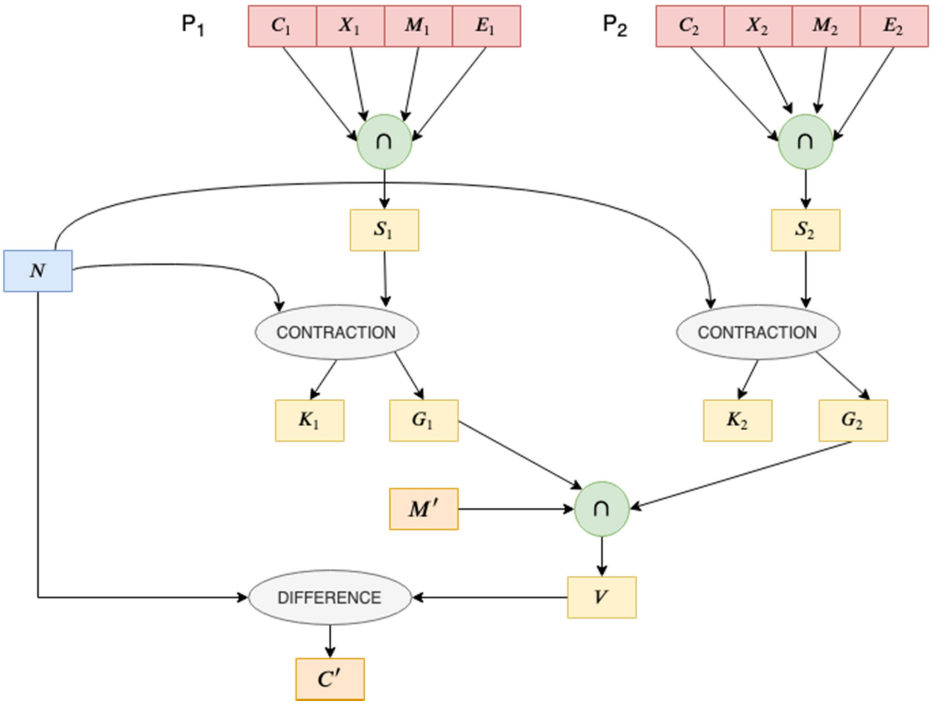

Figura 4.4 – Diagramma operativo per la generazione del campo *C'*

Lo step successivo prevede il calcolo del nuovo campo *X'*, che rappresenta semplicemente la congiunzione di tutti i frammenti descrittivi incompatibili con l'annotazione *N* e calcolati attraverso l'operazione di

Concept Contraction. In particolare, in Figura 4.5 è mostrato il diagramma di flusso dettagliato per il calcolo di *X'* nel caso di |**P**| = 2.

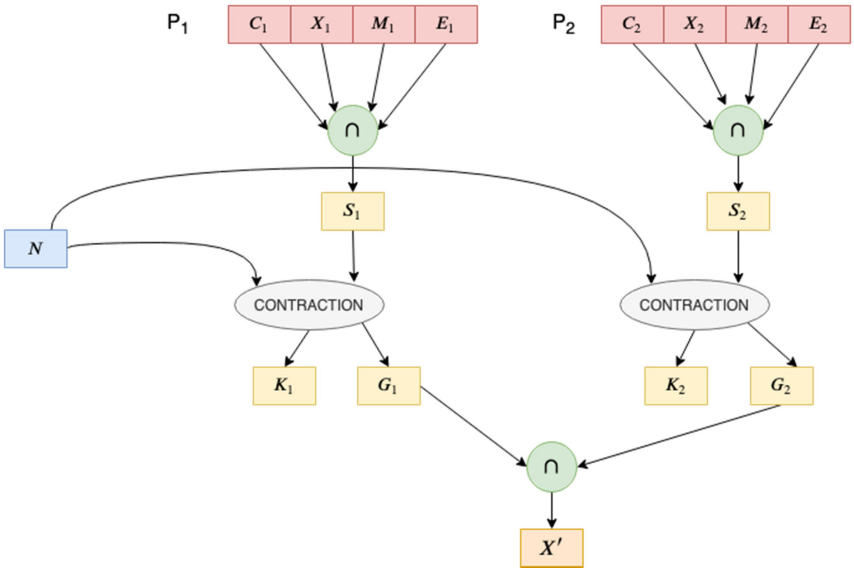

Figura 4.5 – Diagramma operativo per la generazione del campo *X'*

Le operazioni eseguite sono dettagliate di seguito:

1. calcola S_1 che rappresenta la congiunzione logica dei campi C_1, X_1, M_1 e E_1;
2. calcola S_2 che rappresenta la congiunzione logica dei campi C_2, X_2, M_2 e E_2;
3. operazione di Concept Contraction tra *N* e la descrizione S_1, che produce in output le descrizioni K_1 (Keep) e G_1 (Give-Up);
4. operazione di Concept Contraction tra *N* e la descrizione S_2, che produce in output le descrizioni K_2 e G_2;
5. il nuovo campo *X'* è il risultato della congiunzione logica di G_1 e G_2.

L'ultimo campo ad essere calcolato è rappresentato da *E'*. Come descritto nell'algoritmo di Figura 4.2, le descrizioni ricevute dagli altri nodi sono elaborate a coppie in maniera iterativa; da quelle più datate a quelle più recenti. Per ciascuna coppia, l'obiettivo è rimuovere i frammenti mutuamente inconsistenti attraverso l'esecuzione di operazioni di Concept Contraction e Difference. La parte consistente che resta tra le osservazioni più recenti ricevute è utilizzata come argomento in ingresso al servizio di inferenza di Bonus, assieme alla descrizione *N* del nodo Edge, ottenendo in questo modo un frammento di conoscenza che non è implicato da *N* e allo stesso tempo non in contrasto con esso. In questo modo, l'algoritmo di CIP sviluppato consente di integrare la conoscenza pregressa sul contesto con nuova informazione, al fine di

generare una rappresentazione aggiornata che riflette il punto di vista dell'agente che ha realizzato tale integrazione di informazioni.

Anche in questo caso, per completezza in Figura 4.6 è riportato il diagramma di flusso per il calcolo del nuovo campo *E'* in caso di |**P**| = 2, composto dai seguenti passi:

1. calcolo di Q_1 che rappresenta la congiunzione logica dei campi C_1, X_1 e M_1;
2. calcolo di Q_2 che rappresenta la congiunzione logica dei campi C_2, X_2 e M_2;
3. operazione di Concept Contraction tra le descrizioni Q_1 e Q_2, che produce in output le descrizioni K_1 (Keep) e G_1 (Give-Up);
4. operazione di Concept Contraction tra le descrizioni Q_2 e Q_1, che produce in output le descrizioni $InvK_1$ (Keep) e $InvG_1$ (Give-Up);
5. applicazione della Double Difference tra le descrizioni K_1 e $InvK_1$, che produce in output l'annotazione Q_1;
6. il nuovo campo *E'* è il risultato dell'operazione di Bonus effettuata tra le descrizioni *N* e Q_1.

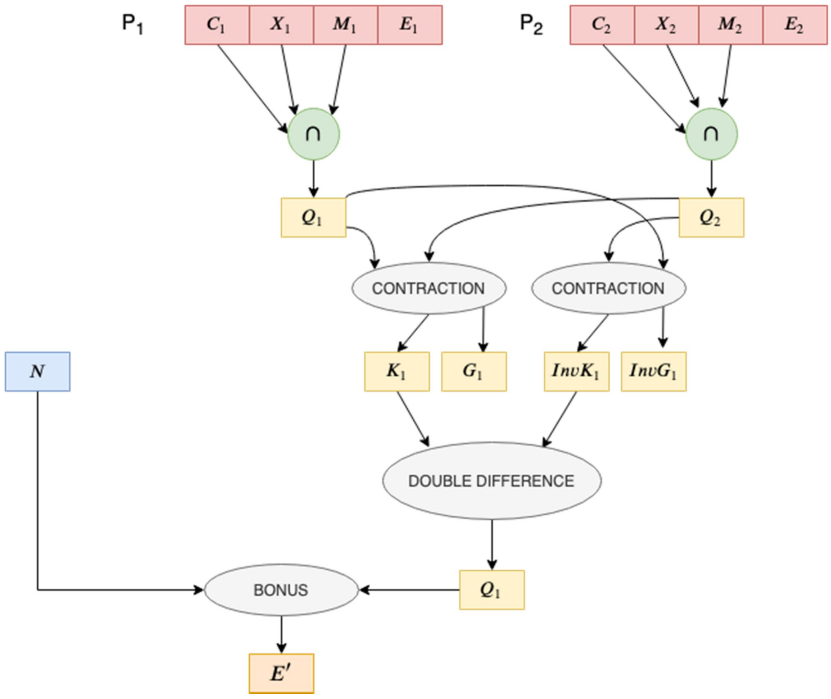

Figura 4.6 – Diagramma operativo per la generazione del campo *E'*

È opportuno notare che Q_1 e Q_2 non comprendono, rispettivamente, i campi E_1 ed E_2: ciò serve a dare un *time to live* di due cicli di CIP e non oltre alle osservazioni compiute da un agente e non confermate da qualcun altro.

Come evidenzia l'algoritmo descritto, il framework proposto soddisfa alcuni dei postulati sulla fusione di informazione descritti in [4.4], tra cui: unanimità, monotonicità dell'informazione, coerenza, efficienza, minimo sforzo. Al contrario, per esplicita scelta di progettazione, esso non è caratterizzato dalle proprietà di:

- imparzialità, poiché privilegia l'informazione più recente;
- insensibilità ad informazioni mancanti, in quanto tratta descrizioni contestuali provenienti da agenti indipendenti, così che il concetto di "dato mancante" (modellato con T) è trattato in modo diverso rispetto a quello di "nessun dato";
- commutatività, poiché integra sempre informazioni dalle più datate alle più recenti.

Queste caratteristiche rendono l'approccio particolarmente indicato per sistemi multi-agente estesi e dinamici, in cui gli agenti possiedono risorse di calcolo grosso modo comparabili: per tale ragione si adatta bene alla fusione di conoscenza tra molteplici nodi di livello edge che raccolgono, elaborano ed annotano i dati provenienti dai dispositivi distribuiti sul campo. Inoltre, la procedura di inferenza sviluppata all'interno dell'algoritmo di CIP è in grado di gestire informazioni incomplete, adattarsi rapidamente ai cambiamenti e si mostra robusta nel gestire informazioni spurie o inaccurate. Ad ogni variazione di dati, l'algoritmo raggiunge uno stato stazionario al più dopo due cicli di osservazione e integrazione dell'informazione [4.3]. Per quanto riguarda la complessità computazionale, le prestazioni dipendono direttamente dalla complessità degli algoritmi di ragionamento di Abduction, Contraction, Bonus e Difference, e di conseguenza dall'espressività della DL scelta. Nel caso $\mathcal{L} = \mathcal{ALN}$ con ontologia $\mathcal{T}$ priva di General Concept Inclusion è possibile eseguire le suddette inferenze mediante algoritmi strutturali [3.3] aventi complessità polinomiale nel tempo (PTIME), da cui deriva che anche l'algoritmo di CIP ha classe di complessità PTIME.

4.3. Benefici attesi e problemi aperti

Risulta difficile effettuare un confronto qualitativo immediato tra il framework proposto per la fusione di informazioni e quello che altre soluzioni esistenti in letteratura offrono in tal senso; questo a causa delle differenze non trascurabili che esistono tra le tecnologie adoperate e gli scenari di riferimento. Tuttavia, in Tabella 4.2 è proposto un confronto tra le caratteristiche principali della soluzione proposta e quelle dei lavori scientifici più rilevanti e simili.

Il framework descritto nel paragrafo precedente non è il solo basato sull'utilizzo di ontologie formalizzate in linguaggi logici derivanti dalle DL, ma altri approcci simili non sfruttano procedure di ragionamento non-standard non-monotone e in questo modo non permettono una fusione della conoscenza "controllata" ed

una certa resilienza nei confronti in informazione incompleta o addirittura inconsistente. In generale, sono in grado di gestire esclusivamente la presenza di informazione consistente. Altri framework analizzati prevedono l'utilizzo di:

- tecniche di machine learning e in particolar modo di reti neurali artificiali;
- reti Bayesiane di tipo probabilistico;
- teoria delle belief function (*a.k.a.* teoria di Dempster-Schafer).

Riferimento	*Radak et al. [4.5]*	*Golestan et al. [4.6]*	*Chen, Shi et al. [4.7]*	*Chen, Hao et al. [4.8]*	*CIP*
Approccio	*Belief function theory*	*Bayesian networks*	*Convolutional Autoencoder Neural Network*	*Convolutional Neural Network*	*KRR*
Rappresentazione del contesto	*Belief mass distribution*	*Multi-entity BN (MEBN) theories*	*Opaca (parametri CANN)*	*Opaca (parametri CNN)*	*Description Logics*
Dominio applicativo	*Generale (probabilistico)*	*Generale (probabilistico)*	*Healthcare*	*Healthcare*	*Generale (ontology-based)*
Modello di fusione dei dati	*HLIF (distributed confidence calculation)*	*HLIF (hybrid MEBN inference)*	*LLIF (CANN training)*	*LLIF (multimodal CNN training)*	*HLIF (non-monotonic reasoning)*
Resilienza verso informazione incompleta o inconsistente	*Si*	*Si*	*No*	*Si (latent factor model)*	*Si*
Orientato a multi-agent-system	*Si*	*Si*	*No*	*No*	*Si*
Orientato ad IoT	*No*	*No*	*No*	*No*	*Si (mobile reasoning)*

Tabella 4.2 – Comparazione tra alcuni approcci cognitivi in letteratura e soluzione di CIP proposta

Gli approcci descritti in [4.5] e [4.6] rientrano nella categoria dei sistemi di elaborazione dati di tipo HLIF e risultano essere abbastanza robusti ed efficaci nei confronti di informazione erronea o mancante, mentre le soluzioni basate sull'utilizzo di reti neurali dettagliate in [4.7] e [4.8] vengono applicate solo in caso di dati ed informazioni di basso livello e non strutturati. La maggior parte dei lavori confrontati è orientata ai

sistemi multi-agente, anche se pochi tengono conto degli specifici e restrittivi requisiti di calcolo necessari in applicazioni IoT.

In sintesi, il framework di context awareness descritto in questo paragrafo garantisce alcune caratteristiche peculiari ed importanti, tra cui:

- una rappresentazione strutturata ed arricchita semanticamente del contesto, basata sull'utilizzo delle tecnologie della rappresentazione della conoscenza. Descrizioni di condizioni o fenomeni importanti sono annotate rispetto ad ontologie, che rappresentano dei vocabolari condivisi relativi ad un particolare dominio di conoscenza, dotate di una semantica formale e machine-understandable.
- un nuovo task di inferenza per Concept Integration, che realizza una operazione di integrazione di osservazioni provenienti da entità indipendenti con risoluzione di eventuali inconsistenze. Tale algoritmo è in grado di adattarsi velocemente ai cambiamenti del contesto ed è robusto nei confronti di eventi spuri dovuti alla mancanza di informazioni o alla presenza di errori nei dati in ingresso. Inoltre, come descritto nel paragrafo precedente, tale algoritmo mostra una complessità nel tempo di tipo polinomiale.

L'algoritmo di integrazione sviluppato è di tipo general-purpose e per questo può essere applicato in diversi ambiti applicativi. In particolare, come spiegato nel paragrafo precedente, le sue caratteristiche lo rendono adatto per la fusione di conoscenza in applicazioni di edge computing. Tuttavia, poiché diverse ontologie possono essere gestite allo stesso tempo da uno medesimo nodo edge mentre la fusione può avvenire solo tra annotazioni che si riferiscono allo stesso dominio, tali applicazioni necessiteranno di una fase preliminare di ontology agreement tra le diverse entità o agenti software.

Un aspetto da approfondire, infine, è quello legato alla sicurezza: a tal proposito è necessario identificare dei meccanismi flessibili e scalabili che siano in grado di garantire l'integrità del dato all'interno di un framework di collaborazione tra nodi eterogenei.

4.4. Bibliografia

[4.1] E. P. Blash, D. A. Lambert, P. Valin, M. M. Kokar, J. Llinas, S. Das, C. Chong, E. Shahbazian, High level information fusion (HLIF): Survey of models, issues, and grand challenges, *IEEE Aerospace and Electronic System Magazine*, 27(9): 4-20, 2012.

[4.2] WRC OWL Working Group. OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C Working Group Note, 2012, https://www.w3.org/TR/owl2-manchester-syntax/.

[4.3] F. Scioscia, M. Ruta, E. Di Sciascio, A swarm of Mini-MEs: reasoning and information aggregation in ubiquitous multi-agent contexts. In 4th OWL Reasoner Evaluation Workshop (ORE 2015), serie CEUR Workshop Proceedings, vol. 1207, CEUR-WS, 2015.

[4.4] D. Dubois, W. Liu, J. Ma, H. Prafe, The basic principles of uncertain information fusion. An organized review of merging rules in different representation frameworks, *Information Fusion*, 32: 12-39, 2016.

[4.5] J. Radak, B. Ducourthial, V. Cherfaoui, S. Bonnet, Detecting road events using distributed data fusion: Experimental evaluation for the icy roads case. *IEEE Transactions on Intelligent Transportation Systems*, 17(1): 184-194, 2016.

[4.6] K. Golestan, B. Khaleghi, F. Karray, M.S. Kamel, Attention assist: A high-level information fusion framework for situation and threat assessment in vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 17(5): 1271-1285, 2016.

[4.7] M. Chen, X. Shi, Y. Zhang, M. Guizani, Deep Features Learning for Medical Image Analysis with Convolutional Autoencoder Neural Network. *IEEE Transactions on Big Data*, 2017.

[4.8] M. Chen, Y. Hao, K. Hwang, Lu Wang, Lin Wang, Disease Prediction by machine learning over Big Data from Healthcare Communities. *IEEE Access*, 5: 8869-8879, 2017.

5. Progettazione di un framework per la valutazione di un grafo argomentativo basato su ragionamento automatico

Questa sezione descrive il framework proposto per la valutazione e l'explanation di un grafo argomentativo mediante formalismi di rappresentazione della conoscenza e servizi di inferenza non standard [5.1]. Il Bipolar Weighted Argumentation Framework (BWAF) [5.2] è il modello di argomentazione di riferimento adottato in virtù della sua maggiore espressività e flessibilità nella valutazione degli argomenti, nell'interazione (ovvero la possibilità di stabilire relazioni ponderate bipolari tra coppie di argomenti) e accettabilità.

La modalità con cui costruire automaticamente il grafo orientato degli argomenti in conflitto in uno specifico contesto e la modalità di valutazione delle relazioni e delle semantiche argomentative sono i due problemi fondamentali delle tecniche di argomentazione. Essi possono essere considerati, per molti versi, ortogonali ed affrontabili separatamente. In linea con la maggior parte della letteratura sui framework di argomentazione astratta, nel framework che si intende proporre, si suppone che il grafo argomentativo sia noto a priori e che il sistema abbia completa conoscenza della topologia di rete di sensori e oggetti intelligenti. La proposta infatti considera un grafo argomentativo ottenuto da messaggi annotati semanticamente scambiati in una rete di dispositivi intelligenti in contesti Semantic Web of Things (SWoT) e si concentra sul processo di argomentazione composto da: (i) valutazione delle relazioni tra argomenti, (ii) valutazione dell'accettabilità dell'argomento, (iii) explanation dei risultati. L'approccio sfrutta servizi di ragionamento non-standard forniti da ragionatori Semantic Web adatti ad ambienti mobili e pervasivi [5.3, 5.4] per trovare un punto di connessione tra la rappresentazione astratta di BWAF e l'argomentazione strutturata. Inoltre, le inferenze adottate consentono l'explanation delle proprietà dell'argomentazione, dando origine a una pipeline di argomentazione SWoT general-purpose interpretabile.

Al fine di definire un grafo orientato e pesato, ad ogni dispositivo è associato un'annotazione semantica che rappresenta la propria conoscenza condivisa sul campo. Le annotazioni sono espresse come descrizioni concettuali rispetto ad un'ontologia di dominio comune $\mathcal{T}$ e sono condivise secondo la topologia di rete. L'insieme delle annotazioni scambiate assume il ruolo dell'insieme di argomenti $\square$ nel BWAF $\square$ e l'insieme delle interazioni tra coppie di dispositivi coincide con l'insieme delle relazioni $\mathcal{R}'$ tra i corrispettivi argomenti. Il tipo di relazione tra coppie di argomenti e il corrispondente peso sono definiti sfruttando i servizi di inferenza per il matchmaking semantico [5.1] secondo il seguente approccio.

Dati due dispositivi intelligenti generici $\square_{\square}$ e $\square_{\square}$ in una rete SWoT, le loro annotazioni – indicate rispettivamente con $\square$ e $\square$ – sono trattate come argomenti astratti, poiché esprimono la conoscenza posseduta dai dispositivi. Se $\square_{\square}$ comunica con $\square_{\square}$ ($\square_{\square} \rightsquigarrow \square_{\square}$), per valutare l'accettabilità di $\square$ e $\square$ l'approccio basato sul matchmaking proposto considera $\square$ come una richiesta e $\square$ come una risorsa, poiché negli scenari SWoT $\square$ può essere usato per "rispondere alle necessità" espresse in $\square$. Questa peculiarità mostra che può esistere una possibile relazione argomentativa con orientamento dell'arco da $\square$ a $\square$.

Una volta definito l'orientamento della relazione, l'algoritmo illustrato in Figura 5.1 esegue la valutazione del suo tipo, discriminando se $\square$ attacca o supporta $\square$. Innanzitutto, è eseguito un check di *compatibilità semantica*: se $\square \sqcap \square$ è consistente, allora la relazione che collega $\square$ a $\square$ è un *supporto*, altrimenti è un *attacco*.

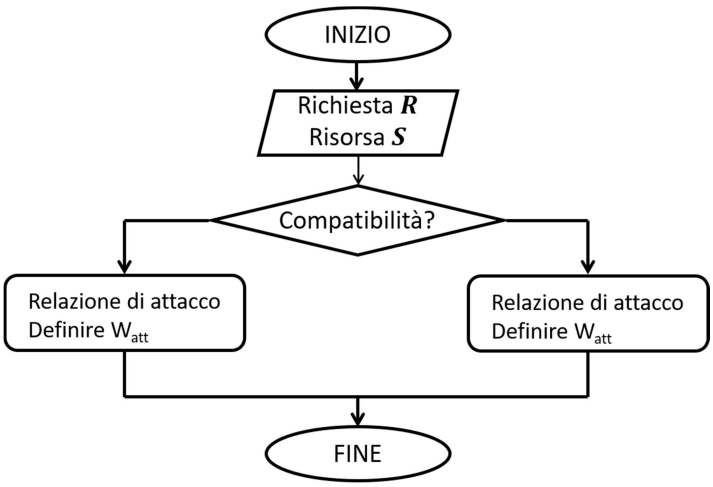

Figura 5.1 – Algoritmo di definizione del tipo di relazione tra coppie di argomenti

A seguito del controllo preliminare di compatibilità, sono state opportunamente determinate, adoperando i servizi di inferenza non standard, due differenti strategie per assegnare ad ogni arco che connette due argomenti il congruo peso. Questo processo va svolto iterativamente per ogni coppia di nodi del grafo in interazione tra loro.

5.1. Valutazione del peso di supporto

Considerando che nel BWAF il peso della relazione di supporto è un numero reale appartenente all'intervallo $]0,1]$, al fine di definire opportunamente una formulazione di peso che possa essere congruente con la nozione di relazione di supporto in un grafo argomentativo sono stati adoperati i servizi di inferenza non-standard di Concept Abduction e Bonus. In particolare, la valutazione quantitativa si basa sulla *norma* $||\square||$ di un'espressione di concetti $\square$ in Description Logic (DL) e sulle *penalty* $\square_{\square}$ *e* $\square_{\square}$ rispettivamente indotte dai servizi di ragionamento non-standard di Concept Abduction e Concept Contraction, così come calcolate dagli algoritmi di inferenza *strutturali* per la DL $\mathcal{ALN}$ definiti in [5.1]. Di qui in avanti si suppone perciò che le annotazioni di tutti i dispositivi nella rete siano state sottoposte ad *unfolding* rispetto agli assiomi di una ontologia di riferimento comune e a normalizzazione in *forma normale congiuntiva*, come precondizione per l'applicazione di tali algoritmi [5.1].

Analizzando rigorosamente il significato concettuale di relazione di **supporto** tra due nodi del grafo, in Figura 5.2 ad esempio si osserva come $\square$ supporti $\square$ e, in considerazione del fatto che in caso di relazione di supporto è garantita la compatibilità semantica tra le due risorse $\square$ e $\square$, hanno rilevanza nella valutazione del peso dell'arco la quantità di informazione che manca in $\square$ per raggiungere la compatibilità completa con $\square$ e l'informazione che $\square$ possiede in aggiunta a quelle che $\square$ ha e di cui non fa dichiarazione esplicita, secondo l'assunzione di mondo aperto. È possibile valutare questi due contributi per mezzo dei servizi di inferenza non-standard, ed in particolare il primo termine è valutato sfruttando la Concept Abduction e il secondo mediante il Bonus.

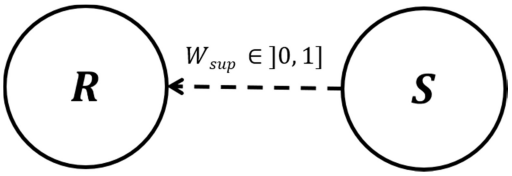

Figura 5.2 – Esempio di relazione di supporto

Il peso del supporto è così definito:

$$W_{sup}(R,S) \stackrel{\text{def}}{=} 1 - \frac{p_a(R,S)}{\|R\|} + \frac{\|Bonus\ (R,S)\|}{\|R\| \cdot \|S\|} \tag{5.1}$$

In evidenza del fatto che il servizio di inferenza non standard di Concept Abduction permette di quantificare l'informazione mancante nella risorsa per soddisfare la completa compatibilità con la richiesta, nella definizione (5.1) per quantificare la rilevanza del supporto che il nodo S esercita su R è stata sottratta al

peso di supporto massimo pari a 1 la penalty della Concept Abduction adoperando come "richiesta" del servizio di inferenza la descrizione dell'annotazione semantica del nodo R e come "risorsa" quella del nodo S, dividendola per la norma di R.

Inoltre, per quantificare l'informazione che il nodo S potrebbe possedere in più rispetto a quella che R ha, è stata calcolata la norma risultante dall'applicazione del Bonus, adoperando come richiesta del servizio di inferenza l'annotazione di R e come risorsa quella di S, dividendola per il prodotto della norma di R ed S. L'operazione di divisione è stata necessaria per normalizzare opportunamente in entrambi i casi i contributi calcolati rispetto al numero di concetti nelle descrizioni di R e S.

Il massimo supporto di S a R avviene quando $S \sqsubseteq R \Leftrightarrow H \equiv \top \Rightarrow p_a(R,S) = 0$, mentre il supporto minimo si ha quando $H = R \Rightarrow \frac{p_a(R,S)}{\|R\|} = 1$.

È importante osservare che i segni che precedono i due contributi sono opposti: il primo termine è utilizzato per valutare la quantità di informazione che non permette il raggiungimento del supporto massimo, e pertanto ha segno negativo; il secondo termine è preceduto dal segno positivo in quanto per definizione il Bonus non è in contrasto con R e in riferimento al fatto che tipicamente nel matchmaking semantico il Bonus rappresenta qualcosa che S dice in più e che, in virtù del principio di *Open World Assumption* (OWA), è potenzialmente utile per R, ma che R nella sua descrizione ha trascurato di specificare.

5.2. Valutazione del peso di attacco

Dalla formalizzazione teorica del BWAF è noto che il peso della relazione di attacco è un numero reale compreso nell'intervallo $[-1,0[$. Come per il supporto, utilizzando opportunamente i servizi di inferenza non standard di Concept Contraction, Concept Abduction e Bonus è stata definita una formulazione che possa fornire l'entità del peso di attacco tra due nodi.

Da quanto riportato in precedenza, una relazione di attacco come quella di Figura 5.3, in cui l'argomento $\square$ attacca $\square$, è discriminata da una di supporto dal fatto che vi è incompatibilità tra le annotazioni semantiche degli individui $\square$ e $\square$.

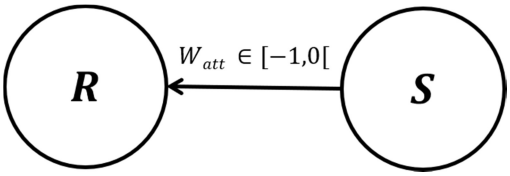

Figura 5.3 – Esempio di relazione di attacco

Considerando questa assunzione, per la definizione di peso di attacco sono sommati algebricamente i quattro seguenti contributi informativi:

- la quantità d'informazione in conflitto tra i due argomenti che partecipano alla relazione;
- la quantità di informazione confermata da entrambi gli argomenti;
- la quantità d'informazione presente nell'argomento attaccato e di cui non vi è definizione esplicita nell'argomento attaccante;
- l'eventuale quantità d'informazione presente nell'attaccante in più rispetto all'argomento attaccato, che comunque non inficia quest'ultimo.

Ad ognuno dei quattro contributi è assegnato un differente segno e peso nella somma algebrica pesata, poiché si vuole dare una rilevanza massima al contributo informativo in contrasto tra i due argomenti e, dato che il peso di una relazione di attacco è un numero reale negativo, il primo contributo è preceduto dal segno meno e ha peso pari ad 1.

I restanti tre termini sono adoperati per indebolire la forza dell'attacco e pertanto sono preceduti da segno positivo e hanno rispettivamente pesi – determinati empiricamente a partire da prove preliminari – pari a $\beta = 0.1, \gamma = 0.05, \delta = 0.05$. Questa scelta è motivata dal fatto che il secondo termine quantifica il contenuto informativo in comune tra i due argomenti in conflitto mentre i restanti due valutano quanto uno dei due argomenti possiede informazioni che l'altro trascura.

Questa interpretazione ha pertanto condotto alla seguente definizione (5.2) di peso dell'attacco:

$$W_{att}(R,S) \stackrel{\text{def}}{=} -\frac{p_c(R,S)}{\|R\|} + \beta\frac{\|Bonus(Bonus(K,S),S)\|}{\|R\|} + \gamma\frac{p_a(K,S)}{\|R\|} + \delta\frac{\|Bonus(R,S)\|}{\|R\|\cdot\|S\|} \tag{5.2}$$

Data la situazione di attacco e la mancata compatibilità tra gli argomenti R e S, al fine di quantificare l'informazione in conflitto è adoperato il servizio di inferenza non-standard di Concept Contraction. Considerato che la penalty della Concept Contraction $p_c(R,S)$ è valutata sulla *Give up* G al fine di quantificare gli elementi in conflitto tra i due argomenti, essa è utilizzata adoperando come richiesta del servizio di inferenza l'annotazione dell'argomento R e come risorsa quella dell'argomento S, dividendo la penalty per la norma di R, poiché quest'ultima risulta pari al peggior valore possibile di $p_c(R,S)$, quando tutte le informazioni in R sono rigettate da S.

La valutazione del secondo termine della definizione (5.2), che consente di quantificare i concetti in comune tra i due argomenti, avviene utilizzando due volte il servizio di Bonus. Il risultato dell'applicazione del Bonus utilizzando come richiesta del servizio di inferenza la descrizione semantica della *Keep* K della

precedente Concept Contraction e come risorsa quella dell'argomento S è impiegato come richiesta nella seconda applicazione del servizio di Bonus, che ha nuovamente come risorsa la descrizione dell'argomento S. L'applicazione del servizio di Bonus annidato consente infatti di individuare i concetti in comune tra i due argomenti originali che partecipano alla relazione e la norma di questo contributo è divisa per la norma di R in modo che il secondo termine sia sempre ≤ 1.

Il contributo che quantifica l'informazione presente nell'argomento attaccato e di cui non vi è definizione esplicita nell'argomento attaccante è determinato dalla penalty della Concept Abduction, adoperando come richiesta del servizio di inferenza la *Keep* K della precedente Concept Contraction e come risorsa l'annotazione di S, divisa per la norma di R. La motivazione per cui viene valutata questa Abduction sta nel fatto che essa quantifica i concetti dell'argomento attaccante che mancano per soddisfare completamente la *Keep* K che descrive la parte non in contrasto tra l'argomento attaccante e quello attaccato.

Infine, per quantificare l'informazione che l'argomento attaccante S possiede in più rispetto all'argomento attaccato R, è stata calcolata la norma risultante dall'applicazione del Bonus, adoperando come richiesta del servizio di inferenza R e come risorsa S, dividendola poi per il prodotto della norma delle due annotazioni.

5.3. Processo di argomentazione

L'approccio proposto può essere integrato vantaggiosamente con qualsiasi classico framework di argomentazione astratta. Tuttavia, considerata la propensione del matchmaking semantico a svolgere analisi accurate e dettagliate, esso risulta particolarmente adatto per la famiglia BWAF (Bipolar Weighted Argumentation Framework), che supporta sia le relazioni di attacco che quelle di supporto, con un peso assegnato per rappresentare il tipo e la forza dell'interazione.

L'analisi della letteratura condotta conferma come il BWAF sia nato dall'esigenza di poter usufruire di due aspetti dell'Argomentazione Astratta mantenuti precedentemente separati: il concetto di supporto tra argomenti (tipico del *Bipolar Argumentation Framework*, BAF), ed il concetto di forza dell'attacco tra argomenti (tipico del *Weighted Argumentation Framework*, WAF). L'idea è quella di fornire un unico tipo di arco tra nodi (argomenti) con un peso, ovvero un valore numerico reale, che indica un attacco per valori compresi nell'intervallo $[-1,0[$, o un supporto per valori compresi nell'intervallo $]0,1]$.

Un BWAF può essere rappresentato come un grafo diretto:

- i cui nodi rappresentano argomenti,

- le cui relazioni rappresentano attacchi (con archi solidi dall'attaccante al nodo attaccato) e supporti (con archi tratteggiati)
- i cui pesi rappresentano la forza relativa delle relazioni.

Per una rappresentazione grafica intuitiva si osservino la Figura 5.2 e la Figura 5.3.

In contesti SWoT, dove è comprovata la massiva presenza di dispositivi dotati di capacità di rilevamento e annotazione dei dati, il grafo argomentativo BWAF si può considerare come derivato fedelmente dalla topologia di rete di oggetti intelligenti nello scenario, utilizzando una base di conoscenza (KB) modellata in DL $\mathcal{ALN}$.

La Terminological Box (TBox) $\mathcal{T}$ contiene tutte le classi utilizzate per descrivere opportunamente i dispositivi e le caratteristiche di contesto dello scenario che si vuole adoperare. Ogni dispositivo è modellato come individuo dell'Assertion Box (ABox) e la sua espressione concettuale è usata come singolo argomento nel BWAF. I servizi di inferenza standard e non-standard menzionati nel paragrafo precedente, disponibili su motori inferenziali orientati a scenari SWoT come *Mini-ME* (*Mini Matchmaking Engine*) [5.1], sono sfruttati per la valutazione di tipo e peso della relazione tra coppie di dispositivi presenti nello scenario (ovverosia, nodi del BWAF) come descritto nelle precedenti sezioni.

A seguito della definizione di un grafo argomentativo bipolare pesato, il framework proposto esegue il processo di argomentazione utilizzando un motore di ragionamento argomentativo compatibile come *ARGUER (ARGuing Using Enhanced Reasoning)* [5.2], capace di calcolare tutte le semantiche proposte in letteratura e di gestire varie generalizzazioni di Argumentation Framework come VAF, BAF, PAF, WAF, SAF, BWAF. Questo sistema è completo sia di interfaccia testuale interattiva che di una modalità automatica in grado di ricevere in input una base di conoscenza da file di testo.

La valutazione dell'argomentazione strutturata basata sul matchmaking semantico viene eseguita per individuare interazioni "influenti" o argomenti "forti" nel framework. Ciò richiede spesso l'utilizzo di semantiche basate su ranking [5.5] a discapito di semantiche basate su estensione.

Il principale svantaggio delle semantiche basate su estensione è infatti quello di non sfruttare appieno il peso delle relazioni: l'ammissibilità per sottoinsiemi di argomenti è definita senza considerare la forza effettiva delle relazioni e come queste forze influenzano la valutazione complessiva. Le semantiche basate su estensione dichiarano tutti gli argomenti giustificati di un framework argomentativo come ugualmente accettati. Le semantiche basate sul ranking tengono invece conto della gradualità degli stessi.

Un fattore incisivo nella valutazione di accettabilità degli argomenti è stimare quanto il peso delle relazioni di attacco e di supporto influenzi la forza complessiva di un cammino tra due argomenti, in quanto spesso

tra due argomenti possono esistere più percorsi diversi che coinvolgono sia relazioni di attacco sia di supporto. In aggiunta, un argomento può essere coinvolto in molti cicli, ognuno dei quali può contenere, a sua volta, argomenti coinvolti in altri cicli e così via.

La semantica che tiene conto della *strength propagation* tra argomenti e classifica gli argomenti confrontandoli con un grado di accettabilità numerica assegnato a ciascuno è la semantica *sp-ranking* [5.2].

Il framework argomentativo proposto integra i formalismi di rappresentazione della conoscenza e il ragionamento automatico, consentendo di valutare il rank di argomenti prodotto dalla semantica *sp-ranking* e di compararlo ai risultati prodotti dalle altre semantiche basate su estensione valutando i benefici che comporta l'adozione di ciascuna di esse. L'accurata analisi dei risultati prodotti consentirà di valutare la potenzialità del ragionamento argomentativo al fine di determinare l'appropriatezza e l'affidabilità di una decisione presa automaticamente dalla rete di dispositivi presenti nello scenario di riferimento.

5.4. Benefici attesi e problemi aperti

Rispetto ai problemi tipici di veridicità e comparazione delle informazioni raccolte da gruppi di entità eterogenee in contesti SWoT e di affidabilità delle decisioni prese o suggerite da reti mobili ad-hoc di dispositivi in maniera autonoma, il framework presentato offre un approccio originale e innovativo, basato sull'integrazione di un framework bipolare pesato per l'argomentazione astratta (BWAF) con una valutazione della tipologia e del peso delle relazioni in un grafo argomentativo per mezzo dei formalismi di rappresentazione della conoscenza e del ragionamento deduttivo basati sulle Logiche Descrittive.

Lo scambio di frammenti di conoscenza e le costanti interazioni machine-to-machine sostengono i protocolli collaborativi per il coordinamento e il supporto alle decisioni context-aware. Tali interazioni dinamiche possono essere considerate come un dialogo o una conversazione in corso tra oggetti, che si comportano come agenti indipendenti.

Questa visione si presta molto bene ai paradigmi dell'argomentazione classica [5.6], come sostenuto in [5.7], dove si mostra che le diverse forme di dialogo umano si adattano ai requisiti di cooperazione e coordinamento delle interazioni sociali tra oggetti nell'Internet of Things (IoT), rappresentabili per mezzo di tecniche di argomentazione. Gli Speaking Objects –oggetti che interagiscono mediante argomentazione– possono scambiare argomenti al fine di cercare informazioni, negoziare su una problematica, persuadere gli altri, deliberare azioni, sforzandosi di raggiungere un consenso sullo stato delle cose e dei propri obiettivi.

Tuttavia, la corretta integrazione dei framework di argomentazione nei contesti SWoT resta ancora un problema aperto.

La metodologia proposta in questa sezione ha la peculiarità di poter modellare gli argomenti di un grafo argomentativo come annotazioni nei linguaggi del Semantic Web, che è un interessante avanzamento rispetto allo stato dell'arte. Le inferenze non-standard e non-monotone, dotate di giustificazione formale dei risultati, consentono una valutazione strutturata delle relazioni tra coppie di argomenti, mediante il calcolo del peso dell'attacco e del supporto con formule opportunamente studiate nel framework proposto. Un vantaggio peculiare dell'approccio è proprio la capacità di fornire una explanation su base logica dei pesi assegnati, giustificando il comportamento del sistema.

L'adozione del framework argomentativo BWAF è motivata dal fatto che esso appare versatile ed efficace, in quanto permette di assegnare a ciascuna relazione un peso che può essere positivo o negativo, consentendo quindi non solo relazioni di attacco pesate tra argomenti astratti, ma anche relazioni di supporto pesate.

La proposta è general-purpose ed è motivata dalla propensione a rendere autonomo e collaborativo il processo decisionale e dalla necessità di valutare l'affidabilità delle informazioni nelle reti SWoT di oggetti intelligenti.

Per quel che riguarda l'accettabilità degli argomenti, mentre le semantiche basate su estensione non sfruttano appieno il peso delle relazioni e dichiarano tutti gli argomenti giustificati di un framework argomentativo come ugualmente accettabili, le semantiche basate sul ranking tengono conto della gradualità, valutando quanto il peso delle relazioni di attacco e di supporto influenzino la forza complessiva di un cammino tra due argomenti. Entrambe le tipologie di semantiche sono utilizzabili nel framework che si propone.

Il particolare approccio che si propone descrive come le potenzialità della rappresentazione della conoscenza e dei servizi di inferenza per il matchmaking semantico possano essere integrate vantaggiosamente nella definizione e valutazione del grafo argomentativo, ed in particolare per un calcolo accurato e coerente dei pesi degli archi tra nodi. È significativa la convergenza tra l'utilizzo dei servizi di inferenza e i risultati che essi producono con la definizione formale di BWAF, in cui i pesi degli archi del grafo assumono notevole rilevanza al fine di permettere a semantiche argomentative basate sul ranking di valutare in modo corretto e realistico grafi complessi.

La proposta avanzata per l'individuazione del peso dell'arco appare, dal punto di vista strutturale, completa ed esaustiva, in quanto consente, grazie alle annotazioni semantiche di ogni nodo del grafo, di quantificare

dettagliatamente come ciascuna parte della descrizione di un nodo interagisca con quella del nodo con cui è in relazione. La definizione di tipologia e peso delle relazioni tra argomenti del grafo, mediante i servizi di inferenza non-standard, si basa infatti sulle proprietà degli algoritmi di inferenza strutturali definiti in [5.1]. È ipotizzabile che al fine di inferire maggiore conoscenza da adoperare ad ogni variazione nel contesto all'interno degli scenari SWoT –per natura altamente volatili– si possa prevedere l'incorporazione (embedding) nel grafo argomentativo, oltre ai pesi numerici, delle explanation semantiche delle relazioni tra gli argomenti. Ciò conferirebbe al grafo un significativo valore aggiunto dal punto di vista informativo, ed inoltre permetterebbe di definire filtri dinamici sugli argomenti nel grafo sulla base di specifici criteri.

Una limitazione significativa del framework illustrato in questo capitolo è che l' algoritmo di valutazione viene attualmente calcolato in modo centralizzato, conoscendo in anticipo l'intero grafo argomentativo. Inoltre, i requisiti relativi alle prestazioni e alle risorse computazionali non risultano ancora ottimizzati: il problema aperto più importante è come evitare di ricalcolare l'intero ranking BWAF ogni volta che cambia il grafo (situazione che può verificarsi molto di frequente negli scenari SWoT). Tutti questi aspetti dovrebbero essere risolti studiando una più profonda integrazione della valutazione delle relazioni e del ranking BWAF, al fine di adattare l'intero framework ai contesti distribuiti.

5.5. Bibliografia

[5.1] M. Ruta, E. Di Sciascio, F. Scioscia, Concept Abduction and Contraction in Semantic-based P2P Environments. *Web Intelligence and Agent Systems*, 9(3): 179-207, 2011.

[5.2] A. Pazienza, S. Ferilli, F. Esposito, Constructing and Evaluating Bipolar Weighted Argumentation Frameworks for Online Debating Systems. In: Advances in Argumentation in Artificial Intelligence Workshop, 16th International Conference of the Italian Association for Artificial Intelligence (A^3I@AI*IA), pp. 111-125, 2017.

[5.3] M. Ruta, F. Scioscia, F. Gramegna, I. Bilenchi, E. Di Sciascio, Mini-ME Swift: the first mobile OWL reasoner for iOS. In: 16th Extended Semantic Web Conference, pp. 298-313, Springer, 2019.

[5.4] F. Scioscia, M. Ruta, G. Loseto, F. Gramegna, S. Ieva, A. Pinto, E. Di Sciascio, Mini-ME matchmaker and reasoner for the Semantic Web of Things. In: Innovations, Developments, and Applications of Semantic Web and Information Systems, pp. 262-294, IGI Global, 2018.

[5.5] E. Bonzon, J. Delobelle, S. Konieczny, N. Maudet, A comparative study of ranking-based semantics for abstract argumentation. In: Thirtieth AAAI Conference on Artificial Intelligence, pp. 914-920, 2016.

[5.6] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2): 321-357, 1995.

[5.7] M. Lippi, M. Mamei, S. Mariani, F. Zambonelli. An argumentation-based perspective over the Social iot, *IEEE Internet of Things Journal*, 5(4): 2537–2547, 2018.

6. Conclusioni

Partendo dalle limitazioni esistenti nelle attuali piattaforme per edge computing, all'interno di questa attività è stato realizzato lo studio di un framework in grado di supportare approcci innovativi per l'apprendimento automatico su base semantica applicato a flussi di dati eterogenei in scenari orientati all'Internet of Things. Il sistema integra concetti derivanti dall'osmotic computing per eseguire la personalizzazione dinamica di differenti microservizi in ambienti intelligenti sfruttando le risorse delle infrastrutture edge e cloud.

Nell'approccio orientato all'analisi dei dati su piattaforma edge, è stato proposto un mapping dei dati grezzi utilizzando etichette concettuali basate su un'ontologia di riferimento che permette un'interpretazione semantica a differenti livelli di astrazione della distribuzione statistica delle informazioni, mentre l'aggregazione congiunta di componenti concettuali consente di costruire automaticamente una rappresentazione ricca e significativa degli eventi durante la fase di addestramento del modello. Infine, l'utilizzo di inferenze non standard per il matchmaking semantico consente un rilevamento di eventi con una elevata specificità trattando il classico problema della classificazione, tipico degli algoritmi di Machine Learning, attraverso un approccio basato sul discovery di risorse. Utilizzando l'approccio proposto, successive valutazioni effettuate all'interno degli scenari di riferimento permetteranno di valutare le prestazioni del framework proposto rispetto alle tecnologie attuali per l'analisi di grandi moli di dati, al fine di evidenziare i vantaggi e i limiti della proposta. I principali benefici attesi sono legati alle minori richieste in termini di risorse di elaborazione, combinati con risultati di classificazione più espressivi e modelli facilmente interpretabili. In generale, il vantaggio principale derivante dall'utilizzo delle tecnologie del Semantic Web è l'estrazione di informazioni significative dai dati rese facilmente condivisibili con ulteriori applicazioni e processi di interesse.

Numerose ed interessanti prospettive future sono aperte partendo dalle soluzioni proposte. Per l'algoritmo di analisi di stream di dati è possibile individuare un'estensione in cui un modello possa essere in continua e dinamica evoluzione utilizzando meccanismi di caching in grado di gestire opportunamente i campioni di training sulla base di finestre temporali. Un'ulteriore estensione dell'algoritmo di classificazione potrebbe mirare a distribuire l'elaborazione su più nodi edge in parallelo utilizzando una fase finale di fusione dei singoli risultati. Ulteriori varianti potrebbero aumentare la flessibilità dell'approccio proposto in fase di classificazione. Ad esempio, potrebbe essere utile studiare la possibilità di creare superclassi dinamiche con un intervallo che combini i valori dei concetti che compongono la descrizione di ciascun evento: ciò eviterebbe di influenzare il risultato degli algoritmi di inferenza in presenza di descrizioni troppo simili.

Inoltre, l'adozione di un linguaggio logico più espressivo come $\mathcal{ALN}$(D) per modellare le ontologie di dominio potrebbe consentire l'introduzione di *datatype properties* per caratterizzare meglio le caratteristiche tipiche dei dati provenienti da sensoristica IoT.

Infine, in relazione all'utilizzo di tecniche di ragionamento argomentative, ulteriori studi potranno riguardare l'integrazione degli output delle inferenze direttamente nelle relazioni dello stesso grafo argomentativo. Ciò migliorerebbe ulteriormente l'interpretabilità dell'approccio, rendendo immediati la condivisione e l'utilizzo del modello e dei risultati. Inoltre, si tratta di una condizione preliminare per un'estensione dell'approccio basata su una modellazione del grafo argomentativo correlata ai vincoli dello scenario, in cui sia possibile abilitare l'attivazione, la disattivazione o la modifica dei nodi in base alle condizioni del contesto e cambiare i risultati dell'argomentazione di conseguenza.